UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

UM PROTÓTIPO DE SOFTWARE ASSISTENTE DE ATUALIZAÇÃO DE VERSÃO DE SOFTWARE

TRABALHO DE ESTÁGIO SUPERVISIONADO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO — BACHARELADO

DENIS CEZAR METZNER

BLUMENAU, AGOSTO/2001

2001/1-22

UM PROTÓTIPO DE SOFTWARE ASSISTENTE DE ATUALIZAÇÃO DE VERSÃO DE SOFTWARE

DENIS CEZAR METZNER

ESTE TRABALHO DE ESTÁGIO SUPERVISIONADO, FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE ESTÁGIO SUPERVISIONADO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Ricardo Azambuja — Orientador na FURB

_____

Rodrigo Norberto Lermen — Orientador na Empresa

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Ricardo Azambuja

_____

Prof. Everaldo Artur Grahl

_____

Prof. Paulo Roberto Dias

DEDICATÓRIA

Em primeiro lugar a DEUS, por ter me guiado e protegido em todos os momentos, a quem agradeço por tudo o que consegui durante toda a minha vida e por mais esta vitória.

À minha família, em especial meus pais pelo esforço sem medida para me proporcionar a melhor formação possível.

À minha noiva, pela infinita paciência por suportar horas e dias de lazer que deixamos de usufruir.

Aos meus amigos que de alguma forma contribuíram diretamente e indiretamente para que mais essa vitória pudesse ser conquistada.

Portanto, dedico este trabalho final a DEUS, meus pais, minha noiva, minha irmã e meus amigos por todo o amor, compreensão, carinho, dedicação, apoio e incentivo dado a mim durante toda minha caminhada acadêmica.

AGRADECIMENTOS

Especialmente a DEUS, meus pais e minha noiva, por todo o apoio e confiança dedicados a mim e por suportarem a minha ausência nas intermináveis horas de dedicação à elaboração e confecção deste trabalho.

Ao professor RICARDO ALENCAR DE AZAMBUJA, pela supervisão, críticas e sugestões dadas no decorrer do trabalho, bem como pela amizade estabelecida.

A HEITOR RODOLFO DE SOUZA, proprietário da empresa Strategies Ltda, que sempre me apoiou e acreditou em mim, por quem tenho profundo respeito, admiração e amizade.

A RODRIGO NORBERTO LERMEN, chefe e colega, que acompanhou todo o andamento do trabalho e orientou-me em muitas dificuldades encontradas.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

RESUMO

O atual estágio do mercado de softwares obriga as empresas se tornarem mais ágeis e competitivas. O desenvolvimento de aplicações que agilizam processos internos e externos da empresa se torna uma fonte de auxílio e recurso indispensável. Pensando nisso e face à necessidade da empresa, o presente trabalho propôs a elaboração de um protótipo de sistema de automatização dos processos de atualização de versão de software. O objetivo desse protótipo é atender a uma necessidade da empresa que é agilizar o processo de fechamento de versão e atualização desta versão no cliente. Esses processos atualmente são efetuados manualmente por um funcionário da empresa desenvolvedora e ambos os processos demandam um alto consumo de tempo e custo. Para a implementação do protótipo, utilizou-se a linguagem de programação Uniface 7.0 com base na metodologia de análise estruturada de sistemas auxiliada pela ferramenta Power Designer.

ABSTRACT

The current apprenticeship of the market of softwares forces the companies if they turn more agile and competitive. The development of applications that they activate internal and external processes of the company becomes a source of aid and indispensable resource. Thinking of that and face to the need of the company, the present work proposed the elaboration of a prototype of system of automation of the processes of modernization of software version. The objective of that prototype is of assisting to a need of the company that is to activate the process of version closing and modernization of this version in the customer. Those processes now are made manually by an employee of the developer company and, both processes demand a high consumption of time and cost. For the implementation of the prototype, the programming language Uniface 7.0 was used with base in the methodology of structured analysis of systems aided by the tool Power Designer.

1 INTRODUÇÃO

O mercado atual tem forçado as empresas desenvolvedoras de softwares ao constante desenvolvimento. Nas parcerias entre empresa desenvolvedora e seus clientes, encontram-se dificuldades no que diz respeito à padronização de software, pois cada cliente possui necessidades diferentes. Face à este fato, a empresa obriga-se a efetuar um controle de versões, para cada cliente, cujo processo de atualização do sistema (implantação de nova versão) gera uma grande mobilização. As versões são criadas e fechadas periodicamente e, os clientes sempre têm o interesse de atualizar o software para a última versão disponível. Para isso, a empresa desenvolvedora do software necessita que, sempre que um cliente solicitar uma atualização, deslocar um funcionário até o endereço do cliente, com a versão desejada, para efetuar as alterações necessárias. Essas alterações são feitas na estrutura do modelo de dados. Tanto para a empresa desenvolvedora quanto para o cliente esse processo gera custos e uma grande perda de tempo.

Segundo informações da empresa onde o estágio foi efetuado, para iniciar o processo de atualização é necessário um levantamento das diferenças na estrutura de modelo de dados entre a versão que está no cliente e, a nova versão a ser atualizada. De posse das diferenças detectadas, deve-se proceder as alterações na estrutura do modelo de dados para que as novas telas, relatórios e processos internos com suas devidas alterações possam surtir efeito positivo ao usuário.

Se o software for de pequeno porte com reduzido número de entidades, telas, relatórios e processos internos no modelo de dados, o trabalho de atualização é rápido. Mas a empresa desenvolvedora possui um software de gestão empresarial, com muitos processos, desde a alta administração da empresa até o chão de fábrica, com aproximadamente 350 entidades e mais de 2500 componentes (entre telas, relatórios e processos internos). Neste caso, a atualização do software é problemática, lenta, trabalhosa e com alto custo.

Segundo Pressman (1995), realisticamente sabe-se que uma massa enorme de problemas e custos adicionais esconde-se sob a superfície em um processo de atualização. A natureza ambígua da mudança é subjacente a todo trabalho de software. A mudança é inevitável quando se constroem sistemas baseados em computador; portanto, deve-se

desenvolver mecanismos para avaliar, controlar e fazer modificações. Dentre estes mecanismos o desenvolvimento estruturado é uma alternativa.

Segundo Keller (1990) o desenvolvimento estruturado de sistemas é uma disciplina que produz uma especificação de sistema concisa, não-ambígua, não-redundante e rigorosa, usando entre outras ferramentas os diagramas de fluxo de dados (D.F.D.). A especificação baseada em D.F.D. é apoiada por uma breve narrativa ainda que estruturada em português, por uma descrição do banco de dados lógico e por um dicionário de dados completo para o projeto. Esta especificação estruturada das necessidades do usuário é convertida em diagramas de estrutura de módulos durante o projeto *(design)* que por sua vez são transformados em programas estruturados durante a implementação. Mas para isso, os diagramas devem ser convertidos para o modelo entidade relacionamento lógico e físico onde a partir desse ponto possam ser criados num banco de dados relacional com o intuito de permitir que as informações reais possam ser armazenadas e utilizadas.

Segundo Kern (1994) um banco de dados é um acervo de informações armazenadas segundo certos critérios de organização. O modelo de dados relacional é uma maneira de conceber a organização de um banco de dados como uma coleção de tabelas e associações entre tabelas.

Cerícola (1995) coloca que essa estruturação fornece fundamentos para um futuro desenvolvimento de aplicação. Quando o desenvolvimento acontecer, a análise de necessidades deve ser conduzida sem preocupação com quaisquer restrições que não sejam aquelas relativas à maneira pela qual a organização faz seus negócios. A partir do momento em que todas as informações foram levantadas, pode-se efetuar a criação da estrutura de um banco de dados relacional e o desenvolvimento da aplicação propriamente dita.

Para o desenvolvimento do protótipo, utilizou-se uma ferramenta de desenvolvimento que permita fazer modelagem de dados utilizando a metodologia de análise estruturada onde, baseado no modelo criado será possível a construção dos componentes do protótipo. Como foi exposto por Compuware (1996), o UNIFACE é um ambiente de desenvolvimento para aplicações corporativas complexas e de negócios críticos. Ele permite todo o desenvolvimento de um projeto de sistemas, desde a modelagem de dados até a construção dos componentes do aplicativo. As linhas de código dos componentes são criadas baseadas em orientação a

eventos. O UNIFACE possui uma grande manutenabilidade e reusabilidade, pois tudo é feita diretamente na modelagem de dados que são aplicadas aos componentes através de herança. Isto reflete em qualidade, consistência e produtividade.

Numa publicação recente de (Sysart, 1998) é colocado que o UNIFACE está disponível para uma série de sistemas operacionais (Macintosh, Motif, OS/2, Windows, etc), caracterizando execução multi-plataforma, implementando apresentação gráfica ou em modo caractere. Possui dispositivos nativos e específicos para os mais importantes bancos de dados e sistemas de arquivos com acessos via ODBC (conforme Poffo (1996) significa conectividade aberta de banco de dados).

Sabendo que o UNIFACE possui como característica o desenvolvimento utilizando banco de dados relacional, o protótipo foi desenvolvido utilizando esta ferramenta na criação dos componentes, juntamente com a metodologia de análise estruturada para efetuar a criação do modelo de dados.

1.1 OBJETIVOS

O objetivo principal foi desenvolver um protótipo de software que automatize o processo de atualização de versão do software no cliente. Para isso, deverá ser criado um processo de fechamento de versão executado na empresa desenvolvedora do software. Esse processo será responsável em criar toda a estrutura de informações da versão a ser atualizada no cliente. Será através desta estrutura criada pelo fechamento de versão que o processo de atualização de versão será executado. Será esse o processo que executará todas as alterações na estrutura do modelo de dados, bem como a atualização das telas, relatórios e processos internos.

a) O objetivo específico do trabalho é desenvolver mecanismos para automatizar os processos de fechamento e atualização de versão. Eles deverão permitir que estes processos sejam executados por qualquer funcionário de ambas as empresas (empresa desenvolvedora e cliente). O processo de fechamento da versão deve ser executado para que as alterações em curso do(s) sistema(s) não “contaminem” a versão que será implantada no cliente. Ele implica em criar uma cópia de telas, relatórios e processos internos bem como fazer um levantamento da estrutura do modelo de dados do(s) sistema(s) que está(ão) sendo fechado(s). O processo de

atualização de versão que será executada no cliente terá como base os arquivos gerados no fechamento de versão. Ele efetuará a cópia das telas, relatórios e processos internos da versão fechada para o cliente bem como irá alterar o modelo de dados conforme o modelo de dados da versão fechada.

1.2 ESTRUTURA

Este trabalho está disposto em capítulos descritos a seguir:

O primeiro capítulo introduz o assunto correspondente ao trabalho, seus objetivos e como está disposta a estrutura do trabalho.

O segundo capítulo descreve a fundamentação teórica necessária para o desenvolvimento do trabalho bem como outras tecnologias utilizadas.

O terceiro capítulo apresenta como foi o desenvolvimento do trabalho. Mostra a análise e especificação de requisitos, especificação do problema utilizando diagramas, ferramentas utilizadas para o desenvolvimento do protótipo bem como a operacionalidade da implementação.

No quarto capítulo apresentam-se as principais conclusões obtidas com o desenvolvimento deste trabalho, suas limitações e sugestões para novas funções do protótipo.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 FERRAMENTA DE DESENVOLVIMENTO UNIFACE 7.0

2.1.1 HISTÓRIA

A ferramenta de desenvolvimento Uniface é um dos produtos da Compuware S.A. que é uma empresa com sede na Holanda e considerada como a quinta maior empresa de software do mundo, ficando atrás apenas da Microsoft, Oracle, Computer Associates e SAP. Atuando no mercado há mais de 44 anos, o seu faturamento anual é superior a dois bilhões de dólares. Conforme Compuware (2000), ela possui mais de 15 mil funcionários, mais de 120 escritórios distribuídos em 46 países, está em 29^{o} lugar entre as empresas de crescimento mais rápido, segundo a revista Fortune de janeiro de 2000. Atualmente, a Compuware S.A. oferece mais de 130 produtos/serviços para mais de 14 mil corporações através do mundo, e dentre os clientes no Brasil estão Bradesco, Correios e Localiza National.

2.1.2 DEFINIÇÕES E CARACTERÍSTICAS

O Uniface é um ambiente de desenvolvimento de aplicações para criar aplicações complexas e de missão crítica, conforme Compuware (1996). Mais especificamente é uma linguagem de quarta geração (4GL) – ou seja, uma linguagem com alto nível de automação para desenvolvimento de aplicações para banco de dados, e que consiste de um ambiente de desenvolvimento e outro de execução. Para acessar os ambientes é necessário utilizar os executáveis IDF.EXE e UNIFACE.EXE respectivamente, explica Compuware (1995).

Compuware (1996) explica os princípios fundamentais e as capacidades de Uniface examinando as dimensões críticas requeridas para desenvolvimento de aplicação estratégica: desenvolvimento e ambiente.

Desenvolvimento: o desenvolvimento dentro do Uniface é empresarial modelo-dirigido. Desenvolvimento modelo-dirigido começa com a especificação de um modelo de aplicação que captura as estruturas, comportamentos, e aspectos de apresentação de uma aplicação. O modelo se torna o “coração” da aplicação e permite que a mesma absorva mudanças em estrutura e lógica. O ambiente de desenvolvimento do modelo provê um nível alto de abstração de considerações de tecnologia e permite para o desenvolvedor gastar a

maioria do esforço no modelo, não escrevendo código. O Uniface é um ambiente de desenvolvimento componente-baseado. Os componentes de Uniface incluem formulários, relatórios e serviços (processos internos). As definições no modelo são reutilizáveis. Para criar uma aplicação que usa as definições no modelo, os desenvolvedores constroem componentes de aplicação que herdam as definições do modelo reutilizando um trabalho que já foi efetuado. Atualizações à nível de modelo são refletidas à nível de componente porque as definições estão unidas. Essa reusabilidade promove qualidade, consistência, e produtividade nas aplicações de Uniface. Podemos visualizar na figura 1 as ligações entre os modelos e objetos que utilizaram estes modelos. Na figura 2 um exemplo de herança e reusabilidade e na figura 3 é mostrado o ambiente de desenvolvimento.

Figura 1 – Ligações entre modelos e objetos que utilizaram esses modelos.

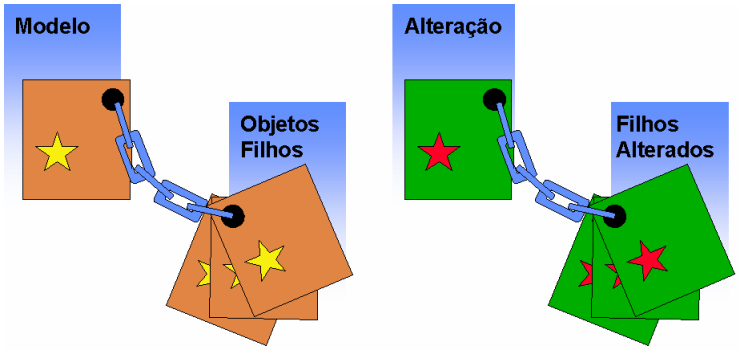


Figura 2 – Herança e reusabilidade.

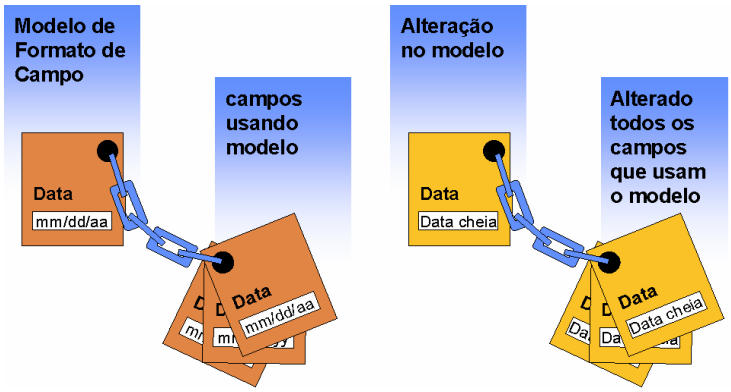
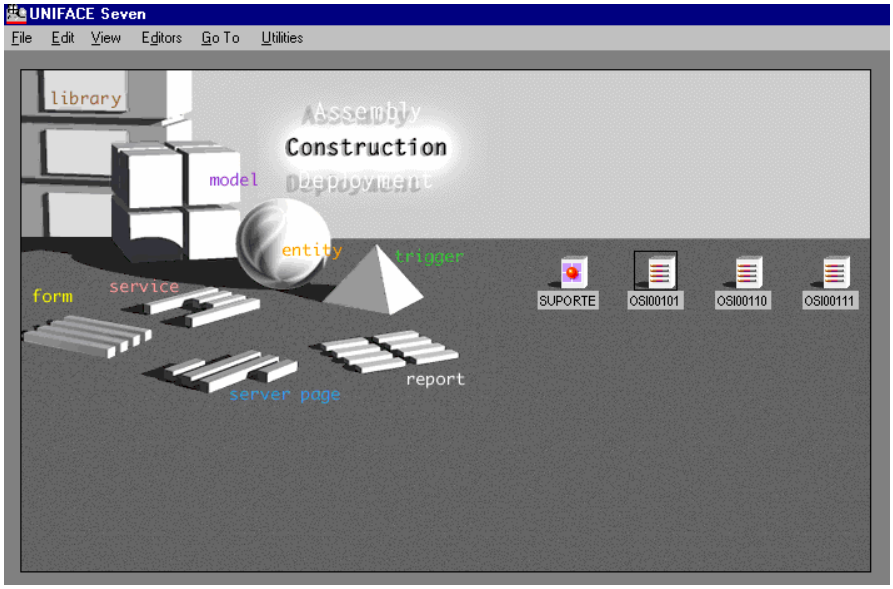


Figura 3 – Ambiente de desenvolvimento.



Ambiente: as aplicações de Uniface são de infraestrutura independentes de ambiente. Isto é, durante o desenvolvimento, o desenvolvedor não precisa considerar o ambiente onde a aplicação será executada. As aplicações de Uniface resultantes podem ser divididas à nível de componente em qualquer arquitetura desejada. Podemos ver na figura 4 que a infra-estrutura-independência de Uniface estende por plataformas múltiplas incluindo hardware, sistema operacionais, rede e banco de dados. Tendo projetado os componentes durante a fase de desenvolvimento, o ambiente final da aplicação é a junção dos componentes no modo exigido e os distribuindo para o lugar mais efetivo na infra-estrutura. Na figura 5 é mostrado um exemplo de ambiente final de uma aplicação.

Figura 4 – Infra-estrutura-independência.

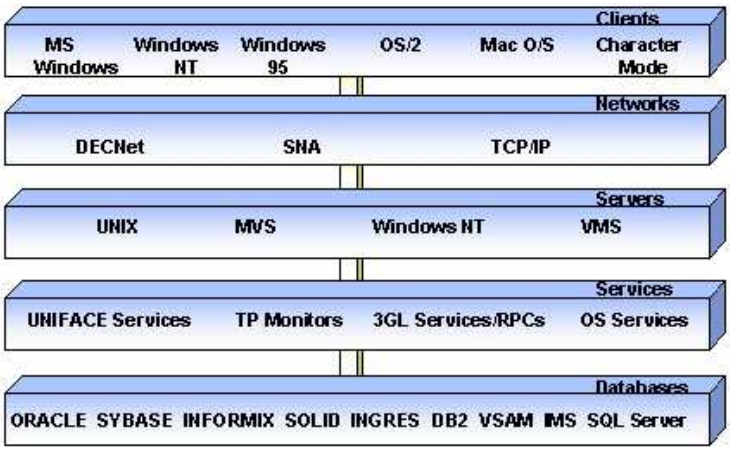
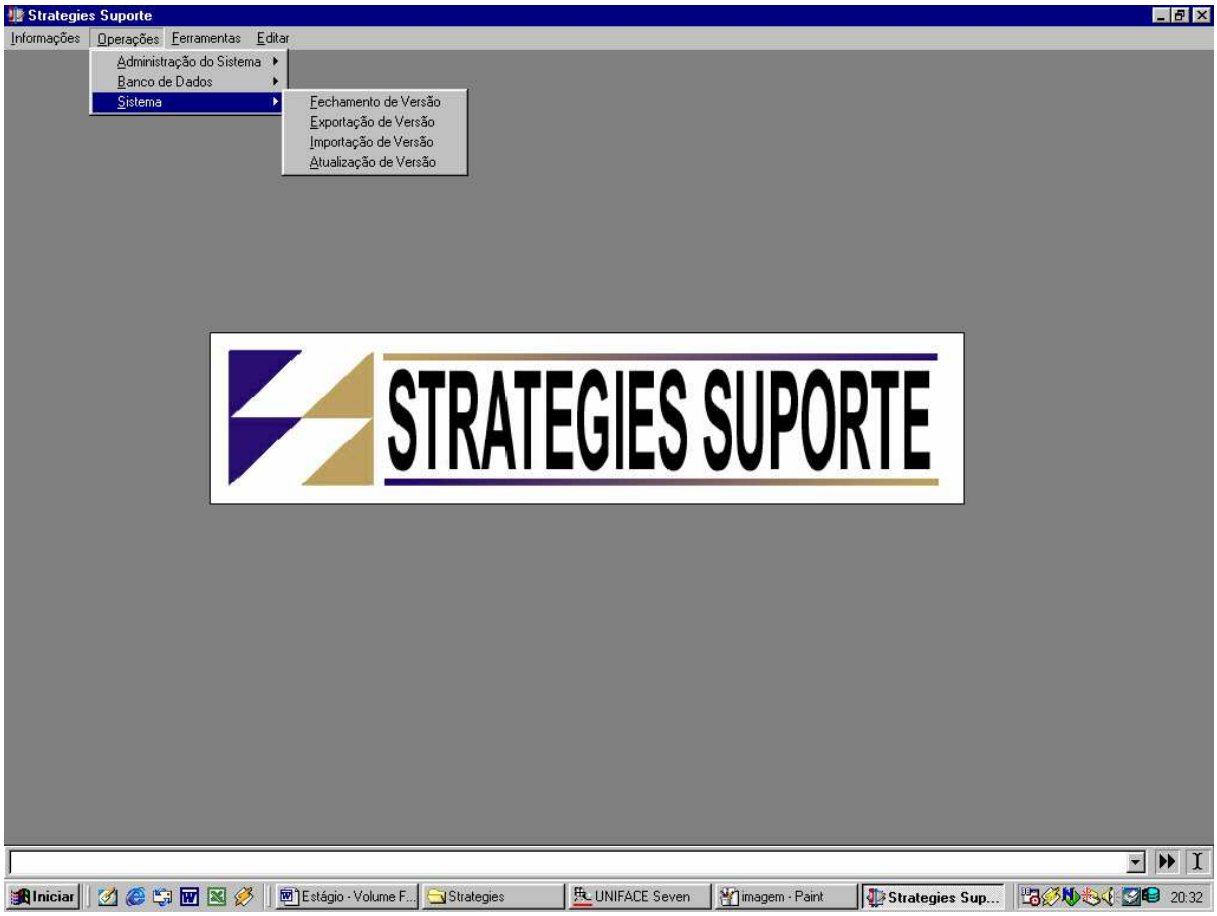


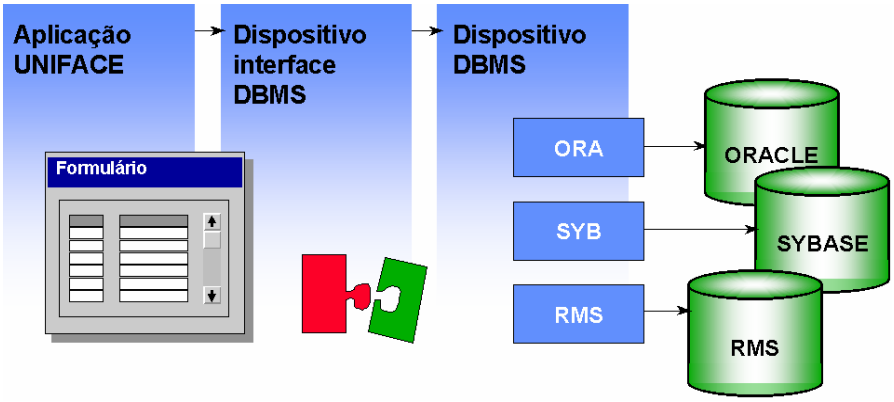
Figura 5 – Ambiente final de uma aplicação.



Compuware (1995) coloca algumas características do Uniface são:

- Independência de banco de dados; Oracle, Sybase, DB2, Informix, SQL Server, Ingres, VSAM, IMS e Solid são os principais bancos de dados para os quais o Uniface tem dispositivo de acesso. O DBMS Driver Interface define todas as características para chamadas ao banco de dados como *logon*, *open*, *select*, *update*, *fetch*, e assim por diante Compuware (1997). Trabalhando em conjunto com o banco de dados garante a integridade das informações. A figura 6 mostra como é feito o acesso aos dados, sendo que a aplicação requisita o acesso ao dispositivo de interface adequado, que encontra o dispositivo do banco de dados desejado, e este efetua a requisição no formato que o banco de dados compreende.

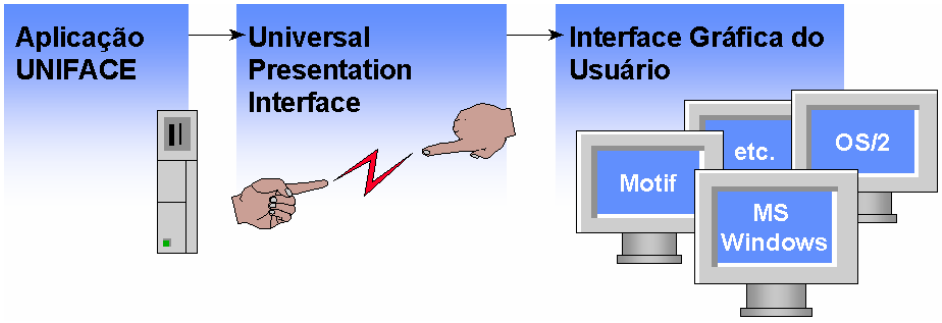
Figura 6 – Acesso ao banco de dados.



- Independência de sistema operacional na máquina servidor, sendo portável para vários sistemas operacionais como novell, windows nt, unix, linux, vms, vax e outros.

- Independência de sistema operacional na máquina cliente, sendo portável para vários sistemas operacionais como windows 95, windows 98, windows 2000, windows nt, linux, mac O/S, OS/2, motif, modo caracter (DOS) e outros. A figura 7 demonstra a conversão para o ambiente do usuário, onde a aplicação do Uniface é mostrada na tela do usuário pela Universal Presentation Interface, que se encarrega de converter as informações no formato do tipo de GUI sendo usado; para isto, há um dispositivo definido para cada tipo de GUI.

Figura 7 – Conversão da aplicação para a interface do usuário.



- Implementação rápida por possuir uma orientação visual.

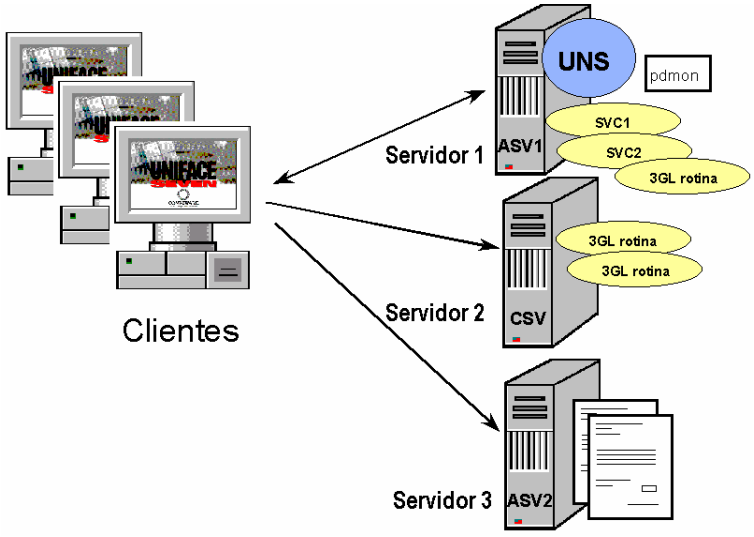
- Permite desenvolver e distribuir aplicações para a internet.

- Ampla utilização de modelos para agilizar o trabalho, modelos estes podem ser de formulário, entidade, campo, neste último caso podendo ser em relação ao armazenamento, à sintaxe ou ao layout de apresentação.

- Permite que várias tabelas sejam distribuídas entre vários servidores, com bancos de dados diferentes, e sistemas operacionais diferentes, sem que seja necessário alterar os programas, apenas efetuando-se alterações nos arquivos de configuração.

- Permite executar tarefas pesadas em qualquer servidor da rede, através de parametrização. Desta forma, relatórios pesados e rotinas de processamento em grupo (batch) podem ser direcionados para a CPU de um servidor dedicado a esta tarefa. A figura 8 mostra que uma rotina pode ser configurada para rodar usando processamento em qualquer um dos três servidores disponíveis.

Figura 8 – Execução em qualquer servidor.



Segundo Sysart (1998) o desenvolvimento no Uniface é orientado/dirigido pelo seu modelo de dados e regras de negócio, ou seja, começa com a especificação de um modelo de aplicação com suas estruturas, características, comportamentos e aspectos de apresentação da aplicação. Em Compuware (1996) é colocado que após criar os modelos lógicos e derivá-los para obter o modelo físico de dados, este modelo físico é introduzido no Uniface e vai reger o desenvolvimento. Na figura 9 vemos que os componentes são baseados no modelo, e na figura 10 como definir uma entidade, com suas propriedades, campos, chaves, relacionamentos e regras de negócio que podem ser inseridas diretamente no modelo.

Figura 9 – Definição do modelo de dados e regras de negócio.

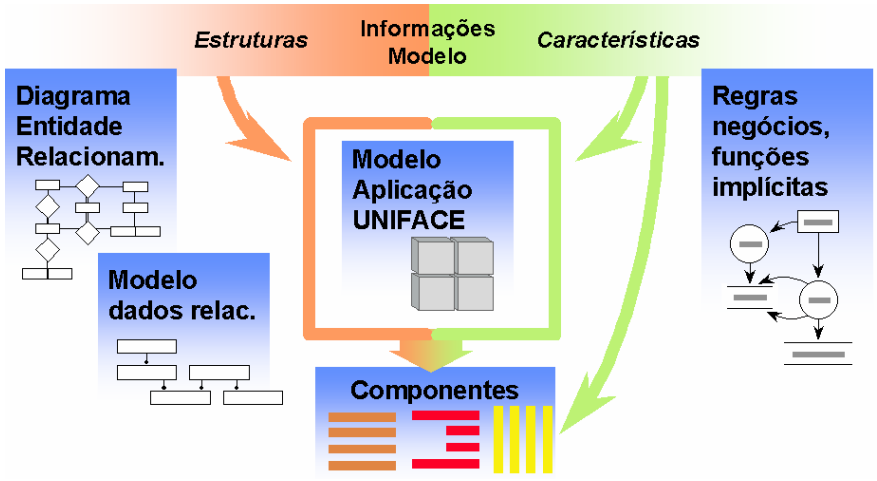


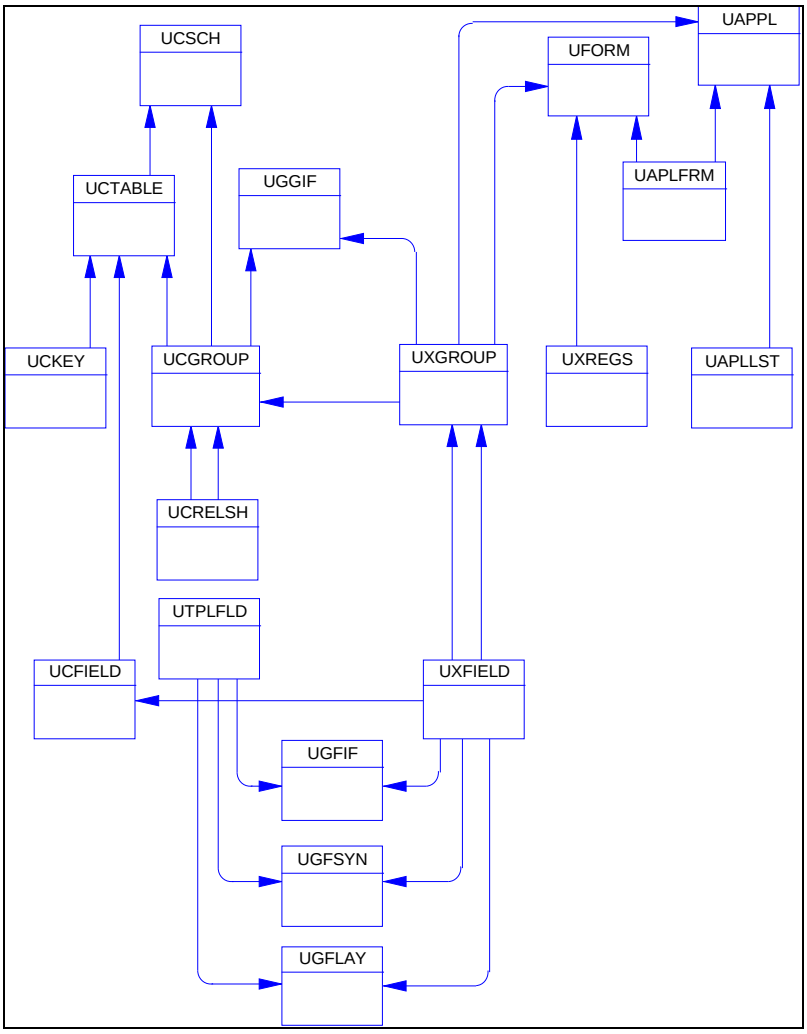
Figura 10 – Criação de entidades, chaves, propriedades e relacionamentos.

O trabalho do Uniface dentro do ciclo de vida do sistema começa após a análise e modelagem dos dados, quando já se tem a definição das estruturas físicas das tabelas normalizadas. Então, o modelo físico pode ser introduzido no Uniface de forma manual, pode ser importado diretamente do banco de dados que contenha estas tabelas criadas, ou importado da ferramenta de análise baseada em UML chamada Rational Rose com o qual possui interface.

Segundo Compuware (1996) o Uniface grava as definições de todos os objetos da aplicação, incluindo o modelo de dados da aplicação, em um banco de dados relacional conhecido como *Repositório.* Todos os registros no repositório são definidos uma vez e usadas várias vezes. Desenvolvimento baseado em repositório promove reusabilidade de definições e as heranças das propriedades entre objetos pais e filhos. Este repositório possui entidades que armazenam informações de campos, subtipos, chaves e relacionamentos de

entidades. Armazenam também informações dos componentes como entidades, variáveis, layout e sintaxe dos componentes das aplicações. Na Figura 11 podemos visualizar o modelo entidade relacionamento físico do *Repositório*.

Figura 11 – Repositório do UNIFACE.



No quadro 1 pode-se visualizar algumas entidades do repositório UNIFACE bem como a sua função e conteúdo armazenado.

2.1.2.1.1.1.1.1 QUADRO 1 – ENTIDADES E FUNÇÕES DO REPOSITÓRIO UNIFACE.

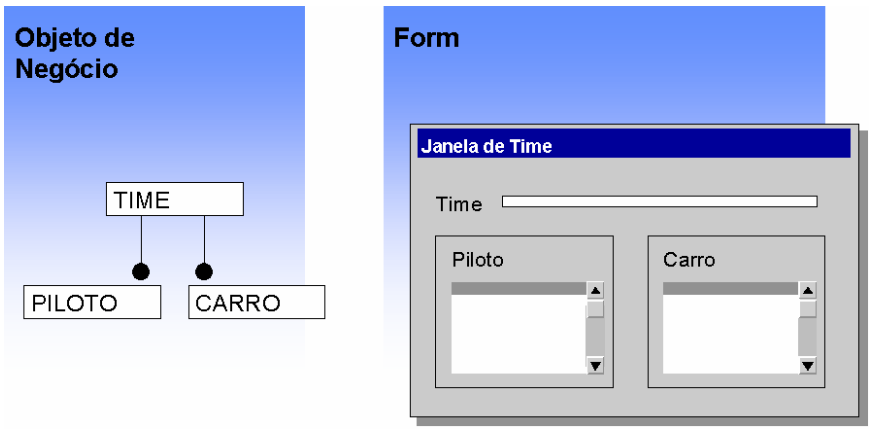
Nome	Definição
UCFIELD	Definições de campos de entidades.

UCGROUP	Definições de subtipos de entidades.
UCKEY	Definições de chaves de entidades.
UCRELSH	Definições de relacionamentos entre subtipos de entidades.
UCSCH	Definições de modelos de aplicações.
UCTABLE	Definições de entidades.
UFORM	Definições de modelos de componentes e componentes.
UXFIELD	Definições de campos de componentes.
UXGROUP	Definições de entidades de componentes.
UXREGS	Definições de variáveis de componentes.
UAPLFRM	Lista dos componentes de inicialização.
UAPPL	Definições dos componentes de inicialização.
UGFIF	Modelos de interface de campos.
UGFLAY	Modelos de layout de campos.
UGFSYN	Modelos de sintaxe de campos.
UGGIF	Modelos de interface de entidades.
UTPLFLD	Modelos de campos.
UAPLLST	Lista de distribuição de aplicação.

Segundo Compuware (1996) a aplicação Uniface é formada por partes chamadas de componentes que fazem o processamento e representam os objetos de negócios. Componentes são divididos em forms (formulários), services (serviços) e reports (relatórios). O Uniface possui um ambiente de editoração para cada componente. Todos os componentes podem conter *operations* (operações) que são similares à métodos e podem ser chamados (ativados) por outros componentes.

Forms: são as telas que fazem a interação com o usuário e seu visual é muito importante. Permitem que o usuário trabalhe com os dados do banco de dados, inserindo, alterando e excluindo. Permitem acessar objetos globais como variáveis e imagens. Podem conter *operations*, mas o escopo dessas operações é local. São executados no cliente, isto é, no computador do usuário. No momento de sua compilação é gerado um arquivo físico com o nome do componente tendo como extensão FRM. Podemos visualizar um exemplo de form na figura 12 que contém objeto de negócio e o form propriamente dito.

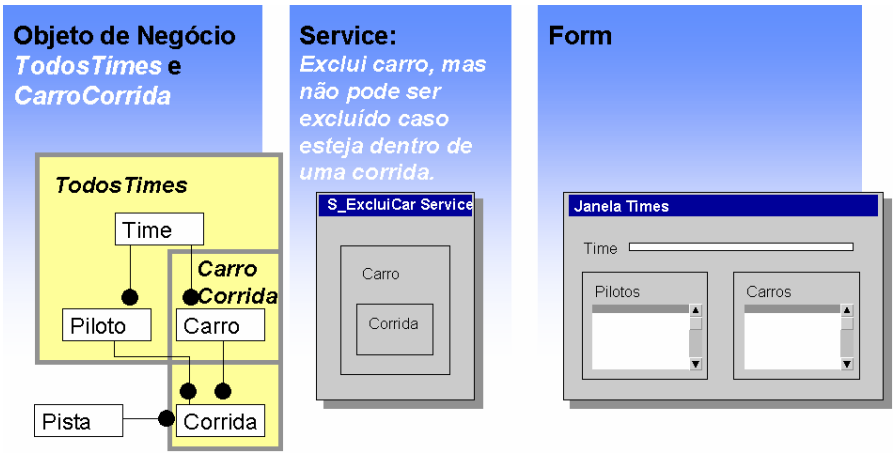
Figura 12 – Exemplo de Form.



Services: são componentes designados para gerenciar os processos de uma aplicação em Uniface. Eles usualmente agrupam todas as entidades necessárias para suportar os processos de um objeto de negócios. Não possuem interação com o usuário, portanto o seu visual não é importante. Porém, possuem uma estrutura de dados e podem efetuar alterações na base de dados. Essas estruturas de dados podem utilizar-se dos relacionamentos entre as entidades para efetuar seus processos. Contém uma ou mais *operations* que podem ser chamados por qualquer componente da aplicação. No momento de sua compilação é gerado

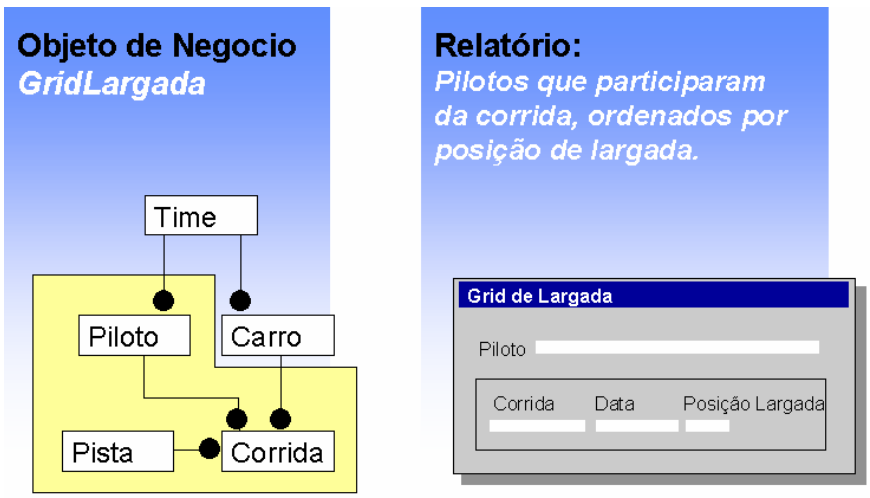
um arquivo físico com o nome do componente tendo como extensão SVC. Na figura 13 podemos visualizar um *service.*

Figura 13 – Exemplo de Service.



Reports: são objetos da aplicação que possuem o propósito de impressão. Não possuem interação com o usuário. Porém, o visual determina a aparência da impressão do relatório. Como os *services*, os *reports* possuem estruturas de dados que podem utilizar-se dos relacionamentos entre as entidades para efetuar sua impressão. Podem Conter uma ou mais *operations* que podem ser chamados por qualquer componente da aplicação. No momento de sua compilação é gerado um arquivo físico com o nome do componente tendo como extensão RPT. Na figura 14 podemos visualizar um *report.*

Figura 14 – Exemplo de Report.



Dynamic Object Library (DOL): Compuware (1996) coloca que o UNIFACE compila os objetos globais em um arquivo binário, somente leitura, com alta-performance, independência-de-plataforma conhecido como *Dynamic Object Library (DOL).* O mecanismo DOL facilita a instalação de aplicações em diferentes sistemas operacionais. A aplicação UNIFACE vai buscar os objetos no momento da inicialização da aplicação. Não vai buscar todos os objetos globais de uma vez, e sim, conforme vai necessitando. Uma grande vantagem disso é que os objetos globais podem ser alterados sem modificar o resultado na aplicação UNIFACE. Só é alterada na aplicação UNIFACE quando for gerado outro arquivo DOL. Para que essa independência aconteça, é necessário informar no arquivo de parâmetros da aplicação UNIFACE que a DOL está em arquivo, e não no repositório. Neste segundo caso, a aplicação já é afetada caso houve alterações em algum objeto global. É nesse arquivo de parâmetros da aplicação que é informada onde o arquivo se encontra e qual o seu nome (geralmente é *uobj.dol*).

Uniface Runtime Repository (URR): Compuware (1996) coloca que o UNIFACE compila todas as ligações entre os componentes em um arquivo binário, somente leitura, conhecido como *Uniface Runtime Repository (URR).* Essas ligações são feitas sempre que um comando *activate* é utilizado em qualquer componente UNIFACE (inclusive menu's). Esse arquivo é consultado no momento em que a aplicação UNIFACE é executada. Uma grande vantagem disso é as ligações podem ser alterados sem modificar o resultado na aplicação UNIFACE. Só é alterada na aplicação UNIFACE quando for gerado outro arquivo URR. Para que essa independência aconteça, é necessário informar no arquivo de parâmetros da aplicação UNIFACE que a URR está em arquivo, e não no repositório. Neste segundo caso, a aplicação já é afetada caso houve alterações em alguma ligação. É nesse arquivo de parâmetros da aplicação que é informada onde o arquivo se encontra e qual o seu nome (geralmente é *udesc.urr*).

Segundo Compuware (1996) a aplicação Uniface é armazenada em usuários de banco de dados relacional (objeto, uniface e dados), separados por tipo de componente da aplicação.

Usuário objeto: neste usuário são armazenados os componentes globais da aplicação. Exemplo: variáveis globais, menus da aplicação, mensagens de tela, imagens, processos, etc.

Usuário uniface: neste usuário são armazenadas todas as informações de todos os componentes da aplicação. Exemplo: Nome do componente, objetos do componente, linhas de código do componente, etc.

Usuário dados: neste usuário são armazenadas todas as informações que a aplicação irá gerar. Exemplo: Cadastro de sistema, Cadastro de usuários, Cadastro de acessos, etc.

Segundo Compuware (1995), o ambiente da aplicação e o ambiente de desenvolvimento necessitam de um arquivo de parâmetros para serem executados. Neste arquivo são especificados vários parâmetros que serão utilizados na execução. No quadro 2 pode-se visualizar um exemplo de arquivo ASN.

2.1.2.1.1.1.1.2 QUADRO 2 – EXEMPLO DE ARQUIVO ASN.

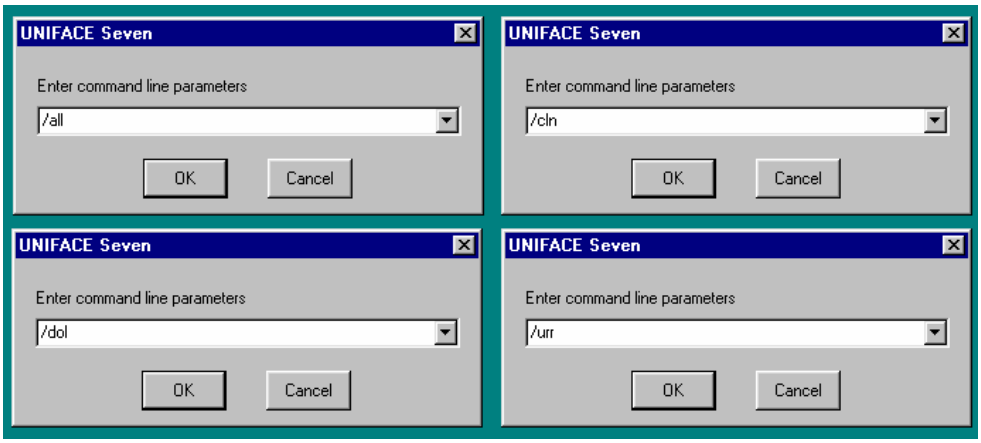
```
[SETTINGS]
; Linguagem utilizada
$LANGUAGE=BRZ ; Brasileira
; Arquivo de dados de teclado
$KEYBOARD=SUPORTE_MSWINX
; Busca de objetos globais
$SEARCH_OBJECT=DBMS_FIRST ; busca primeiro do banco de dados
; Busca de assinaturas
$SEARCH_DESCRIPTOR=DBMS_FIRST ; busca primeiro do banco de dados
; Parar quando der erro de divisão por zero?
$STOP_ON_DIV_BY_ZERO=FALSE ; falso
; Caracter de divisão decimal.
$NUMERIC_LAYOUT_CHAR=,
; Aplicação em modo de testes.
$TESTMODE_COMPONENTS
; Geração de log no arquivo especificado.
$PUTMESS_LOGFILE   c:\suporte.log

[FILES]
usys:idf.aps        usys:..\ud\idf.aps
usys:ur*.svc        usys:..\ud\ur*.svc
usys:uu*.svc        usys:..\ud\uu*.svc
usys:uu*.frm        usys:..\ud\uu*.frm
usys:uv*.frm        usys:..\ud\uv*.frm
usys:idf*.frm       usys:..\ud\idf*.frm
usys:pv*.frm        usys:..\ud\pv*.frm
usys:usys*.frm      usys:usys*.frm
usys:*.dis          usys:*.dis
usys:*.dsc          usys:*.dsc
usys:*.trx          usys:..\trx\*.trx
usys:udbg.aps       usys:udbg.aps
udbg:udbg*.frm      usys:udbg*.frm
gpm.aps             usys:..\usys\gpm.aps
; Localização dos arquivos de dados globais e assinaturas
usys:uobj.dol       c:\strategi\suporte\tools\uobj.dol
usys:udesc.urr      c:\strategi\suporte\tools\udesc.urr
```

```
; Localização dos componentes do tipo serviço
*.svc               c:\strategi\suporte\element\*.svc
; Localização dos componentes do tipo relatório
*.rpt               c:\strategi\suporte\element\*.rpt
; Localização dos componentes do tipo formulário
*.frm               c:\strategi\suporte\element\*.frm
*.aps               c:\strategi\suporte\element\*.aps
*.pro               c:\strategi\suporte\pro\*.pro
*.p*                c:\strategi\suporte\printer\*.p*
*.hlp               c:\strategi\suporte\help\*.hlp
*.dbf               c:\strategi\suporte\database\*.dbf

; Nome dos usuários que o uniface deve conectar para buscar
; os usuários objeto, uniface e dados.
[PATHS]
$UNIFACE  ORA:|UNIFACE_SUPORTE|UNIFACE_SUPORTE
$OBJETO   ORA:|OBJETO_SUPORTE|OBJETO_SUPORTE
$ORACLE   ORA:|DADOS_SUPORTE|DADOS_SUPORTE
$TEMP     ORA:?|?|?

$IDF=$UNIFACE
$SYS=$UNIFACE
$UUU=$OBJETO
$ORA=$ORACLE
$UD0=$TEMP

; Permite que seja definido qual usuário algumas tabelas devem
; ser armazenadas.
[ENTITIES]
*.sysenv                    usys:*.*
tbentidade.suporte          $ORACLE:tbentidade
tbcampo.suporte             $ORACLE:tbcampo
tbrelacionamento.suporte    $ORACLE:tbrelacionamento
tbalteracao.suporte         $ORACLE:tbalteracao
tab.oracle                  $TEMP:tab
user_tab_columns.oracle     $TEMP:user_tab_columns
```

Segundo Compuware (1995), o Uniface permite através de comandos digitados diretamente na linha de comando que aparece ao iniciar o ambiente de desenvolvimento, efetuar tarefas como compilar todos os componentes (/all), efetuar limpeza de todos os componentes (/cln), gerar arquivo de dados globais (/dol) e gerar arquivo de assinaturas (/urr).

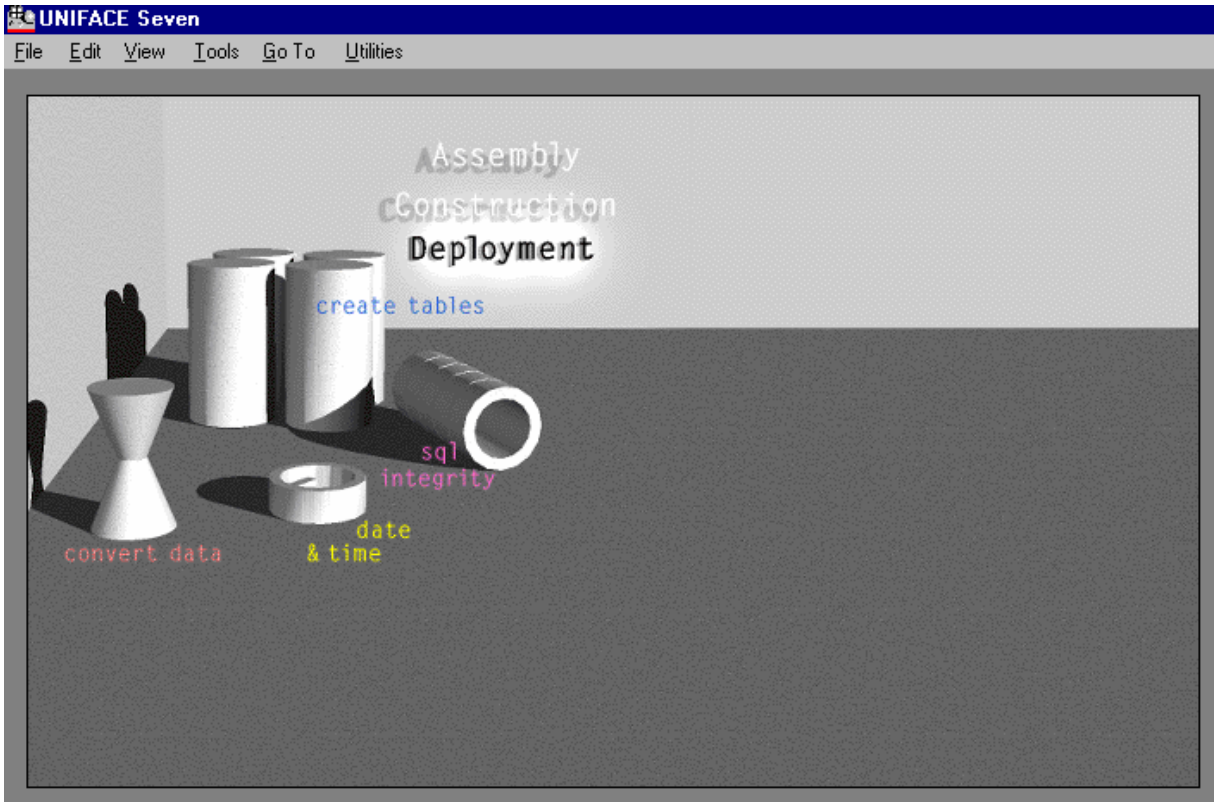
Na figura 15 pode-se visualizar os comandos para efetuar tais tarefas.

Figura 15 – Exemplos de comandos do uniface.



Compuware (1995) também explica que o Uniface permite criar *scripts* que podem ser executados dentro do usuário de dados para criar entidades e relacionamentos, isto é, criar o modelo de dados dentro de um usuário de banco de dados (usuário dados) a partir de uma estrutura que foi criada dentro do uniface. Na figura 16 pode-se visualizar a opção que deve ser utilizada para efetuar a criação destes *scripts.*

Figura 16 – Ambiente para criação de *scripts* de estrutura de modelos de dados.



2.2 OUTRAS TECNOLOGIAS UTILIZADAS

2.2.1 BANCO DE DADOS ORACLE 8I

Segundo Hursch (1991), Oracle é um sistema de gerenciamento de banco de dados relacional (RDBMS – relational database management system) produzido pela Oracle Corporation de Belmont, Califórnia. Abrangem uma grande faixa de sistemas operacionais e hardware.

Segundo Cerícola (1995) o Oracle é um sistema gerenciador de banco de dados relacional com funções completas baseadas no ANSI SQL3. São seus componentes: gerador de aplicações, dicionário de dados, gerador de relatórios, planilhas eletrônicas, pré-compiladores para linguagens hospedeiras, utilitários, gráficos, cálculos estatísticos, armazenagem de dados de vários tipos, como data, números inteiros, valores numéricos com ou sem decimais, ponto flutuante, caracteres variáveis ou fixos, binários, fotografias, textos, som, imagem, etc.

2.2.2 LINGUAGEM MANIPULAÇÃO DE DADOS SQL PLUS 3.0

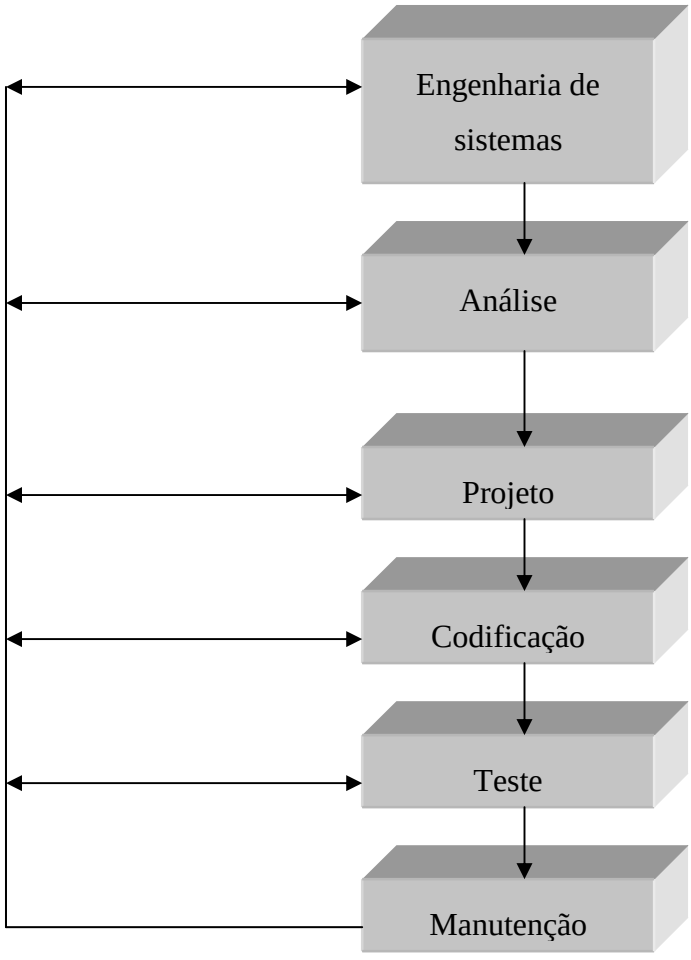
Segundo Fernandes (2000), a SQL (structured query language) é uma linguagem para interface com bancos de dados relacionais, isto é, todos os usuários e programas que desejam realizar alguma tarefa no banco de dados devem fornecer comandos escritos nesta linguagem.

Segundo Saraiva (1999) a SQL é uma linguagem baseada no inglês e usa palavras como select, insert, e delete (selecionar, inserir e excluir) como parte de seu conjunto de comandos. É uma linguagem não-procedural, você especifica qual informação você quer e não como traze-la. Em outras palavras, você não especifica qual vai ser o método de acesso aos dados. Todos os comandos SQL utilizam o “optimizer” que determina a maneira mais rápida de recuperar os dados.

3 DESENVOLVIMENTO DO TRABALHO

Segundo Pressman (1995) a figura 17 ilustra o paradigma do ciclo de vida clássico da engenharia de software, Ás vezes chamado de *modelo cascata*, o paradigma do ciclo de vida requer uma abordagem sistemática, seqüencial ao desenvolvimento do software, que se inicia no nível do sistema e avança ao longo da análise, projeto, codificação, teste e manutenção. Modelado em função do ciclo da engenharia convencional, o paradigma do ciclo de vida abrange as seguintes atividades:

Figura 17 – Modelo de ciclo de vida básico conhecido como cascata.



Com base neste ciclo de vida, o desenvolvimento do protótipo teve as seguintes fases:

a) Levantamento dos processos. (fechamento de versão e atualização de versão)

b) Descrição dos objetivos do protótipo.

c) Lista de eventos.

d) Diagrama de contexto.

e) Diagramas entidade relacionamento (lógico e físico).

f) Diagrama de fluxo de dados (D.F.D.).

g) Implementação do protótipo utilizando Uniface 7.0.

h) Instalação e testes do protótipo.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Segundo Pressman (1995) a análise de requisitos é uma tarefa da engenharia de software que efetua a ligação entre a alocação de software em nível de sistema e o projeto de software. A análise de requisitos possibilita que o engenheiro de sistemas especifique a função e o desempenho do software, indique a interface do software com outros elementos do sistema e estabeleça quais são as restrições de projeto que o software deve enfrentar. A análise de requisitos permite que o engenheiro de software aprimore a alocação de software e construa modelos do processo, dos dados e dos domínios comportamentais que serão tratados pelo software. A análise de requisitos proporciona ao projetista de software uma representação da informação e da função que pode ser traduzida em projeto procedimental, arquitetônico e de dados. Finalmente, a especificação de requisitos proporciona ao cliente os critérios para avaliar a qualidade logo que o software for construído e ao desenvolvedor facilitará o processo de prototipação da automatização da versão.

O protótipo será construído utilizando a ferramenta de desenvolvimento de aplicação UNIFACE na versão 7.0. Sendo o objetivo do protótipo a automatização do processo de atualização de versão no cliente, ele deverá atender (automatizar) todas as atividades que fazem parte de uma atualização de versão. Sem esquecer que para isso o processo de fechamento de versão também deverá ser atendido, pois a segunda etapa depende única e exclusivamente da primeira. Os detalhes das etapas serão descritos no item especificação.

Suas principais características deverão ser a automatização completa de todos os processos de cada etapa (fechamento e atualização), isto é, quanto mais automatizado os

processos “rodarem” (sem a intervenção do usuário) melhor será a qualidade final de cada processo.

3.2 ESPECIFICAÇÃO

A especificação do protótipo foi desenvolvida com base nas informações levantadas junto ao usuário do departamento de suporte da empresa onde o estágio foi efetuado. Das informações levantadas, iniciou-se o desenvolvimento das etapas de criação do diagrama de contexto, do diagrama de fluxo de dados, do modelo entidade relacionamento lógico e do modelo entidade relacionamento físico e para isso, foi utilizado a ferramenta case Power Designer.

3.2.1 ETAPA 1 – LEVANTAMENTO DOS PROCESSOS (FECHAMENTO E ATUALIZAÇÃO DE VERSÃO)

3.2.1.1 FECHAMENTO DE VERSÃO

O processo de fechamento de versão é executado na empresa desenvolvedora e é efetuado sempre que um conjunto de alterações foi efetuado em qualquer sistema que a empresa desenvolvedora possui. Esse conjunto de alterações é definido entre empresa desenvolvedora e seus clientes e ocorrem devido a alterações na administração nos clientes e idéias novas que vão sendo criadas a partir das parcerias. Segundo a empresa onde o estágio foi efetuado existe uma periodicidade de fechamento de nova versão de dois meses. As etapas no processo de fechamento de versão são os seguintes:

a) Efetuar a verificação de espaço disponível do servidor (geralmente deve ter 200 MBytes livres) para criar nova versão fechada do sistema.

    a. Caso o servidor não possuir esse espaço disponível, deve-se efetuar a escolha de uma versão (deve ser uma versão antiga onde nenhum cliente está utilizando) para ser excluída com o intuito de liberar espaço no servidor.

        i. Escolhida a versão, efetuar a exportação dos usuários de banco de dados que são pertencentes à versão que está sendo excluída (uniface, objeto e dados). Essa exportação gerará um arquivo com extensão DMP

que servirá como cópia de segurança caso for no futuro for necessário utilizar esta versão novamente e deve ser gravado em mídia (CD, fita, etc);

ii. Efetuar cópia de segurança dos diretórios de arquivos da versão (telas, relatórios e processos internos) a ser excluída em mídia (CD, fita, etc);

iii. Excluir versão (usuário do banco de dados, diretórios de telas, relatório e processos internos);

b) Efetuar a criação da nova tablespace da versão e seus respectivos usuários (uniface, objeto e dados);

c) Efetuar a importação da versão através de um arquivo com extensão DMP que é criado todos os dias (através de processos de cópia de segurança que são executados pelo servidor). Este arquivo é utilizado como cópia de segurança do ambiente de desenvolvimento (versão atual) e quando uma versão é fechada, o arquivo serve para efetuar a importação do sistema em outro usuário que acaba de ser criado.

d) Criar novo arquivo ***asn*** (arquivo de características) do Uniface que servirá como arquivo de acesso à versão que está sendo fechada. É dentro deste arquivo que são especificados em qual diretório se encontra os componentes da versão, sua biblioteca de dados, seu arquivo de assinaturas, etc.

e) Iniciar a ferramenta de desenvolvimento Uniface 7.0 (utilizando novo arquivo ***asn***) e na linha de comando inicial digitar a palavra ***/cln*** com o intuito de efetuar um clean-up (limpeza) em todos os componentes do sistema;

f) Iniciar a ferramenta de desenvolvimento Uniface 7.0 (utilizando novo arquivo ***asn***) e na linha de comando inicial digitar a palavra ***/all*** com o intuito de efetuar uma compilação em todos os componentes e objetos do sistema;

g) Iniciar a ferramenta de desenvolvimento Uniface 7.0 (utilizando novo arquivo ***asn***) e na linha de comando inicial digitar a palavra ***/dol*** com o intuito de gerar em arquivo uma biblioteca com todos os objetos globais do sistema. Ex. Menus, mensagens, imagens, processos globais, variáveis globais, etc;

h) Iniciar a ferramenta de desenvolvimento Uniface 7.0 (utilizando novo arquivo ***asn***) e na linha de comando inicial digitar a palavra ***/urr*** com o intuito de gerar em arquivo todas as assinaturas entre os componentes, isto é, todas as "ligações" entre os componentes que são criadas através do comando ***activate*** *(comando utilizado para efetuar chamadas e processos de outros componentes)*;

i) Iniciar a ferramenta de desenvolvimento Uniface 7.0 (utilizando novo arquivo ***asn***) e não informar nada na linha de comando inicial com o intuito de entrar no ambiente de desenvolvimento. Através da opção de ***Deployment → create tables → Oracle***, para cada modelo de dados que compõem o sistema que está sendo fechado, deverá ser criado o arquivo de criação das entidades;

j) Iniciar a ferramenta de desenvolvimento Uniface 7.0 (utilizando novo arquivo ***asn***) e não informar nada na linha de comando inicial com o intuito de entrar no ambiente de desenvolvimento. Através da opção de ***Deployment → create script → Oracle***, para cada modelo de dados que compõem o sistema que está sendo fechado, deverá ser criada o arquivo de criação de relacionamento entre as entidades;

k) Iniciar a ferramenta de manipulação de dados SQL PLUS e conectar como administrador de banco de dados. Utilizar para isso o usuário ***system*** com a senha ***manager*** com o nome do banco (na empresa desenvolvedora é ***oracle***);

l) Criar dentro do banco de dados um usuário chamado ***temp*** com a senha ***temp*** com o intuito de dentro dele, executar os arquivos de criação de entidades e relacionamentos criados nos processos anteriores;

m) Ainda utilizando a ferramenta de manipulação de dados SQL PLUS, executar dentro do usuário ***temp*** um arquivo chamado ***describe nome_do_arquivo.lst*** (geralmente *describe strategi.lst)* que vai gerar um arquivo texto com todas as estruturas das entidades geradas a partir dos arquivos de criação executados dentro do usuário ***temp***;

n) Copiar para a nova estrutura de diretórios todos os diretórios e arquivos que fazem parte do sistema que está sendo fechado. Exemplo: diretório de componentes, diretórios das estruturas de entidades/relacionamentos, diretório dos arquivos de

biblioteca e assinaturas, diretório de ajuda do sistema, diretório com arquivo de rotinas novas, etc.

3.2.1.2 ATUALIZAÇÃO DE VERSÃO

O processo de atualização de versão é executado no cliente e efetuado alguns dias após uma nova versão ter sido fechada. O cliente tem o interesse em atualizar seu software para a versão mais atual, pois o mesmo pode ter solicitado alterações no software e necessita utilizá-las. Os processos de atualização de versão são os seguintes:

a) O funcionário de posse de toda a estrutura de diretórios que a versão possui, deve iniciar o processo de atualização de versão efetuando uma cópia dos dados do cliente através de uma exportação do usuário de dados. Em outras palavras, vai gerar (através de um software da Oracle) um arquivo de extensão DMP com todos os dados do cliente;

b) Copiar toda a estrutura de diretórios da versão para sua estrutura de diretórios do software;

c) Conectar, via SQL PLUS, dentro do usuário de dados do cliente e rodar os arquivos de criação das entidades;

d) Executar dentro do SQL PLUS, o comando ***describe nome_do_arquivo.lst*** (geralmente *describe nome_da_empresa.lst)* que vai gerar um arquivo texto com a atual estrutura do modelo de dados do cliente;

e) Executar, a partir do prompt do sistema operacional, um arquivo chamado ***compara*** que vai efetuar a comparação dos dois arquivos de estrutura de dados (o arquivo da nova versão e o arquivo da versão atual que o cliente está utilizando). Deve-se passar os dois arquivos como parâmetro para o arquivo executável. (Ex. ***compara strategi.lst nome_da_empresa.lst***).Esse arquivo executável gera um arquivo chamado ***tabelas.dif*** que possui todas as diferenças detectadas entre os dois arquivos passados como parâmetros;

f) Verificar no arquivo ***tabelas.dif*** as diferenças encontradas.

g) Agora inicia o processo mais demorado de uma atualização, pois para cada diferença detectada o atualizador deve efetuar as alterações no modelo de dados para que o mesmo fique de acordo com a nova versão que está sendo instalada. Para facilitar o trabalho deve-se eliminar todos os relacionamentos das tabelas, para que ao manusear com os dados, o SQL PLUS não acuse nenhum problema de integridade referencial. Efetuar isso usando o comando ***dropar***. Esse processo irá gerar um arquivo na raiz do computador que está sendo utilizado chamado ***drop*** contendo todos os comandos de exclusão de relacionamento. Dentro do SQL PLUS executar o arquivo ***drop*** para que todos os relacionamentos sejam eliminados.

h) Com base no arquivo ***tabelas.dif*** deve-se efetuar as alterações necessárias no modelo de dados. As diferenças detectadas podem ser as seguintes:

   a. Inserir entidade nova. Solução: executar o comando de criação de tabela da entidade nova encontrado dentro do arquivo de criação de entidades;

   b. Entidade excluída. Solução: excluir a entidade usando o comando ***drop table nome_da_entidade***;

   c. Inserir campo novo no fim da entidade. Solução: efetuar a alteração na entidade utilizando o comando ***alter table nome_da_entidade add nome_do_campo tipo_de_dado(tamanho_do_campo)***;

   d. Inserir campo novo no fim da entidade obrigatório. Solução: efetuar a alteração na entidade utilizando o comando ***alter table nome_da_entidade add nome_do_campo tipo_de_dado(tamanho_do_campo) not null***;

   e. Inserir campo novo no fim da entidade obrigatório e com relacionamento. Solução: efetuar a alteração na entidade utilizando o comando ***alter table nome_da_entidade add nome_do_campo tipo_de_dado(tamanho_do_campo) not null*** e executar o comando de criação dos relacionamentos para esta entidade encontrados dentro do arquivo de criação de relacionamentos;

   f. Inserir campo novo no meio da entidade. Solução: criar entidade temporária com os dados utilizando o comando ***create table xnome_da_entidade as select***

**** from nome_da_entidade unrecoverable***. Excluir a entidade e cria-la novamente usando o comando de criação de entidade encontrado no arquivo de criação de entidades e jogar os dados dentro da nova entidade buscando da entidade temporária. Excluir a entidade temporária com o comando ***drop table xnome_da_entidade***;

g. Inserir campo novo no meio da entidade obrigatório. Solução: criar entidade temporária com os dados utilizando o comando ***create table xnome_da_entidade as select * from nome_da_entidade unrecoverable***.Excluir a entidade e cria-la novamente usando o comando de criação de entidade encontrado dentro do arquivo de criação de entidades e jogar os dados dentro da nova entidade buscando da entidade temporária. Excluir a entidade temporária com o comando ***drop table xnome_da_entidade***;

h. Inserir campo novo no meio da entidade obrigatório e com relacionamento. Solução: criar entidade temporária com os dados utilizando o comando ***create table xnome_da_entidade as select * from nome_da_entidade unrecoverable***. Excluir a entidade e cria-la novamente usando o comando de criação de entidade encontrado dentro do arquivo de criação de entidades e jogar os dados dentro da nova entidade buscando da entidade temporária. Excluir a entidade temporária com o comando ***drop table xnome_da_entidade***. Executar o comando de criação de relacionamentos encontrado dentro do arquivo de criação dos relacionamentos;

i. Campo se tornou obrigatório. Solução: alterar todas as linhas de dados jogando um valor para este campo e alterar sua estrutura com o comando ***alter table nome_da_entidade modify nome_do_campo not null***;

j. Campo se tornou obrigatório e com relacionamento. Solução: alterar todas as linhas de dados jogando um valor para este campo e alterar sua estrutura com o comando ***alter table nome_da_entidade modify nome_do_campo not null***. Executar o comando de criação de relacionamentos encontrado dentro do arquivo de criação de relacionamentos;

k. Campo deixou de ser obrigatório. Solução: alterar a entidade usando o comando ***alter table nome_da_entidade modify nome_do_campo null***;

l. Campo aumentou tamanho. Solução: alterar a entidade usando o comando ***alter table nome_da_entidade modify nome_do_campo tipo_de_dado(tamanho_do_campo)***;

m. Campo diminuiu tamanho. Solução: criar entidade temporária com os dados utilizando o comando ***create table xnome_da_entidade as select * from nome_da_entidade unrecoverable***. Excluir a entidade e cria-la novamente usando o comando de criação de entidade encontrado dentro do arquivo de criação de entidades e jogar os dados dentro da nova entidade buscando da entidade temporária, sem esquecer de jogar para o campo que diminui o tamanho, os valores com seu tamanho de gravação diminuído. Excluir a entidade temporária com o comando ***drop table xnome_da_entidade***;

n. Campo aumentou precisão. Solução: alterar a entidade usando o comando ***alter table nome_da_entidade modify nome_do_campo tipo_de_dado(tamanho_do_campo)***;

o. Campo diminuiu precisão. Solução: criar entidade temporária com os dados utilizando o comando ***create table xnome_da_entidade as select * from nome_da_entidade unrecoverable***. Excluir a entidade e cria-la novamente usando o comando de criação de entidade encontrado dentro do arquivo de criação de entidades e jogar os dados dentro da nova entidade buscando da entidade temporária, sem esquecer de jogar para o campo que diminui a precisão, os valores com sua precisão diminuída. Excluir a entidade temporária com o comando ***drop table xnome_da_entidade***;

p. Campo mudou tipo de dado. Solução: criar entidade temporária com os dados utilizando o comando ***create table xnome_da_entidade as select * from nome_da_entidade unrecoverable***. Excluir a entidade e cria-la novamente usando o comando de criação de entidade encontrado dentro do arquivo de criação de entidades e jogar os dados dentro da nova entidade buscando da

entidade temporária, sem esquecer de jogar para o campo que alterou o tipo de dado, a sua conversão correspondente;

q. Campo excluído da entidade. Solução: utilizar o comando ***alter table nome_da_entidade drop column nome_do_campo***;

r. Incluído campo no relacionamento. Solução: serão alterados quando forem executados os arquivos de criação de relacionamentos entre entidades;

s. Excluído campo no relacionamento. Solução: serão alterados quando forem executados os arquivos de criação de relacionamentos entre entidades;

i) Executar os arquivos de criação de relacionamentos entre entidades;

j) Alterar o arquivo ***asn*** para utilizar os novos arquivos de biblioteca de dados e assinaturas entre os componentes do sistema.

3.2.2 ETAPA 2 – DESCRIÇÃO DOS OBJETIVOS DO PROTÓTIPO

O protótipo tem como objetivo automatizar o processo de fechamento e atualização de versão. Isto é, automatizar todos os processos que fazem parte das etapas de fechamento e atualização de versão. Esses processos foram apurados junto ao departamento de suporte da empresa onde o estágio foi efetuado.

3.2.3 ETAPA 3 – LISTA DE EVENTOS

A lista de eventos é composta de 13 eventos e os mesmos foram relacionados abaixo.

a) Empresa é cadastrada.

b) Sistemas da empresa são cadastrados.

c) Usuários do protótipo são cadastrados.

d) Usuários do banco de dados são cadastrados.

e) Usuário Strategies cadastra módulos.

f) Usuário Strategies cadastra rotinas.

g) Usuário Strategies cadastra tipo de alteração.

h) Usuário Strategies cadastra banco de dados.

i) Usuário Strategies cadastra sistema.

j) Usuário Strategies cadastra modelos do sistema.

k) Usuário Strategies cadastra versão.

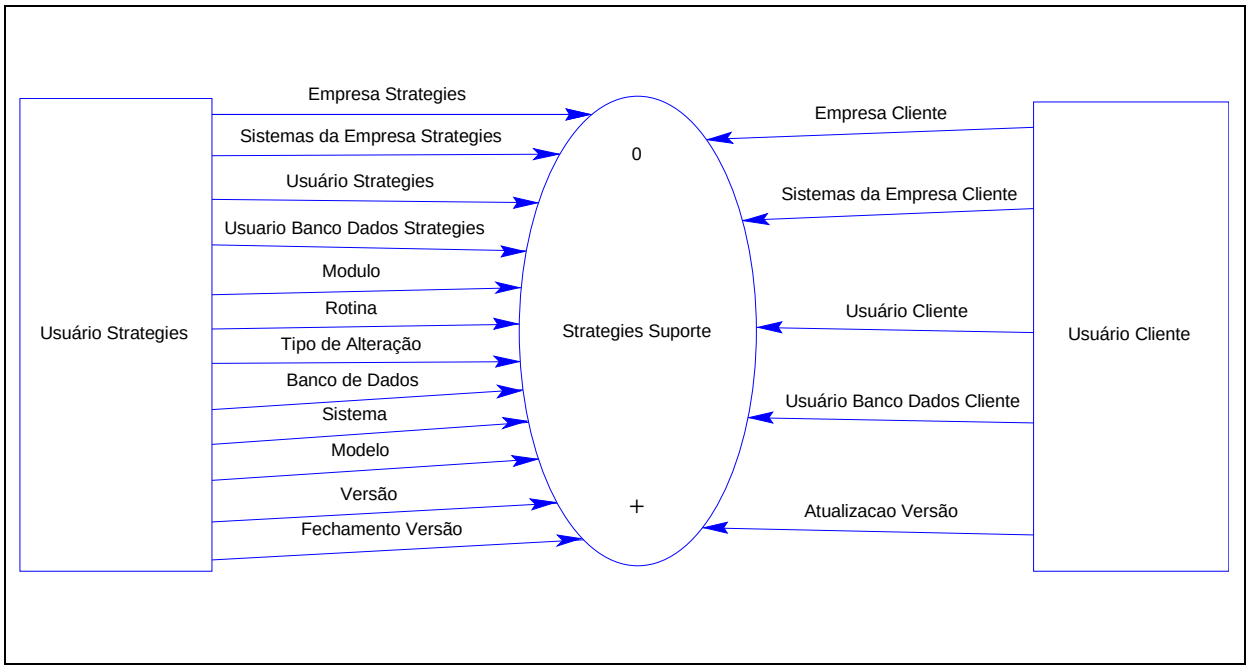
l) Usuário Strategies efetua fechamento de versão.

m) Usuário Cliente efetua atualização de versão.

3.2.4 ETAPA 4 – DIAGRAMA DE CONTEXTO

O diagrama de contexto do protótipo pode ser visualizado na figura 18.

Figura 18 – Diagrama de contexto.

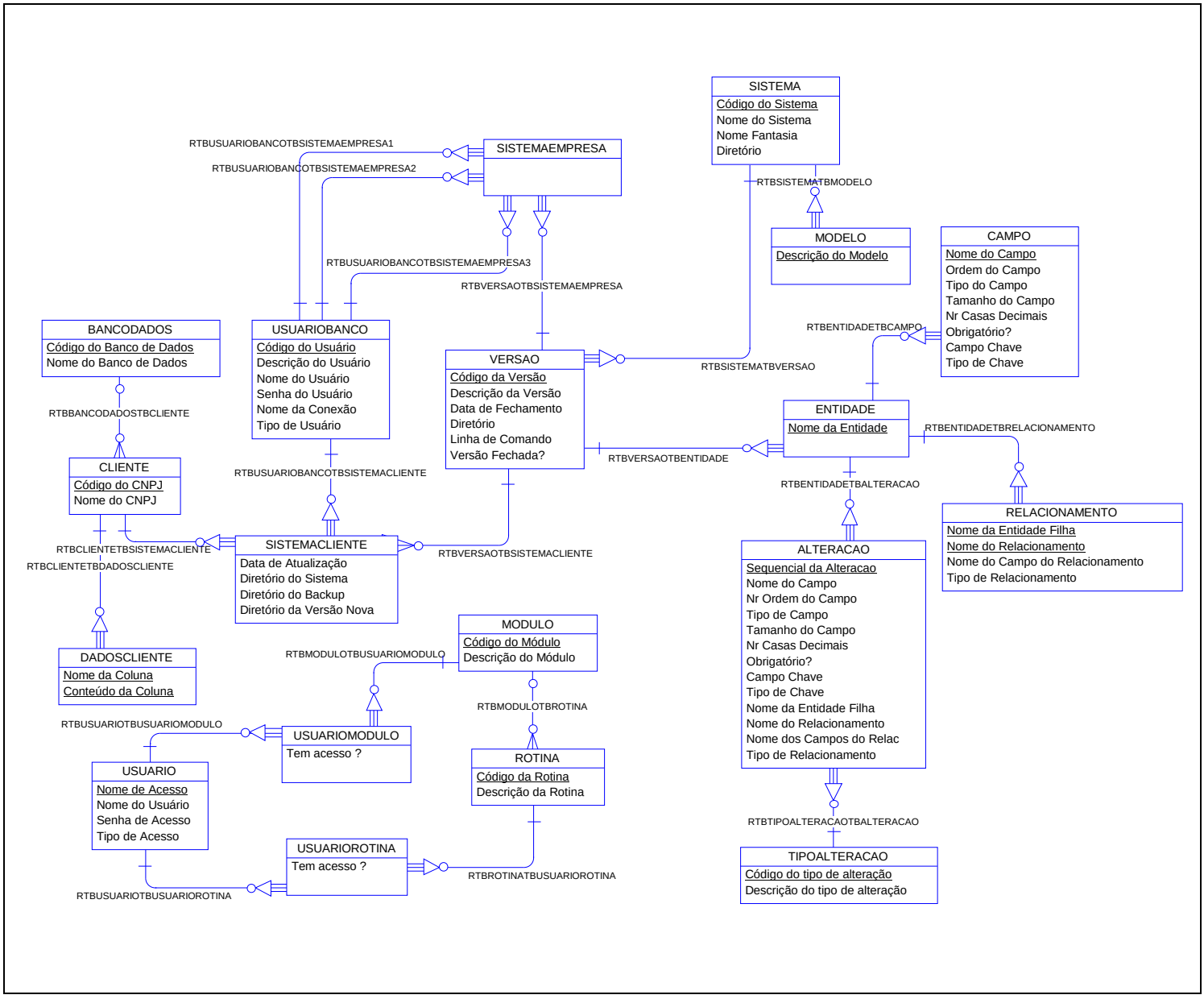


3.2.5 ETAPA 5 – MODELO ENTIDADE RELACIONAMENTO

3.2.5.1 MODELO LÓGICO

O modelo de dados entidade relacionamento lógico pode ser visualizado na figura 19.

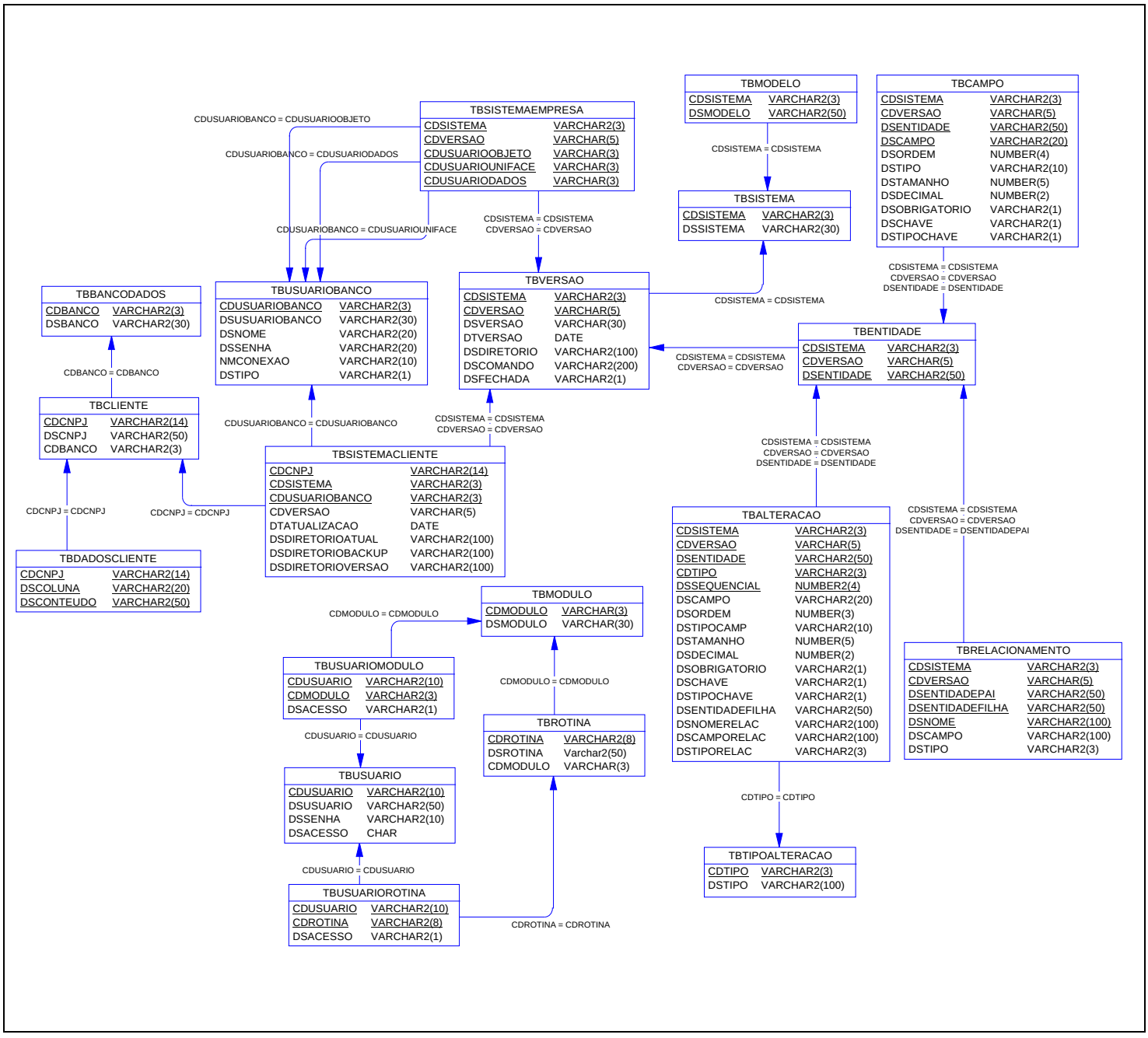
Figura 19 – Modelo de dados entidade relacionamento lógico.



3.2.5.2 MODELO FÍSICO

O modelo de dados entidade relacionamento físico pode ser visualizado na figura 20.

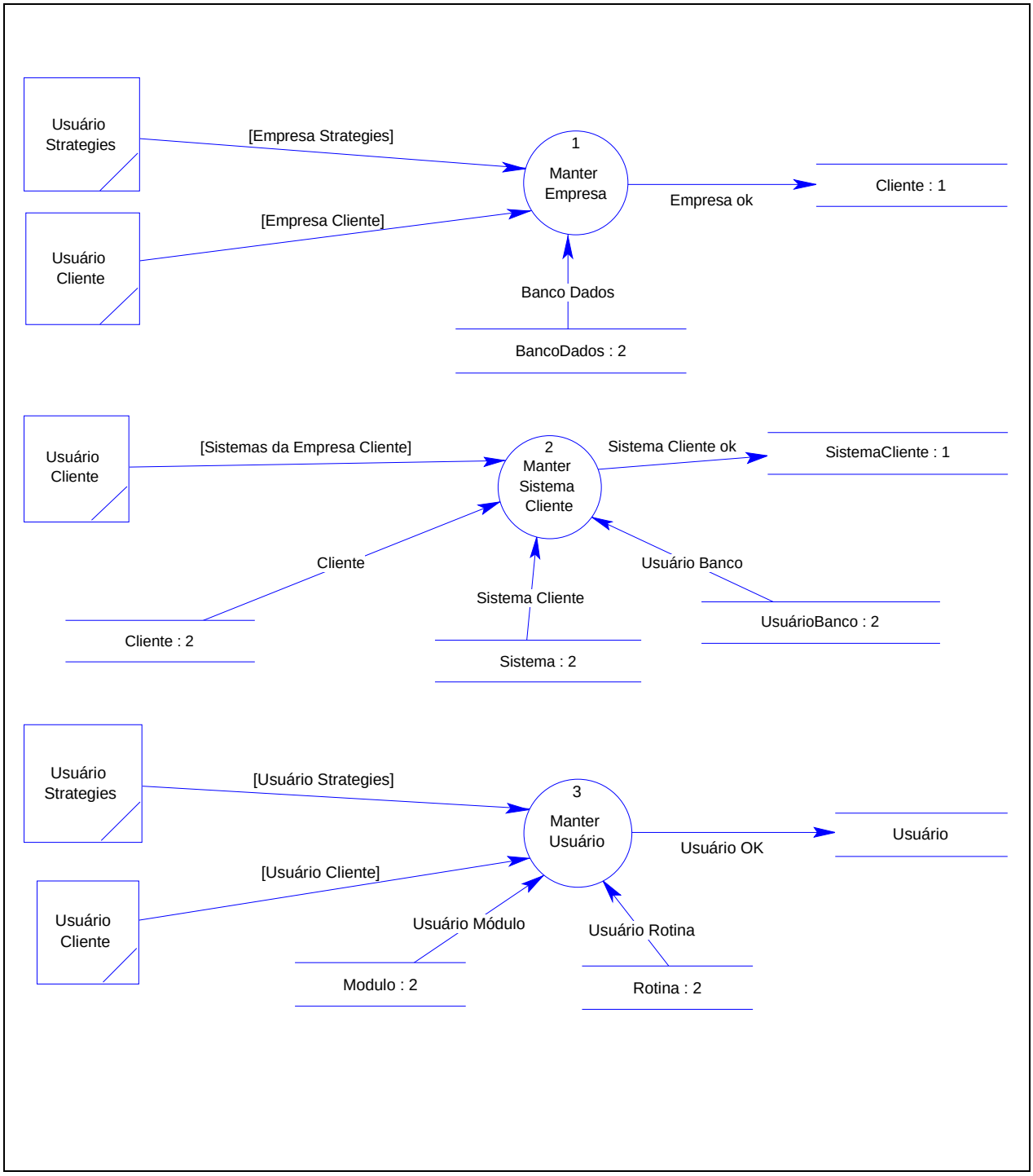
Figura 20 – Modelo de dados entidade relacionamento físico.

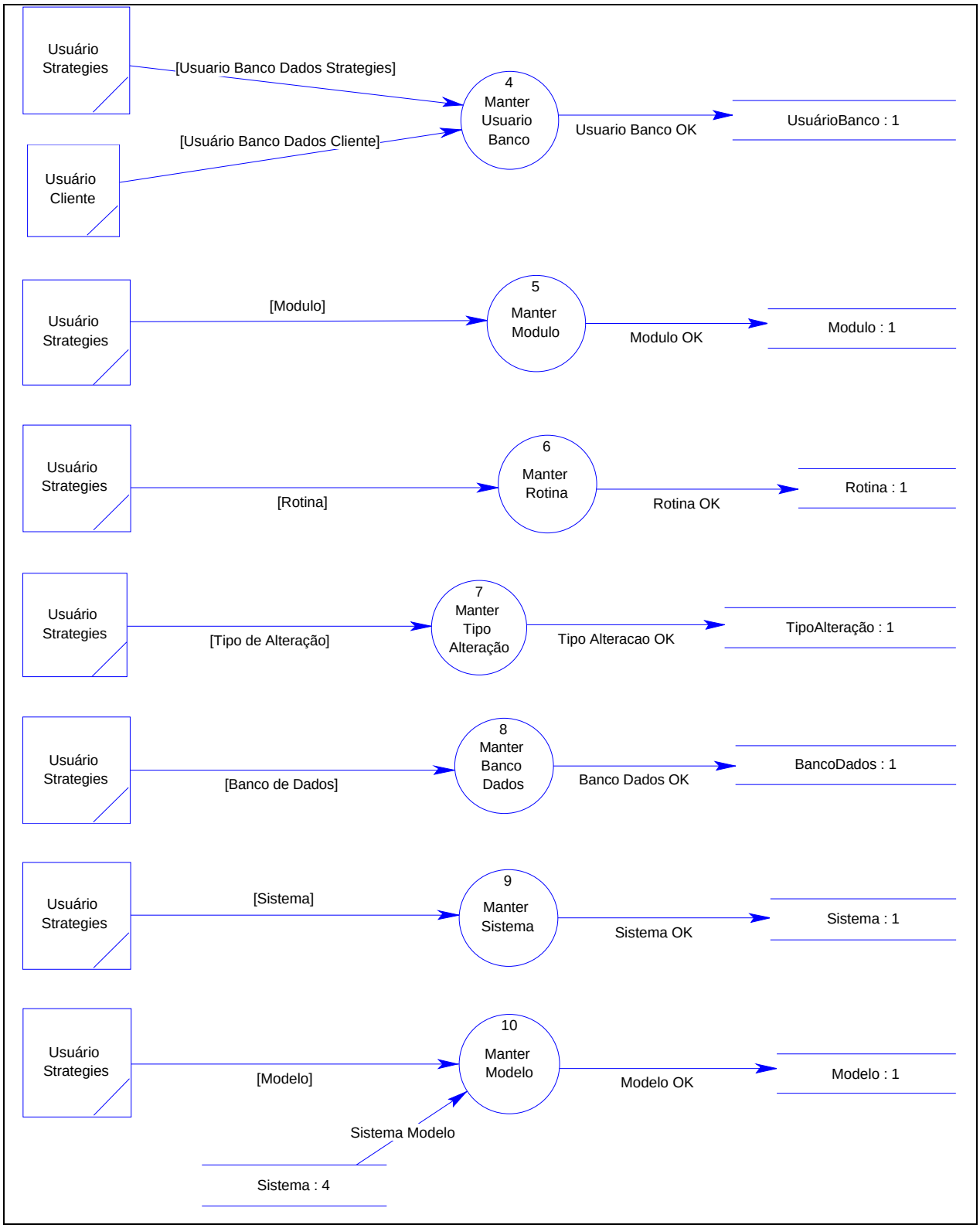


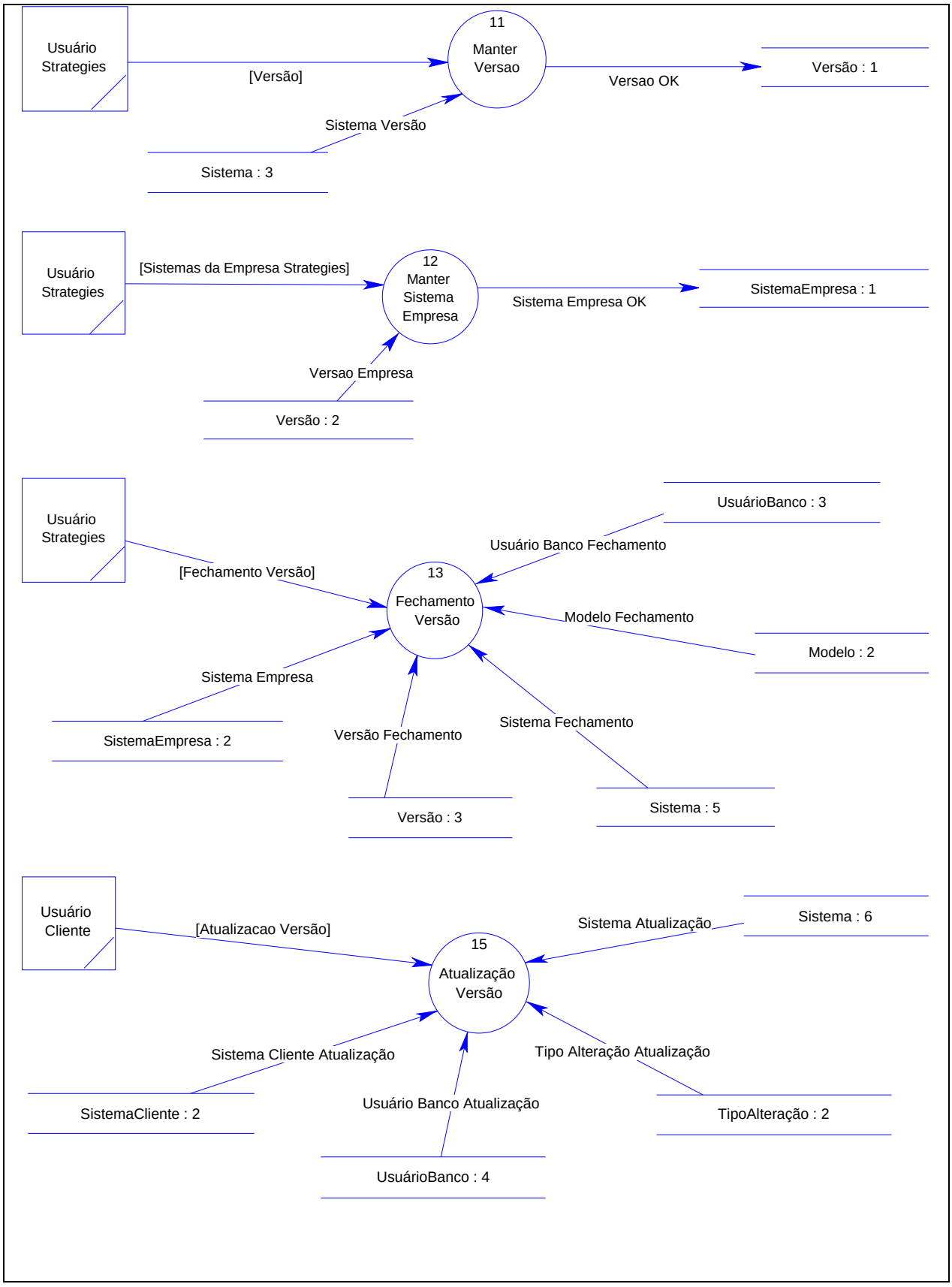
3.2.6 ETAPA 6 – DIAGRAMA DE FLUXO DE DADOS (D.F.D.)

Os diagramas de fluxo de dados podem ser visualizados na figura 21.

Figura 21 – Diagrama de fluxo de dados.



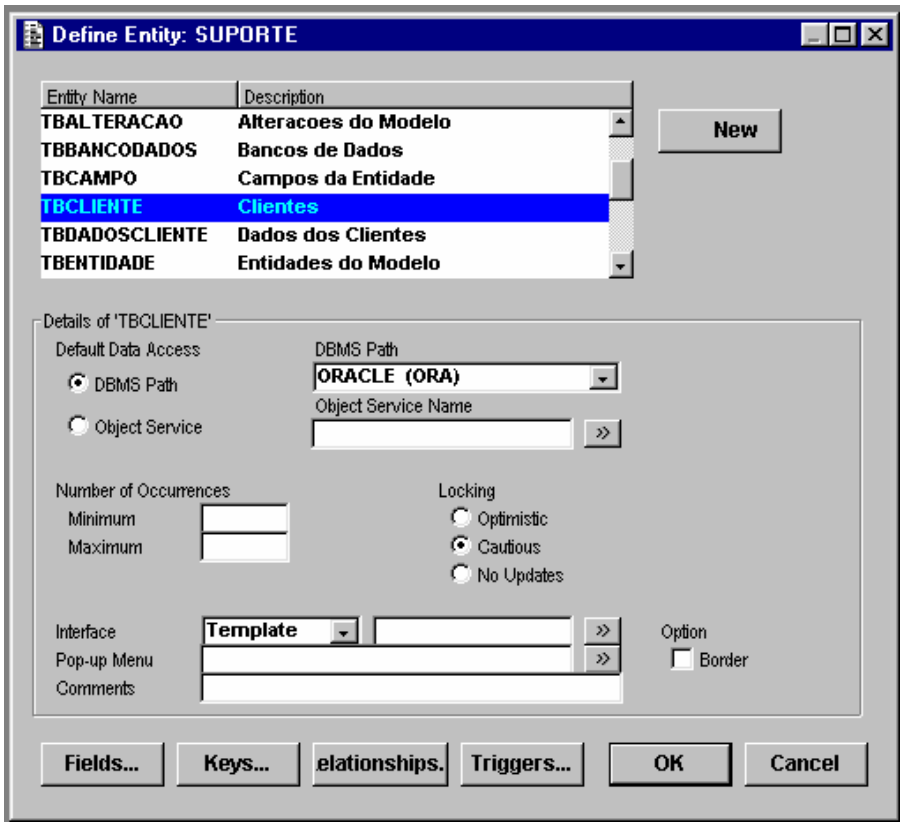

Usuário Strategies
[Usuario Banco Dados Strategies]
4 Manter Usuario Banco
Usuario Banco OK
UsuárioBanco : 1
Usuário Cliente
[Usuário Banco Dados Cliente]
Usuário Strategies
[Modulo]
5 Manter Modulo
Modulo OK
Modulo : 1
Usuário Strategies
[Rotina]
6 Manter Rotina
Rotina OK
Rotina : 1
Usuário Strategies
[Tipo de Alteração]
7 Manter Tipo Alteração
Tipo Alteracao OK
TipoAlteração : 1
Usuário Strategies
[Banco de Dados]
8 Manter Banco Dados
Banco Dados OK
BancoDados : 1
Usuário Strategies
[Sistema]
9 Manter Sistema
Sistema OK
Sistema : 1
Usuário Strategies
[Modelo]
10 Manter Modelo
Modelo OK
Modelo : 1
Sistema Modelo
Sistema : 4


Usuário Strategies
[Versão]
11
Manter Versao
Versao OK
Versão : 1
Sistema Versão
Sistema : 3
Usuário Strategies
[Sistemas da Empresa Strategies]
12
Manter Sistema Empresa
Sistema Empresa OK
SistemaEmpresa : 1
Versao Empresa
Versão : 2
Usuário Strategies
UsuárioBanco : 3
Usuário Banco Fechamento
[Fechamento Versão]
13
Fechamento Versão
Modelo Fechamento
Modelo : 2
Sistema Empresa
SistemaEmpresa : 2
Versão Fechamento
Sistema Fechamento
Versão : 3
Sistema : 5
Usuário Cliente
[Atualizacao Versão]
Sistema Atualização
Sistema : 6
15
Atualização Versão
Sistema Cliente Atualização
Tipo Alteração Atualização
SistemaCliente : 2
Usuário Banco Atualização
TipoAlteração : 2
UsuárioBanco : 4

3.3 IMPLEMENTAÇÃO

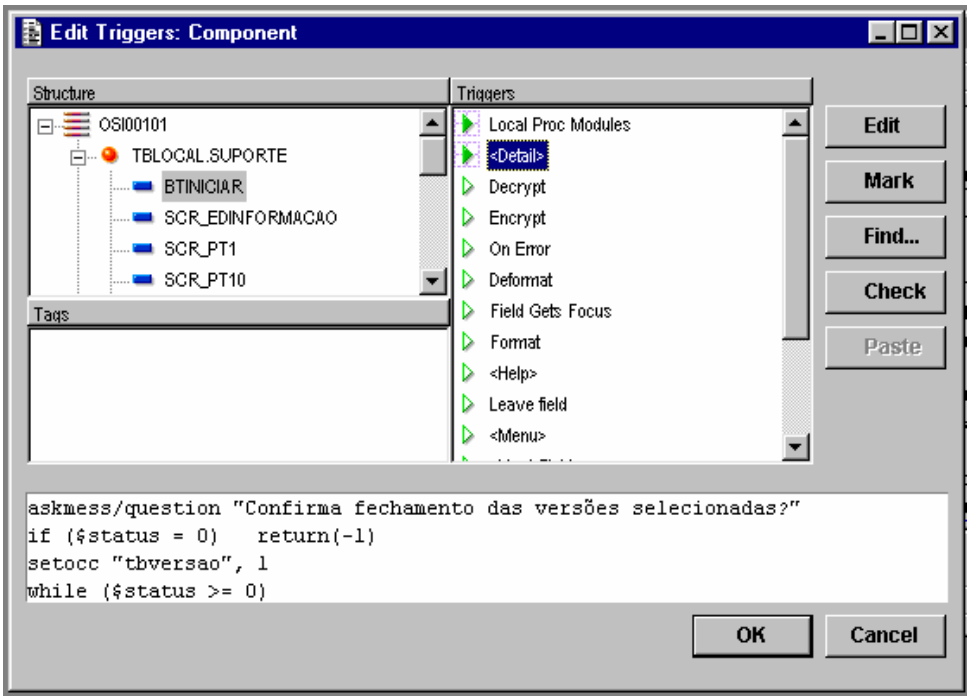
Toda a implementação do protótipo foi feita utilizando a ferramenta de desenvolvimento UNIFACE versão 7.0. Desde a parte de criação das entidades até a codificação dos programas. A criação de entidades no UNIFACE foi feita paralelamente ao desenvolvimento do modelo entidade relacionamento físico. Isso foi necessário porque as entidades devem estar criadas no UNIFACE para que seja possível a criação das linhas de códigos nos componentes da aplicação. Na figura 22 pode-se visualizar o ambiente de criação de entidade bem como suas propriedades (campos, chaves, relacionamentos, etc).

Figura 22 – Ambiente de criação de entidades e suas propriedades.



O desenvolvimento de linhas de código utilizando UNIFACE é semelhante à ferramenta de desenvolvimento DELPHI. Ambos utilizam o conceito de eventos. No UNIFACE são conhecidos como *triggers*. Eles podem ser de formulário, de entidade ou de campo. Para cada tipo existem eventos específicos. Na figura 23 pode-se ver como é o editor de *triggers* do UNIFACE.

Figura 23 – Editor de *trigger*.



Cada *trigger* tem sua função específica, isto é, um evento específico. No momento em que o evento acontece, se o *trigger* relacionado possuir linhas de código, as mesmas são executadas. Pode-se ver no quadro 3 um exemplo de linhas de código retirado da tela de fechamento de versão, no *trigger* <*detail*> *(evento clicar)* onde o mesmo efetua a pergunta se o usuário confirma o fechamento das versões selecionadas e caso a resposta do usuário for positiva a aplicação inicia o processo.

3.3.1.1.1.1.1.1 QUADRO 3 – EXEMPLO DE LINHAS DE CÓDIGO NO *TRIGGER <DETAIL>*.

```
askmess/question "Confirma fechamento das versões selecionadas?"
if ($status = 0)   return(-1)
setocc "tbversao", 1
while ($status >= 0)
   if (scr_ckflag.tbversao = "T")

      ; Busca o dados dos usuarios de repositorio dos
      ; sistemas selecionados.
      clear/e "tbsistemaempresa"
      cdsistema.tbsistemaempresa = cdsistema.tbversao
      cdversao.tbsistemaempresa  = cdversao.tbversao
      retrieve/e "tbsistemaempresa"
      if ($status < 0)
         askmess/question "Os usuários de banco de dados de
%%dssistema.tbsistema não estão cadastrados.","Cancelar tudo","Continuar"
```

```
         if ($status = 1)
            message/info "Processo de fechamento das versões cancelado."
            return(-1)
         endif
      else

         ; Inicia as imagens das tarefas a serem efetuadas.
         call Lp_Imagens
         message "Fechando sistema %%dssistema.tbsistema versão
%%cdversao.tbversao"
         scr_edinformacao.tblocal = "%%dssistema.tbsistema Versão
%%cdversao.tbversao"
         show

         ; Verifica espaço disponivel no servidor
         message "Verificando espaço disponível no servidor..."
         call Lp_Espacodisponivel

         ; Cria repositório
         message "Criando repositório..."
         call Lp_Repositorio

         ; Busca modulos e rotinas
         message "Buscando módulos e rotinas..."
         call Lp_Modulos

         ; Efetua Clean Up de todos os componentes
         message "Efetuando Clean Up de todos os componentes..."
         call Lp_Cleanup

         ; Analiza todos os modelos de dados
         message "Analizando todos os modelos de dados..."
         call Lp_Analise

         ; Efetua compilação geral
         message "Efetuando compilação geral..."
         call Lp_Compilacao

         ; Gera dol
         message "Gerando arquivo de biblioteca de dados (dol)..."
         call Lp_Dol

         ; Gera urr
         message "Gerando arquivo de assinaturas (urr)..."
         call Lp_Urr

         ; Busca estrutura dos modelos de dados
         message "Buscando estrutura do(s) modelo de dados..."
         call Lp_Estrutura

         ; Efetua cópia dos componentes
         message "Efetuando cópia dos componentes..."
         call Lp_Copia

         ; Cria o arquivo VERSAO.INI com informações da versão fechada
         message "Criando arquivo identificador..."
         call Lp_Arquivo
         dsfechada.tbversao = "T"
         store/e "tbversao"
```

```
         if ($status < 0)
            message/error "Erro de gravação na entidade TBVERSAO"
            return(-1)
         endif
         message/info "Fechamento sistema %%dssistema.tbsistema versão
%%cdversao.tbversao efetuada com êxito."
         commit
      endif
   endif
   setocc "tbversao", $curocc(tbversao) + 1
endwhile
message " "
exit
```

Pode-se visualizar no quadro 4 um exemplo de linhas de código retirado da tela de fechamento de versão, no *trigger <Local Proc Modules> (lugar especifico para guardar procedimentos)* onde efetua-se busca em dados dentro do repositório UNIFACE da estrutura atual das entidades e suas propriedades como campos, chaves etc.

3.3.1.1.1.1.1.2 QUADRO 4 – EXEMPLO DE LINHAS DE CÓDIGO NO *TRIGGER <LOCAL PROC MODULES>*.

```
; Busca estrutura dos modelos de dados
entry Lp_Estrutura

   variables
      string conexao
      string nome
      string senha
      string usuario
   endvariables

   scr_pt9.tblocal = "^U_SETA"
   show
   conexao = nmconexao.sttbusuariobanco2
   nome    = dsnome.sttbusuariobanco2
   senha   = dssenha.sttbusuariobanco2
   usuario = "%%conexao|%%nome|%%senha"
   open "%%usuario", "$UD0"
   if ($status < 0)
      message "Erro abrindo usuário Uniface de %%dssistema.tbsistema."
      return(-1)
   endif
   ; Busca os modelos de dados do sistema selecionado
   clear/e "tbmodelo"
   cdsistema.tbmodelo = cdsistema.tbversao
   retrieve/e "tbmodelo"
   if ($status < 0)
      message/error "Os Modelos de %%dssistema.tbsistema não estão
cadastrados."
      return(-1)
   endif
   ; Para cada modelo cadastrado, busca seus dados dentro do repositorio
   setocc "tbmodelo", 1
```

```
   while ($status >= 0)
      retrieve/e "tbentidade"
      setocc "tbentidade", -1
      erase/e "tbentidade"
      if ($status < 0)
         message/error "Erro excluindo ocorrencias entidade TBENTIDADE"
         return(-1)
      endif
      ; Busca informacoes das entidades do modelo
      clear/e "ucsch"
      u_vlab.ucsch = dsmodelo.tbmodelo
      retrieve/e "ucsch"
      if ($status < 0)
         message/error "O modelo %%dsmodelo.tbmodelo %%%não está cadastrado
na modelagem de dados do UNIFACE."
         return(-1)
      endif
      u_vlab.ucsch = u_vlab.ucsch
      ; Para cada modelo, busca as entidades
      setocc "uctable", 1
      while ($status >= 0)
         creocc "tbentidade", -1
         dsentidade.tbentidade = u_tlab.uctable
         ; Para cada entidade, busca seus campos
         setocc "ucfield", 1
         while ($status >= 0)
            creocc "tbcampo", -1
            dscampo.tbcampo   = u_flab.ucfield
            dsordem.tbcampo   = u_fseq.ucfield
            dstipo.tbcampo    = u_dtyp.ucfield
            if (u_dtyp.ucfield = "D" | u_dtyp.ucfield = "I")
               dstamanho.tbcampo = 0
               dsdecimal.tbcampo = 0
            elseif (u_dtyp.ucfield = "B")
               dstamanho.tbcampo = 1
               dsdecimal.tbcampo = 0
            else
               ; Se o campo tplintf da entidade ucfield estiver preenchido
               ; é porque usa template e tem que pegar da tabela de
template
               if (tplintf.ucfield != "")
                  clear/e "ugfif"
                  u_mlab.ugfif = tplintf.ucfield
                  retrieve/e "ugfif"
                  if ($status < 0)
                     message/error "Template de interface de campo
%%tplintf.ucfield não encontrado no repositorio do Uniface.%%%^%%%Entidade
%%u_tlab.uctable Campo %%u_flab.ucfield"
                  endif
                  ; Procura o primeiro digito no campo Ex. N14.4
                  scan u_mod.ugfif, '#'
                  if ($result > 0)
                     ; $1 recebe 14.4
                     $1 = u_mod.ugfif[$result]
                  endif
                  clear/e "ugfif"
               ; Nào tem template, pega direto do campo u_intf.
               else
                  ; Procura o primeiro digito no campo Ex. N14.4
```

```
                  scan u_intf.ucfield, '#'
                  ; $1 recebe 14.4
                  $1 = u_intf.ucfield[$result]
               endif
               ; Procura o ponto (casa decimal)
               scan $1, '.'
               ; Se não achou, joga no tamanho o $1 e no decimal zero.
               if ($result = 0)
                  dstamanho.tbcampo = $1
                  dsdecimal.tbcampo = 0
               ; Se achou
               else
                  ; o $2 recebe o que está a partir do ponto
                  $3 = $result + 1
                  $2 = $1[$3]
                  ; o $1 recebe ele mesmo se o ponto atrás
                  $1 = $1[1:$result]
                  ; O tamanho recebe o $1 que é o inteiro
                  dstamanho.tbcampo = $1
                  ; O decimal recebe o $2 que é o que mostro nro casas dec.
                  dsdecimal.tbcampo = $2
               endif
            endif
            scan u_syn.ucfield, 'MAN'
            if ($result > 0)
               dsobrigatorio.tbcampo = "T"
            else
               dsobrigatorio.tbcampo = "F"
            endif
            setocc "ucfield", $curocc(ucfield) + 1
         endwhile
         store/e "tbentidade"
         if ($status < 0)
            message/error "Erro de gravação na entidade TBENTIDADE"
            return(-1)
         endif
         setocc "uctable", $curocc(uctable) + 1
      endwhile
      setocc "tbmodelo", $curocc(tbmodelo) + 1
   endwhile
   scr_pt9.tblocal = "^U_ACCEPT"
   scr_slprogressao.tblocal = ((100*11)/13)
   show
   ; Mover os arquivos gerados para um diretório da versao fechada.
   close "$UD0"
end
```

3.3.2 OPERACIONALIDADE DA IMPLEMENTAÇÃO

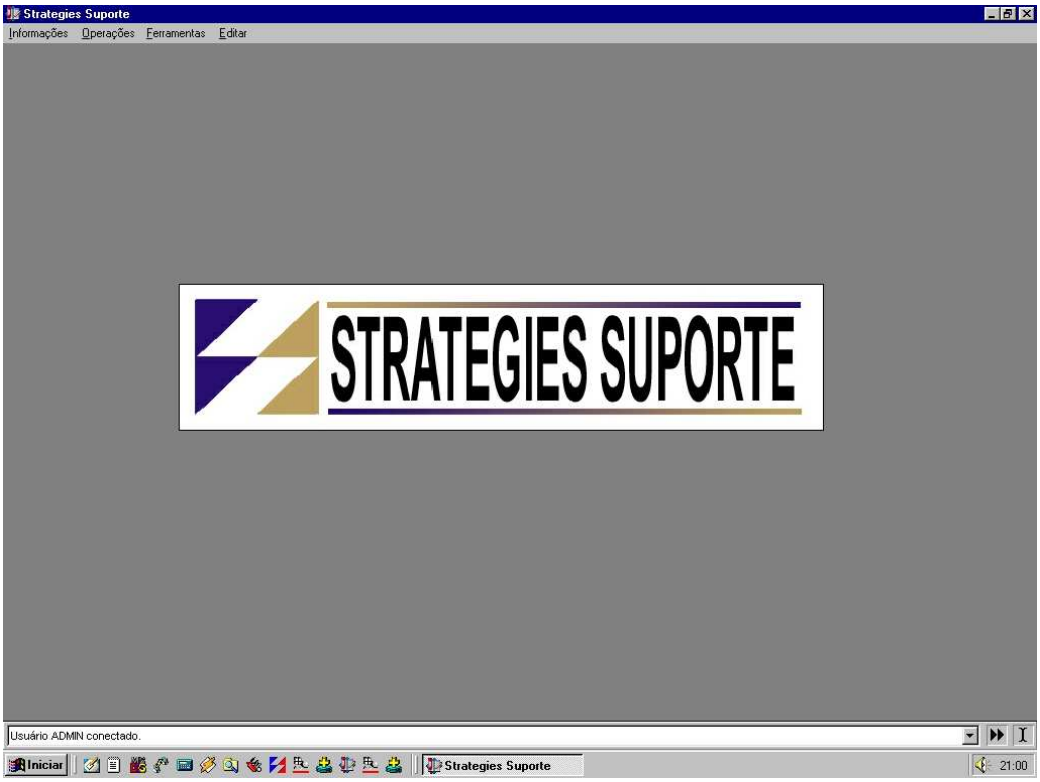
Serão apresentados os componentes desenvolvidos com o UNIFACE com o intuito de atender o objetivo do protótipo. Na figura 24 pode-se ver que o nome do protótipo é Strategies Suporte. Foi colocado esse nome pelo motivo de ser um software que será mais utilizado pelo departamento de suporte.

A figura 24 mostra a primeira tela que faz a interação com o usuário. É a tela de acesso ao protótipo. Nesta tela devem ser informados o nome do usuário, sua senha e o tipo de impressão que vai usar (Matricial, DeskJet, Arquivo ou E-mail). Essa tela tem como objetivo manter a segurança das informações do protótipo.

Figura 24 – Tela para conexão com Strategies Suporte.

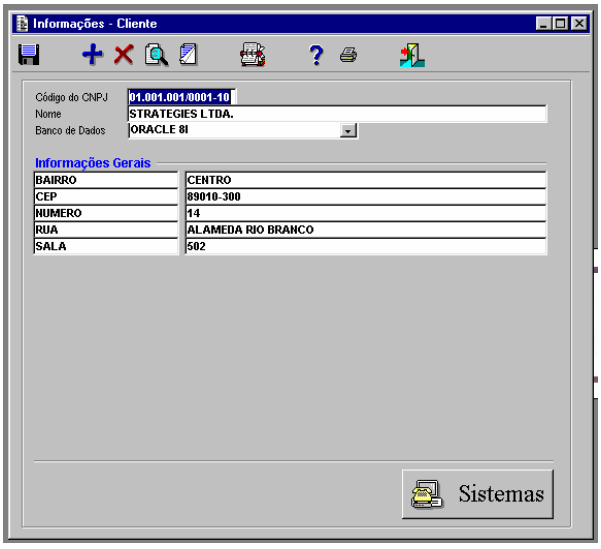
Após a identificação do usuário no Strategies Suporte, a figura 25 mostra como é tela principal do protótipo. É a partir desta tela que acessos outras telas do protótipo através dos menus de *Informações*, *Operações*, *Ferramentas* e *Editar*.

Figura 25 – Tela geral do Strategies Suporte.



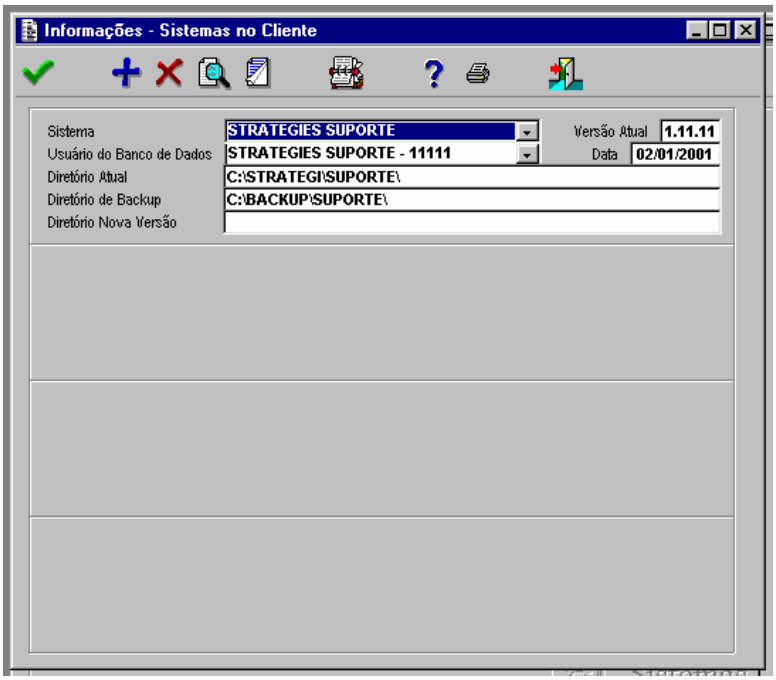
A figura 26 mostra a tela onde são mantidas as informações do cliente. Isto é, da empresa que está utilizando o Strategies Suporte. As “informações gerais” possuem a estrutura de campo e conteúdo pelo motivo de permitir que vários campos de informações da empresa possam ser incluídos sem afetar a estrutura da entidade no banco de dados.

Figura 26 – Informações do Cliente.



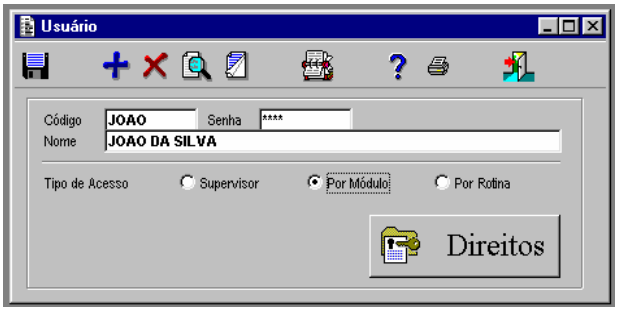
Pode-se ver na figura 27 a tela onde informações dos sistemas que a empresa usa são informados. É a partir dessa tela que informações como *diretório atual, diretório de backup e diretório da nova versão* são informadas. Essas informações serão utilizadas no processo de atualização de versão.

Figura 27 – Informações dos sistemas que o cliente possui.



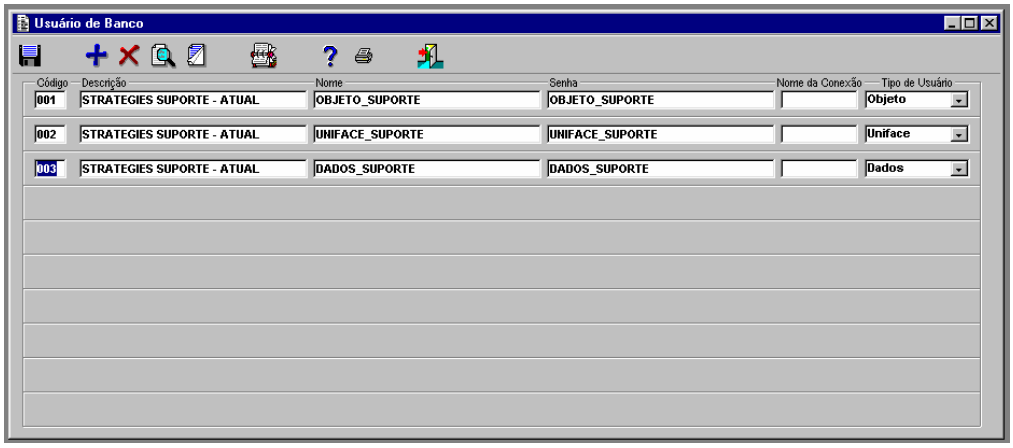
Podemos ver na figura 28 onde é efetuado o cadastro de usuários do Strategies Suporte. Esta tela o usuário “administrador” informa o nome do usuário, senha, nome completo e o tipo de acesso que o usuário pode ter. Esse tipo de acesso pode ser *supervisor, por módulo ou por rotina*. O botão de direitos é mostrado somente quando o usuário tiver acesso por módulo ou por rotina e serve para informar quais módulos / rotinas que o usuário terá acesso.

Figura 28 – Informações dos usuários do Strategies Suporte.



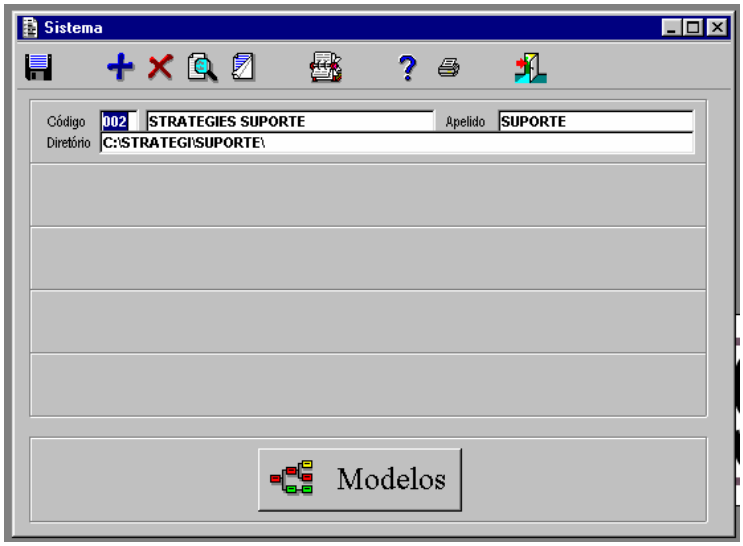
A figura 29 mostra a tela onde o cadastro de usuário de banco de dados é efetuado. Esta tela é de fundamental importância e necessita que todas as informações sejam corretas. É com base nessas informações que o processo de fechamento de versão e atualização de versão são executados. Devem ser informados o código do usuário, descrição do usuário, nome do usuário, senha do usuário, nome de conexão e o seu tipo de usuário (Uniface, Objeto e Dados).

Figura 29 – Informações dos usuários de banco de dados.



A figura 30 mostra a tela onde são cadastrados os sistemas que a empresa Strategies possui. Somente usuários administradores da Strategies (senha Admin) terão acesso a essa tela. Nesta tela existe um botão onde são cadastrados os modelos de dados que cada sistema possui, porque isso servirá para o fechamento de versão.

Figura 30 – Informações dos Sistemas da Empresa Strategies.



A figura 31 mostra a tela onde são cadastradas as versões. Somente usuários administradores da Strategies (senha Admin) terão acesso a essa tela. Nesta tela devem ser informados dados como *sistema, código da versão, descrição da versão, data da versão, diretório da versão e linha de comando no ícone do uniface.* Existe um botão chamado "*detalhes"* onde são informados os usuários de banco desta versão (*Usuário Objeto, Uniface e Dados*). A figura 32 mostra a tela de cadastro de usuários Objeto, Uniface e Dados. Esse cadastro é de fundamental importância, porque é a partir dele que o processo de fechamento de versão processará.

Figura 31 – Informações de versão dos sistemas.

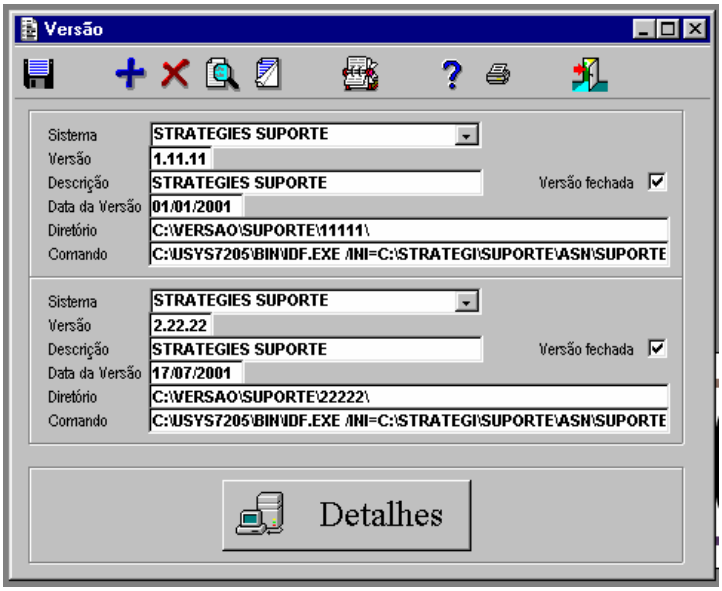
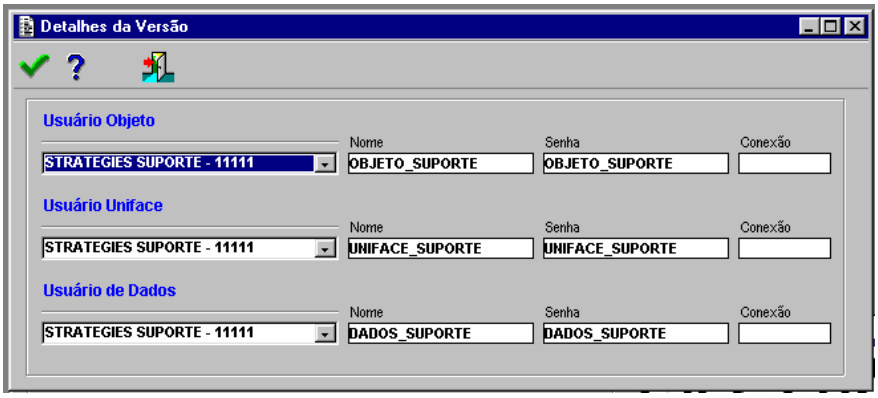


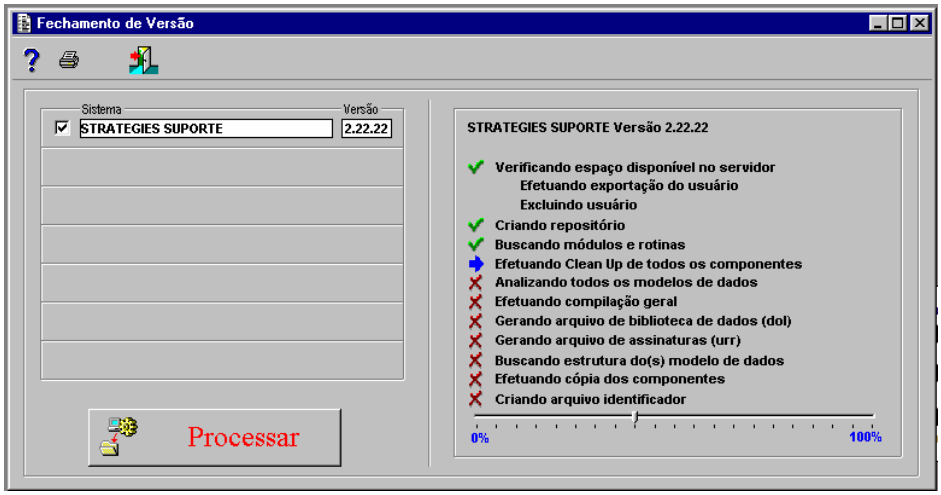
Figura 32 – Usuários Objeto, Uniface e Dados.



A figura 33 mostra a tela de fechamento de versão. Somente usuários administradores da Strategies (senha Admin) terão acesso a essa tela. É nela que os processos de fechamento

de versão são executados. O usuário seleciona quais versões deseja efetuar o fechamento e clica no botão *processar*. Após isso, inicia-se o processo de fechamento de versão e no quadro à direita é mostrado o *status* de todas as etapas do processo.

Figura 33 – Operação de fechamento de versão.



O processo de fechamento de versão efetua os seguintes processos:

a) Criação da nova tablespace da versão e seus respectivos usuários (uniface, objeto e dados);

b) Importação da versão através de um arquivo com extensão DMP;

c) Busca os módulos e rotinas que pertencem à versão do sistema que está sendo fechada;

d) Clean-up (limpeza) de todos os componentes;

e) Análise de todos os modelos de dados que pertencem à versão do sistema que está sendo fechado;

f) Compilação de todos os componentes do sistema (telas, relatórios e processos internos);

g) Geração do arquivo que contêm biblioteca de dados (DOL);

h) Geração do arquivo que contêm as assinaturas entre os componentes (URR);

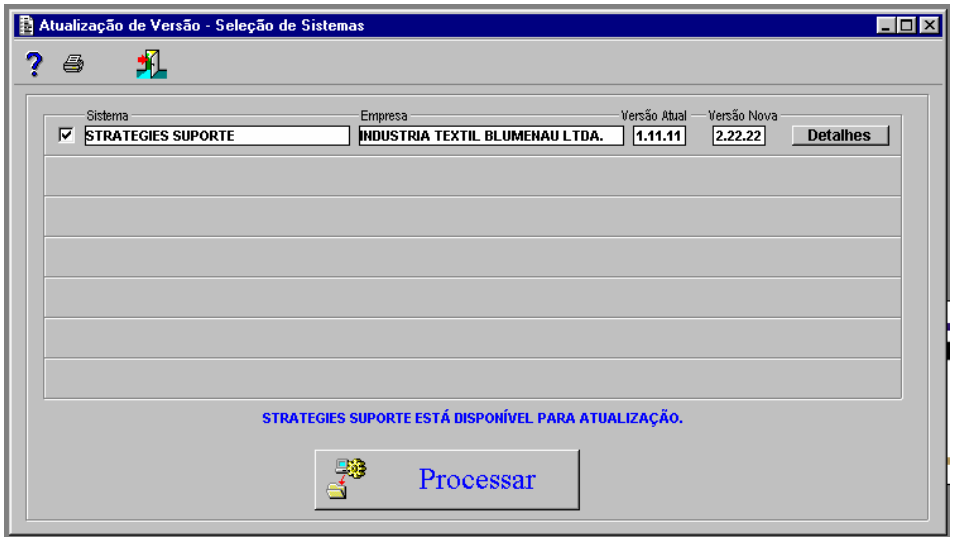
i) Busca a estrutura dos modelos de dados (entidades, campos, relacionamentos, etc) do sistema que está sendo fechado;

j) Cópia dos componentes (já compilados) para sua nova estrutura de diretórios da versão;

k) Criação de arquivo identificador que será utilizado pelo cliente na hora de atualizar a versão que está sendo fechada.

Na figura 34 pode-se visualizar a tela inicial do processo de atualização de versão. É nela que os processos de atualização de versão são executados. O usuário seleciona quais os sistemas deseja efetuar a atualização e clica no botão *processar*. Após isso, é aberta outra tela, conforme a figura 35, onde é mostrado o *status* de todas as etapas do processamento de atualização de versão. Na figura 36 podemos visualizar a tela de *status* de atualização após ter efetuado todo o processo.

Figura 34 – Tela inicial da atualização de versão.



Conforme os sistemas selecionados pelo usuário, a segunda tela de atualização de versão mostra para cada sistema, todos os processos que ele executa. Nessa tela pode existir a interação com o usuário, dependendo da alteração que deve ser efetuada no banco de dados. Um exemplo é no caso de um campo de uma determinada entidade se tornar obrigatório, nesse caso, o Strategies Suporte abre uma tela solicitando o valor que deve ser preenchido no campo antes de alterá-lo para obrigatório.

Figura 35 – Tela de status da atualização de versão durante uma atualização.

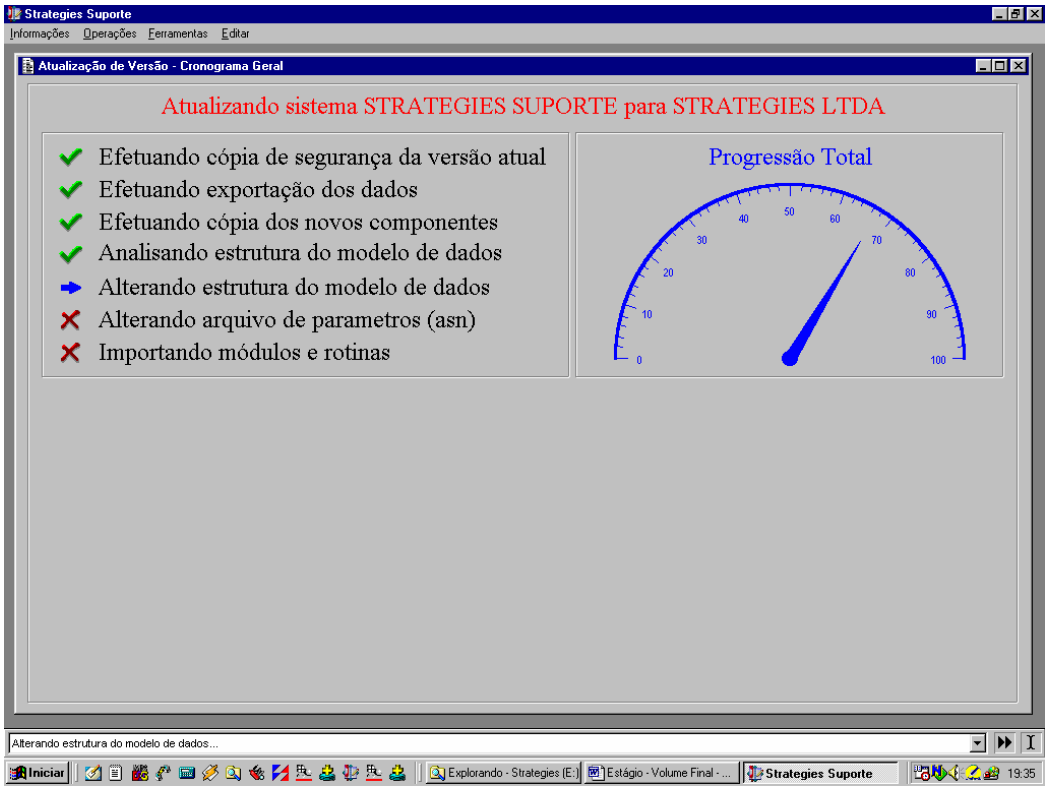
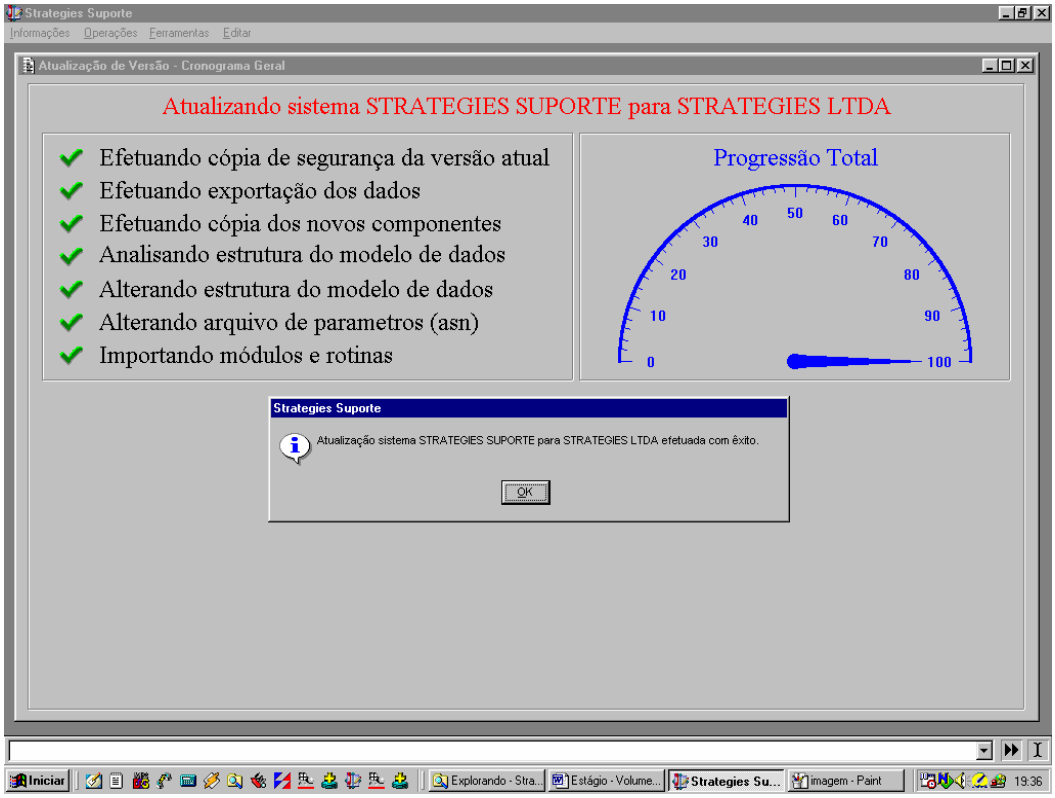


Figura 36 – Tela de status da atualização de versão após uma atualização.



3.4 RESULTADOS

Para que seja possível ter uma boa noção dos resultados obtidos, seria preciso que o protótipo desenvolvido fosse utilizado intensamente pelos clientes da empresa onde o estágio foi efetuado. Sendo que o mesmo foi somente testado e utilizado na empresa, com dados e informações fictícios, não foi possível verificar o seu comportamento utilizando uma grande base de dados. Porém, baseado em testes realizados na empresa, o protótipo agilizou e facilitou em muito os processos de fechamento de atualização de versão que eram feitos manualmente. Estatisticamente, eram despendidos aproximadamente duas horas para o fechamento de versão e para o processo de atualização de versão aproximadamente quatro horas. Acredita-se que com os processos automatizados esse tempo reduza para 45 minutos e 30 minutos respectivamente porque as etapas que despendiam a maior parte do tempo foram automatizadas. Exemplo: A etapa de levantamento de estrutura do modelo de dados da versão que está sendo fechada e etapa de alterações da estrutura do modelo de dados no ambiente do cliente.

Conforme levantamento de informações efetuado no departamento de suporte da empresa onde o estágio foi efetuado, observou-se que o processo de fechamento de versão contava com quatorze etapas. Com a automatização do processamento, o numero de etapas diminuiu para onze etapas. No processo de atualização de versão, a melhoria foi ainda mais significante, pois, no processo manual eram dez o número de etapas a serem efetuadas e com a automatização do processo, o número diminuiu pela metade.

4 CONCLUSÕES

Apesar do protótipo desenvolvido ainda ser sub-dimensionado em relação à necessidade da empresa (onde o estágio foi efetuado), pode-se dizer que os objetivos principais propostos neste trabalho foram atingidos. Os processos de fechamento e atualização de versão foram automatizados, não completamente, porém, etapas que consumiam muitos recursos foram automatizadas. Em função de limitações da ferramenta UNIFACE, como chamadas de sistemas operacionais e chamadas de executáveis que não fazem parte da aplicação, algumas etapas não foram automatizadas até o momento. Um exemplo disso foi a operação de copiar arquivos de um diretório para outro. Para que esse processo fosse automatizado, foi necessário desenvolver uma DLL utilizando a ferramenta DELPHI para efetuar tal tarefa.

Esse protótipo era uma grande necessidade da empresa (onde o estágio foi efetuado) justamente porque havia a necessidade de agilizar os processos internos e externos do departamento de suporte. A partir de agora, cuja implantação será feita nos clientes, poderão ser verificadas as limitações, as vantagens e as facilidades que o sistema proporcionará utilizando informações de uma grande base de dados. Com certeza o sistema sofrerá ajustes inevitáveis, mas com certeza se tornará uma nova ferramenta de auxílio para o departamento de suporte da empresa Strategies Ltda.

4.1 EXTENSÕES

Quando surgiu a idéia do desenvolvimento do sistema onde o estágio foi efetuado, uma idéia se destacou: “O foco do sistema é manter um funcionário do departamento de suporte 24 horas por dia no cliente”. Essa idéia despertou o interesse do que o sistema poderia oferecer. Em função disto sugere-se para novas versões:

- Controle e *backup* dos dados;
- Monitoração do banco de dados;
- Ferramenta de auxílio para manipulação de dados;

REFERÊNCIAS BIBLIOGRÁFICAS

CERÍCOLA, Vincent Oswald. **Oracle banco de dados relacional e distribuído** ferramentas para desenvolvimento**.** São Paulo: Makron Books do Brasil, 1995.

COMPUWARE. **Uniface V6.1 designer´s guide.** Netherlands: Compuware Corporation, 1995.

COMPUWARE. **Introduction to UNIFACE.** Netherlands: Compuware Corporation, 1996.

COMPUWARE. **Compuware Corporation**, São Paulo, dez. 2000. Disponível em: <http://www.compuware.com.br/Compuware_Corporate_Information.htm>. Acesso em: 22 mar. 2001.

HURSCH, Jack L; Carolyn J. Hursch. **Usando Oracle versão 6.0.** Rio de Janeiro: Campus, 1991.

KELLER, Robert. **Análise estruturada na prática.** São Paulo: McGraw-Hill, 1990.

KERN, Vinícius M. **Bancos de dados relacionais.** São Paulo: Érica, 1994.

POFFO, Ranieri Astrogildo. **Protótipo de uma aplicação com interface web baseado em "links" ODBC para acesso a banco de dados relacional**. 1996. 79 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

PRESSMAN, Roger S. **Engenharia de software.** São Paulo: Makron Books do Brasil, 1995.

SYSART. **UNIFACE,** São Paulo, jul. 1998. Disponível em: <http://www.sdweb.com.br/sysart/uniface.htm>. Acesso em: 26 fev. 2001.