CENTRO UNIVERSITÁRIO FEEVALE
INSTITUTO DE CIÊNCIAS EXATAS E TECNOLOGICAS
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

UM MODELO DE FERRAMENTA PARA A GERAÇÃO DO ARQUIVO SINTEGRA

por

Jaques Metz

Professor orientador: Prof. Ms. Ricardo Ferreira de Oliveira

Novo Hamburgo, novembro de 2006.

CENTRO UNIVERSITÁRIO FEEVALE
INSTITUTO DE CIÊNCIAS EXATAS E TECNOLOGICAS
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

UM MODELO DE FERRAMENTA PARA A GERAÇÃO DO ARQUIVO SINTEGRA

por

Jaques Metz

Trabalho de Conclusão submetido à avaliação,
como requisito parcial para a obtenção do grau de Bacharel
em Ciência da Computação

Professor orientador: Prof. Ms. Ricardo Ferreira de Oliveira

Novo Hamburgo, novembro de 2006.

AGRADECIMENTOS

Agradeço e dedico esta conquista a meus pais, que proveram meus estudos e me incentivaram nas horas difíceis. Sem eles, esta etapa de minha vida não teria se completado.

Aos meus colegas de serviço, principalmente ao sócio e grande amigo Luciano pelas sugestões e incentivos, que foram muito importantes.

Aos amigos, pela compreensão nas horas de ausência. Ao professor orientador, pela confiança e troca de experiências. Obrigado.

RESUMO

Atualmente, as organizações têm utilizado a tecnologia EDI – *Electronic Data Interchange* (intercâmbio eletrônico de dados) – para trocar informações, no que diz respeito às transações de compra e venda efetuada entre as mesmas. Da mesma forma, as organizações devem enviar à Secretaria Estadual da Fazenda (SEFAZ) de seu estado de origem o chamado arquivo SIntegra – Sistema Integrado de Informações sobre Operações Interestaduais com Mercadorias e Serviços – que é um arquivo EDI contendo informações sobre operações de compra e venda de mercadorias e serviços. Desenvolvedores de *software* na área de comércio necessitam grande conhecimento da legislação vigente para implementar em seus sistemas a rotina de geração do arquivo SIntegra.

O problema surge justamente no fato de que os desenvolvedores de software devem conhecer a legislação para criar sua rotina de geração do arquivo SIntegra, ou adquirir alguma biblioteca de vínculo dinâmico (DLL) existente no mercado que gere o mesmo. Contudo, estas bibliotecas, apesar de gerarem o arquivo formatado e validado, exigem do desenvolvedor muitas linhas de programação.

Utilizando a tecnologia ETL – *Extract, Transform & Load* (extrair, transformar e carregar) – aliada aos conceitos de Programação Orientada a Objetos (POO), é possível desenvolver uma rotina de geração do arquivo SIntegra que exija do desenvolvedor pouco ou nenhum conhecimento sobre a legislação do SIntegra. Além disso, o desenvolvedor vai necessitar de apenas algumas linhas de código para chamar a rotina principal de geração do arquivo, ao contrário das soluções já existentes, onde a criação do arquivo exige a chamada de vários métodos.

Assim sendo, este trabalho tem por objetivo levantar e relacionar os diversos aspectos legais e funcionais da legislação concernente ao SIntegra, além de modelar e implementar uma DLL que disponha de toda a funcionalidade necessária à geração do arquivo SIntegra. Procurar-se-á focar nos aspectos de fácil utilização e configuração por parte dos desenvolvedores. Para tanto, pretende-se fazer uso das tecnologias ETL e EDI, implementando e utilizando os conceitos da POO.

Palavras-chave: SIntegra, ETL e Desenvolvimento de componentes Delphi.

ABSTRACT

Title: "A TOOL PROTOTYPE FOR THE SINTEGRA FILE GENERATION"

Actually, companies have made use from EDI – Electronic Data Interchange - technology to interchange information about buy and sell transactions between themselves. By the same way, this companies need to send to their State Treasury Bureau the SIntegra file, which is a EDI file with information about buy and sell transactions. Software developers need a great know-how about legislation to develop in their computer systems the SIntegra file generation routine.

The problem appears right here, when software developers need to know the legislation to implement the SIntegra file generation routine, or acquire some third-party library that will do the job. However, despite this libraries generate the formatted and validated file, they demand many lines of code.

Making use of the ETL – *Extract, Transform & Load* – technology, joint by the concepts of Object Oriented Programming (OOP), is possible to develop a routine to generate the SIntegra file that demand from developer none or a minimum know-how about SIntegra legislation. Moreover, the software developer will need a few lines of code to call the main routine, in opposite of the third-party solutions, which need to call many methods to generate the SIntegra file.

In this way, this project is intended to lift and list the many legal and functional aspects of legislation about SIntegra, and also modeling and implementing a library in Delphi programming language that has all the functionality necessary to generate the

SIntegra file. It will focus on ease of use and configuration on developer's side. For this, it will made use of ETL and EDI technologies, implementing and using the OOP concepts.

Keywords: SIntegra, ETL and Delphi component development.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

EDI	*Electronic Data Interchang*
SEFAZ	Secretaria Estadual da Fazenda
DLL	*Dynamic Link Library*
ETL	*Extract, Transform & Load*
POO	Programação Orientada a Objetos
OOP	*Object Oriented Programming*
VIES	*VAT Information Exchange System*
VAT	*Value Added Tax*
UE	União Européia
SINTEGRA	Sistema Integrado de Informações sobre Operações Interestaduais com Mercadorias e Serviços
XML	*Extended Markup Language*
SQL	*Structured Query Language*
UML	*Unified Modeling Language*
SGDB	Sistema de Gerenciamento de Banco de Dados
PNAFE	Programa Nacional de Apoio à Administração Fiscal para os Estados Brasileiros
ICMS	Imposto sobre Circulação de Mercadorias e Prestação de Serviços

CNPJ	Cadastro Nacional de Pessoa Jurídica
CFOP	Código Fiscal de Operação
IPI	Imposto sobre Produtos Industrializados
PDV	Ponto de Venda
ABNT	Associação Brasileira de Normas Técnicas
UF	Unidade de Federação
ONU	Organização das Nações Unidas

SUMÁRIO

INTRODUÇÃO

Tema

Este trabalho trata do uso dos conceitos da tecnologia ETL – *Extract, Transform & Load* (Extrair, Transformar e Carregar), além da avaliação de soluções já existentes no mercado, para a modelagem e implementação de uma ferramenta para a geração do arquivo Sintegra focada na facilidade de uso e configuração por parte dos desenvolvedores de *software*.

Motivação

Atualmente as organizações têm trocado informações a respeito de suas transações de compra e venda eletronicamente, através de arquivos EDI – *Electronic Data Interchange* (troca eletrônica de dados), a fim de diminuir o volume de inserção de dados manuais em seus sistemas de controle (UNI5, 2006). Freqüentemente, essas organizações não têm o mesmo fornecedor de *software*, o que dificulta a troca de informações num padrão comum. Para tanto, empresas de *software* especializaram-se na tecnologia EDI, e criaram um serviço que possibilita a troca de arquivos por meio eletrônico sem que os fornecedores de *software* tenham que desenvolver os *layouts* de importação e exportação de arquivos de seus concorrentes: as organizações interessadas em trocar informações de compra e venda com seus clientes e/ou fornecedores contratam os serviços das empresas especializadas em EDI.

Esta, por sua vez, recebe de seus clientes o *layout* dos arquivos gerados nos sistemas de controle destes e, com o auxílio de ferramentas de ETL – *Extract, Transform & Load*, desenvolve as ferramentas que farão o trabalho de transformar os arquivos que estão num *layout* "A", para arquivo no formato do *layout* "B", e vice-versa (HENRY et al., 2005).

Já com o arquivo SIntegra é um pouco diferente. O SIntegra teve seu início em 1997, e foi baseado no VIES – *VAT Information Exchange System* (Sistema de troca de Informações sobre VAT[1]), sistema este implantado na União Européia (UE) em 1992. Através do VIES, foi abolido o controle físico de mercadorias nas fronteiras dos países membros da UE (SINTEGRA, 2006). O arquivo SIntegra deve ser enviado eletronicamente – assim sendo, ele é um arquivo EDI – por organizações que efetuam compra e/ou venda de bens e serviços. Este arquivo deve ser enviado à Secretaria Estadual da Fazenda (SEFAZ) do estado origem da organização, e esta, por sua vez, cruza as informações obtidas com as outras secretarias estaduais. O *layout* dos arquivos, não importando o estado em que se encontra a organização, é único, ou seja, se uma empresa desenvolvedora de *software* atende mais de um estado brasileiro, não precisa se preocupar com a legislação estadual, pois o *layout* do arquivo SIntegra obedece à legislação federal. A problemática surge justamente neste ponto: a legislação. Pelo fato do arquivo SIntegra ser o instrumento de um sistema integrado – SIntegra é o acrônimo de "Sistema Integrado de Informações sobre Operações Interestaduais com Mercadorias e Serviços" – que visa controlar se as organizações estão fornecendo às secretarias estaduais da fazenda as informações relativas às operações de compra e venda de bens e serviço. Assim sendo, o desenvolvedor que desejar implementar a rotina de geração do arquivo SIntegra deve ter conhecimento bastante avançado da legislação, principalmente no que diz respeito a modelos de documentos fiscais, operações fiscais, entre outros.

O objetivo deste trabalho é levantar as características e funcionalidades relacionadas à geração do arquivo SIntegra além de elaborar uma biblioteca em forma de componente Delphi que possa eximir o desenvolvedor da necessidade de conhecer a legislação que permeia o arquivo SIntegra, ou mesmo a estrutura interna do mesmo. Como se sabe, já existe no mercado soluções sob a forma de bibliotecas de vínculo dinâmico (DLL's), ou até mesmo classes de objetos Delphi – que serão analisadas durante o trabalho – que geram o arquivo, mas estas soluções exigem que o desenvolvedor tenha conhecimento da estrutura interna do arquivo, e implementar a rotina pode desprender muito tempo do desenvolvedor.

[1] VAT - VALUE ADDED TAX (Imposto sobre Valor Agregado – corresponde ao ICMS (Imposto sobre Circulação de Mercadorias e Serviços)).

A estrutura interna do arquivo SIntegra é bastante complexa, pois ele pode ser formado por vários tipos de registro, dependendo das informações relacionadas às operações de compra e venda de mercadorias e serviços da organização que for gerar o arquivo.

Com o uso de alguns conceitos da tecnologia ETL – esta tecnologia é a base para a implementação de repositórios de dados (data warehouses) (KIMBALL e CASERTA, 2004) – é possível extrair dados relevantes dos sistemas de origem, transformá-los, corrigir os possíveis erros, para finalmente disponibilizar a informação no formato necessário.

> *"Um sistema ETL bem modelado extrai as informações dos sistemas de origem, reforça a qualidade dos dados e os padrões de consistência, padronizando os dados, assim diferentes fontes de dados podem ser usadas em conjunto, e finalmente disponibiliza os dados em um formato pronto para ser utilizado, do modo que desenvolvedores podem desenvolver seus sistemas utilizando estes dados, assim como usuários finais podem tomar decisões."*
> *(KIMBALL e CASERTA, 2004).*

Deste modo, a biblioteca a ser desenvolvida vai extrair as informações necessárias, e a partir destas vai gerar o arquivo SIntegra com seus devidos registros. As informações necessárias podem ser extraídas e/ou alocadas de modos diferentes – por essa razão os sistemas ETL são muitas vezes nomeados de *staging areas*, ou "áreas de alocação temporária" (KIMBALL e CASERTA, 2004). Pensando nisso, o protótipo deve comportar mais de um tipo de entrada de dados – neste caso, propõe-se o suporte a arquivos estruturados XML – *Extended Markup Language*, bem como o acesso direto a bancos de dados através de consultas no padrão SQL – *Structured Query Language*.

A adaptabilidade da qual disporá esta biblioteca é um principais agentes motivadores deste trabalho, visto que o mercado de *software* necessita de uma solução de fácil adaptação aos sistemas em desenvolvimento, bem como aos sistemas já desenvolvidos. Visto que a legislação brasileira freqüentemente sofre alterações, é conveniente que os desenvolvedores não tenham que se preocupar se sua rotina de geração do arquivo SIntegra está de acordo com a legislação vigente. Assim sendo, uma solução para o problema da geração do arquivo SIntegra em forma de protótipo é desejável.

Objetivos

Este trabalho tem como objetivo geral modelar e implementar um componente na linguagem Delphi para a geração do arquivo SIntegra, de fácil configuração e utilização pelos desenvolvedores de *software*, visando contribuir para simplificar e documentar as transações de informações sobre operações de compra e venda de mercadorias e serviços.

Como objetivos específicos, podem ser destacados:

- Avaliar as soluções para a geração do arquivo SIntegra já existentes no mercado: fazer um estudo sobre o funcionamento e estrutura dos protótipos já existentes no mercado, a fim de identificar pontos positivos e negativos na sua implementação e utilização, para posterior utilização, quando da modelagem e implementação da solução a ser proposta neste trabalho;
- Estudar a tecnologia ETL e casos de uso: para utilizar os conceitos da tecnologia ETL na modelagem e implementação da biblioteca de geração do arquivo SIntegra proposta neste trabalho, é necessário o estudo da mesma. Como a implementação de sistemas ETL se divide em milhares de subcasos, que dependem do tipo de fonte de dados, regras de negócio, sistemas de *software* pré-existentes, entre outros (KIMBALL e CASERTA, 2004), deve ser feito um estudo sobre qual metodologia ETL melhor se aplica à proposta do trabalho;
- Caracterizar a estrutura do arquivo SIntegra: Por se tratar de um arquivo estruturado, o arquivo SIntegra possui vários tipos de registros diferentes. Para compreender a complexidade que envolve a geração do arquivo SIntegra, deve-se caracterizar cada tipo de registro;
- Modelar e Implementar o protótipo para geração do arquivo SIntegra: após avaliar as soluções existentes no mercado, definir a melhor metodologia de uso do ETL a ser aplicada e caracterizar os tipos de registro do arquivo SIntegra, criar os modelos e diagramas UML do componente Delphi a ser desenvolvido, e após validação dos modelos, implementar a solução. Após a implementação

do componente, o mesmo será testado em um sistema de controle real, para posterior avaliação dos resultados.

Estrutura do Texto

O capítulo 1 apresenta o funcionamento de sistemas ETL, que vão servir de base para a implementação do componente para a geração do arquivo SIntegra.

O capítulo 2 apresenta o arquivo SIntegra, sua estrutura interna e quem é obrigado a enviá-lo às Secretarias Estaduais da Fazenda. Ao explicar a estrutura interna do mesmo, é possível diagnosticar pontos críticos na implementação da rotina de geração deste, principalmente no que diz respeito à legislação concernente ao SIntegra.

O capítulo 3 apresenta as soluções para a geração do arquivo SIntegra já existentes no mercado. Estas serão avaliadas através de um formulário de métricas de *software*, para apontar pontos positivos e negativos na implementação e utilização das mesmas. A partir destes resultados, será possível apresentar a proposta de um novo componente para gerar o arquivo SIntegra.

É justamente sobre a proposta do novo componente que o capítulo 4 vai falar. Neste capítulo será explanado qual será a proposta de funcionamento do componente, e qual ou quais vantagens este vai apresentar em relação às soluções já existentes no mercado. Além disso, serão apresentados os modelos UML do novo componente a ser implementado, de forma a facilitar o entendimento do mesmo. A última seção do capítulo explana sobre a implementação do componente modelado, tratando mais especificamente do seu funcionamento interno e de como foram usados os conceitos de ETL para a implementação do mesmo.

Finalmente, o capítulo 5 faz uma avaliação do componente proposto e agora implementado neste trabalho, fazendo um comparativo entre as soluções existentes no mercado e o componente desenvolvido neste trabalho.

1 SISTEMAS ETL

Sistemas ETL são muito importantes quando é necessário reunir uma grande quantidade de informações, e destas extrair dados que auxiliem na tomada de decisão nas organizações. Nos sistemas de controle das organizações existe uma enorme quantidade de dados, mas muitas vezes estes dados são dispostos em relatórios separados, não é feito o vínculo adequado entre estes, não sendo possível demonstrar ao nível estratégico da organização a real situação da mesma. Desta forma, não é possível tomar decisões a respeito ou mesmo traçar uma estratégia para o futuro da organização.

Este é o objetivo dos sistemas ETL: extrair, limpar, padronizar e entregar uma ou mais fontes de dados em uma fonte de armazenamento de dados relacional (banco de dados) que suporte e implemente a consulta e análise destes dados com o propósito da tomada de decisão (KIMBALL e CASERTA, 2004).

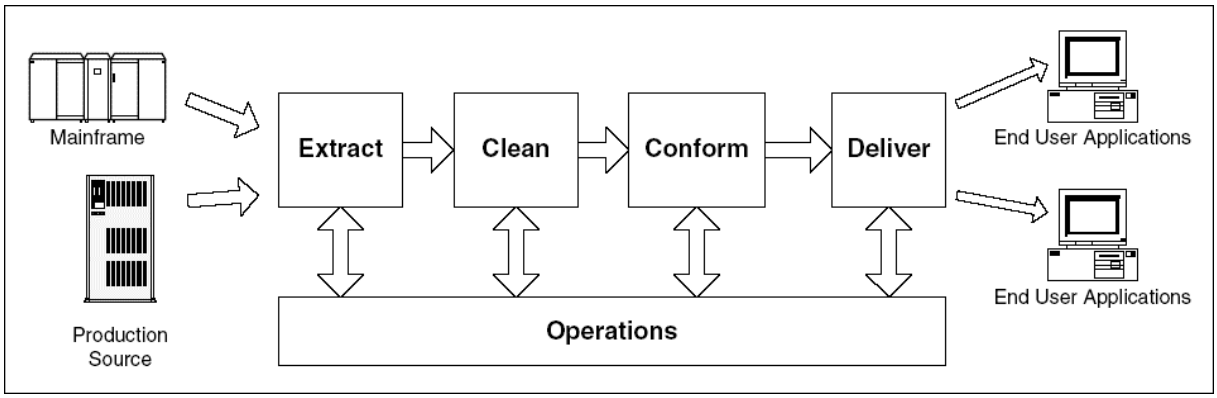

Figura 1.1 Processo ETL (KIMBALL e CASERTA, 2004).

Vale lembrar que sistemas ETL são diferentes de sistemas de *data mining*: enquanto que em sistemas ETL os dados são extraídos de arquivos estruturados (arquivos XML,

arquivos de texto com valores separados por vírgula etc.) ou diretamente de tabelas de bancos de dados relacionais, em sistemas de *data mining* os dados são extraídos de textos ou documentos não estruturados, ou seja, os dados são extraídos com ferramentas específicas que identificam nestes textos ou documentos padrões, para então extrair os dados desejados. A extração dos dados é feita baseada na análise do contexto, diferentemente dos sistemas ETL, onde primeiramente é feito um estudo aprofundado de todas as fontes de dados, em busca dos dados realmente necessários à extração.

Este trabalho não fará uso de sistemas ETL, apenas usará os conceitos deste tipo de ferramenta para desenvolver o componente para a geração do arquivo SIntegra. Assim sendo, é necessário explanar sobre as etapas de um sistema ETL que serão utilizadas na modelagem e implementação do mesmo, que são: *Extract* (extração) - *Transform* (transformação) – *Load* (carga).

1.1 *Extract* – a extração dos dados

Em sistemas ETL a fase de extração dos dados é a mais complexa e demorada. Isto porque é nesta fase que se define quais serão os dados que se deseja extrair das diversas fontes de dados, bem como o modo como estes dados serão extraídos. Quando projetos de repositórios de dados – *data warehouses* - são iniciados em organizações, é notável que um dos primeiros desafios encontrados pelos programadores é o de integrar os dados dos diversos sistemas existentes na mesma, para que estes possam produzir resultados. Sem dúvida, um *data warehouse* sem dados coerentes e estruturados não tem utilidade nenhuma. Assim, o primeiro passo da integração deve ser a correta extração dos dados dos sistemas de origem (KIMBALL e CASERTA, 2004).

Cada fonte de dados tem um conjunto específico de características, que precisam ser mensuradas e controladas, de modo que ocorra a extração dos dados no processo de ETL. É comum que existam vários tipos de sistemas de controle nas organizações, tais como: controle de vendas, controle de estoques, controle de produção entre outros. Como se não bastasse que estes sistemas não fossem integrados, eles geralmente são incompatíveis, tanto física quanto logicamente. Desta forma, o processo de ETL deve integrar sistemas que possuem:

- Sistemas Gerenciadores de Bancos de Dados (SGBD) diferentes;

- Sistemas Operacionais diferentes;
- Hardware diferenciado;
- Protocolos de comunicação diferentes.

Antes de iniciar o processo de extração dos dados, deve-se primeiramente planejar o mesmo. Para tanto, é necessário fazer um mapa lógico dos dados, que documentará a relação entre os atributos originais – dos sistemas de origem, e os atributos de destino – que receberão os dados no processo de carga (*Load*). O mapa lógico de dados deve conter informações referentes às tabelas e atributos de origem, bem como as tabelas e atributos de destino, que receberão os dados após a extração e transformação dos dados originais. A definição do mapa lógico dos dados é imprescindível em sistemas ETL, partir diretamente para a implementação acarretaria em perda de tempo – integrar dados de diversas fontes de dados sem nenhuma análise prévia inevitavelmente produziria erros de implementação e, sucessivamente o re-trabalho; além disso, formar-se-ia uma lacuna na documentação do sistema, tornando sua manutenção mais difícil (KIMBALL e CASERTA, 2004).

O mapa lógico dos dados é geralmente apresentado sob a forma de uma tabela ou planilha (figuras 1.2 e 1.3), contendo os seguintes componentes (KIMBALL e CASERTA, 2004):

- Nome da tabela destino: o nome atribuído à tabela destino no repositório de dados;
- Nome do atributo destino: nome do atributo que receberá os valores no repositório de dados;
- Tipo de tabela: indica o tipo de tabela destino. A tabela pode ser do tipo factual - contém apenas um evento ou registro, ou uma dimensão - contém uma série de eventos ou registros;
- Tipo de SCD (*slowly changing dimension*, ou "dimensão alterada lentamente"): Para dimensões, este componente indica o tipo alteração que pode ser feita em cada dimensão. Os tipos podem ser 1 - *Overwrite* (Sobrescrita), 2 - *Partitioning History* (as alterações são guardadas em um histórico) e 3 - *Alternate Realities* (houve uma alteração, mas o registro original é válido em algumas circunstâncias). O tipo SCD pode variar nos diversos atributos de uma dimensão;

- Base de dados de origem: nome da base de dados onde os dados de origem residem. Este componente geralmente define a string de conexão ao banco de dados, bem como pode ser o nome do arquivo tal qual está definido no sistema de arquivos;
- Nome da tabela de origem: nome da tabela de onde os dados são provenientes. Em alguns casos serão necessárias mais de uma tabela. Nestes casos, serão listadas todas as tabelas necessárias para popular a respectiva tabela destino no repositório de dados;
- Nome do atributo de origem: o atributo ou os atributos necessários para popular o atributo destino. Serão listados todos os atributos necessários para carregar o atributo destino. As associações dos atributos de origem serão documentadas no componente Transformação;
- Transformação: especifica a manipulação necessária aos dados de origem para chegarem ao formato esperado ao final do processo ETL. Este componente geralmente é descrito em SQL ou pseudocódigo.

Com a definição do mapa lógico dos dados, o desenvolvedor tem em mãos tudo o que precisa para iniciar a implementação de seu sistema ETL, pois já foi definido de onde os dados serão extraídos (base de dados de origem), quais dados serão extraídos (tabelas e atributos de origem), como os dados serão extraídos (transformação), e finalmente onde os dados serão alocados (tabelas e atributos destino).

Target				
Table Name	Column Name	Data Type	Table Type	SCD Type
EMPLOYEE_DIM	EMPLOYEE_KEY	NUMBER	Dimension	1
EMPLOYEE_DIM	EMPLOYEE_ID	NUMBER	Dimension	1
EMPLOYEE_DIM	BIRTH_COUNTRY_NAME	VARCHAR2(75)	Dimension	1
EMPLOYEE_DIM	BIRTH_STATE	VARCHAR2(75)	Dimension	1
EMPLOYEE_DIM	DISPLAY_NAME	VARCHAR2(75)	Dimension	1
EMPLOYEE_DIM	BIRTH_DATE	DATE	Dimension	1
EMPLOYEE_DIM	SALUTATION	VARCHAR2(12)	Dimension	1
EMPLOYEE_DIM	FIRST_NAME	VARCHAR2(30)	Dimension	1
EMPLOYEE_DIM	LAST_NAME	VARCHAR2(30)	Dimension	1
EMPLOYEE_DIM	MARITAL_STATUS	VARCHAR2(12)	Dimension	2
EMPLOYEE_DIM	DIVERSITY_CATEGORY	VARCHAR2(30)	Dimension	1
EMPLOYEE_DIM	GENDER	VARCHAR2(12)	Dimension	1
EMPLOYEE_DIM	EMPLOYEE_STATUS	VARCHAR2(24)	Dimension	1
EMPLOYEE_DIM	POSITION_CODE	VARCHAR2(12)	Dimension	2
EMPLOYEE_DIM	POSITION_CATEGORY	VARCHAR2(30)	Dimension	2
EMPLOYEE_DIM	HIRE_DATE	DATE	Dimension	1
EMPLOYEE_DIM	DEPARTMENT_CODE	VARCHAR2(12)	Dimension	2
EMPLOYEE_DIM	DEPARTMENT_NAME	VARCHAR2(75)	Dimension	2
EMPLOYEE_DIM	PART_TIME_FLAG	VARCHAR2(1)	Dimension	1
EMPLOYEE_CONTRACT_FACT	EFFECTIVE_DATE_KEY	NUMBER	Fact	N/A
EMPLOYEE_CONTRACT_FACT	END_DATE_KEY	NUMBER	Fact	N/A
EMPLOYEE_CONTRACT_FACT	CURRENCY_KEY	NUMBER	Fact	N/A
EMPLOYEE_CONTRACT_FACT	RATE_TYPE_KEY	NUMBER	Fact	N/A
EMPLOYEE_CONTRACT_FACT	PROJECT_KEY	NUMBER	Fact	N/A
EMPLOYEE_CONTRACT_FACT	EMPLOYEE_ROLE_KEY	NUMBER	Fact	N/A
EMPLOYEE_CONTRACT_FACT	CONTRACT_TYPE_KEY	NUMBER	Fact	N/A
EMPLOYEE_CONTRACT_FACT	CONTRACT_NUMBER	NUMBER	Fact	N/A
EMPLOYEE_CONTRACT_FACT	RATE_AMOUNT_LOCAL	NUMBER	Fact	N/A
EMPLOYEE_CONTRACT_FACT	RATE_AMOUNT_USD	NUMBER	Fact	N/A

Figura 1.2 Mapa Lógico dos Dados – parte 1 (KIMBALL e CASERTA, 2004).

Source				Transformation
Database Name	Table Name	Column Name	Data Type	
			NUMBER	Surrogate key.
HR_SYS	EMPLOYEES	EMPLOYEE_ID	NUMBER	Natural Key for employee in HR system
HR_SYS	COUNTRIES	NAME	VARCHAR2(75)	select c.name from employees e, states s, countries c where e.state_id = s.state_id and s.country_id = c.country
HR_SYS	STATES	DESCRIPTION	VARCHAR2(255)	select s.description from employees e, states s where e.state_id = s.state_id
HR_SYS	EMPLOYEES	FIRST_NAME	VARCHAR2(75)	select initcap(salutation) ll ' 'llinitcap(first_name) ll' 'll initcap(last_name) from employee
HR_SYS	EMPLOYEES	DOB	DATE	trunc(DOB)
HR_SYS	EMPLOYEES	SALUTATION	VARCHAR2(12)	initcap(salutation)
HR_SYS	EMPLOYEES	FIRST_NAME	VARCHAR2(30)	initcap(first_name)
HR_SYS	EMPLOYEES	LAST_NAME	VARCHAR2(30)	initcap(last_name)
HR_SYS	MARITAL_STATUS	DESCRIPTION	VARCHAR2(12)	select nvl(m.name,'Unknown') from employee e marital_status m where e.marital_status_id = m.marital_status_id
HR_SYS	EMPLOYEES	EEO_CLASS	VARCHAR2(30)	decode(eeo_class,null, 'Not Stated', decode(eeo_class,'N', 'Not Stated',eeo_class))
HR_SYS	EMPLOYEES	SEX	VARCHAR2(12)	nvl(sex, 'Unknown')
HR_SYS	EMPLOYEES	STATUS	VARCHAR2(24)	select es.name from employee e employee_status es where e.employee_status_id = m.employee_status_id
HR_SYS	POSITIONS	POSITION_CODE	VARCHAR2(12)	select p.code from employees e, positions p where p.position_id = e.position_id
HR_SYS	POSITIONS	POSITION_CATEGORY	VARCHAR2(30)	select p.category from employees e, positions p where p.position_id = e.position_id
HR_SYS	EMPLOYEES	DATE_HIRED	DATE	trunc(date_hired)
HR_SYS	DEPARTMENTS	CODE	VARCHAR2(12)	select d.code from employee p, employee_department pd, departments d where p.employee_id= pd.employee_id and pd.department_id = d.department_id
HR_SYS	DEPARTMENTS	DESCRIPTION	VARCHAR2(75)	select d.description from employee p, employee_department pd, departments d where p.employee_id= pd.employee_id and pd.department_id = d.department_id
HR_SYS	EMPLOYEES	PERCENTAGE	VARCHAR2(1)	select decode(sign(percentage-100),-1,'Y','N') from employee
DW_PROD, HR_SYS	DATE_DIM, EMPLOYEE_CONTRACT	DATE_KEY	NUMBER	where employee_contract.eff_date = dw_prod.date_dim.cal_date
DW_PROD, HR_SYS	DATE_DIM, EMPLOYEE_CONTRACT	DATE_KEY	NUMBER	where employee_contract.end_date = dw_prod.date_dim.cal_date
DW_PROD, HR_SYS	CURRENCY_DIM, EMPLOYEE_CONTRACT	CURRENCY_KEY	NUMBER	where employee_contract.currency_code = dw_prod.currency_dim.currency_code
DW_PROD, HR_SYS	RATE_TYPE_DIM, EMPLOYEE_CONTRACT	RATE_TYPE_KEY	NUMBER	where employee_contract.rate_type_id = dw_prod.rate_type_dim.rate_type_id
DW_PROD, HR_SYS	PROJECT_DIM, EMPLOYEE_CONTRACT	PROJECT_KEY	NUMBER	where employee_contrac.project_code = dw_prod.project_dim.project_code
DW_PROD, HR_SYS	EMPLOYEE_ROLE_DIM, EMPLOYEE_CONTRACT	EMPLOYEE_ROLE_KEY	NUMBER	where employee_contract.employee_role = employee_role_dim.contractor_role_name
DW_PROD, HR_SYS	CONTRACT_TYPE_DIM, EMPLOYEE_CONTRACT	CONTRACT_TYPE_KEY, CONTRACT_TYPE_ID	NUMBER	where dw_prod.employee_contract.contract_type = contract_type_dim.contract_type_id
DW_PROD, HR_SYS	EMPLOYEE_CONTRACT	CONTRACT_NUMBER	NUMBER	Degenerate Dimension
DW_PROD, HR_SYS	EMPLOYEE_CONTRACT	AMOUNT	NUMBER	sum(amount)
DW_PROD, HR_SYS	EMPLOYEE_CONTRACT	AMOUNT	NUMBER	select ec.amount * avg(cc.conversion_rate) from employee_contract ec, currency_conversion cc where ec.currency = cc.from_currency and cc.effective_Date between ec.effective_date and ec.end_date group by ec.amount

Figura 1.3 Mapa Lógico dos Dados – parte 2 (KIMBALL e CASERTA, 2004).

1.2 *Transform* – a transformação dos dados

A fase de transformação dos dados é considerada a principal etapa de um processo ETL, pois é nesta etapa que os dados agregam valor. As outras fases de extração e carga dos dados são obviamente necessárias, mas elas apenas movem e reformatam os dados. Já nesta fase de transformação, os dados recebem o tratamento necessário e se pode ter uma noção de

onde os mesmos serão utilizados. Nesta fase é desejável que se obtenham três resultados: um relatório de definição dos dados, um relatório com os eventos de erro, e o relatório de auditoria.

Para elaborar o relatório de definição dos dados, deve ser feita uma minuciosa análise das fontes de dados. O resultado desta análise geralmente é apresentado sob a forma de um repositório de meta-dados, descrevendo:

- Definições em geral;
- Objetos de Negócio;
- Domínios;
- Fontes de dados;
- Definições de entidades;
- Sinônimos;
- Regras para os dados;
- Regras para os valores;
- Questões relevantes que devem ser levadas em conta.

Esta análise da definição dos dados não apenas gera uma boa avaliação quantitativa das fontes de dados originais, como também influencia o conteúdo do relatório de eventos de erro e do relatório de auditoria.

O relatório de eventos de erro reporta todo e qualquer erro de qualidade dos dados. Isto quer dizer que, na etapa de transformação de um sistema ETL é verificado se os dados originais possuem qualidade suficiente para serem utilizados. Assim, cada erro gerado é gravado em uma tabela, para posterior utilização no relatório.

Mas afinal, o que é qualidade nos dados? Para Kimball e Caserta (2004), dados com qualidade devem ter as seguintes características:

- Corretude. Os valores e a descrição dos dados devem descrever verdadeira e fielmente os objetos associados a estes;
- Não-ambigüidade. Os valores e a descrição dos dados devem ter apenas um significado, ou seja, não podem ser ambíguos;
- Consistência. Os valores nos dados devem utilizar uma notação constante para

compreender seu significado. É conveniente que se estabeleça um padrão e que este seja obedecido, por exemplo, com siglas de estado (RS – Rio Grande do Sul, SP – São Paulo etc.);

- Dados Completos. Dois aspectos devem ser observados:
    1. O primeiro é garantir que os valores individuais nos dados estejam atribuídos (não nulos) para cada registro;
    2. O segundo aspecto deve garantir que o conjunto de registros esteja completo, ou seja, garantir que não houve perda de dados no fluxo da informação.

Desse modo, o processo de transformação de um sistema ETL tem por objetivo melhorar a qualidade dos dados extraídos. Esse processo de melhoria na qualidade dos dados obtém-se através de uma série de checagens de erro (*screens*), feitas com base em regras pré-estabelecidas. Ao encontrar um erro de qualidade de dados, o erro é gravado na tabela de eventos de erro, e é feito seu diagnóstico. Conforme as regras pré-estabelecidas, o erro pode ser diagnosticado como grave, médio ou leve. Ao passar para o próximo estágio de checagem de erros, o sistema ETL verifica se não há erros graves. Caso eles existam, o processo ETL é abortado e é necessária intervenção do administrador do sistema para sanar o problema. A figura 1.4 ilustra o processo de checagem de erros.

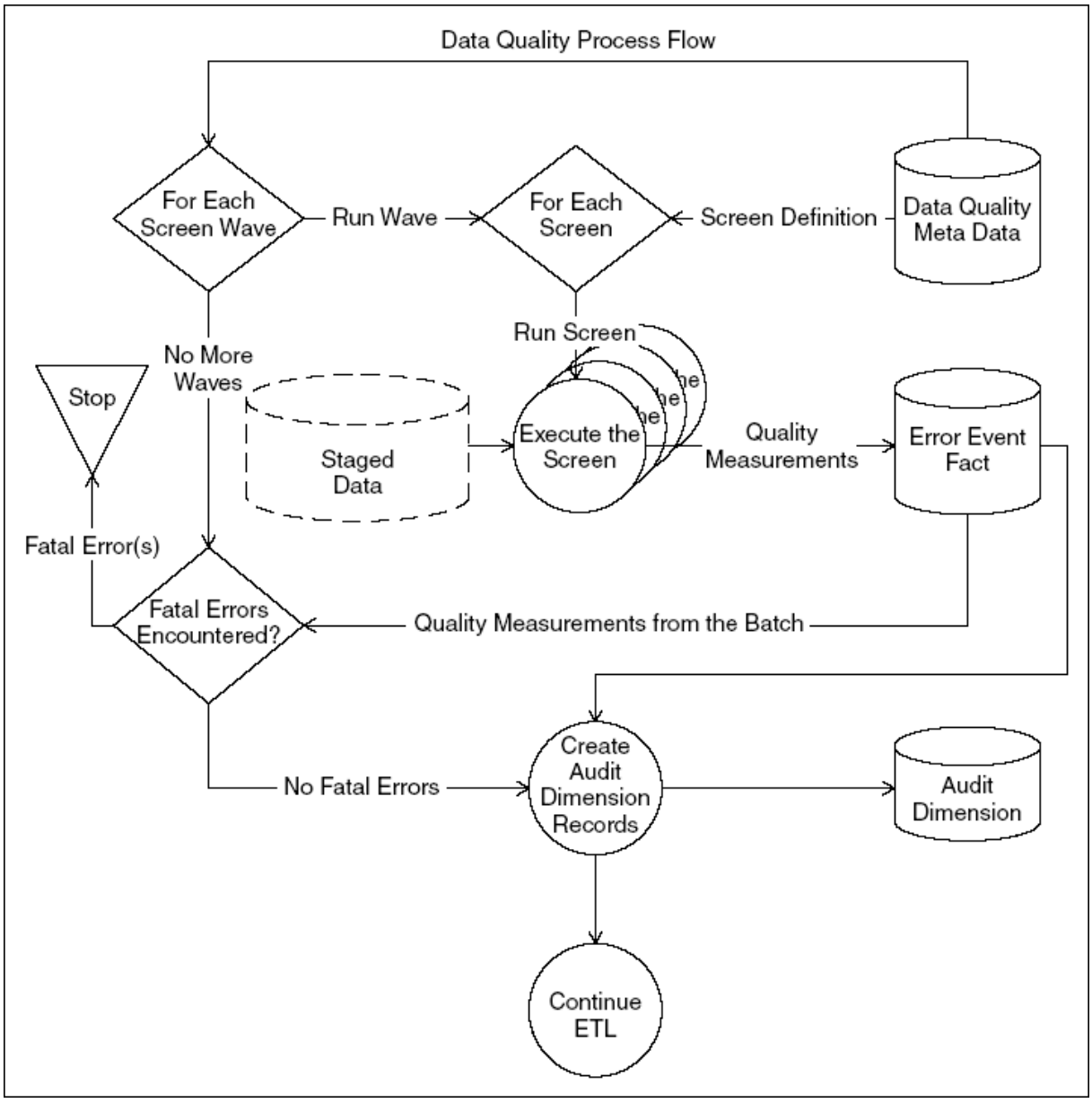

Figura 1.4 Processo de checagem de erros (KIMBALL e CASERTA, 2004).

Finalmente, tem-se o relatório de auditoria. Este relatório é gerado a partir dos eventos de erro obtidos no processo anterior, onde se verificou a qualidade dos dados. Deve-se observar que os eventos de erro gerados são erros de registro, individuais, e não podem ser apresentados no relatório de auditoria, sem que antes haja uma análise destes, para aí sim gerar um relatório que descreva todo o contexto, sempre baseado no conceito de qualidade dos dados. Este relatório, que é gerado ao final do processo de transformação dos dados, deve conter as ações de reparo e/ou mudanças que foram efetuadas nos registros para que estes pudessem ser aproveitados.

1.3 *Load* – a carga dos dados

A carga dos dados é a etapa final de um processo de ETL. É nesta etapa que os dados serão carregados para sua tabela de destino, e ali estarão disponíveis para o uso. Como existem inúmeras estruturas de dados e técnicas de modelagem de dados existentes, esta etapa do processo deve ser muito bem conduzida, para que o sistema ETL tenha as seguintes características: replicabilidade, usabilidade, escalabilidade e manutenibilidade (KIMBALL e CASERTA, 2004).

Nesta etapa do processo ETL, os dados geralmente são carregados em dois tipos de estrutura de dados: as tabelas dimensionais e as tabelas factuais. As tabelas dimensionais contêm a descrição das diversas entidades utilizadas para gerar a tabela factual, ou seja, a tabela dos dados. Assim, é possível saber de qual fonte de dados um determinado registro é proveniente, pois cada registro na tabela factual faz referência a um registro na tabela dimensional (KIMBALL e CASERTA, 2004).

Neste trabalho far-se-á uso dos conceitos de ETL – extração, transformação e carga – para a implementação de um componente Delphi que facilite o processo de geração do arquivo SIntegra. Desta forma, pretende-se automatizar ao máximo o processo de geração deste arquivo, que exige um amplo conhecimento da legislação brasileira.

2 O ARQUIVO SINTEGRA

SIntegra é o acrônimo de "Sistema Integrado de Informações sobre Operações Interestaduais com Mercadorias e Serviços". O principal objetivo desse sistema por parte dos contribuintes é o de simplificar o fornecimento de informações relativas às operações de compra, venda e prestação de serviços. Do lado dos fiscos estaduais, o objetivo é o de propiciar maior agilidade e confiabilidade ao tratamento das informações recebidas dos contribuintes e à troca de dados entre as diversas unidades de federação (SINTEGRA2, 2006).

O SIntegra foi concebido com a intenção de dar consistência a diversas iniciativas de modernização dos sistemas tributários estaduais que constituem o PNAFE – Programa Nacional de Apoio à Administração Fiscal para os Estados Brasileiros. Este programa visa contribuir para o equilíbrio financeiro das unidades federadas através de medidas de aperfeiçoamento de suas máquinas arrecadadoras e de seus mecanismos de controle das despesas. Dadas as necessidades locais diferenciadas, os projetos variam de uma UF (Unidade de Federação) para outra. Contudo, o fato de que todas as Ufs dependem fortemente da arrecadação do ICMS, que tem características semelhantes de uma UF para outra, e de que uma parcela significativa do mesmo provem de operações de compra e venda interestaduais – que caem num terreno de jurisdição conjunta – caracterizou a necessidade de promover iniciativas conjuntas no que diz respeito ao controle exercido sobre as operações interestaduais de compra e venda, nascendo assim no ano de 1997, o SIntegra (SINTEGRA2, 2006).

O arquivo SIntegra é um arquivo EDI. A seguir, descrever-se-á mais detalhadamente o que é um arquivo EDI.

2.1 Conceito de EDI

A sigla EDI abrevia *Electronic Data Interchange* ou, em português, Intercâmbio Eletrônico de Dados, não é uma norma, mas sim uma filosofia de relacionamento entre organizações, aproveitando as facilidades de comunicação e da informática. A ONU - Organização das Nações Unidas - define EDI como um "processo de transferência eletrônica entre organizações parceiras de negócios, computador-a-computador, de transações comerciais ou administrativas, usando um formato padrão pré-estabelecido" (ONU, 2006). Toda essa transferência de informações poderá ser feita automaticamente através dos computadores das organizações, independente da plataforma de *Hardware* e *Software* utilizada.

Mais formalmente, pode-se descrever o EDI como a troca de dados estruturados, baseados em formatos de mensagens normalizadas entre sistemas computacionais por meios eletrônicos. Dados estruturados significam um método não ambíguo de representar dados existentes num documento, seja ele uma fatura, uma encomenda ou qualquer outro tipo de documento. O método para assegurar a correta interpretação da informação recebida pelo sistema computacional é definido pela padronização. Quanto ao conceito, é importante ressaltar que é a transferência eletrônica de dados sem a intervenção humana (COL et al, 1996).

O EDI difere de outros sistemas de informação, pois integra sistemas de informações de organizações parceiras comerciais, envolvendo transações de negócios em formatos padronizados e utilizando serviços de comunicação de dados. O EDI é uma decisão de negócios e uma nova forma de fazer negócios. Para que seja executado um processo de EDI entre dois parceiros, é necessário que sejam executadas as seguintes tarefas exibidas no fluxo abaixo:

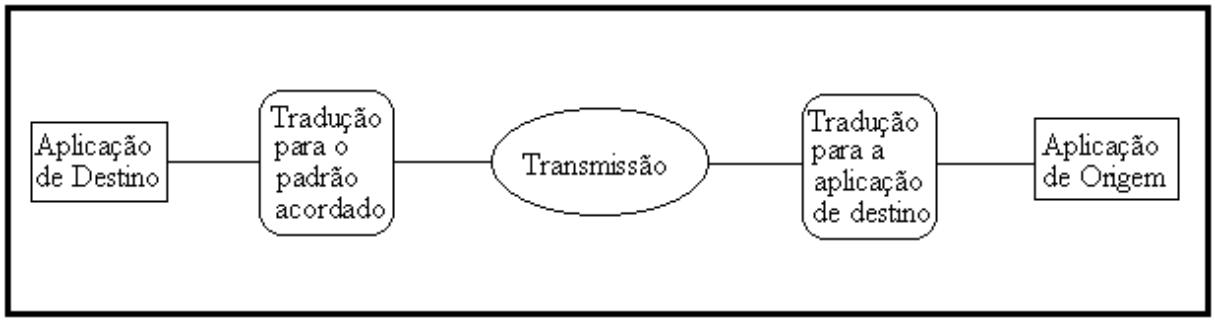

Figura 2.1 Fluxo dos dados EDI.

1. criação da transação pela aplicação de origem;
2. tradução da transação para o formato padronizado, de forma a que possa ser reconhecido pelo outro parceiro.
3. transmissão, através das facilidades de comunicações, para a aplicação destinatária do documento. Visando diminuir os custos de transmissão envolvidos, alguns softwares de tradução realizam a compactação e a posterior descompactação dos dados. Além disso, alguns softwares possuem funções adicionais como geração de senhas, criptografia dos dados, compactação, comprovação de recebimento, geração de log, etc.
4. tradução da transação recebida no formato padronizado para o formato da aplicação de destino;
5. processamento da transação pela aplicação destinatária.

Atualmente alguns sistemas referidos como usando a tecnologia EDI não implementam todos os tópicos acima mencionados e, dessa forma não podem ser considerados verdadeiramente como EDI.

O EDI é uma ferramenta que permite às organizações estabelecer as comunicações com os seus parceiros comerciais de uma forma mais efetiva, mas também totalmente diferente dos métodos normais.

EDI é a troca de dados de forma padronizada entre aplicações de sistemas de informática de organizações com negócios em comum com uma intervenção mínima manual. EDI é uma evolução natural que combina a potência computacional com serviços de telecomunicações, tudo isso processado de maneira padronizada, uma vez que os sistemas e organizações envolvidos são de portes variados e com equipamentos diferentes.

Muitas organizações optaram pelo EDI como um método rápido, econômico e seguro para o envio de ordens de compra, faturas, notificações de transporte e outros documentos utilizados freqüentemente em transações comerciais.

O EDI diferencia-se do simples envio de mensagens eletrônicas ou compartilhamento de arquivos por rede ou modem. A transferência dos arquivos requer que as aplicações, tanto do lado do emissor como do receptor do documento, sejam compatíveis com o formato do documento enviado. O emissor deve utilizar uma determinada aplicação para criar um formato de arquivo idêntico à sua aplicação comercial. Mas não há a necessidade do emissor e

do receptor possuir sistemas idênticos de processamento de documentos.

É necessário reforçar o conceito de que o EDI é uma ferramenta que, além do componente tecnológico de mudança, é considerado um processo organizacional e uma forma inevitável de, no presente e futuro, estabelecerem as relações entre os parceiros comerciais, definindo-se como a transferência eletrônica de informações estruturadas entre os sistemas de computadores de diferentes organizações de acordo com normas pré-estabelecidas.

O limite do EDI está na criatividade de cada instituição, na visão de como usar os recursos da teleinformática com qualidade e eficiência em benefício do atendimento ao cliente.

2.2 Benefícios do EDI

Quando é necessária a troca de informações entre parceiros comercias, essa troca comumentemente é feita através de relatórios, *e-mails* etc., principalmente quando estes parceiros não possuem um sistema de controle integrado. Todas essas informações são re-digitadas e posteriormente processadas por uma outra aplicação no parceiro comercial, sendo este um processo lento, caro e não merecedor de confiança.

Através da aplicação de padrões de mensagens EDI, as organizações podem até mesmo se comunicar com sistema de computadores diferentes e incompatíveis, gerando, assim, rapidez, eficiência e precisão.

Com uma implantação bem sucedida do EDI nas organizações, proporcionam-se grandes benefícios, distribuídos em dois grandes focos, apresentados a seguir:

Foco Estratégico:

1. agilidade: A comunicação de grandes volumes de dados comerciais entre os computadores reduz notavelmente os tempos previstos. Considerando que a variável tempo é um aspecto fundamental no funcionamento e na evolução dentro de uma organização, o mesmo deverá ser aproveitado ao máximo, evitando que a organização tenha dispêndios desnecessários e maior retorno do seu investimento, garantindo assim, maior satisfação por parte dos usuários do sistema;
2. aumento de produtividade: Nas organizações dirigidas pelo EDI que são

caracterizadas por um rígido e eficiente controle de estoque das suas entradas e saídas, irá existir um maior controle das necessidades de produção, de compras e de entregas, resultando em significativas reduções nos níveis de estoque, mantendo sua rotatividade no nível ideal. Tudo isso, considerado num conjunto e em aplicação paralela com o uso do EDI, demonstra a eficiência e ganho de produtividade da organização, evitando, dessa forma, gastos com pessoal desnecessários no controle de seus estoques. O EDI, portanto, é um componente chave nos elos de ligação entre cliente, fornecedor e transportador na fabricação *just-in-time*. Redução dos tempos de resposta na cadeia de distribuição;

3. melhor serviço a clientes e relacionamento com fornecedores: É decorrente da execução mais rápida e correta dos seus pedidos, colocando, assim, a organização em uma posição de maior competitividade. Através de entregas a tempo, notificação prévia de envio, redução ao mínimo dos tempos de transporte e alfândega, bem como nos pagamentos de bens ou serviços.

Foco Operacional:

1. eliminação de erros: Conforme a enorme quantidade de informações necessárias ao funcionamento da organização, não há garantia de que todas elas serão realmente informadas, ou sequer, corretamente processadas nos sistemas próprios da organização. Caso esses erros venham a acontecer, geram-se problemas de ordem econômica e financeira que irão prejudicar a organização ou até mesmo causar danos irreversíveis. Portanto, com a utilização do EDI, o mesmo elimina os inevitáveis erros resultantes da entrada manual dos dados, tendo, assim, a qualidade no processamento devido à precisão dos mesmos;

2. redução de custos: Devido à eliminação da re-digitação e da existência de documentos eletrônicos, ocorre uma considerável redução de custos operacionais, tais como, funcionários, fax, correios, papéis impressos. Isto é uma redundância administrativa e uma grande perda de tempo que resulta em gastos adicionais.

Para as organizações que já operam com o sistema de EDI, cabe às mesmas reordenarem e readaptarem os seus métodos de trabalhos e táticas administrativas através de

análises e estudos periódicos dentro da organização para, então, localizarem as suas falhas e serviços desnecessários ou improdutivos ou ainda repetitivos, atingindo, assim, o máximo dos benefícios propostos pela transferência eletrônica de dados.

Para aquelas organizações que pretendem se estabelecer e pretendem utilizar o sistema de transferência eletrônica de dados, podem e devem considerar o EDI no seu sistema de planejamento de constituição e no manuseio de gerenciamento de informações, reduzindo a quantidade de investimentos em alguns setores que serão completamente absorvidos pela utilização do EDI.

Com base nos aspectos mencionados, elaborou-se uma pesquisa, realizada em 1993 na Inglaterra sobre as principais razões que levaram as organizações a implementar o EDI e os seus benefícios diretos. Seguem abaixo os gráficos, contendo os resultados encontrados:

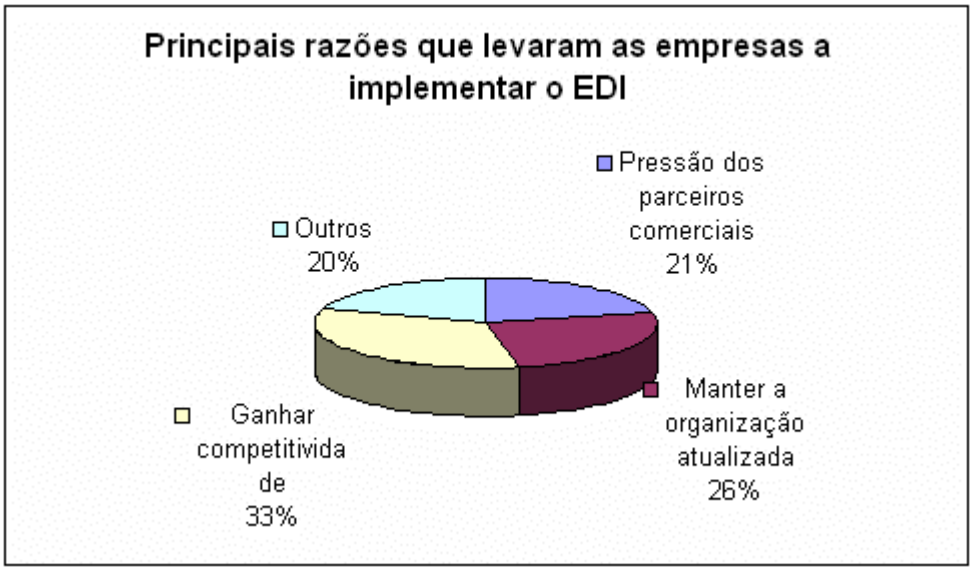

Figura 2.2 (EANBRASIL, 2006).

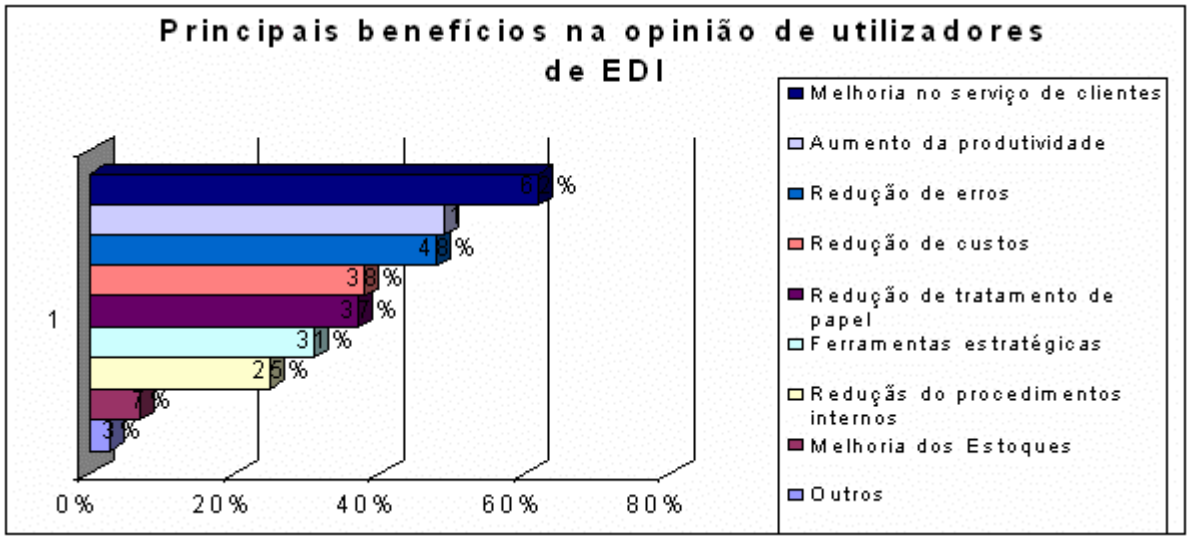

Figura 2.3 (EANBRASIL, 2006).

Os dados apresentados refletem, sem margem para dúvidas, o enorme sucesso que o EDI pode introduzir nas organizações que o pretendem implementar. É necessário salientar que, nesta pesquisa, as organizações estavam apenas 10% conectadas via EDI com suas parceiras comerciais.

Em tese, o uso do EDI é extremamente promissor e atrativo, mas podem surgir alguns problemas para se atingir os resultados esperados. A idéia geral de EDI baseia-se em ganho de velocidade nas transações comercias, no entanto os documentos EDI são transmitidos em arquivos de lote e ficam armazenados nas caixas postais dos destinatários, até que eles as retirem. Nesse momento há uma perda de tempo acentuada.

Existe uma dificuldade de se realizar um estudo de custo x benefício, pois os benefícios são dificilmente mensuráveis, sendo mais um entrave para expansão do EDI. Em muitas organizações, esse benefício pode ser dimensionado, baseado exclusivamente na economia devido aos freqüentes erros de digitação existente no modo convencional. No entanto, mesmo sendo difícil dimensionarem-se os benefícios, a não adoção de EDI, hoje, poderá implicar na futura perda de clientes.

2.3 Padronização

Para ocorrer a comunicação de dados estruturados, é necessária que se adote a normalização dos mesmos, criando-se, deste modo, uma interface entre os parceiros

interessados na comunicação que, no caso do EDI, são computadores pertencentes à mesma organização ou a diferentes organizações que têm relações comerciais.

EDI não é uma comunicação do tipo correio eletrônico com mensagem de formato livre, ou seja, a mensagem tem uma formatação rígida o que resulta na necessidade de padronização a fim do emissor e receptor da mensagem se entenderem. Os padrões para EDI, tal como uma linguagem, consistem numa gramática (sintaxe, regras de estruturação de elementos de dados em segmentos e de segmentos numa mensagem) e um vocabulário de palavras (lista de elementos de dados, lista de segmentos e lista de mensagens) (MHS, 1997).

A comunicação sofreria uma ruptura caso os parceiros do intercâmbio não seguissem normas previamente estabelecidas, passando-se de uma intolerável montanha de papel para um lixo eletrônico, especialmente no EDI internacional.

Organizações distintas têm necessidades, processos, formas, sistemas de computadores, *softwares* e sofisticações técnicas diferentes. Para que estas informações geradas e transmitidas de uma organização para outra possam ser interpretadas automaticamente, é essencial para o EDI a linguagem de comunicação.

Ao implementar o EDI, muitas organizações são tentadas a definir sua própria linguagem EDI, impondo-a aos seus parceiros, esquecendo, assim, a necessidade de levar em conta questões como sua integração com os processos internos da organização e a maneira de trocar os dados de acordo com as necessidades dos parceiros.

Para que os documentos eletrônicos e os dados fluam harmoniosamente entre as organizações e sejam corretamente interpretados, é preciso que sejam respeitadas certas regras, as quais definem o conteúdo de informação, isto é, os dados dos documentos e a forma como eles são transmitidos.

No caso do arquivo SIntegra, o padrão EDI para o mesmo é definido através de convênios distintos. Um convênio nada mais é do que a especificação dos tipos de registros que o arquivo SIntegra pode conter, bem como sua formatação e regras específicas para a geração do arquivo. O primeiro conjunto de regras (convênio) para o arquivo SIntegra foi o Convênio ICMS 57/95, de 30 de Junho de 1995. Desde lá, já sofreu várias alterações. Atualmente, os contribuintes devem obedecer ao conteúdo do Convênio ICMS 76/03. Assim, este trabalho basear-se-á neste convênio, e no seu respectivo manual de orientação para a geração do arquivo SIntegra.

2.4 Convênio ICMS 76/03

O Convênio ICMS 76/03 foi publicado no Diário Oficial da União em 16 de outubro de 2003, e descreve como os contribuintes devem entregar eletronicamente seus dados à Secretaria Estadual da Fazenda, através do arquivo SIntegra. Este arquivo é formado por vários tipos de registro, que serão ligeiramente descritos a seguir. Este trabalho não descreverá minuciosamente os tipos de registro constantes no arquivo SIntegra, por não ser este o foco do presente trabalho. Conforme o Manual de Orientação do Convênio ICMS 76/03 (2003), seguem os registros do arquivo SIntegra.

2.4.1 Registro 10

Este registro contém as informações da empresa que está fornecendo o arquivo, ou seja, os dados de identificação – razão social, CNPJ, cidade etc. – do contribuinte. Este é o primeiro registro do arquivo, e ele é único, ou seja, só pode haver um registro deste tipo no arquivo.

2.4.2 Registro 11

No registro 11 constam as informações complementares, relativas ao Registro 10 – endereço, bairro, telefone etc. Assim como o registro 10, só pode haver um registro tipo 11 no arquivo.

2.4.3 Registro 50

Este registro deve apresentar o total da Nota Fiscal emitidas nos modelos 1 e 1-A; Nota Fiscal de Produtor, modelo 4 (a critério de cada unidade da Federação); Nota Fiscal/Conta de Energia Elétrica, modelo 6; Nota Fiscal de Serviço de Comunicação, modelo 21; e Nota Fiscal de Serviço de Telecomunicações, modelo 22.

Para fins de esclarecimento, os modelos fiscais citados anteriormente dizem respeito ao tipo de atividade que o contribuinte exerce. Segue abaixo uma tabela contendo todos os

possíveis modelos de documento fiscal:

Tabela 2.1 Modelos de Documentos Fiscais. (ICMS76/03, 2003).

CÓDIGO	MODELO
24	Autorização de Carregamento e Transporte, modelo 24
14	Bilhete de Passagem Aquaviário, modelo 14
15	Bilhete de Passagem e Nota de Bagagem, modelo 15
16	Bilhete de Passagem Ferroviário, modelo 16
13	Bilhete de Passagem Rodoviário, modelo 13
10	Conhecimento Aéreo, modelo 10
11	Conhecimento de Transporte Ferroviário de Cargas, modelo 11
09	Conhecimento de Transporte Aquaviário de Cargas, modelo 9
08	Conhecimento de Transporte Rodoviário de Cargas, modelo 8
17	Despacho de Transporte, modelo 17
25	Manifesto de Carga, modelo 25
26	Conhecimento de Transporte Multimodal de Cargas, modelo 26
01	Nota Fiscal, modelo 1
06	Nota Fiscal/Conta de Energia Elétrica, modelo 6
03	Nota Fiscal de Entrada, modelo 3
21	Nota Fiscal de Serviço de Comunicação, modelo 21
04	Nota Fiscal de Produtor, modelo 4
22	Nota Fiscal de Serviço de Telecomunicações, modelo 22
07	Nota Fiscal de Serviço de Transporte, modelo 7
02	Nota Fiscal de Venda a Consumidor, modelo 02
20	Ordem de Coleta de Carga, modelo 20
18	Resumo Movimento Diário, modelo 18

Assim, este registro deve especificar as informações de totalização do documento fiscal, relativamente ao ICMS – Imposto sobre Circulação de Mercadorias e Serviços.

No caso de documentos com mais de uma alíquota de ICMS e/ou mais de um Código Fiscal de Operação – CFOP, deve ser gerado para cada combinação de ‘alíquota’ e ‘CFOP’ um registro tipo 50, com os valores correspondentes ao total da combinação alíquota X CFOP, de tal forma que as somas dos valores dos registros que representam uma mesma nota fiscal, corresponderão aos valores totais da mesma.

2.4.4 Registro 51

O registro 51 destina-se a registrar os totais de Notas Fiscais modelos 1 e 1-A relativas ao IPI – Imposto sobre Produtos Industrializados. Como já foi dito, nem todas as organizações vão gerar o arquivo SIntegra com todos os tipos de registro possíveis. Por exemplo, o registro 51 só é gerado por organizações que importam ou produzem produtos industrializados.

2.4.5 Registro 53

O registro 53 contém informações de total de documento fiscal, quanto à substituição tributária. Entenda-se por substituição tributária quando "a Lei Estadual atribui a contribuinte do imposto ou a depositário a qualquer título a responsabilidade pelo seu pagamento" (PORTAL, 2006).

2.4.6 Registro 54

Neste registro temos os itens (produtos) da nota fiscal. Para cada registro 54 deve haver um registro 50 correspondente, tal qual em um sistema de controle de vendas, onde obrigatoriamente deve haver um registro na tabela de Notas Fiscais para um ou mais registros na tabela de Itens de Notas Fiscais.

2.4.7 Registro 55

Registro de Guia Nacional de Recolhimento, que diz respeito ao imposto devido por substituição tributária. Neste caso, todas as organizações que gerarem em seu arquivo SIntegra o registro 54, terão de gerar no mínimo um registro 55.

2.4.8 Registro 56

Registro complementar relativo às operações com veículos automotores novos realizadas por montadoras, concessionárias e importadoras.

2.4.9 Registro 60

Este registro é destinado a informar as operações e prestações realizadas com os documentos fiscais emitidos por equipamento emissor de cupom fiscal, os quais são: Cupom Fiscal, Cupom Fiscal – PDV, Bilhete de Passagem Rodoviário, modelo 13; Bilhete de Passagem Aquaviário, modelo 14; Bilhete de Passagem e Nota de Bagagem, modelo 15; Bilhete de Passagem Ferroviário, modelo 16; e Nota Fiscal de Venda a Consumidor, modelo 2.

2.4.10 Registro 61

Registro dos documentos fiscais descritos a seguir, quando não emitidos por equipamento emissor de cupom fiscal: Bilhete de Passagem Rodoviário, modelo 13; Bilhete de Passagem Aquaviário, modelo 14; Bilhete de Passagem e Nota de Bagagem, modelo 15; Bilhete de Passagem Ferroviário, modelo 16; Nota Fiscal de Venda a Consumidor, modelo 2; e Nota Fiscal de Produtor, modelo 4 (a critério de cada unidade da Federação);

2.4.11 Registro 70

Registro de total de Nota Fiscal de Serviço de Transporte, modelo 7; de Conhecimento de Transporte Rodoviário de Cargas, modelo 8; de Conhecimento de Transporte Aquaviário de Cargas, modelo 9; de Conhecimento Aéreo, modelo 10; e de Conhecimento de Transporte Ferroviário de Cargas, modelo 11, destinado a especificar as informações de totalização do documento fiscal, relativamente ao ICMS.

2.4.12 Registro 71

Registro de Informações da carga transportada referente a Conhecimento de Transporte Rodoviário de Cargas, modelo 8; Conhecimento de Transporte Aquaviário de Cargas, modelo 9; Conhecimento Aéreo, modelo 10; e Conhecimento de Transporte Ferroviário de Cargas, modelo 11.

2.4.13 Registro 74

Registro de Inventário (a critério de cada Unidade Federada). O registro de inventário trata da movimentação de estoque de cada produto constante no registro 54.

2.4.14 Registro 75

O registro de Código de Produto e Serviço apresenta os registros de produtos e serviços constantes no arquivo SIntegra.

2.4.15 Registro 76

O registro 76 traz as informações de Nota Fiscal de Serviços de Comunicação, modelo 21, e Nota Fiscal de Serviços de Telecomunicações, modelo 22. Este registro é gerado por organizações prestadoras de serviços de comunicação e telecomunicação – operadoras de telefonia, etc.

2.4.16 Registro 77

Registro de serviços de comunicação e telecomunicação. Este registro diz respeito ao registro 76, neste registro que serão especificados os serviços prestados pelas organizações deste ramo.

2.4.17 Registro 90

Registro de totalização do arquivo, destinado a fornecer dados indicando a quantidade de registros. É o registro finalizador do arquivo, também chamado de *trailler*. Assim como o registro 10, só pode haver um registro tipo 90.

São estes os possíveis tipos de registro em um arquivo SIntegra. Os registros gerados

vão depender da realidade em que a organização se enquadra e do(s) tipo(s) de documento(s) fiscal(is) que ela emite, ou seja, uma organização dificilmente vai gerar um arquivo SIntegra com todos os tipos de registro. Isto é o que torna a implementação desta rotina difícil, pois os desenvolvedores devem prever todas as possibilidades possíveis. A problemática é justamente esta: como prever todas as possibilidades sem o devido conhecimento da legislação? É por isto que as soluções disponíveis no mercado apenas formatam e validam o formato e conteúdo dos registros, e não automatizam o processo da geração do arquivo como um todo.

Deste modo, para que seja possível propor uma nova solução para a rotina de geração do arquivo SIntegra, a seguir serão avaliadas duas soluções já existentes no mercado: a biblioteca Sintegra32Dll e o componente SIntegra – Projeto Sultan.

3 SISTEMAS DISPONÍVEIS NO MERCADO

Para este trabalho optou-se por avaliar uma solução proprietária – a biblioteca Sintegra32Dll, desenvolvida pela TKS Software – e uma solução *open source* – o componente SIntegra, desenvolvido pelo Projeto Sultan. Ambas as soluções foram submetidas a um formulário de avaliação, onde foram avaliadas questões gerais como portabilidade, desempenho, e outras questões pertinentes à geração do arquivo SIntegra, como automatização do processo de geração do arquivo – até que ponto a solução avaliada automatiza o processo de geração do arquivo, ou o programador deve chamar sempre um rotina diferente para gerar os tipos de registro, etc. O formulário de avaliação das duas soluções encontra-se no Anexo A.

3.1 Método de Avaliação

Para proceder a avaliação das soluções para a geração do arquivo SIntegra anteriormente citadas foi elaborado um formulário baseado na NBRISO/IEC9126-1, que é a norma brasileira para avaliação de produto de *software*, homologada pela ABNT – Associação Brasileira de Normas Técnicas (ABNTNET, 2006). Esta norma define seis características de qualidade de produto de *software*. São elas:

- Funcionalidade: conjunto de atributos que evidenciam a existência de um conjunto de funções e suas propriedades específicas. Fazem parte deste quesito de qualidade outras sub-características, como adequação, acurácia, interoperabilidade, conformidade e segurança de acesso;

- Confiabilidade: conjunto de atributos que evidenciam a capacidade do *software* de manter seu nível de desempenho sob condições estabelecidas, durante um período de tempo estabelecido. São sub-características de confiabilidade: maturidade, tolerância a falhas e recuperabilidade;
- Usabilidade: conjunto de atributos que evidenciam o esforço necessário para poder-se utilizar o *software*, bem como o julgamento individual deste uso, por um conjunto implícito ou explícito de usuários. Atrelada a esta característica estão as sub-características de inteligibilidade, apreensibilidade e operacionalidade;
- Eficiência: conjunto de atributos que evidenciam o relacionamento entre o nível de desempenho do *software* e a quantidade de recursos usados, sob condições estabelecidas. São sub-características de eficiência: comportamento em relação ao tempo e comportamento em relação aos recursos;
- Portabilidade: conjunto de atributos que evidenciam a capacidade do *software* em ser transferido de um ambiente para outro. Adaptabilidade, capacidade de ser instalado, conformidade, modificabilidade e analisabilidade são sub-características de portabilidade;
- Manutenibilidade: conjunto de atributos que evidenciam o esforço necessário para fazer modificações especificadas no *software*. Como sub-características de manutenibilidade, podem ser citadas a estabilidade e a testabilidade.

Assim, com base nestas seis características de qualidade de software, o formulário foi elaborado com o seguinte conteúdo:

Tabela 3.1 Formulário de Avaliação de Software.

	N	P	T
A solução avaliada gera todos os possíveis tipos de registro do arquivo SIntegra?			
A solução avaliada faz algum tipo de crítica nos dados de entrada?			
O arquivo gerado pela solução avaliada é confiável?			
A solução avaliada possui uma interface amigável para com o desenvolvedor?			
O desempenho da solução avaliada foi considerado:			
A solução avaliada pode ser utilizada em mais de um ambiente de programação?			
Em caso de necessidade de manutenção do código,			

a necessidade de reescrita de código foi:			
Em caso de erro, a solução avaliada possui algum tipo de registro de erros?			
A solução avaliada exige do desenvolvedor o conhecimento aprofundado da linguagem de programação?			
É necessária a instalação da solução avaliada no ambiente de programação?			
A solução avaliada é estável, podendo ser utilizada em sistemas comerciais?			

Os valores de avaliação do formulário – N, P e T – foram escolhidos propositadamente, pois como as perguntas do formulário são de tipos diferentes, este tipo de valor é o que melhor expressa quantitativamente as respostas: "N" representa que o quesito não foi atendido; "P", atendeu parcialmente o quesito; e "T", atendeu plenamente o quesito.

Além do formulário de avaliação, as soluções existentes – a biblioteca Sintegra32Dll e o componente Sintegra do Projeto Sultan – foram testadas em um ambiente de programação. Ambas as soluções têm em seu pacote de distribuição uma aplicação de demonstração, que mostra como os desenvolvedores devem utilizar as mesmas. Assim sendo, estes aplicativos de demonstração serão usados como parte da avaliação das soluções.

A seguir, serão apontados os mais importantes pontos, sejam eles positivos ou negativos, averiguados quando da avaliação das soluções anteriormente citadas.

3.2 Biblioteca Sintegra32Dll

A biblioteca Sintegra32Dll, desenvolvida pela TKS Software, é uma das soluções disponíveis no mercado mais utilizadas para a geração do arquivo SIntegra. A Sintegra32dll foi implementada sob a forma de uma DLL – *Dynamic Link Library*, ou Biblioteca de Vínculo Dinâmico. Antes de prosseguir com a avaliação da mesma, explanar-se-á sobre o que é uma DLL, suas características, bem como as vantagens e desvantagens de seu uso.

3.2.1 DLL's

Uma DLL é um arquivo executável que atua como uma biblioteca de funções

compartilhável. Através do vínculo dinâmico é possível a chamada a uma função que não faz parte do código do programa executável. O código da função está localizado na DLL, que pode conter uma ou mais funções compiladas e alocadas separadamente dos processos que fazem uso destas. Além disso, o uso de DLL's facilita o compartilhamento de dados e recursos. Assim, mais de uma aplicação pode acessar simultaneamente o conteúdo de uma DLL em memória (MSDN, 2006).

Existem duas maneiras de vincular uma DLL a um aplicativo. A primeira, chamada de vínculo dinâmico, permite ao módulo executável incluir apenas a informação ou função necessária em runtime. A segunda, chamada de vínculo estático, retorna todas as funções referenciadas na DLL, disponibilizando as mesmas para o uso durante a execução do aplicativo, mesmo que não sejam utilizadas todas as funções da DLL (MSDN, 2006). Conseqüentemente, este tipo de vinculação entre o programa executável e a DLL acarreta em alocação de memória desnecessária.

O uso do vínculo dinâmico ao invés do vínculo estático oferece inúmeras vantagens. São elas:

- Usa menos memória e reduz a troca de paginação: isto acontece porque vários processos podem fazer uso de uma DLL simultaneamente, compartilhando a mesma. Em contrapartida, o vínculo estático força o sistema a carregar uma cópia da DLL em memória para cada aplicação que assim a utilizar;
- Reduz o espaço em disco utilizado: ao compartilhar as funções de uma DLL, as aplicações não necessitam ter uma cópia específica da DLL para sua utilização, basta uma cópia em disco, e esta será utilizada por todos os aplicativos;
- Atualizações facilitadas: quando as funções de uma DLL sofrem alterações, os aplicativos que fazem uso desta não precisam ser necessariamente recompilados, desde que os argumentos e valores de retorno das funções não mudem. Desse modo, basta atualizar a DLL e o aplicativo estará atualizado.

Voltando a Sintegra32dll, justamente por ser uma DLL ela pode ser usada por qualquer ferramenta de desenvolvimento que suporte esta tecnologia, por isso sua ampla utilização. Além de sua grande portabilidade, ela possui uma boa performance – o fato de ser uma DLL poderia prejudicá-la neste quesito, pois o programa que vai chamar os métodos deve primeiramente carregar a biblioteca na memória do computador, antes de fazer uso de seus métodos.

Pode-se facilmente notar que o ponto forte desta biblioteca é a portabilidade – como já foi dito, o fato de ser uma DLL faz com que seja possível utilizar a Sintegra32Dll em qualquer ambiente de programação que tenha suporte a DLL's. Contudo, este fato faz com que seja necessário passar todos os parâmetros para a geração de um registro do arquivo SIntegra. O método vai receber todos estes parâmetros, e como resultado vai retornar o registro (linha) do arquivo SIntegra formatado. Deste modo, o desenvolvedor que desejar utilizar a Sintegra32Dll deverá criar uma rotina que chame estes métodos, e posteriormente grave os registros gerados em um arquivo texto. Isto demanda uma grande quantidade de linhas de programação, o que é um ponto negativo.

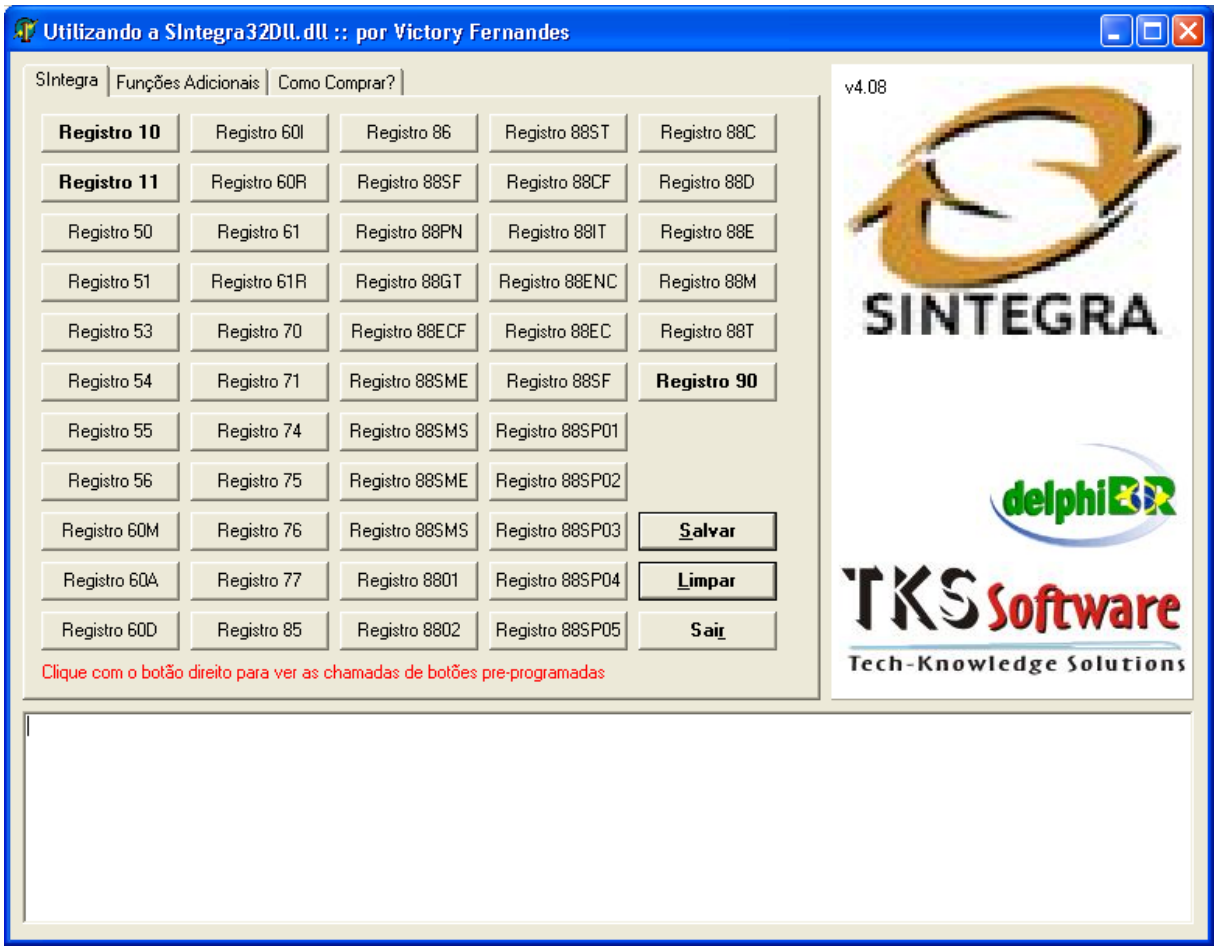

Figura 3.1 Aplicativo de Demonstração da Sintegra32Dll.

3.3 Componente SIntegra - Projeto Sultan

O Componente Sintegra, desenvolvido pelo Projeto Sultan faz parte de um esforço da comunidade participante desse projeto, em desenvolver um componente *open source* para a

geração do arquivo SIntegra. Como o componente é *open source*, o seu desenvolvimento depende dos desenvolvedores que fazem parte do projeto, e colaboram para o progresso do projeto.

O componente SIntegra do Projeto Sultan foi desenvolvido na linguagem Object Pascal, portanto, pode ser usado apenas no ambiente de programação Delphi. Isto lhe traz uma desvantagem em termos de portabilidade em comparação a DLL Sintegra32Dll. Em contrapartida, confere-lhe a possibilidade de usar todas as funções e métodos disponíveis na ferramenta de programação. Ao contrário da Sintegra32Dll, o SIntegra – Projeto Sultan tem uma rotina principal de geração do arquivo, chamada GerarArquivo(). Contudo, a entrada dos dados é análoga a DLL da TKS Software: todos os atributos devem ser atribuídos pelo programador em tipos de dados (objetos). Cada tipo de registro – registro tipo 10, 11, 50 etc. - tem uma classe equivalente, que deve ser preenchida com os seus atributos. Após a chamada do método principal, o componente gera os tipos de registro de acordo com as informações constantes nos objetos internos do componente. Assim como a Sintegra32Dll, esse tipo de implementação também demanda uma grande quantidade de linhas de programação.

Outro ponto negativo do componente SIntegra – Projeto Sultan é que ainda não foram implementados todos os tipos de registro possíveis no Convênio ICMS 76/03. Uma causa bem provável é o fato de o componente ser *open source*, e o seu desenvolvimento depender da boa vontade dos participantes do projeto.

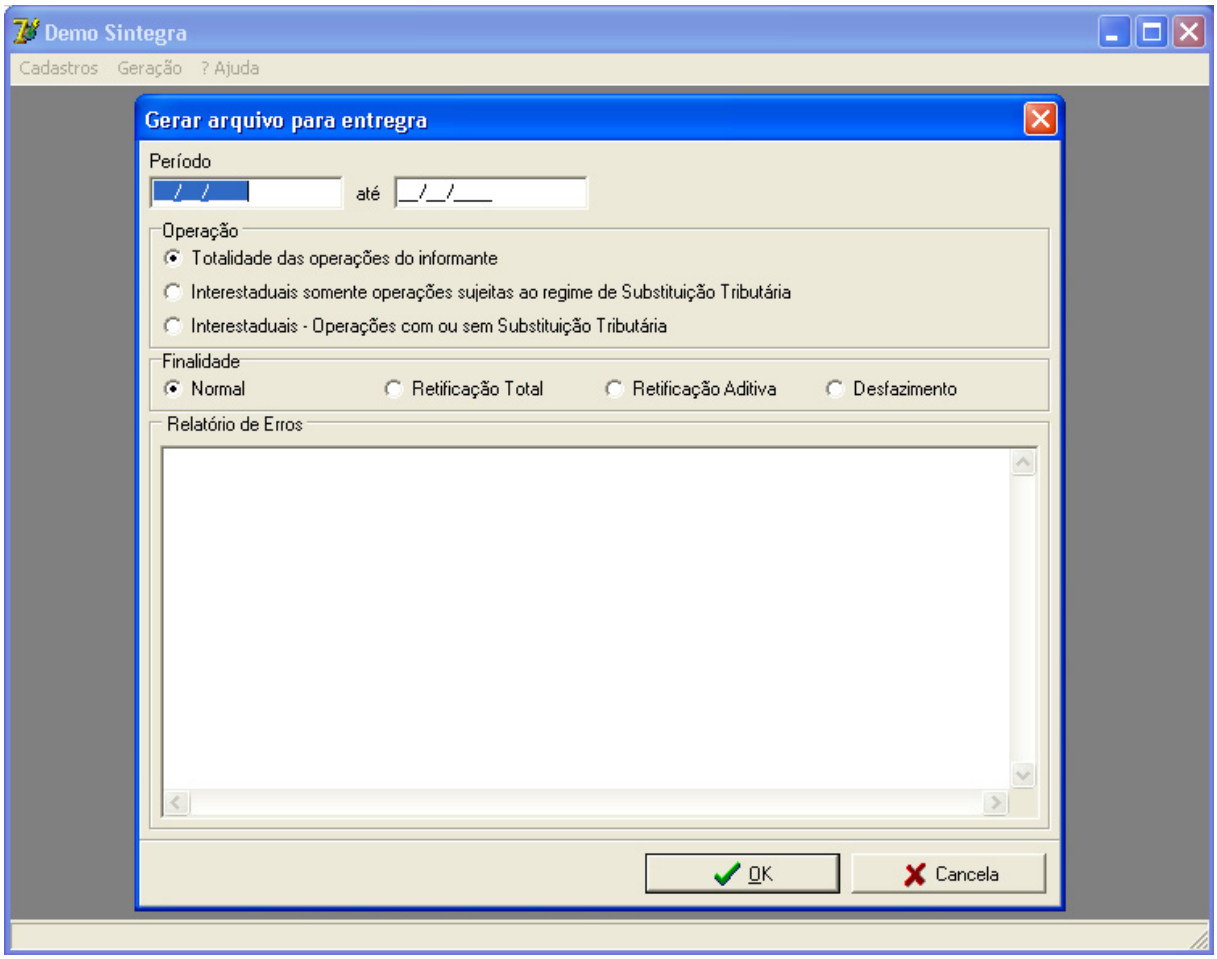

Figura 3.2 Aplicativo de Demonstração do Componente SIntegra – Projeto Sultan.

3.4 Resultados

Baseado na avaliação dos aplicativos de demonstração e no formulário de avaliação da DLL Sintegra32Dll e do componente SIntegra – Projeto Sultan, é possível chegar aos seguintes resultados:

- As duas soluções avaliadas demandam muitas linhas de código para implementar a rotina de geração do arquivo SIntegra. Isto dificulta a manutenção do mesmo – principalmente no caso de atualizações;
- A biblioteca Sintegra32Dll exige do desenvolvedor um profundo conhecimento sobre os tipos de registro do arquivo SIntegra, pois seus métodos geram diretamente os tipos de registro do arquivo SIntegra, e para tanto deve-se passar todos os atributos necessários para a geração do registro. Assim sendo, o

desenvolvedor deve necessariamente ter o conhecimento de quais atributos passar como parâmetro para o método;

- O componente Sintegra – Projeto Sultan já trata os tipos de registro de uma forma mais familiar ao desenvolvedor: por exemplo, o registro 50 - que registra os totais de Notas Fiscais - o componente em questão instancia um objeto chamado "NF", contendo todos os atributos necessários para a geração do seu devido registro – no caso, o registro 50;

Com base nos problemas apontados – necessidade de muitas linhas de código, necessidade de conhecimento da legislação do SIntegra – é possível propor o modelo de um novo componente para a geração do arquivo SIntegra, focado na facilidade de uso por parte dos desenvolvedores – o componente a ser proposto deve ser de fácil implementação, exigindo assim um mínimo de linhas de programação – e na transparência de uso – com o uso dos conceitos de ETL, a única tarefa do desenvolvedor vai ser definir quais os atributos que serão extraídos da fonte de dados, e o componente então fará uso destes para gerar o arquivo SIntegra.

4 PROPOSTA PARA UM NOVO COMPONENTE

Como já foi dito anteriormente, desenvolver a rotina de geração do arquivo SIntegra requer um amplo conhecimento da legislação. Mesmo utilizando uma das soluções avaliadas anteriormente, o desenvolvedor deve necessariamente conhecer a estrutura interna do arquivo SIntegra – os tipos de registro que o compõe – pois tanto a biblioteca Sintegra32Dll quanto o componente do Projeto Sultan apenas geram as linhas do arquivo. Para isso, deve-se passar como parâmetro todos os dados necessários para a geração de um registro do arquivo. Segue abaixo o trecho de código para a geração de um registro tipo 50, utilizando a biblioteca Sintegra32Dll:

```
Registro50('34.261.131/0001-44',      //CNPJ
                      'ISENTO',       //Insc_Est
                      '01/01/2005',   //Data_Emissao_Recebimento
                      'BA',           //UF,
                      '1A',           //Modelo
                      '1',            //Serie
                      '1',            //Nro
                      '5111',         //CFOP
                      'P',            //Emitente
                      '0',            //Valor_Total
                      '0',            //Base_ICMS
                      '0',            //Valor_ICMS
                      '',             //Isenta
                      '0,00',         //Outras
                      '0',            //Aliquota
                      'N'             //Situacao
                     );
```

Abaixo, outro trecho de código para a geração do mesmo registro tipo 50, utilizando componente do Projeto Sultan:

```
with NF.New do
  begin
    { Dados da nota fiscal }
```

```
    CNPJ                := Dados.Registro50CNPJCPF.Value;
    InscEstadual        := Dados.Registro50InscEstadual.Value;
    Data                := Dados.Registro50Data.Value;
    UF                  := Dados.Registro50UF.Value;
    Modelo              := Dados.Registro50Modelo.Value;
    Serie               := Dados.Registro50Serie.Value;
    Numero              := Dados.Registro50Numero.Value;
    Frete               := Dados.Registro50Frete.Value;
    Seguro              := Dados.Registro50Seguro.Value;
    DespAcessoria       := Dados.Registro50DespAcessorias.Value;

    case Dados.Registro50ModFrete.Value[1] of
      'C': ModFrete := mdfCIF;
      'F': ModFrete := mdfFOB;
      'O': ModFrete := mdfOUTROS;
    end;

    case Dados.Registro50Emitente.Value[1] of
      'P': TipoEmitente := tpeProprio;
      'T': TipoEmitente := tpeTerceiros;
    end;

    case Dados.Registro50Situacao.Value[1] of
      'N': Situacao := nfNormal;
      'S': Situacao := nfCancelado;
      'E': Situacao := nfExtNormal;
      'X': Situacao := nfExtCancelado;
    end;
  end;
```

Como se pode notar através da análise dos trechos de código acima, ambas as soluções passam como parâmetro os atributos necessários à criação do tipo de registro. É justamente isto que o novo componente visa: em momento algum o desenvolvedor estará implementando ou declarando alguma rotina que vai gerar um tipo de registro específico; o desenvolvedor apenas chamará o método principal, chamado “Exportar”.

Isto será possível utilizando os conceitos de ETL. O componente aqui proposto reproduzirá o comportamento de um sistema ETL, ou seja, vai extrair os dados necessários a partir de uma fonte de dados, após fará a transformação destes, com base nas regras do Convênio ICMS 76/03, e finalmente fará a carga dos valores transformados para o arquivo destino – o arquivo SIntegra.

Primeiramente, o desenvolvedor terá que definir uma consulta SQL que vai trazer como resultado os dados necessários para a geração do arquivo SIntegra, ou carregar um arquivo XML contendo os dados para o mesmo. O resultado da consulta SQL ou o arquivo XML atuarão como a fonte de dados do sistema ETL embutido no componente aqui proposto.

Após a definição da fonte de dados, o desenvolvedor terá que definir como os dados serão extraídos desta. Isto corresponde ao processo de extração. Este processo é muito

importante, pois é nesta etapa que serão definidos quais atributos serão extraídos da fonte de dados para a geração do arquivo SIntegra. Vale salientar que, diferentemente das soluções anteriormente avaliadas, o desenvolvedor não irá atribuir diretamente os valores constantes na fonte de dados para a geração do arquivo SIntegra, apenas irá definir quais atributos serão utilizados para a criação dos tipos de registro do arquivo. A rotina interna do componente será a responsável pela atribuição e utilização destes valores. Os valores extraídos da fonte de dados serão armazenados em estruturas internas, mais precisamente classes de objetos. Para cada registro extraído da fonte de dados instanciar-se-á o objeto correspondente. Estes objetos por sua vez serão armazenados em listas de objetos, para posterior utilização no processo de carga do arquivo SIntegra.

Definidos os atributos a serem extraídos da fonte de dados, o processo de geração do arquivo SIntegra praticamente é automatizado, pois o componente vai transformar – aqui se inicia o processo de transformação dos dados – os atributos originais definidos no processo de extração, fazendo todas as rotinas internas necessárias (colocar os dados em conformidade com os tipos de registro do arquivo SIntegra), para finalmente gerar o arquivo SIntegra (a carga dos dados). A transformação dos dados acontece praticamente ao mesmo tempo que a carga dos dados no arquivo destino, pois os valores armazenados nos objetos internos estão no seu formato original, e precisam ser transformados, conforme as regras de formatação EDI constantes no Convênio ICMS 76/03.

Para exemplificar melhor o funcionamento do componente, a seção 4.1 traz os diagramas UML do mesmo.

4.1 Modelagem do Componente

Na figura 4.1, o diagrama de Classes apresenta as principais classes do componente aqui proposto. A classe principal (TSWSintegra) tem apenas um método público (Exportar), que será o método a ser chamado para a geração do arquivo SIntegra.

Como pode ser notado no diagrama de classes, o componente SIntegra proposto tem sua estrutura de classes sob a forma de uma árvore binária, com a classe TSWSintegra sendo a raiz da estrutura, ou classe-pai. Desta forma, será possível ao desenvolvedor acessar as

classes-filho do componente, para definição das entidades que serão responsáveis pela extração dos dados – as propriedades constantes em TSWSintegra.ETL.Entidades. Além disso, o método Exportar, responsável pela geração do arquivo SIntegra terá de acessar as classes-filho durante a execução deste método, principalmente no que diz respeito ao acesso a classe TETL, que reproduz o comportamento de um sistema ETL em si.

A classe TETL é onde estão modelados os métodos internos que farão a extração dos valores da fonte de dados. Para cada tipo de registro existe um método privado correspondente, que vai percorrer os registros da fonte de dados, instanciar um objeto interno que vai receber os valores extraídos. Este objeto será armazenado em sua lista de objetos correspondente. Estas listas serão utilizadas para a geração do arquivo SIntegra.

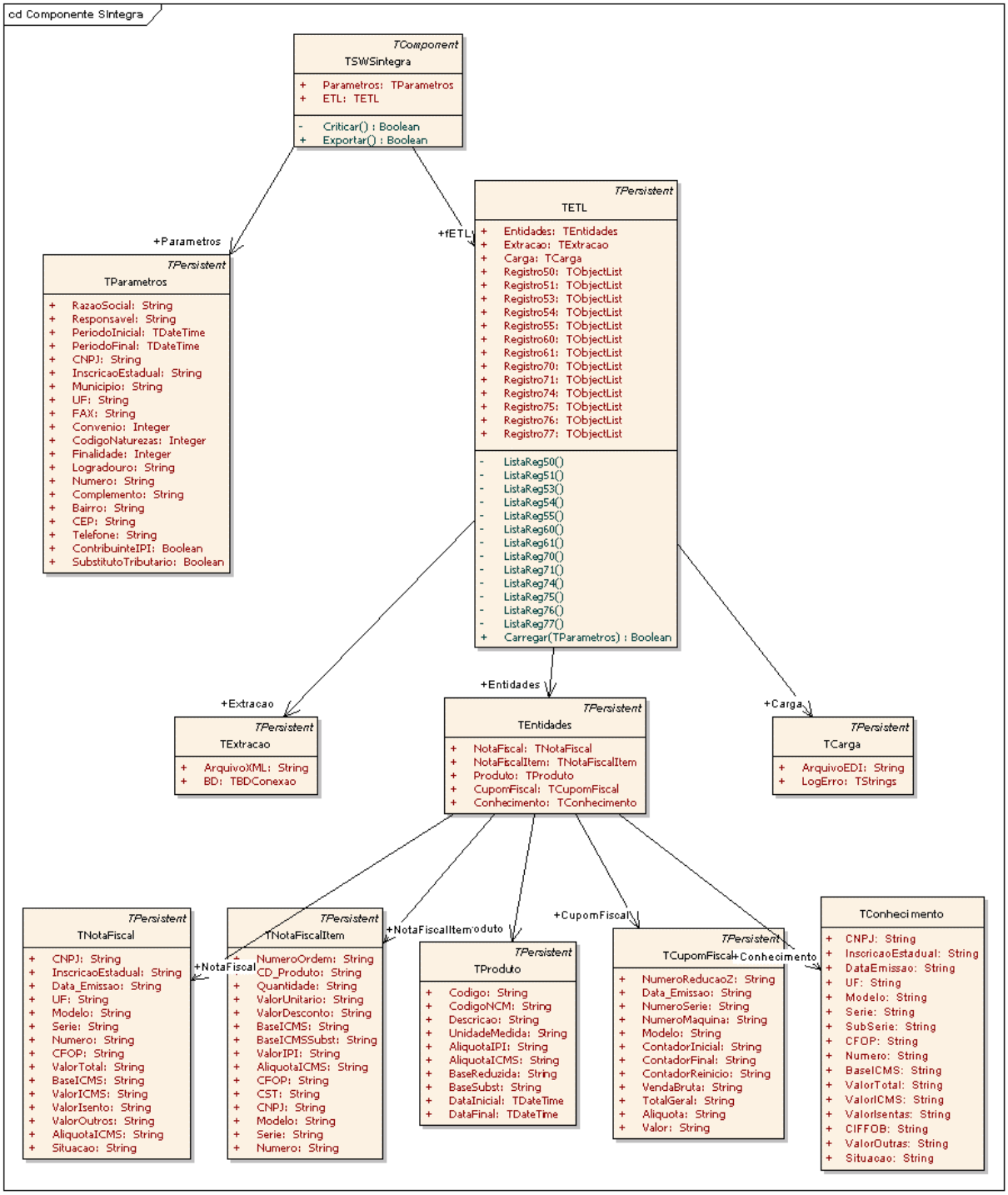

Figura 4.1 Diagrama de Classes.

A figura 4.2 apresenta o diagrama de atividades, que mostra as atividades desenvolvidas para a geração do componente.

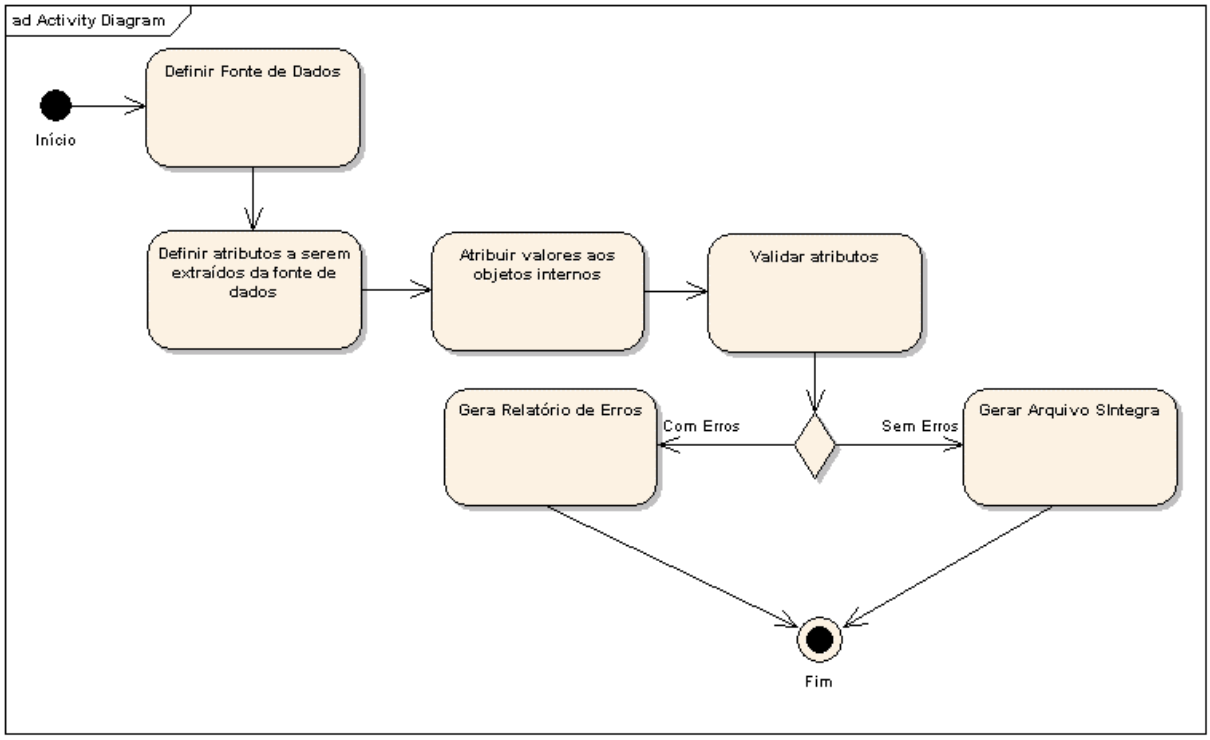

Figura 4.2 Diagrama de Atividades.

No diagrama de atividades tem-se uma visão mais clara do que o componente aqui modelado vai implementar. Vale salientar que as únicas atividades que cabem ao desenvolvedor são aquelas relativas à definição da fonte de dados, bem como a definição dos atributos a serem extraídos desta. Assim como num sistema ETL onde os processos de extração, transformação e carga dos dados são automatizados, o componente aqui modelado também pretende que estes processos também serão automatizados. Com isso, ter-se-á um ganho em relação às soluções já existentes no mercado, pois o desenvolvedor estará isento de implementar tanto a extração dos dados, a transformação destes, bem como implementar a carga dos dados, ou seja, a criação e gravação dos tipos de registro no arquivo SIntegra.

A atividade de validação dos atributos, feita pelo método Criticar, tem por função identificar valores inconsistentes nos atributos extraídos da fonte de dados. Ao contrário de um sistema ETL, que é capaz de tornar um dado inconsistente válido – desde que o sistema tenha sido configurado previamente para isso – o processo de validação do componente aqui modelado apenas identifica os atributos inconsistentes. Caberá ao usuário fazer o ajuste no atributo, de modo a torná-lo válido.

Na seção 4.2 descrever-se-á a implementação do componente aqui modelado. Com a descrição da implementação, será possível visualizar e compreender melhor como os

conceitos de ETL foram utilizados neste trabalho. Assim, será possível mensurar o ganho que o uso dos conceitos desta tecnologia trará aos usuários do componente aqui proposto.

4.2 Implementação do Componente

Após a modelagem, partiu-se para a implementação do componente. O componente foi implementado utilizando o ambiente de programação Borland® Delphi 7, que dá suporte aos conceitos da POO utilizados para a implementação deste.

As grande maioria das classes do componente são herdadas de TPersistent, que é uma classe Delphi nativa usada para a implementação de classes de objetos. Este tipo de classe é próprio para esta implementação, pois trata-se da classe base para implementar objetos no Delphi, tendo apenas métodos construtores e destrutores implementados. Além disso, foram utilizadas outras classes de objetos Dephi para o acesso a base de dados e para a manipulação de Strings.

A seguir, explanar-se-á sobre a implementação do componente ora modelado.

4.2.1 *TSWSintegra*

A *TSWSintegra* é a super-classe do componente, que engloba todas as demais classes do mesmo. Esta classe é herdada de *TComponent*, que é a classe base para a implementação de componentes em Delphi.

Por ser a super-classe, é nesta classe que foi implementado o método público *Exportar*, que é o responsável pela geração do arquivo SIntegra em si. Este método formata os campos conforme o layout e os grava no arquivo destino. Para criar os tipos de registro do arquivo o componente acessa a estrutura interna do mesmo, onde foram alocados os atributos extraídos da base de dados. Esta estrutura será melhor explicada na seção 4.2.3.

A classe *TSWSintegra* possui também o método privado *Criticar*. Este método é chamado pelo método *Exportar*, para fazer a validação dos atributos constantes na propriedade Parametros, que é da classe *TParametros*. A classe *TParametros*, por sua vez,

contém os atributos que vão formar os registros 10 e 11 do arquivo SIntegra. Estes atributos dizem respeito a organização que está gerando o arquivo – razão social, endereço, município etc. – e também informações específicas do arquivo, como período ao qual o arquivo se refere, código do convênio etc. Se o método *Criticar* citado anteriormente encontrar alguma inconsistência nos atributos da classe *TParametros*, o componente gera um registro de erro e aborta o processo de geração do arquivo, iniciado pelo método *Exportar*. Deste modo, evitar-se-á a geração de arquivos com incosistências.

Existe ainda a propriedade ETL na classe *TSWSintegra*. Esta propriedade pertence a classe *TETL*, a qual será comentada a seguir.

4.2.2 *TETL*

Esta classe implementa o mecanismo ETL do componente. Como foi explanado no capítulo 1, sistemas ETL dividem-se em três principais etapas: a extração dos dados de uma ou mais fontes de dados, seguido da transformação destes, e finalmente a carga dos dados para a apresentação dos resultados. Analogamente, o componente aqui implementado segue essa estrutura de funcionamento. Para que haja o entendimento do funcionamento do mecanismo ETL do componente aqui implementado, explanar-se-á primeiramente sobre as propriedades que este possue: as propriedades Entidades (*TEntidades*), Extracao (*TExtracao*) e Carga (*TCarga*).

4.2.3 *TEntidades*

A classe *TEntidades* implementa o mapa lógico dos dados do mecanismo ETL do componente. A partir da definição do mapa lógico, o componente será capaz de extrair da fonte de dados – no caso desta implementação, uma consulta SQL ou um arquivo XML – o valor do atributo definido no mapa lógico de dados. A classe *TEntidades* possui seis propriedades. São elas: NotaFiscal (*TNotaFiscal*), NotaFiscalItem (*TNotaFiscalItem*), Produto (*TProduto*), CupomFiscal (*TCupomFiscal*), Conhecimento (*TConhecimento*) e ConhecimentoNF (*TConhecimentoNF*). Estas propriedades por sua vez, possuem atributos onde serão definidos os respectivos nomes dos atributos da fonte de dados a serem extraídos. A definição deste mapa lógico de dados é de suma importância, pois dele vai depender a

correta extração dos dados da fonte de dados e a subseqüente geração do arquivo SIntegra. A figura 4.3 ilustra a definição do mapa lógico de dados para a extração das informações referentes aos produtos.

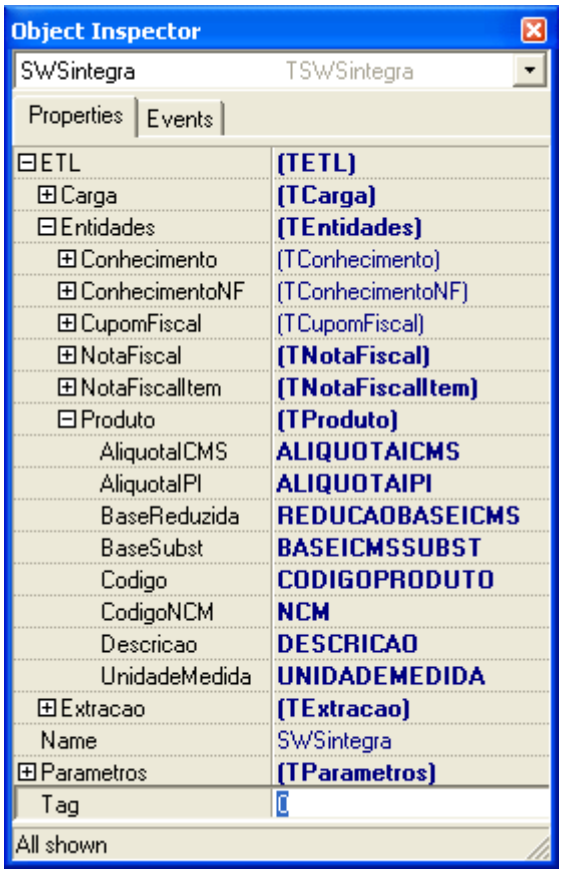

Figura 4.3 Definição do mapa lógico para a Entidade Produto.

Ao serem extraídas da fonte de dados, as informações são armazenadas em objetos internos, e estes por sua vez, são armazenados em listas de objetos. Para cada mapa lógico de dados foi implementado uma classe de objeto respectiva, que vai receber e armazenar os valores extraídos da fonte de dados. Estas classes foram implementadas seguindo os padrões de desenvolvimento (*Design Patterns*). Nesse padrão de desenvolvimento existem classes de objetos responsáveis pelo armazenamento dos valores dos objetos, denominandas *Value Objects*, ou VO's (SUN, 2006). Por exemplo, para armazenar os dados extraídos referentes aos produtos, foi implementada uma classe *TProdutoVO*, conforme trecho de código abaixo:

```
TProdutoVO = class(TObject)
private
   ...
published
  property DataInicial: TDateTime;
  property DataFinal: TDateTime;
  property Codigo: String;
```

```
  property CodigoNCM: String;
  property Descricao: String;
  property UnidadeMedida: String;
  property AliquotaIPI: Real;
  property AliquotaICMS: Real;
  property BaseReduzida: Real;
  property BaseSubst: Real;
end;
```

Estas classes de objetos VO serão utilizadas no processo de extração dos dados, conforme descrito na seção 4.2.5.

4.2.4 *TExtracao*

A classe *TExtracao* é a responsável pela definição da fonte de dados. Assim como em sistemas ETL, que permitem a extração de diversas fontes de dados, o componente aqui implementado tem suporte a dois tipos de fontes de dados: é possível extrair os dados de um arquivo XML ou diretamente de uma base de dados, através de uma consulta SQL. Para tanto, esta classe possui duas propriedades: a primeira, chamada ArquivoXML, deve ser utilizada quando pretende-se gerar o arquivo SIntegra a partir da extração dos dados constantes em um arquivo XML; a segunda, chamada BD (*TBDConexao*), deve ser utilizada quando o arquivo SIntegra será gerado com os dados obtidos através de uma consulta SQL a uma base de dados. Vale lembrar que a utilização de uma das opções é mutuamente exclusiva, ou seja, não é possível gerar o arquivo SIntegra a partir de um arquivo XML e uma consulta SQL, apenas uma das fontes será utilizada. Para utilizar um arquivo XML, basta definir o nome do mesmo. Já para utilizar consultas SQL, é preciso configurar a propriedade BD, conforme descrito a seguir.

4.2.4.1 *TBDConexao*

Esta classe foi implementada visando o uso de consultas SQL para a obtenção dos dados que gerarão o arquivo SIntegra. Para uma maior flexibilidade, esta classe permite ao usuário utilizar diferentes classes de componentes de conexão a bancos de dados. É possível utilizar os componentes BDE, IBX ou dbExpress para a conexão ao banco de dados. Para tanto, basta definir o tipo apropriado na propriedade Tipo. Para a conexão ao banco de dados

propriamente dita, temos a propriedade Query. O usuário deve associar uma instância de um componente que realiza consultas SQL a essa propriedade, que o componente se encarregará de executar a consulta e retornar os respectivos resultados da mesma.

Para a definição da consulta SQL, houve uma mudança na modelagem do componente. No modelo original, havia apenas uma propriedade para a definição da consulta SQL, chamada SQL. Visando uma maior flexibilidade, foram incluídas mais três propriedades para a definição de consultas SQL. Assim, as propriedades foram denominadas conforme sua utilização:

- SQLEntradas: consulta SQL para as notas fiscais de entrada;
- SQLSaidas: consulta SQL para as notas fiscais de saída;
- SQLConhecimentos: consulta SQL para os conhecimentos de frete;
- SQLCupomFiscal: consulta SQL para as informações relativas a cupom fiscal.

Cabe ao usuário do componente definir se será necessário utilizar uma ou mais consultas SQL para gerar o arquivo SIntegra.

A seguir, descrever-se-á a última propriedade da classe *TETL*: a propriedade Carga (*TCarga*).

4.2.5 *TCarga*

Como o próprio nome sugere, a classe *TCarga* representa a última etapa de um sistema ETL: a carga dos dados transformados para apresentação dos resultados. Neste caso, a apresentação dos resultados dá-se através da geração do arquivo SIntegra. Assim, nesta classe deve ser definido o nome do arquivo que será gerado. Essa definição é feita na propriedade ArquivoEDI. Esta propriedade recebeu este nome porque o arquivo SIntegra nada mais é do que um arquivo EDI. Além desta propriedade, a propriedade LogErro presente nesta classe, é responsável por armazenar todos os erros de validação gerados ao se chamar o método Exportar na super-classe *TSWSintegra*. Todo e qualquer erro de validação é armazenado nesta propriedade, e impede a geração do arquivo SIntegra.

Voltando à classe *TETL*, explanar-se-á agora sobre a implementação de seus métodos públicos e privados. O método *Extrair(Parametros)* é chamado pelo método *Exportar* da

super-classe *TSWSintegra*, e faz a função de extração do mecanismo ETL. Ele é responsável por verificar quais consultas SQL foram definidas, e subseqüentemente executá-las. Com os resultados das consultas SQL armazenados em memória, o método *Extrair* inicia a extração dos dados propriamente dita.

A extração dos dados é efetuada por métodos privados chamados *ListaReg50*, *ListaReg54*, *ListaReg60*, *ListaReg70*, *ListaReg71* e *ListaReg75*. Cada um destes métodos vai extrair os dados conforme o mapa lógico definido pelo usuário do componente, e serão armazenados em instâncias de objetos VO. Estes objetos VO serão por sua vez armazenados em listas de objetos, para posterior geração do arquivo. Por exemplo, o método *ListaReg75* é responsável pela extração das informações relativas aos produtos. Para isso, o método vai buscar o mapeamento lógico para a extração das informações na propriedade Entidades.Produto. A seguir, instancia-se um objeto da classe *TProdutoVO* que vai receber os valores dos atributos extraídos da seguinte forma:

```
ProdutoVO := TProdutoVO.Create;

ProdutoVO.AliquotaICMS :=
Query.FieldByName(Entidades.Produto.AliquotaICMS).AsFloat;

ProdutoVO.AliquotaIPI :=
Query.FieldByName(Entidades.Produto.AliquotaIPI).AsFloat;

ProdutoVO.BaseReduzida :=
Query.FieldByName(Entidades.Produto.BaseReduzida).AsFloat;

ProdutoVO.BaseSubst :=
Query.FieldByName(Entidades.Produto.BaseSubst).AsFloat;

ProdutoVO.Codigo := Query.FieldByName(Entidades.Produto.Codigo).AsString;

ProdutoVO.CodigoNCM :=
Query.FieldByName(Entidades.Produto.CodigoNCM).AsString;

ProdutoVO.Descricao :=
Query.FieldByName(Entidades.Produto.Descricao).AsString;

ProdutoVO.UnidadeMedida :=
Query.FieldByName(Entidades.Produto.UnidadeMedida).AsString;
```

Após a extração dos valores e armazenamento no objeto VO, é chamado o método privado *Criticar(VO)*, onde é feita a validação dos valores dos atributos do objeto VO. Caso exista alguma inconsistência no objeto VO, é gerado um registro na propriedade Carga.LogErro. Mas, ao contrário do método Criticar chamado pelo método Exportar da

super-classe *TSWSintegra*, a presença de incosistências não aborta o processo de extração, justamente com o intuito de extrair todos os registros retornados na consulta SQL e validá-los. Deste modo, o usuário terá o registro de todas as inconsistências presentes nos seus dados, podendo corrigi-las de uma única vez. Após o método *Criticar(VO)*, o objeto VO é adicionado a uma lista de objetos. Este processo é semelhante em todos os demais métodos privados menciados acima (*ListaReg50*, *ListaReg54* etc.).

É importante salientar que o processo de extração dos dados implementado no método Extrair é o grande diferencial proposto neste componente, em relação às soluções existentes no mercado. Este método exime o desenvolvedor de implementar a atribuição dos valores retornados na consulta SQL aos objetos que serão armazenados nas listas de objetos, para posterior geração do arquivo SIntegra. Eximindo o desenvolvedor desta implementação, não existe a necessidade do mesmo ter conhecimento a respeito da legislação concernente ao SIntegra – outra proposta do componente ora implementado neste trabalho.

Terminado o processo de extração dos dados e, não tendo sido encontrada nenhuma inconsistência pelo método Criticar, o método *Exportar* continua. Neste momento inicia-se a geração do arquivo SIntegra propriamente dita. Fazendo uma analogia a sistemas ETL, neste momento tem-se a transformação dos dados. No caso específico do componente aqui implementado, a transformação dá-se no momento em que os valores dos objetos VO armazenados em listas de objetos internas são formatados conforme o *layout* SIntegra. Ao mesmo tempo ocorre o processo de carga dos valores formatados no arquivo destino, o arquivo SIntegra. Para isso, o método percorre as listas de objetos e, à medida que lê os objetos VO da lista, formata (transforma) os valores dos atributos o objeto VO conforme o *layout* e imediatamente grava o registro no arquivo destino. Após percorrer todas as listas de objetos, o arquivo destino é finalizado com um registro finalizador chamado *trailer*. Ao finalizar o arquivo, chega-se ao fim do método *Exportar*.

É perceptível que o componente aqui implementado tem o mesmo comportamento de um sistema ETL, automatizando ao máximo o processo de extração, transformação e carga dos valores para a geração do arquivo SIntegra. Deste modo, pretende-se reduzir o tempo desprendido para implementar a rotina de geração do arquivo SIntegra, bem como propiciar aos desenvolvedores uma alternativa para geração do mesmo sem a necessidade do conhecimento da legislação concernente ao SIntegra.

Com o componente implementado, é necessário avaliar se o mesmo oferece vantagem

em relação às soluções existentes no mercado. O próximo capítulo trata da avaliação do componente SWSintegra.

5 AVALIAÇÃO DO COMPONENTE

Após a implementação do componente proposto, é necessário avaliar se o mesmo atingiu seus objetivos – ser de fácil utilização e configuração para seu usuário, o desenvolvedor de software. Além disso, o componente recém-implementado não pode exigir do seu usuário um conhecimento muito avançado sobre a legislação concernente ao SIntegra, pois como pode ser visto no capítulo 3, a estrutura de tipos de registro do arquivo SIntegra é bastante complexa. Se, para utilizar o componente proposto e implementado neste trabalho, fosse necessário esse conhecimento ao desenvolvedor, este trabalho não teria sentido, pois já existem ferramentas no mercado que geram o arquivo SIntegra, contudo, estas requerem o conhecimento da legislação para seu uso.

Para a avaliação do componente SIntegra desenvolvido, deve-se necessariamente compará-lo às soluções já citadas neste trabalho – a Sintegra32dll, da TKS Software e o componente SIntegra, do Projeto Sultan. Desta forma, será possível não apenas comparar se o resultado obtido – o arquivo SIntegra – é satifatório, mas sim comparar qualitativamente e quantitativamente as soluções pré-existentes e o componente aqui implementado.

Desta forma, será possível comparar se o arquivo SIntegra gerado pelas três soluções possui o mesmo conteúdo, além de analisar e avaliar a qualidade do código-fonte das três implementações. Finalmente, observar-se-á se o componente aqui proposto traz vantagens ao desenvolvedor, no que diz respeito a tempo de implementação e posterior manutenção do código-fonte. Para tanto, foi elaborado um caso de teste, descrito a seguir.

5.1 Caso de Teste

Primeiramente, a solução implementada foi submetida ao formulário de avaliação de *software,* definido no capítulo 3. O resultado da avaliação encontra-se no Anexo A.

Por se tratar de um componente, não é possível testar ou avaliar o mesmo de forma modular, sem que ele esteja inserido em um programa de computador (mesmo que seja um programa elaborado com o fim exclusivo de testar o componente). Da mesma forma, as outras soluções citadas também precisam da estrutura de uma aplicação para o seu funcionamento. Assim, o caso de teste elaborado consiste na implementação de uma aplicação que vai gerar o arquivo SIntegra a partir de uma base de dados com a seguinte estrutura:

- Cadatro de Clientes;
- Cadastro de Produtos;
- Cadastro de Notas Fiscais;
- Cadastro de Itens de Notas Fiscais;
- Cadastro de Empresas.

Estes cadastros encontram-se devidamente populados, de modo a simular a base de dados de um sistema de controle de vendas e emissão de notas fiscais, assim obter-se-á a geração do arquivo SIntegra o mais próximo possível da realidade.

Com a base de dados populada, implementar-se-á três aplicativos distintos, cada um destes utilizando uma das soluções abordadas neste trabalho: a Sintegra32dll, o componente SIntegra do Projeto Sultan e o componente SIntegra proposto e implementado neste trabalho. Cada aplicativo implementado deverá gerar o respectivo arquivo SIntegra, e este será validado utilizando o Validador SIntegra versão 5.2.0 – o Validador SIntegra é um aplicativo para validação e edição do arquivo SIntegra, cedido pela Secretaria da Fazenda. Pretende-se que os arquivos gerados, além serem validados e aceitos pelo Validador SIntegra, sejam analisados minuciosamente, para que haja a certeza que as três rotinas de geração do arquivo SIntegra geram os arquivos exatamente iguais, sem distorções nos valores dos atributos. Essa análise será feita através da comparação binária entre os arquivos gerados – como os arquivos gerados são no formato texto, essa análise torna-se simples, através do uso de ferramentas de

comparação de texto. A ferramenta escolhida para este trabalho foi o aplicativo FC.EXE[2], desenvolvido pela MicroSoft, disponível em qualquer versão do sistema operacional Windows.

Como já foi dito anteriormente, o caso de teste não se restringirá a avaliar o arquivo SIntegra gerado pelas três soluções aqui abordadas, mas também avaliará quesitos que interessam ao desenvolvedor de *software*, como quantidade de linhas necessárias para a implementação da rotina de geração do arquivo SIntegra, qualidade do código – o código é de fácil compreensão por parte do desenvolvedor, para posterior manutenção, tempo necessário para a implementação da rotina e o último quesito a ser avaliado, mas não menos importante: se houve a necessidade de conhecimento da legislação concernente ao SIntegra para a implementação da rotina de geração do arquivo ou não. São apenas alguns fatores, mas são determinantes para o desenvolvedor atingir em sua implementação características muito desejáveis ao software: código-fonte padronizado, com um bom grau de manutenibilidade a um baixo custo de desenvolvimento.

5.2 Resultados do Caso de Teste

Após a implementação do aplicativo para gerar o arquivo SIntegra a partir da base de dados citada anteriormente, analisar-se-á os seus resultados. Quanto à geração do arquivo SIntegra, as três soluções – Sintegra32dll, Projeto Sultan e o componente desenvolvido neste trabalho – geraram arquivos exatamente iguais. Os arquivos gerados pelas soluções anteriormente citadas foram validados pelo programa Validador SIntegra versão 5.2.0, e não apresentaram nenhuma inconsistência que pudesse invalidar os mesmos. Deste modo, poder-se-á afirmar que as três soluções podem ser utilizadas em sistemas de controle reais, que necessitem a implementação da rotina de geração do arquivo SIntegra.

Contudo, este caso de teste não se propõe apenas a analisar o resultado final – o arquivo SIntegra – e sim o modo como atingiu-se este resultado, ou seja, a implementação da rotina de geração utilizando cada uma das soluções. Para tanto, analisar-se-á separadamente cada rotina, e posteriomente far-se-á uma comparação entre as três implementações.

[2] FC.exe é um *software* que perfaz uma comparação binária entre dois arquivos elencando os trechos onde os dois diferem.

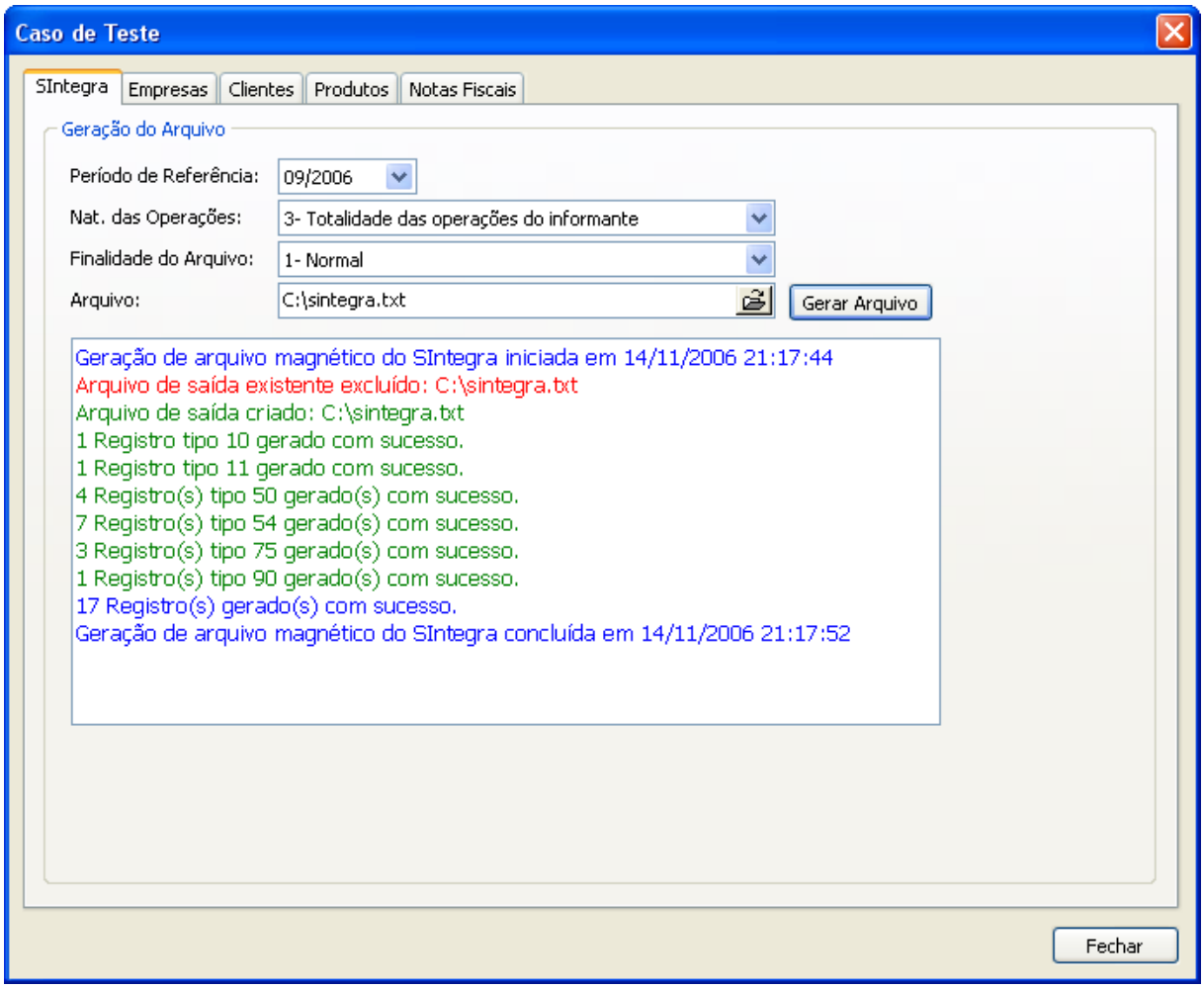

Figura 5.1 Aplicativo de Teste.

5.2.1 Sintegra32dll

Das três soluções utilizadas para implementar a rotina de geração do arquivo SIntegra, a biblioteca Sintegra32dll foi a mais trabalhosa. Por se tratar de uma DLL, não existe um método principal a ser chamado. Deste modo, existe uma função para a criação de cada tipo de registro, que deve ser chamada para a criação dos mesmos. Além disso, cabe ao desenvolvedor implementar as rotinas de criação e escrita no arquivo destino. Como já foi dito neste trabalho, lidar diretamente com os conceitos de tipos de registros do arquivo SIntegra exige do desenvolvedor o conhecimento da legislação concernente ao SIntegra.

Voltando ao aspecto técnico da implementação, o uso da biblioteca Sintegra32dll praticamente impossibilita o desenvolvedor da rotina a utilizar-se de conceitos da POO, ou seja, a rotina é extremamente procedural. Isto porque, como já foi dito anteriormente, não

existe a possibilidade de chamar um método principal que gerará o arquivo, mesmo porque não há como armazenar as informações que vão gerar os registros do arquivos em alguma estrutura interna da biblioteca, para posteriormente chamar o método gerador do arquivo. Para gerar os tipos de registros do arquivo, o desenvolvedor deve criar uma estrutura de laço, que chame a rotina de criação do tipo de registro n vezes, até que todos os registros tenham sido gravados no arquivo destino.

Outro aspecto relevante na implementação da rotina de geração do arquivo SIntegra utilizando a Sintegra32dll foi a quantidade de consultas SQL necessárias para gerar o arquivo. Como foi dito anteriormente, para este caso de teste foi utilizada uma base de dados devidamente populada, simulando um sistema de controle de vendas e emissão de notas fiscais de uma organização. Para a geração de cada tipo de registro foi necessária pelo menos uma consulta SQL. A partir da base de dados, o arquivo gerado teve registros dos tipos 10, 11, 50, 54, 75 e 90. Conforme a realidade da organização, a variedade de registros gerados no arquivo é maior, ou seja, mais consultas SQL serão necessárias.

Todos os aspectos relacionados na implementação do caso de teste utilizando a biblioteca Sintegra32dll levam a crer que a utilização desta solução para a geração do arquivo Sintegra gera um código fonte extremamente volumoso e de difícil manutenção. A manutenção de software é reconhecida por vários autores (PRESSMANN, 2001) (HANNA, 1993) (SNEED, 2003) como a atividade que demanda o maior volume de esforço dentre todas as atividades de engenharia de software. Sabe-se que o layout do arquivo SIntegra é constantemente alterado, para estar em conformidade com a legislação fiscal e tributária vigente. Deste modo, o desenvolvedor que optar por utilizar-se da biblioteca Sintegra32dll terá bastante trabalho para manter seu código fonte, pois além de reescrever os métodos chamadores dos tipos de registros, terá de alterar suas consultas SQL. Por outro lado, a utilização da Sintegra32dll dá ao desenvolvedor a portabilidade de poder utilizar a biblioteca em questão em qualquer ambiente de programação que suporte o vínculo a bibliotecas dinâmicas, o que não é possível com as demais soluções citadas abordadas neste trabalho.

5.2.2 Projeto Sultan

No caso de teste utilizando o componente desenvolvido pelo Projeto Sultan, pôde-se observar uma estrutura interna baseada em Interfaces, que serve como repositório dos dados

que vão gerar os registros do arquivo SIntegra. Assim, o desenvolvedor primeiramente deve alimentar essa estrutura com os dados provenientes da base de dados, para depois chamar o método GerarArquivo(Arquivo). Pode-se dizer que Interfaces são estruturas semelhante a objetos e listas de objetos, contudo ela oferece vantagens, principalmente no que diz respeito a instanciação de novos itens de uma lista, bem como seu uso. Em Interfaces, é possível instanciar um novo item de uma lista e atribuir valores aos seus atributos, nem necessariamente saber qual a posição que este objeto ocupa na lista. Diferentemente de uma lista de objetos, onde primeiramente se instancia o objeto, atribui-se os valores às suas propriedades, e finalmente adicioná-lo à lista. Vale lembrar que o componente do Projeto Sultan é desenvolvido por uma comunidade de desenvolvedores com o objetivo de oferecer aos demais desenvolvedores uma solução open source para a geração do arquivo SIntegra. Por esta razão, pode-se considerar o componente em um bom nível de maturidade, pois apresenta a estrutura de Interfaces, e foi desenvolvido sob os conceitos de orientação a objetos.

Mesmo havendo uma melhora no que diz respeito à sua estrutura interna, o componente do Projeto Sultan também necessita de uma ou mais consultas SQL para alimentar a estrutura interna citada anteriormente, que vai gerar os tipos de registro do arquivo SIntegra. A diferença é que os valores retornados da consulta SQL são atribuídos aos objetos internos do componente. Diferentemente da biblioteca Sintegra32dll, que gera os tipos de registro do arquivo à medida que lê os registros retornados da consulta SQL, o componente do Projeto Sultan lê os registros da consulta SQL e os armazena em sua estrutura interna para posterior validação e geração do arquivo SIntegra. Deste modo, o problema de manutenção de código persiste, pois, caso hajam mudanças no layout, haverá a necessidade de alterar tanto as consultas SQL como as atribuições aos valores da estrutura interna de Interfaces.

5.2.3 Componente SWSIntegra

Nesta seção analisar-se-á o caso de teste utilizando o componente modelado e implementado neste trabalho, ora denominado SWSIntegra. Como já foi explanado no capítulo anterior, o componente SWSIntegra foi desenvolvido baseando-se nos conceitos da POO, visando minimizar o trabalho do desenvolvedor que tiver a necessidade de gerar o arquivo SIntegra em seus sistemas. Assim como no componente desenvolvido pelo Projeto Sultan, o componente SWSIntegra utiliza uma estrutura interna de objetos para armazenar os valores retornados da consulta SQL. Entretanto, ao contrário das outras soluções analisadas

neste caso de teste, o componente SWSIntegra necessita apenas de uma consulta SQL para gerar o arquivo SIntegra. A consulta SQL é estruturada de tal forma a retornar de uma única vez todos os valores necessários para a geração dos diversos tipos de registro. Cabe ao desenvolvedor alimentar a estrutura ETL existente no componente, que vai ser responsável por extrair da consulta SQL os valores dos atributos especificados na estrutura ETL, e fazer a atribuição destes valores aos objetos internos do componente que vão armazenar estas informações para posterior geração do arquivo.

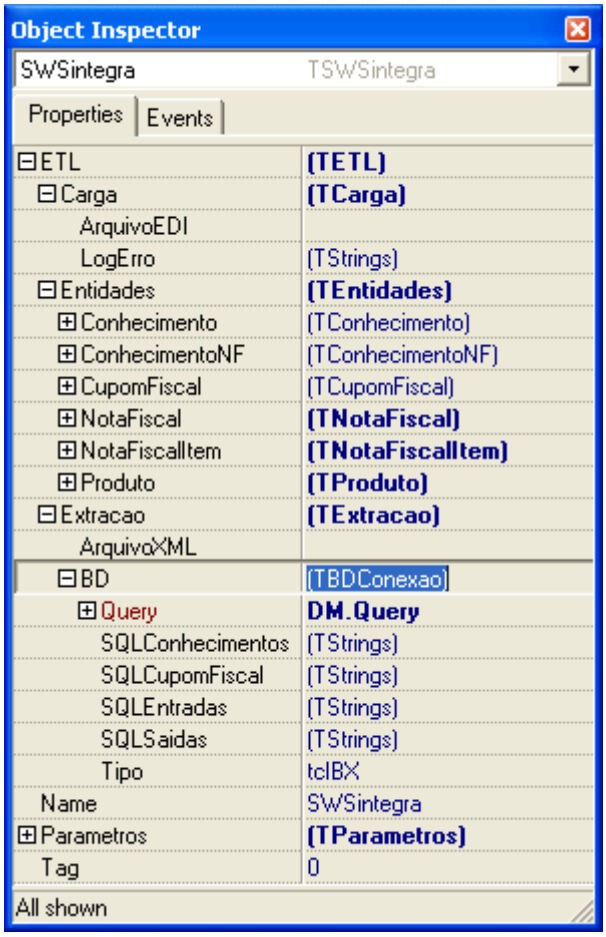

Figura 5.2 Estrutura ETL do Componente SWSintegra

Pelo fato da consulta SQL retornar todos os dados necessários à geração do arquivo SIntegra, a estrutura interna do componente é alimentada sob demanda: ao contrário do Projeto Sultan, onde a estrutura interna é alimentada seguindo a seqüência de geração dos tipos de registro – os registros são gerados obedecendo a ordem crescente dos tipos de registro (10, 11, 50, 54, 75 e finalmente registro 90, neste caso) - no componente SWSIntegra são gerados simultaneamente mais de um tipo de registro, conforme a necessidade, ou seja, pode-se gerar um registro 50, 54 e 75 ao ler um único registro da consulta SQL. O principal diferencial notado ao implementar o caso de teste utilizando o componente SWSIntegra é que

a atribuição dos valores retornados na consulta SQL a estrutura interna do componente é transparente ao desenvolvedor, ou seja, o único trabalho do desenvolvedor é definir a consulta SQL e respectivamente quais campos desta consulta serão atribuídos aos objetos internos do componente.

Pode-se notar um ganho na produtividade do desenvolvedor ao utilizar o componente SWSIntegra desenvolvido neste trabalho, pois caso haja a necessidade de manutenção do código fonte, o desenvolvedor terá apenas um trabalho: alterar a sua consulta SQL e, se houver necessidade, alterar a estrutura ETL para extrair os valores recém incorporados à consulta.

CONCLUSÃO

Através dos estudos realizados, adquiriram-se conhecimentos sobre a tecnologia ETL, tecnologia utilizada de forma eficiente em sistemas de apoio à decisão, bem como em repositórios de dados (*data warehouses*). Foram estudadas as etapas de um sistema ETL - extração, transformação e carga dos dados - para posteriormente modelar e implementar o componente SWSintegra, objetivo principal deste trabalho.

Quanto ao arquivo SIntegra, primeiramente foi feito um estudo sobre arquivos EDI, tecnologia esta desenvolvida com o intuito de padronizar a troca eletrônica de dados. Por se tratar de um arquivo EDI, torna-se necessário o estudo desta tecnologia, para que houvesse o entendimento de como o arquivo SIntegra é formado, e porque este foi assim estruturado. Havendo a compreensão da tecnologia EDI, partiu-se para o estudo da estrutura interna do arquivo SIntegra.

Como é sabido, já existem no mercado ferramentas para auxiliar os desenvolvedores na geração do arquivo SIntegra. Assim, era necessário que a solução proposta neste trabalho trouxesse uma vantagem para os desenvolvedores. A vantagem encontrada ao utilizar o componente modelado e implementado neste trabalho é justamente a junção das duas tecnologias abordadas: EDI e ETL. Assim como as soluções existentes no mercado, o componente proposto neste trabalho implementa a geração do arquivo EDI SIntegra, baseado no Convênio ICMS 76/03. Contudo, a solução aqui implementada faz uso dos conceitos de ETL para automatizar o processo de extração dos dados da fonte de dados, necessários à geração do arquivo. Deste modo, o desenvolvedor fica isento de implementar estas rotinas, que exigem deste o conhecimento da legislação concernente ao SIntegra.

É importante salientar que a solução proposta neste trabalho teve um fim específico - a geração do arquivo SIntegra - mas esta abordagem pode ser utilizada de maneira mais genérica. Desta forma, poder-se-ia implementar uma ferramenta de integração de sistemas.

Como trabalho futuro, pretende-se implementar as rotinas de geração dos registros restantes (registros da classe 60 e 80). Estes registros não foram implementados porque são utilizados em casos muito específicos, portanto sua utilização é menor. Ademais, seria impossível implementar as rotinas de extração e transformação para todos os possíveis registros do arquivo SIntegra.

Enfim, o componente aqui proposto introduz ao mercado de *software* uma alternativa para a geração do arquivo SIntegra, tendo como principais atrativos a facilidade de implementação, baixo custo de implementação e isentar o desenvolvedor de ter conhecimento a respeito da legislação SIntegra.

REFERÊNCIAS BIBLIOGRÁFICAS

ABNTNET. **Associação Brasileira de Normas Técnicas.** Disponível em: <http://www.abntnet.com.br>. Acesso em: 15 jun 2006.

COL, Ana; HELLER, Lucia S.; MELLO, Regina. **EDI : Automação Comercial.** 1996.

EANBRASIL. **EAN Brasil**. Disponível em <http://www.eanbrasil.org.br>. Acesso em: 15 out 2006.

HANNA, M. **Maintenance Burden Begging for Remedy**. Software Magazine, April 1993, pp. 53-63.

HENRY, Scott et al. **Engineering Trade Study: Extract, Transform Load Tools for Data Migration.** Proceedings of the 2005 Systems and Information Engineering Design Symposium, 2005.

ICMS76/03. **Manual de Orientação do Convênio ICMS 76/03.** Disponível em <http://www.sintegra.gov.br>. Acesso em: 15 mai 2006.

KIMBALL, Ralph; CASERTA, Joe. **The Data Warehouse ETL Toolkit.** Wiley Publishing, Inc., 2004.

MHS, Manuel; CORREIA. **EDI-MHS : A comunicação empresarial global.** Érica, 1997.

MSDN. **Microsoft Deleloper's Network.** Disponível em <http://msdn.microsoft.com>. Acesso em: 01 nov 2006.

ONU. **Organização das Nações Unidas - Electronic Data Interchange (EDI).** Disponível em <http://www.unece.org/cefact/refer/comp/intro3.htm>. Acesso em: 15 nov 2006.

PORTAL. **Portal Tributário.** Disponível em <http://www.portaltributario.com.br>. Acesso em: 15 jun 2006.

PRESSMANN, Roger S. **Software Engineering: a practitioner's approach**. New York: McGraw Hill, 5th ed., 2001. 860p.

PRODANOV, Cléber C. **Manual de metodología científica.** 3ª ed. Novo Hamburgo: Feevale, 2003. 77p.

SINTEGRA. **Apresentação Sintegra.** Disponível em: http://www.sintegra.gov.br/. Acesso em: 18 mar 2006.

SINTEGRA2. **SIntegra – Informações Gerais.** Disponível em <http://www.sintegra.gov.br>. Acesso em: 25 mai 2006.

SNEED, H.M. **Critical Success Factors in Software Maintenance**. In: International Conference on Software Maintenance, 2003, 9p.

SUN. **Design Patterns.** Disponível em <http://java.sun.com/blueprints/patterns/TransferObject.html>. Acesso em: 05 nov 2006.

UNI5. **UNI5 – Supply Chain Integration.** Disponível em http://www.uni5.com. Acesso em: 20 mar 2006.

ANEXOS

ANEXO A – FORMULÁRIO DE AVALIAÇÃO DE SOFTWARE

Primeira Avaliação:

Nome: Sintegra32Dll

Versão: 4.0

Fabricante: TKS Software

URL: http://www.igara.com.br

Sistema Operacional: Microsoft Windows XP Professional

Ferramenta de programação: Borland Delphi Enterprise 7

Avaliação:

	N	P	T
A solução avaliada gera todos os possíveis tipos de registro do arquivo SIntegra?			X
A solução avaliada faz algum tipo de crítica nos dados de entrada?			X
O arquivo gerado pela solução avaliada é confiável?			X
A solução avaliada possui uma interface amigável para com o desenvolvedor?	X		
O desempenho da solução avaliada foi considerado:			X
A solução avaliada pode ser utilizada em mais de um ambiente de programação?			X
Em caso de necessidade de manutenção do código, a necessidade de reescrita de código foi:			X
Em caso de erro, a solução avaliada possui algum tipo de registro de erros?		X	
A solução avaliada exige do desenvolvedor o conhecimento aprofundado da linguagem de programação?			X
É necessária a instalação da solução avaliada no ambiente de programação?	X		
A solução avaliada é estável, podendo ser utilizada em sistemas comerciais?			X

Comentários: A SIntegra32Dll gera todos os tipos de registro, mas para tanto existe um método para cada tipo de registro. Assim, fica a cargo do desenvolvedor criar sua rotina própria, que vai chamar os métodos da Sintegra32Dll para gerar o seu arquivo.

Segunda Avaliação:

Nome: SIntegra

Versão: 3.0

Fabricante: Projeto Sultan

URL: http://www.projetosultan.org

Sistema Operacional: Microsoft Windows XP Professional

Ferramenta de programação: Borland Delphi Enterprise 7

Avaliação:

	N	P	T
A solução avaliada gera todos os possíveis tipos de registro do arquivo SIntegra?		X	
A solução avaliada faz algum tipo de crítica nos dados de entrada?			X
O arquivo gerado pela solução avaliada é confiável?		X	
A solução avaliada possui uma interface amigável para com o desenvolvedor?		X	
O desempenho da solução avaliada foi considerado:		X	
A solução avaliada pode ser utilizada em mais de um ambiente de programação?		X	
Em caso de necessidade de manutenção do código, a necessidade de reescrita de código foi:			X
Em caso de erro, a solução avaliada possui algum tipo de registro de erros?		X	
A solução avaliada exige do desenvolvedor o conhecimento aprofundado da linguagem de programação?			X
É necessária a instalação da solução avaliada no ambiente de programação?	X		
A solução avaliada é estável, podendo ser utilizada em sistemas comerciais?		X	

Comentários: O Componente SIntegra – Projeto Sultan não gera todos os tipos de registro SIntegra. Assim, dependendo da necessidade não é possível utilizá-lo. Diferentemente da avaliação anterior, o desenvolvedor não precisa chamar um método para cada tipo de registro, mas existe a necessidade de atribuição manual dos dados de entrada.

Terceira Avaliação:

Nome: TSWSIntegra

Versão: 1.0

Fabricante: Jaques Metz

URL:

Sistema Operacional: Microsoft Windows XP Professional

Ferramenta de programação: Borland Delphi Enterprise 7

Avaliação:

	N	P	T
A solução avaliada gera todos os possíveis tipos de registro do arquivo SIntegra?		X	
A solução avaliada faz algum tipo de crítica nos dados de entrada?			X
O arquivo gerado pela solução avaliada é confiável?			X
A solução avaliada possui uma interface amigável para com o desenvolvedor?			X
O desempenho da solução avaliada foi considerado:			X
A solução avaliada pode ser utilizada em mais de um ambiente de programação?		X	
Em caso de necessidade de manutenção do código, a necessidade de reescrita de código foi:	X		
Em caso de erro, a solução avaliada possui algum tipo de registro de erros?			X
A solução avaliada exige do desenvolvedor o conhecimento aprofundado da linguagem de programação?		X	
É necessária a instalação da solução avaliada no ambiente de programação?		X	
A solução avaliada é estável, podendo ser utilizada em sistemas comerciais?			X

Comentários: O Componente TSWSIntegra mostrou ser de fácil utilização por parte do desenvolvedor. Apesar de não implementar todos os tipos de registro, a solução avaliada pode ser usada em ambientes reais. Além disso, a implementação dos tipos de registro restantes está planejada.