UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

PROTÓTIPO DE FERRAMENTA DE AUXÍLIO À MIGRAÇÃO DE SISTEMAS COBOL PARA O AMBIENTE DELPHI

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

THOR ODEBRECHT

BLUMENAU, NOVEMBRO/1999

1999/2-38

PROTÓTIPO DE FERRAMENTA DE AUXÍLIO À MIGRAÇÃO DE SISTEMAS COBOL PARA O AMBIENTE DELPHI

THOR ODEBRECHT

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

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Wilson Pedro Carli — Orientador na FURB

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Wilson Pedro Carli

_____

Prof. Everaldo Artur Grahl

_____

Prof. Roberto Heinzle

DEDICATÓRIA

Aos meus pais, pela paciência e apoio durante a minha fase de estudos.

AGRADECIMENTOS

Aos meus pais, que possibilitaram meu ingresso na faculdade e sempre apoiaram meus estudos.

Ao professor Wilson Pedro Carli, que soube coordenar o trabalho com competência, apoiando sempre que necessário.

A minha namorada, Anarela Bernardi, que sempre me incentivou e deu forças para que este trabalho fosse realizado.

A todos que de forma direta ou indireta auxiliaram na realização deste trabalho.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS

ASCII - *American Standard Code II*

BAA - *Business Area Analysis*

BDE - *Borland Database Engine*

BNF - *Backus-Naur Form*

CARE - *Computer Aided Software Reengineering*

CASE - *Computer Aided Software Engineering*

DFD - Diagrama de Fluxo de Dados

HCI - *Human-Computer Interface*

MDB - *Microsoft DataBase*

TXT - Extensão de arquivo em formato *ASCII*

RESUMO

A migração de sistemas é uma das áreas da reengenharia de sistemas que abrange um conjunto de ações para tornar os sistemas atuais mais adequados às novas tecnologias e ambientes. Este trabalho tem como objetivo o estudo de conceitos e técnicas de reengenharia de sistemas e também a especificação e a implementação de um protótipo de ferramenta que auxilie a migração de sistemas em *COBOL* para o ambiente *DELPHI 4*, utilizando como banco de dados o *MS-ACCESS 97*.

ABSTRACT

Systems migration is a stage of the systems reengineering that embraces a set of actions to turn the current systems more adapted to the new technologies and environments. This work has as objective to study some concepts and techniques of systems reengineering and also the especification and the implementation of a tool prototype that aids to migrate COBOL systems to the DELPHI 4 environment, using as database the MS-ACCESS 97.

1 INTRODUÇÃO

1.1 ORIGEM

A necessidade de manutenção e melhoramento dos *softwares* e sistemas de informação se faz presente cada vez mais no cotidiano das organizações. O trabalho empregado na manutenção de sistemas antigos, que geralmente possuem uma interface baseada em caracteres, pode ser considerado um grande problema para os desenvolvedores de *software*, já que estas ferramentas possuem uma interação homem-computador (*HCI – Human-Computer Interface*) que pode ser considerada primitiva.

Para que os sistemas baseados em caracteres evoluam e comecem a usufruir de uma interface mais amigável com os usuários, eles devem sofrer mudanças, tais como, mudanças de ambiente operacional (sistema operacional, banco de dados e ferramentas afins) [CAV98]. Porém, o volume de sistemas que uma organização possue, geralmente é elevado, o que pode tornar inviável a manutenção de seus sistemas para que eles funcionem utilizando uma nova interface, pois o dinheiro e o tempo empregado para esta manutenção pode comprometer o funcionamento da organização.

Para tornar possível uma mudança de ambiente, uma organização ou empresa pode utilizar-se dos conceitos de reengenharia de sistemas, criando uma ferramenta que automatize o processo de migração de seus sistemas.

1.2 JUSTIFICATIVA

Este trabalho pretende auxiliar e simplificar o processo de migração de sistemas, desenvolvendo uma ferramenta que automatize e torne mais ágil o processo de conversão de arquivos e telas de sistemas.

Como pode ser visto em [BAU97] e em [CAV98], existe uma preocupação em se utilizar da tecnologia e de metodologias para facilitar a migração de sistemas e bancos de

dados, em conseqüência, notou-se a necessidade de ser gerada uma ferramentas para auxílio a migração de sistemas, tendo como enfoque a migração de sistemas COBOL *Microbase*, para o ambiente *DELPHI* 4, utilizando como banco de dados o *MS-ACCESS* 97.

A escolha do COBOL *MicroBase* foi baseada na grande quantidade de sistemas legados desenvolvidos com o auxílio desta ferramenta, que existem em funcionamento atualmente no mercado. O *DELPHI* 4 foi escolhido por ser um ambiente de desenvolvimento que se encontra em evolução no mercado.

1.3 OBJETIVO

O objetivo deste trabalho é demonstrar um estudo sobre técnicas de reengenharia de sistemas e implementar um protótipo de ferramenta de auxílio à migração de sistema em *COBOL* para o ambiente *DELPHI* 4.

1.4 ORGANIZAÇÃO DO TEXTO

O presente trabalho encontra-se dividido em 6 capítulos, conforme discriminado abaixo:

O capítulo 1, conforme já descrito, trata da origem, dos objetivos e justificativa do trabalho.

O capítulo 2 descreve a manutenção de *software*, explicando algumas de suas características e enquadrando a reengenharia em seu contexto.

O capítulo 3 aborda conceitos da reengenharia de sistemas e sua importância como tecnologia, apresentando cada área em que está dividida.

O capítulo 4 descreve simplificadamente os ambientes de programação que foram utilizados no trabalho.

O capítulo 5 trata sobre a ferramenta de migração, apresentando sua especificação, detalhes técnicos e seu funcionamento.

No capítulo 6 serão apresentadas as conclusões e algumas sugestões para a continuidade deste trabalho.

2 MANUTENÇÃO DE SOFTWARE

2.1 DEFINIÇÃO

De acordo com [PRE95], pode-se definir manutenção descrevendo quatro atividades que são levadas a efeito depois que um programa é liberado para uso. A primeira atividade de manutenção ocorre porque não é razoável presumir que a atividade de testes de *software* descobrirá todos os erros latentes num grande sistema de *software*. Durante o uso de qualquer programa grande, erros ocorrerão e serão relatados ao desenvolvedor. O processo que inclui o diagnóstico e a correção de um ou mais erros é denominado manutenção corretiva.

A segunda atividade que contribui para uma definição de manutenção ocorre por causa da rápida mudança que é encontrada em cada aspecto da computação. Novas gerações de hardware parecem ser anunciadas num ciclo de 24 meses, novos sistemas operacionais ou novos lançamentos de antigos sistemas aparecem regularmente, equipamentos periféricos e outros elementos de sistema são freqüentemente atualizados ou modificados. A vida útil dos aplicativos, por outro lado, pode facilmente ultrapassar 10 anos, vivendo mais do que o ambiente de sistema para o qual foram originalmente desenvolvidos. Por conseguinte, a manutenção adaptativa (uma atividade que modifica o *software* para que ele tenha uma interface adequada com o ambiente mutante) tanto é necessária como é um lugar-comum.

A terceira atividade que pode ser aplicada a uma definição de manutenção ocorre quando um pacote de *software* é bem-sucedido. À medida que o *software* é usado, recomendações de novas capacidades, de modificações em funções existentes e ampliações gerais são recebidas dos usuários. Para satisfazer os pedidos nessa categoria, a manutenção perfectiva é levada a efeito. Essa atividade é responsável pela maior parte de todo o esforço despendido em manutenção de *software*.

A quarta atividade de manutenção ocorre quando o *software* é modificado para melhorar a confiabilidade ou a manutenibilidade futura, ou para oferecer uma base melhor para futuras ampliações. Freqüentemente denominada manutenção preventiva, essa atividade é caracterizada pelas técnicas de engenharia reversa e reengenharia.

2.2 CARACTERÍSTICAS

Para que se possa entender as características da manutenção de *software*, [PRE95] considera o tema a partir de três pontos de vista :

a) As atividades exigidas para realizar a fase de manutenção e o impacto de uma abordagem (ou falta dela) de engenharia de *software* sobre a eficácia de tais atividades;
b) Os custos associados à fase de manutenção;
c) Os problemas que são freqüentemente encontrados quando a manutenção de *software* é levada a efeito.

2.3 MANUTENÇÃO ESTRUTURADA VERSUS NÃO-ESTRUTURADA

Se o único elemento disponível de uma configuração de *software* for o código-fonte, a atividade de manutenção inicia-se com uma penosa avaliação do código, freqüentemente complicada pela documentação interna ruim. Características sutis, tais como estrutura de programa, estrutura de dados globais, interfaces do sistema, desempenho e/ou restrições de projeto, são difíceis de ser verificadas e geralmente são mal-interpretadas. As ramificações das mudanças que finalmente são feitas no código são difíceis de ser avaliadas. Testes de regressão (a repetição de testes passados para garantir que as modificações não introduziram falhas no software anteriormente operacional) são impossíveis de ser realizados porque não existe nenhum registro de testes. Sendo assim, está sendo realizada a manutenção não-estruturada e se paga o preço (em esforço perdido e frustração humana) que acompanha um software que não tenha sido desenvolvido usando-se uma metodologia bem definida.

Se existir uma configuração de software completa, a tarefa inicia-se com uma avaliação da documentação de projeto. As características estruturais, de desempenho e de interface importantes do *software* são determinadas. O impacto das modificações ou correções exigidas é avaliado, e uma abordagem é planejada. O projeto é modificado e revisado. Um novo código-fonte é desenvolvido, testes de regressão são levados a efeito, usando-se as informações contidas na especificação de teste, e o software é liberado novamente. Essa

seqüência de eventos constitui a manutenção estruturada e ocorre como resultado de uma aplicação anterior de uma metodologia de engenharia de software. Não obstante a existência de uma configuração de *software* não garanta uma manutenção isenta de problemas, a quantidade de esforço desperdiçado é reduzida e a qualidade global de uma mudança ou correção é aumentada.[PRE95]

2.4 CUSTO DE MANUTENÇÃO

Segundo [PRE95], o custo da manutenção de *software* tem aumentado firmemente durante os últimos anos. Durante a década de 1970, a manutenção era responsável por um índice entre 35% e 40% do orçamento de *software* para uma organização de sistemas de informação. Esse valor pulou para aproximadamente 60% durante a década de 1980. Nesta década de 1990, muitas empresas podem gastar aproximadamente 80% de seus orçamentos de *software* em manutenção.

O custo da manutenção em termos de dinheiro é a preocupação mais óbvia. Porém, outros custos menos tangíveis podem, por fim, ser uma causa de preocupação maior. Um custo intangível da manutenção de *software* é a oportunidade de desenvolvimento que é postergada ou perdida porque os recursos disponíveis devem ser canalizados para tarefas de manutenção. Outros custos intangíveis incluem :

a) Insatisfação do cliente quando solicitações aparentemente legítimas de reparo ou modificação não podem ser encaminhadas oportunamente quanto ao tempo;
b) Redução da qualidade global do *software* como resultado de mudanças que introduzem erros latentes no *software* mantido;
c) Sublevações causadas durante os esforços de desenvolvimento quando o pessoal precisa ser “empurrado” para trabalhar numa tarefa de manutenção.

2.5 PROBLEMAS

De acordo com [PRE95], a maioria dos problemas associados à manutenção de *software* pode remeter-se a deficiências na maneira segundo qual o *software* foi planejado e desenvolvido. Aplica-se aqui a clássica situação do “pague agora ou pague muito mais depois”. A falta de controle e disciplina nas atividades de desenvolvimento da engenharia de *software* quase sempre traduz-se em problemas durante a manutenção do *software*.

Entre muitos problemas clássicos que podem estar associados à manutenção de *software* encontram-se os seguintes :

a) Freqüentemente é difícil ou impossível rastrear a evolução do *software* através de muitas versões ou lançamentos;
b) Geralmente é difícil ou impossível rastrear o processo através do qual o *software* foi criado;
c) Muitas vezes é excepcionalmente difícil entender o programa “de outra pessoa”. A dificuldade aumenta à medida que o número de elementos de uma configuração de *software* diminui;
d) A “outra pessoa” freqüentemente não está por perto para explicar. A mobilidade entre o pessoal da área de *software* é elevada. Não podemos contar com uma explicação pessoal do desenvolvedor de *software* quando a manutenção for necessária;
e) A documentação não existe ou é muito ruim. O reconhecimento de que o *software* deve ser documentado é um primeiro passo, mas a documentação deve ser compreensível e consistente com o código-fonte para ter algum valor;
f) A maioria dos *softwares* não é projetada para sofrer mudanças. A menos que um método de projeto acomode mudanças mediante conceitos tais como independência funcional ou classes de objetos, as modificações no software são difíceis e propensas a erros;
g) A manutenção não é vista como um trabalho muito glamouroso. Grande parte dessa percepção vem do elevado nível de frustração associado ao trabalho de manutenção.

Todos os problemas anteriormente descritos podem, em parte, ser atribuídos ao grande número de programas atualmente existente que foi desenvolvido sem levar em consideração a engenharia de *software*. Uma metodologia disciplinada não deve ser vista como uma panacéia. Porém, a engenharia de *software* oferece pelo menos soluções parciais para cada problema associado à manutenção.

3 REENGENHARIA DE SISTEMAS

Para um melhor entendimento sobre a reengenharia de sistemas é necessário que sejam esclarecidos seus conceitos, objetivos propostos pela abordagem e os métodos utilizados.

Algumas razões consideradas por [FUR95], como dificultadoras para se atingir o sucesso do processo de reengenharia:

a) resistência à mudança;
b) falta de metodologia de trabalho;
c) falta de infra-estrutura;
d) dificuldades para identificar ou definir o ambiente-alvo;
e) dificuldades para integração com outros sistemas;
f) dificuldades para identificar ou definir necessidades de negócios;
g) capacitação inadequada dos profissionais envolvidos;
h) falta de patrocinadores;
i) falta de ferramentas automatizadas;
j) falta de entendimento da organização com relação ao alcance dos benefícios da reengenharia.

Agora [FUR95], relaciona algumas regras básicas que permitem viabilizar processo de reengenharia de sistemas:

a) seja realístico;
b) estabeleça objetivos claros;
c) aprenda com quem já fez;
d) faça parcerias;
e) aplique ciência e não senso comum;
f) equilibre pesquisa e desenvolvimento com retorno de investimento;
g) mantenha a gerência informada;
h) tente conduzir em paralelo uma reengenharia de processo de negócio;
i) transição é um fator crítico de sucesso;
j) sucesso é a melhor estratégia de marketing.

Alguns objetivos propostos pela reengenharia de sistemas, segundo [FUR95] :

a) criar um inventário dos sistemas existentes;
b) aumentar a produtividade no processo de manutenção através da aplicação de ferramentas automatizadas (*CAREs* ou *CASEs*) e da captura de informações sobre sistemas existentes;
c) melhorar o tempo de resposta às solicitações de manutenção, com redução de custos e eliminação de erros;
d) reutilizar componentes de sistemas existentes para auxiliar no desenvolvimento de novas BAAs (*Business Area Analysis* ou Análise de Área de Negócio) na engenharia progressa;
e) melhorar o gerenciamento do ativo de sistemas e a compreensão desses sistemas para manutenção e teste;
f) possibilitar a conversão e migração tecnológica, facilitando a transição para novos ambientes operacionais;
g) reforçar a aderência a padrões estabelecidos;
h) proteger e estender a vida da aplicação.

3.1 DEFINIÇÃO E ÁREAS DA REENGENHARIA

A Reengenharia de Sistemas pode ser definida como um conjunto de técnicas e ferramentas orientadas à avaliação, reposicionamento e transformação de sistemas de informação existentes, com o objetivo de estender-lhes a vida útil e ao mesmo tempo, proporcionar-lhes uma melhor qualidade técnica e funcional. [FUR95].

Segundo [BER98]: “A reengenharia é uma transformação sistemática de um sistema existente em uma nova forma para realizar melhorias de qualidade na operação, capacidade de sistema, funcionalidade, performance, manutenção e suportabilidade num menor custo, tempo ou risco para o cliente”. Esta definição enfatiza que o foco da reengenharia está na melhoria de sistemas existentes com um maior retorno de investimento que poderia ser obtido através do esforço de um novo desenvolvimento.

A Reengenharia da Informação [FUR95], pode ser subdividida em cinco áreas principais:

a) reestruturação;
b) engenharia reversa;
c) reutilização;
d) engenharia de redesenvolvimento;
e) migração.

A definição dos processos e do escopo de cada área da reengenharia varia de acordo com cada autor. A estrutura acima apresentada é abordada por [FUR95] e utilizada neste estudo por ser apresentada de forma clara e simplificada.

3.1.1 REESTRUTURAÇÃO

De acordo com [FUR95], a reestruturação é a transformação de uma representação para outra no mesmo nível de abstração, preservando o comportamento externo do sistema em questão (funcionalidade e semântica). A transformação da reestruturação é sempre na aparência. Como por exemplo, alteração de código para melhorar a sua estrutura.

O termo reestruturação ficou popular a partir da transformação de código de programas de forma não-estruturada (espaguete) para a forma estruturada. Por tanto, o processo de reestruturação trata do processo de padronização de nomes de dados e estruturação de programas sem alteração do seu funcionamento. Existem basicamente dois tipos de reestruturação:

a) reestruturação de código fonte;
b) reestruturação de dados.

A reestruturação de código fonte é o processo de análise dos fluxos de controle e lógica de programação com geração de uma versão estruturada de código fonte original sem alteração de sua funcionalidade. Assim, um programa seria candidato à reestruturação quando há uma má programação (excessiva quantidade de condicionais encadeadas, muitas variáveis globais, código automodificável, múltiplas possibilidades de saídas, módulos gigantescos), ao

se encontrar dificuldades ou impossibilidade para manter ou testar, alta taxa de erro, necessidade de profissionais específicos, sistemas estrategicamente importantes, caros e ou freqüentemente alteráveis.

Já a reestruturação de dados visa eliminar redundâncias de nomes para a mesma estrutura lógica de dados, adotando-se um nome padrão para os elementos em nível de análise de área de negócio. A partir da reestruturação de dados pode-se obter os seguintes benefícios:

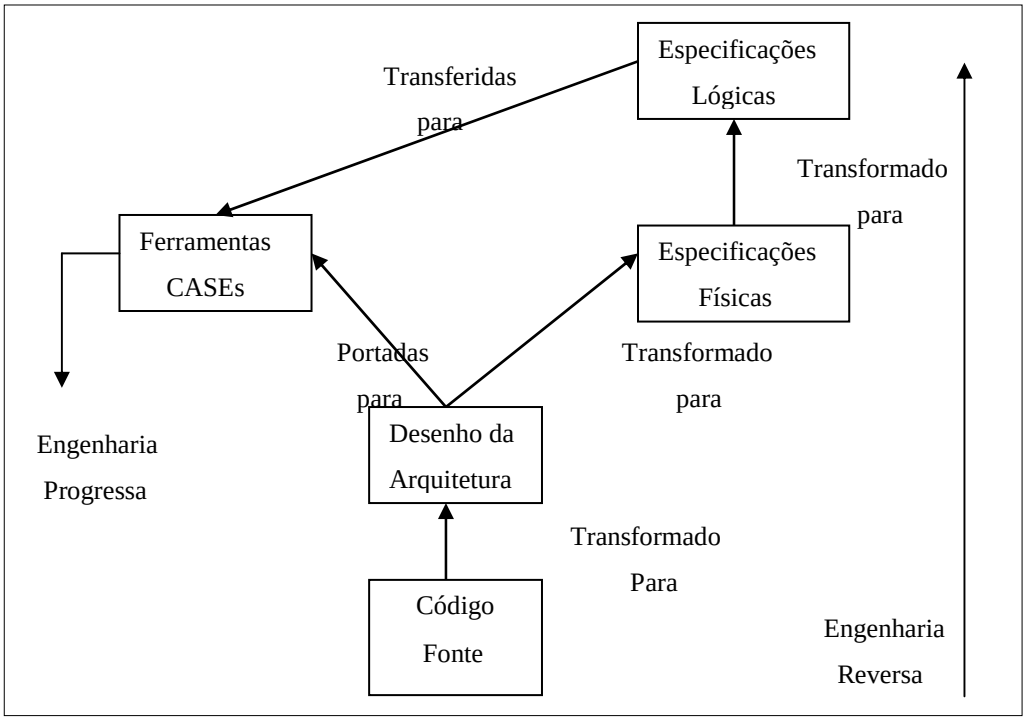
a) criação de uma base de dados normalizada e estável;
b) melhoria da compreensão das interfaces de dados entre sistemas;
c) integração dos sistemas de informação;
d) consolidação de funções afins no tratamento de dados;
e) melhoria do gerenciamento do ambiente sistêmico frente as solicitações de alteração;
f) reposicionamento dos dados para permitir a aplicação de novos enfoques técnicos;
g) utilização de nomes de dados padronizados para novos desenvolvimentos;
h) melhoria da consistência e do significado dos dados.

As atividades de reestruturação de dados poderão ser assistidas por ferramentas automatizadas, disponíveis em vários casos, embora a racionalização e a padronização de nomes de dados não possam ser feitas de forma totalmente automatizada, pois, o entendimento do significado dos dados normalmente requer o julgamento humano. Outra consideração importante acerca dos dados é que estes representam atualmente um dos principais ativos das organizações. Qualquer substituição de sistemas deve prever a transição dos velhos (independentemente de quão arcaico possa ser) para os novos arquivos, garantindo-se dessa forma a continuidade dos dados.

3.1.2 ENGENHARIA REVERSA

Para [FUR95], é o processo de derivar as especificações lógicas dos componentes do sistema a partir de sua descrição física, com o auxílio de ferramentas automatizadas. A engenharia reversa trata dados e processos, conforme pode-se verificar através da figura 3.1, apesar de que ultimamente as ferramentas têm sido mais efetivas para dados.

Figura 3.1 – Passos da Engenharia Reversa .



Fonte : [FUR95].

Como benefícios da engenharia reversa pode-se destacar:

a) melhorar a compreensão dos sistemas existentes;
b) fornecimento automático de documentação atualizada dos sistemas existentes;
c) fornecimento de um meio eficiente para análise de dados e processos;
d) aceleração do processo de manutenção de sistemas;
e) agilização da conversão de sistemas e diminuição dos esforços de migração;
f) possibilidade de manutenção de sistemas em nível de desenho;
g) posicionamento dos sistemas para serem suportados por ferramentas CASEs;
h) recuperação do conhecimento sobre os sistemas existentes;
i) nivelamento do investimento feito nos sistemas existentes;

j) possibilidade de aplicação de métodos *top-down* de desenvolvimento para os sistemas existentes (*Forward Engineering*).

Tipicamente, a entrada para a engenharia reversa são o código-fonte, o dicionário de dados e os DDLs (*Definition Data Language* ou Linguagem de Definição de Dados). O resultado, por sua vez, dependerá diretamente da ferramenta utilizada, porém, alguns desses produtos deveriam ser gerados:

a) pelo lado dos dados:
   - desenho de banco de dados físico;
   - estrutura física de dados;
   - diagrama de entidades e relacionamentos;
   - modelo de dados normalizado.

b) pelo lado dos processos:
   - especificações do desenho físico;
   - diagrama de estrutura.

Engenharia reversa não significa a utilização de ferramentas automatizadas que fazem tudo por si próprias através de um simples apertar de botões. Apesar de existirem boas ferramentas em quase todas as faixas de classificação, a reengenharia continua sendo uma atividade trabalhosa.

São candidatas à engenharia reversa as aplicações que não foram projetadas utilizando-se técnicas de modelagem funcional ou de dados, ou aplicações nas quais a documentação original não foi atualizada juntamente com o código-fonte. A engenharia reversa também pode ser empregada para apresentar modelos lógicos que representam o que a aplicação faz (em vez de como faz), através do uso de ferramentas que permitem sumarizar milhares de linhas de código numa simples imagem gráfica.

3.1.3 REUTILIZAÇÃO

Segundo [FUR95], a redundância é uma ocorrência comum nos sistemas existentes, principalmente naqueles projetados sem o auxílio de técnicas de modelagem. Redundância é o oposto da reutilização. Reutilização é o múltiplo uso de uma dada função da aplicação, por outro lado, redundância é a implementação múltipla, repetida desta mesma função. A redundância resulta em maiores custos de manutenção e teste. Quando as técnicas de desenvolvimento são eficientemente utilizadas, a redundância é minimizada, e até mesmo eliminada, na transição dos modelos lógicos de análise para o modelo físico de desenho. A reutilização apresenta benefícios potenciais que podem promover mudanças revolucionárias na prática da construção de *software*. É uma mudança fundamental de paradigma no campo do *software* onde atualmente é uma exceção, e não a regra, o desenvolvimento a partir da estaca zero.

Para a reutilização de componentes de *software*, normalmente é criado um repositório de componentes reutilizáveis que, de forma isolada ou combinada, podem ser utilizados na construção de novos sistemas. Entre os benefícios da reutilização de componentes de *software*, podem ser destacados:

a) aceleração e simplificação do processo de desenvolvimento;
b) melhoria da confiabilidade e da qualidade do software;
c) redução de custos.

O custo em se descobrir componentes reutilizáveis para se criar e manter um repositório com suas informações não poderá ser justificado economicamente, a menos que os componentes sejam utilizados muitas vezes, pelo menos três vezes, para se atingir um ponto mínimo de *payback*. Quatro aspectos são fundamentais para a implementação desse enfoque:

a) a criação de um repositório contendo os componentes de *software* para reutilização;
b) inclusão de atividades relativas ao reaproveitamento de software na metodologia de desenvolvimento de sistemas estabelecida;
c) seleção de ferramentas automatizadas para suportar o desenvolvimento baseado na reutilização de componentes de *software*;

d) compromisso gerencial e aceitação dos profissionais de desenvolvimento de sistemas em se utilizar dessa prática.

3.1.4 ENGENHARIA DE REDESENVOLVIMENTO

Do ponto de vista de [FUR95], é a forma mais sofisticada da Reengenharia da Informação, pois, pode englobar características da reestruturação, engenharia reversa, reutilização e migração, entretanto, o aspecto mais importante é o fato de possibilitar a mudança de funcionalidade no sistema. A operação típica é a de se elevar para níveis de desenho e análise o ativo físico de programas e dados, e promover ajustes nas funções do sistema e/ou no modelo de dados que sustentam o processamento das informações. Geralmente, os programas são traduzidos para diagramas de estrutura contendo módulos com o código-fonte e os dados em diagramas de entidades e relacionamentos. As interfaces com o usuário (telas e relatórios) também são recuperados e posicionados para ajustes.

Uma vez tendo sido incorporados novos requerimentos nesses modelos, gera-se em seguida o aplicativo para a mesma plataforma de *hardware* e *software* básico ou para a uma nova plataforma-alvo preestabelecida. A engenharia do redesenvolvimento viabiliza a utilização da tecnologia *CASE* (tipicamente orientada para engenharia progressa) para o tratamento de sistemas existentes.

3.1.5 MIGRAÇÃO

De acordo com [FUR95], é o processo de mudança de plataforma tecnológica a partir de um ativo de sistemas existente. A migração é feita através de *softwares* específicos (*CARES*) de conversão e tradução de código-fonte, e não envolve mudanças na funcionalidade do sistema, apenas na incorporação de características operacionais da plataforma-alvo. Os benefícios da migração parecem óbvios, uma vez que se preserva os esforços de desenvolvimento e integração de sistemas numa época de mudanças rápidas nas premissas tecnológicas.

Às vezes, o processo de migração por si não tem a capacidade de tirar total proveito das características operacionais e de desempenho da plataforma-alvo. Principalmente se houver uma mudança radical de plataforma de *hardware* e *software* básico. Mesmo assim, permite aos sistemas migrados que executem normalmente no novo ambiente, tornando o processo de migração transparente ao usuário.

Os esforços de migração são normalmente extenuantes se feitos manualmente, podendo levar meses ou anos para se concluir um ciclo. Os profissionais tendem a permanecer alocados em caráter *full-time* no processo de migração, sem a possibilidade de oferecer novas funções de sistemas à organização. Além disso, a conversão manual poderá ser tediosa e, em alguns casos profissionalmente frustrante tanto para técnicos como para usuários. Já a migração assistida por ferramentas de conversão e tradução oferece uma redução de tempo e esforço podendo encurtar de forma significativa o período de migração. Porém, ainda não estão disponíveis em larga quantidade no mercado, ferramentas de migração para suportar todas as plataformas tecnológicas. Do ponto de vista comercial, existe um maior interesse em disponibilizar ferramentas para as plataformas mais utilizadas, que possam teoricamente, oferecer um melhor retorno de investimento. Quanto às plataformas menos empregadas, os esforços de migração têm grande probabilidade de continuarem sendo feitos manualmente.

O foco deste trabalho é a migração, e o protótipo implementado no mesmo é baseado nos fundamentos desta técnica.

3.2 METODOLOGIAS PARA REENGENHARIA

Uma metodologia adequada para o processo de reengenharia pode determinar o sucesso ou não do projeto de reengenharia de uma organização. É através da metodologia empregada que serão previstas as várias fases de desenvolvimento, seus custos e particularidades, bem como, as estratégias a serem empregadas. A metodologia para reengenharia de sistemas possui vários tópicos padrões para diferentes projetos de reengenharia, porém, além destes procedimentos, a metodologia deve ser capaz de prever e fornecer subsídios para as situações particulares de cada projeto, tornando-se específica para cada situação. [BAU97]

A escolha dos métodos deve levar em consideração como comenta [FUR95], a disponibilidade de ferramentas de suporte (apesar de que a metodologia é algo independente de *hardware* e *software*), o risco de se adotar tal enfoque e o nível de esforço, tempo e custos associados. Faz parte disto, também a necessidade de se obter experiências com consultores ou empresas especializadas no mercado.

Desta forma, deve-se considerar alguns aspectos importantes quanto ao uso de metodologia para a reengenharia:

a) estabelecer pontos comuns de referência para as diversas atividades da reengenharia;
b) criar uma estrutura integrada de trabalho que define as intersecções e o fluxo existente entre tecnologias aparentemente distintas;
c) definir claramente os benefícios de curto, médio e longo prazo para cada atividade da reengenharia;
d) servir de guia para a reconciliação das atividades de reengenharia com as de engenharia;
e) formar a base para uma estratégia customizada de redesenvolvimento.

3.2.1 METODOLOGIA DE FURLAN

Como o objetivo principal deste trabalho é o desenvolvimento de uma ferramenta, só é apresentada uma metodologia para reengenharia, para que ao menos se possa obter uma idéia das partes que a integram. Para isso foi escolhida a metodologia de Furlan, que é descrita a seguir.

A metodologia proposta por [FUR95] é dividida em três fases distintas, abordando o ciclo completo da reengenharia e estabelecendo que cada atividade da reengenharia tenha valor por si própria para que se possa obter a análise custo/benefício por atividade.

A primeira fase é a de Análise e Inventário, que consiste basicamente num levantamento técnico, funcional e de nível de flexibilidade. Normalmente é composta pelas atividades abaixo :

a) carga do repositório com base nos sistemas existentes (engenharia reversa);
b) modelagem de dados *bottom-up*;
c) modelagem funcional *bottom-up*;
d) análise de código fonte (métricas);
e) análise de portfólio de aplicações;
f) estudo de viabilidade e determinação de estratégias para a reengenharia.

A segunda fase é a de Reposicionamento, que consiste na identificação de regras, reestruturação e documentação. Normalmente apresenta as seguintes atividades:

a) externalização de regras de negócio;
b) reestruturação de código fonte;
c) racionalização, padronização, reutilização e remoção de dados e processos;
d) redocumentação.

A terceira fase é a da Transformação, que consiste na transformação do velho sistema para o novo. É composto pelas seguintes atividades :

a) conversão de interfaces;
b) conversão/*upgrade* de linguagem de programação;
c) separação/reagrupamento de código fonte;
d) conversão de banco de dados;
e) migração de plataforma de *hardware* e *software*.

4 AMBIENTES DE PROGRAMAÇÃO

Os ambientes de programação utilizados neste trabalho, são apresentados abaixo, com o objetivo de esclarecer em que ambiente a ferramenta proposta se enquadra.

A ferramenta foi desenvolvida no ambiente visual *Delphi* 4, utilizando o banco de dados *MS-ACCESS* 97. Esta ferramenta foi desenvolvida basicamente para auxiliar na migração de sistemas escritos na linguagem de programação *COBOL MicroBase*.

4.1 COBOL

De acordo com [CAR93], a linguagem *COBOL* é uma das mais utilizadas, ainda mais se levar em consideração os fenômenos “Downsizing” e “Rightsizing” que tratam da correta migração de aplicações desenvolvidas e executadas em ambientes de grande porte para ambientes de microcomputadores, cujo custo de utilização é tremendamente menor.

O *COBOL* utilizado como base deste trabalho é o *COBOL MicroBase* versão 2.02, e é destinado ao uso em equipamentos compatíveis com IBM-PC, com sistema operacional MSDOS ou compatível.

4.2 DELPHI

É um ambiente visual de programação, que utiliza a linguagem de programação Object Pascal, e foi desenvolvida pela empresa Inprise, a antiga Borland. A versão utilizada neste trabalho foi a 4 – *Client/Server Suite*.

Segundo [CAN96], existem muitos ambientes de programação com os quais se pode trabalhar, mas o *Delphi* é surpreendente por uma série de motivos. As dez razões principais para se utilizar o *Delphi* são :

a) compiladores Borland Pascal e C++ anteriores;

b) os componentes e ferramentas de terceiros;
c) o editor, o depurador, o browser e outras ferramentas;
d) a disponibilidade de códigos-fonte das bibliotecas;
e) a abordagem baseada em formulários e orientada a objeto;
f) um compilador rápido;
g) o suporte a banco de dados;
h) a estreita integração com a programação Windows;
i) tecnologia de componentes do *DELPHI*;
j) a linguagem *Object* Pascal.

Como explica [CAN96], os aplicativos de bancos de dados do *Delphi* não têm acesso direto às fontes de dados que eles referenciam. O *Delphi* faz interface com o *Borland Database Engine* (BDE), que tem acesso direto a diversas fontes de dados, incluindo *dBASE*, *Paradox* e na versão 5.01 do BDE, o *MS-ACCESS* (por meio de drivers apropriados).

Para se fazer o acesso a um banco de dados em *MS-ACCESS 97* pelo *Delphi*, deve-se utilizar a ferramenta *BDE Administrator*, gerando um alias e configurando o BDE conforme exemplo mostrado nas figuras 4.1 e 4.2.

Figura 4.1 – Exemplo de alias configurado para acesso ao *MS-ACCESS 97*.

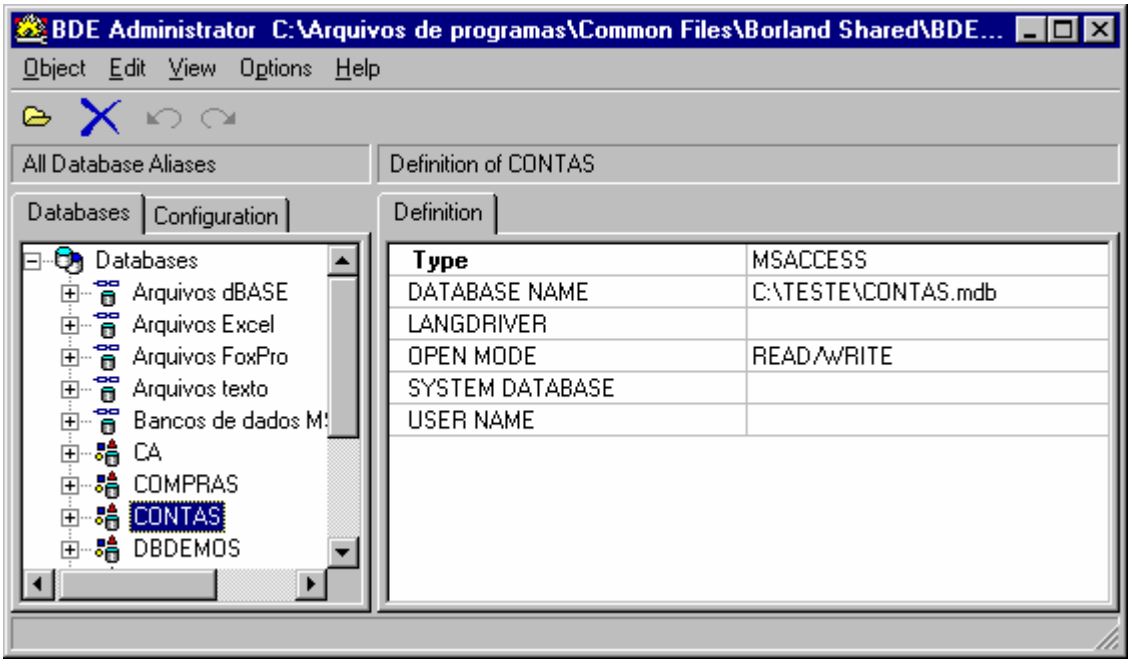
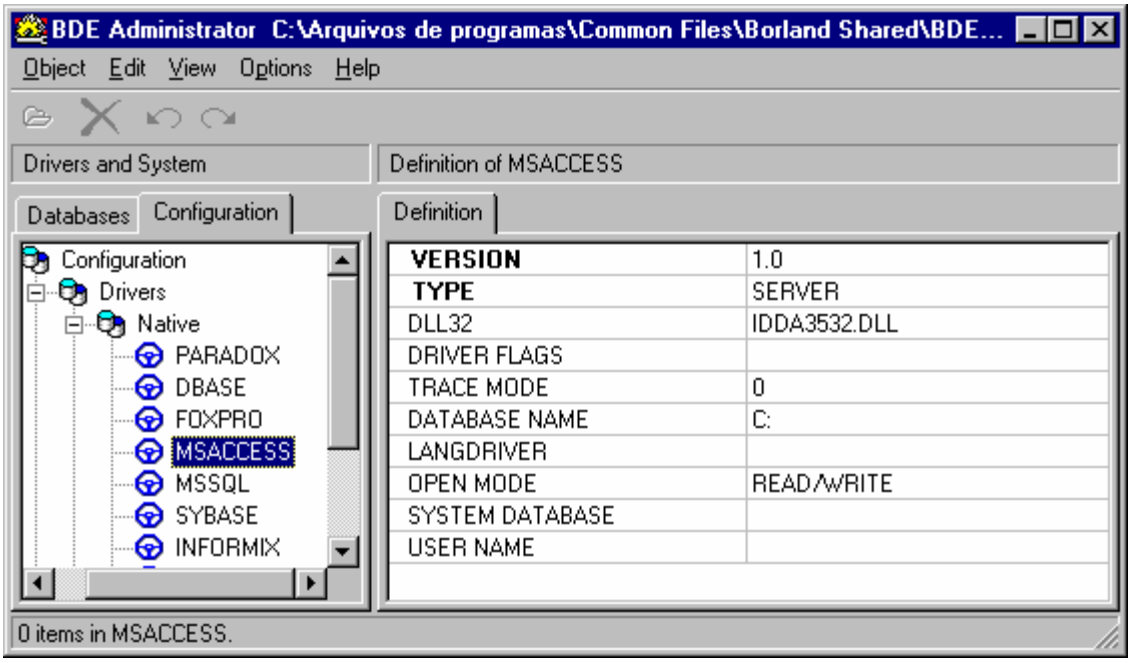


Figura 4.2 – Exemplo de como configurar acesso ao driver *MS-ACCESS 97*.



4.3 MS-ACCESS 97

O *MS-ACCESS 97*, é parte integrante do *Microsoft Office 97*, e serve para gerenciar vários bancos de dados. Pode ser utilizado em microcomputadores que possuam o sistema operacional *Windows 95*, *Windows 98* ou ainda *Windows NT 4.0*.

O *MS-ACCESS 97*, de acordo com [PER97], é um programa de banco de dados que lhe permite :

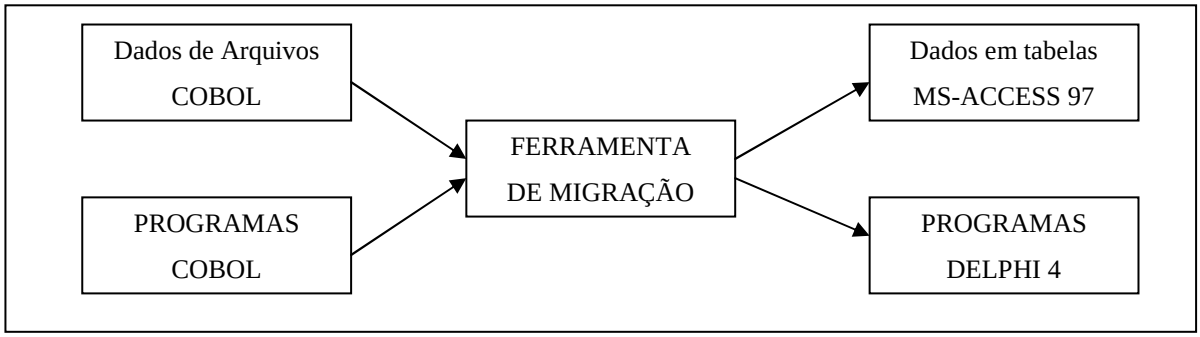
a) armazenar uma quantidade de informações quase ilimitada;
b) organizar informações de uma maneira que faça sentido para o seu modo de trabalho;
c) recuperar informações com base em critérios de seleção especificados pelo usuário;
d) criar formulários que facilitem a entrada de dados;
e) gerar relatórios significativos e complexos que possam combinar dados, texto, gráficos e até mesmo som.

5 FERRAMENTA DE AUXÍLIO À MIGRAÇÃO DE SISTEMAS

Depois de terem sido apresentados os aspectos referentes aos problemas com a manutenção de *software* e as soluções que a reengenharia de sistemas pode proporcionar para minimizar tais problemas, percebeu-se a necessidade do desenvolvimento de uma ferramenta de apoio à migração de sistemas.

Neste caso, a área de atuação da ferramenta projetada se aplica a todas as empresas de desenvolvimento de *software* que possuem no mercado sistemas desenvolvidos na linguagem *COBOL MicroBase*, e que pretendem migrar seus sistemas para o ambiente de desenvolvimento visual *DELPHI 4*, utilizando o banco de dados *MS-ACCESS 97*.

Figura 5.1 – Funcionamento da ferramenta.



A figura 5.1 ilustra as entradas do sistema e em seguida as suas saídas, para que se possa ter uma noção mais abrangente da idéia principal do funcionamento da ferramenta. A seguir serão apresentados os aspectos técnicos referentes à ferramenta, para que se possa entender melhor seu funcionamento, abordando desde a sua especificação até como o usuário deve proceder para fazer uso da mesma.

5.1 ESPECIFICAÇÃO

Para a análise e definição desta ferramenta foi utilizada a Análise Estruturada de Sistemas, que segundo [GAN91], compõe-se de um conjunto de técnicas e ferramentas, em constante evolução, nascido do sucesso da programação e do projeto estruturados. Seu conceito fundamental é a construção de um modelo lógico (não físico) de um sistema, utilizando técnicas gráficas capazes de levar usuários, analistas e projetistas a formarem um quadro claro e geral do sistema e de como suas partes se encaixam para atender às necessidades daqueles que dele precisam. Também foi utilizada a técnica de fluxogramação, para representar a lógica do funcionamento da ferramenta.

Para um melhor entendimento sobre a descrição de um arquivo *COBOL* dentro de um programa fonte, foi realizada a especificação da FD de um arquivo, utilizando a notação BNF (Backus-Naur Form).

5.1.1 SYSTEM ARCHITECT

Para facilitar a fase de análise da ferramenta de migração, foi utilizada a ferramenta CASE *System Architect*, que permite implementar graficamente alguns dos modelos lógicos necessários para a especificação de um sistema. É uma ferramenta compatível com o sistema operacional Windows, que pode ser encontrada no meio acadêmico e empresarial.

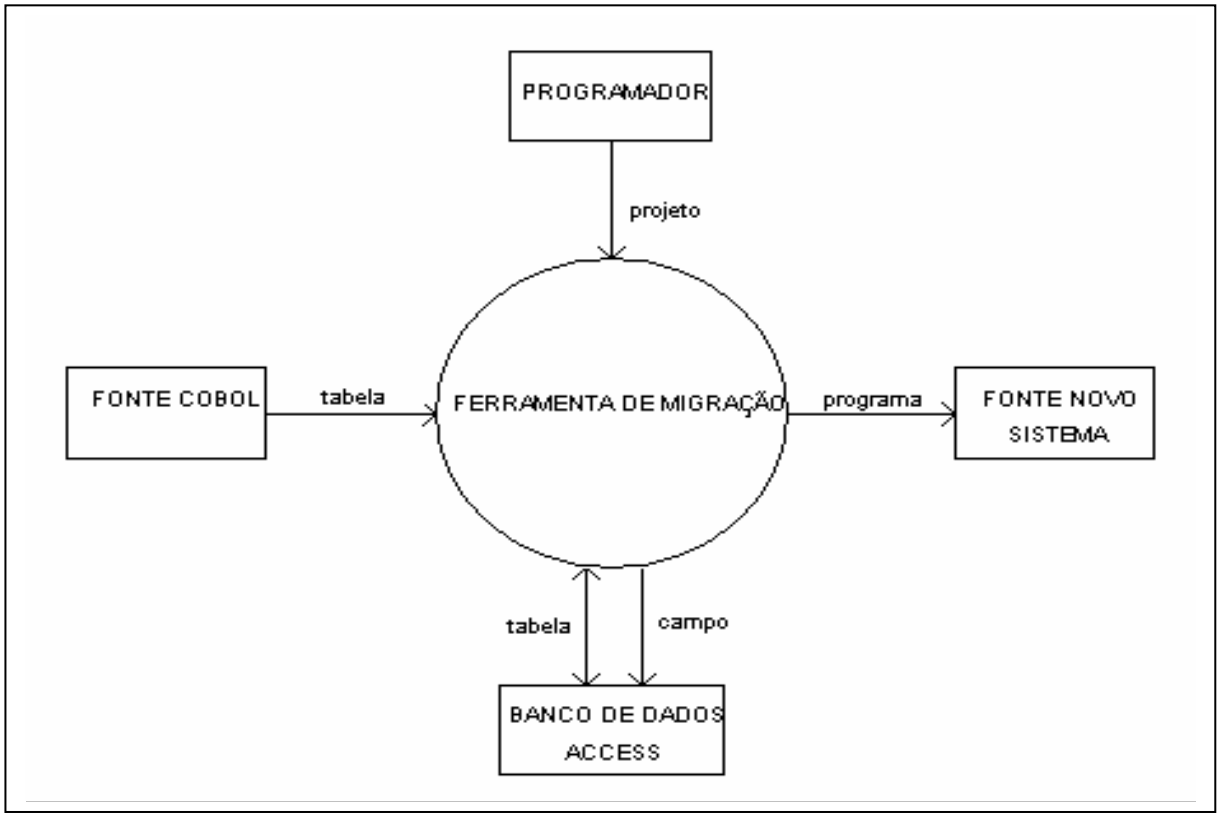
Conforme encontrado em [POP99], a ferramenta *System Architect* têm como características uma boa funcionalidade e grande flexibilidade, pois permite gerar e alterar graficamente vários tipos existentes de diagramas, bem como, gerar documentação sobre os sistemas e também a própria base de dados para o novo sistema.

5.1.2 DIAGRAMA DE CONTEXTO

O diagrama de contexto, de acordo com [YOU90], é um caso especial do diagrama de fluxo de dados, no qual uma única bolha representa o sistema inteiro. Esta ferramenta da análise estruturada realça diversas características importantes do sistema:

a) As pessoas, organizações ou sistemas com os quais nosso sistema se comunica são conhecidos como terminadores;
b) Os dados que nosso sistema recebe do mundo exterior e que devem ser processados de alguma maneira;
c) Os dados produzidos pelo nosso e enviados para o mundo exterior;
d) Os depósitos de dados que são compartilhados por nosso sistema e terminadores. Esses depósitos de dados ou são criados na parte externa do sistema e usados por nosso sistema ou são criados por nosso sistema e usados externamente ao sistema;
e) Os limites entre o nosso sistema e o resto do mundo.

Figura 5.2 – Diagrama de Contexto da ferramenta.



Na figura 5.2 é apresentado o diagrama de contexto referente à ferramenta de auxílio à migração. Através dele pode-se observar que o 'PROGRAMADOR' fornece à ferramenta qual o projeto que deve ser utilizado, com essa informação a ferramenta extrai, da entidade externa 'FONTE SISTEMA COBOL', os dados da tabela desejada para a migração. Com a tabela extraída anteriormente, a ferramenta gera uma tabela dentro da entidade externa 'BANCO DE DADOS ACCESS', através da definição dos campos da mesma. Por final a ferramenta possibilita a geração de um programa, que forma a entidade externa 'FONTE DELPHI NOVO SISTEMA'.

5.1.3 FLUXOGRAMA

Segundo [MAR91], um fluxograma de um sistema é fundamentalmente a documentação de operações, usada para mostrar ao operador de computador como processar um sistema ou programa.

Na figura 5.3, é apresentado o fluxograma da ferramenta, que permite visualizar graficamente a lógica de funcionamento do sistema. Nela pode-se observar a ordem das operações utilizadas para possibilitar o funcionamento da ferramenta. A seguir é descrita a utilização passo-a-passo da ferramenta, conforme o fluxograma.

Para iniciar a utilização da ferramenta o usuário deve entrar com um projeto, se for um novo projeto, deve gerar o mesmo e gerar o arquivo ACCESS correspondente a ele. Se não for um projeto novo então continua normalmente.

Em seguida deve manter os dados referentes ao projeto, e abri-lo para utilização.

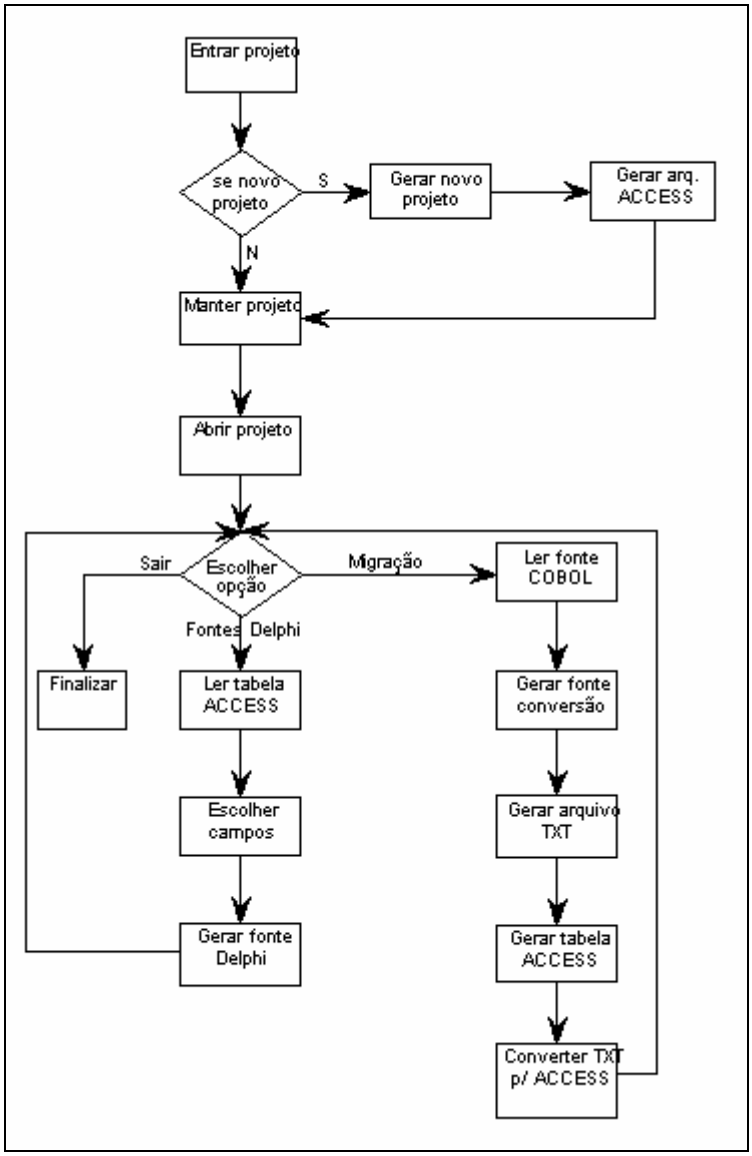
Após aberto o projeto, então pode escolher uma das três opções do menu : 'Migração', 'Fontes Delphi' e 'Sair'.

Escolhendo 'Sair', o usuário finaliza a execução da ferramenta.

Escolhendo ‘Migração’, o usuário permite à ferramenta ler um fonte *COBOL*, depois, gerar um fonte de conversão *COBOL*, que por final gera um arquivo TXT. Em seguida a ferramenta pode gerar a tabela respectiva no banco de dados *MS-ACCESS 97*, e converter o arquivo TXT para a tabela em *ACCESS*. Após efetuar estas ações, retorna para a escolha das opções do menu principal da ferramenta.

Escolhendo ‘Fontes Delphi’, o usuário permite à ferramenta ler uma tabelas *ACCESS*. Em seguida escolhe os campos da tabela que deverão ser utilizados no novo sistema, e possibilita a ferramenta gerar os fontes do novo sistema em *Delphi*. Após efetuar estas ações, retorna para a escolha das opções do menu principal da ferramenta.

Figura 5.3 – Fluxograma da ferramenta.



5.1.4 BNF

De acordo com [AHO95], uma linguagem de programação pode ser definida pela descrição da aparência de seus programas (a sintaxe da linguagem) e do que os mesmos significam (a semântica da linguagem). Para especificar a sintaxe de uma linguagem, é apresentada uma notação, chamada de gramática livre de contexto ou BNF (Forma de Backus-Naur). Além de especificar a sintaxe da linguagem, uma gramática livre de contexto pode ser utilizada como auxílio para guiar a tradução de programas. Uma gramática descreve naturalmente a estrutura hierárquica de muitas construções das linguagens de programação.

Para tornar possível a implementação de um protótipo de ferramenta de migração de arquivos *COBOL*, deve-se possuir um entendimento detalhado sobre a sintaxe de representação das informações referentes aos campos do arquivo, dentro do programa fonte. Para que então possa ser feito com que a ferramenta traduza as informações referentes aos campos, permitindo assim utilizá-las para possibilitar a migração dos dados para uma nova base em *ACCESS*.

Como a sintaxe de um arquivo *COBOL* se encontra representada dentro da cláusula FD de um programa fonte *COBOL*, convém demonstrar aqui a especificação desta cláusula utilizando a notação BNF.

Deve ser observado que na especificação não foram considerados os tipos de campos do *COBOL* que possuem as cláusulas *REDEFINES* e *OCCURS*, pois como a ferramenta neste trabalho é tratada à nível de protótipo, não suporta estas construções da linguagem *COBOL*. A seguir o quadro 1 expressa o modelo utilizado neste trabalho da especificação da FD de um programa fonte *COBOL* utilizando a notação BNF :

Quadro 1 – Modelo em notação BNF de uma FD do *COBOL*.

```
<registro>        ::= <fd> <campos>
<fd>              ::= FD <id>  <label> <value> <id>
<label>           ::= LABEL RECORD IS STANDARD
<value>           ::= VALUE OF FILE-ID IS
<campos>          ::= <nivel> <id> PIC <tipo> . <campos>∗ |
                      <nivel> <id> . <campos>∗ | ∈
<nivel>           ::= <num> <num>
<tipo>            ::= <numerico> | <alfanumerico>
<numerico>        ::= S <num,9> (  ‘(‘ <num>∗  ‘)’ ) ( V <num,9>* ) ( COMP-3 ) |
                      <num,9> (  ‘(‘ <num>∗  ‘)’ ) ( V <num,9>* ) |
                       <num,9>∗ ( V <num,9>* )
<alfanumerico>    ::= X* |
                      X ( ‘(’ <num> <num> ‘)’ )
<id>              ::= ( <alfabeto> * ) ( <num>* ) ( <hifen>* )
<alfabeto>        ::= ( a | b | .. | z ) ( A | B | .. | Z )
<num>             ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<hifen>           ::= -
```

5.1.5 ANÁLISE SINTÁTICA

Após realizada a especificação em BNF, da FD de um arquivo *COBOL*, pode-se dizer que foi construída uma gramática para a escrita desta seção do programa fonte *COBOL*. Para complementar a pesquisa realizada neste trabalho, foi feito um estudo de como esta especificação pode ajudar no desenvolvimento de uma ferramenta de tradução de programa fonte, conhecido como analisador sintático, ou *parser*. Para um melhor entendimento sobre o assunto, deve-se possuir uma noção do que é análise sintática, e qual o papel de um *parser*.

Segundo [AHO95], a análise sintática, envolve o agrupamento dos *tokens* (sequencia de caracteres tendo um significado coletivo) do programa fonte em frases gramaticais, que são usadas pelo compilador, a fim de sintetizar a saída.

De acordo com [AHO95], a sintaxe das construções de uma linguagem de programação pode ser descrita pelas gramáticas livres de contexto ou pela notação BNF. As gramáticas oferecem vantagens significativas tanto para os projetistas como para os escritores de compiladores :

a) uma gramática oferece, para uma linguagem de programação uma especificação sintática precisa e fácil de entender;
b) para certas classes gramaticais, podemos construir automaticamente um analisador sintático (*parser*) que determine se um programa-fonte está sintaticamente bem formado;
c) uma gramática propriamente projetada implica uma estrutura de linguagem de programação útil à tradução correta de programas-fonte em código-objeto e também à detecção de erros;
d) as linguagens evoluíram ao longo de um certo período de tempo, adquirindo novas construções e realizando tarefas adicionais. Essas novas construções podem ser mais facilmente incluídas quando existe uma implementação baseada numa descrição gramatical da linguagem.

O *parser* obtém uma cadeia de *tokens* proveniente do analisador léxico e verifica se a mesma pode ser gerada pela gramática da linguagem-fonte, devendo também relatar erros de sintaxe de uma forma inteligível.

Existem três tipos gerais de analisadores sintáticos, porém os mais utilizados nos compiladores são classificados como *top-dowm* ou *bottom-up*, como indicado por seus nomes, os analisadores sintáticos *top-dowm* constroem árvores do topo (raiz) para o fundo (folhas), enquanto que os *bottom-up* começam pelas folhas e trabalham árvore acima até a raiz.

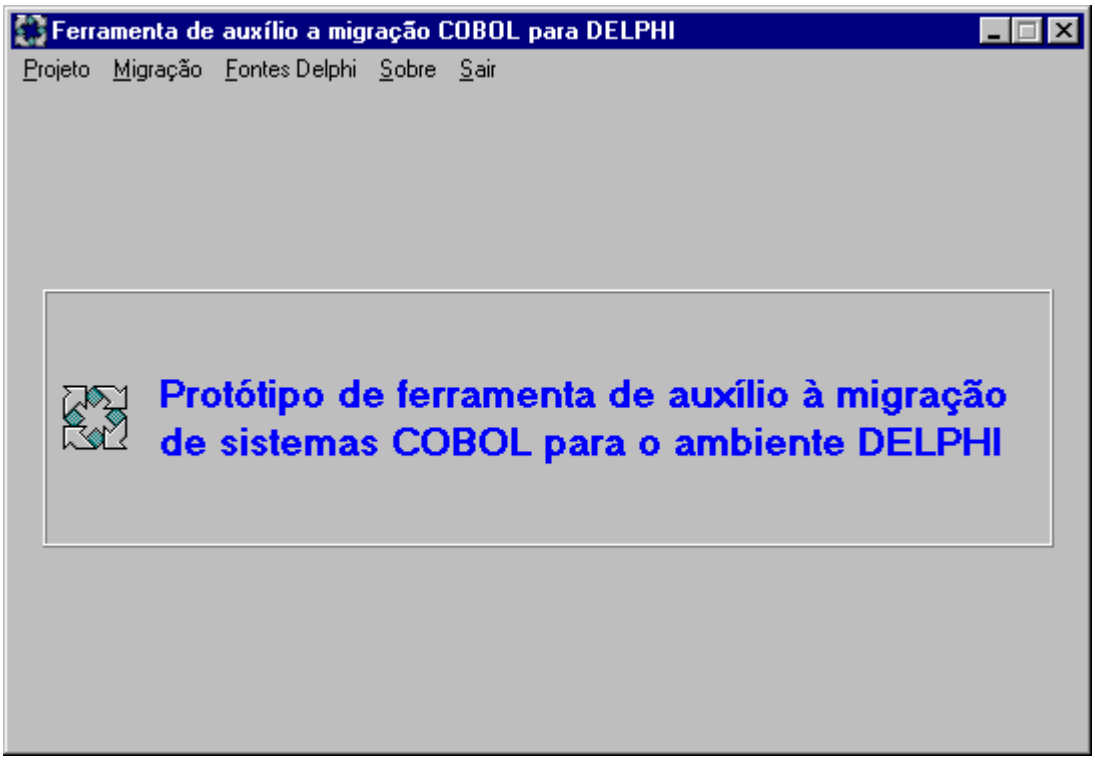
5.2 UTILIZAÇÃO DA FERRAMENTA

Nesta parte do trabalho será mostrado o funcionamento da ferramenta de auxílio à migração de sistemas, exibindo as telas e descrevendo as suas funções, para que o usuário que venha a trabalhar com a ferramenta saiba como proceder em cada situação do programa.

Na figura 5.4, é mostrada a tela principal do sistema, que é o ponto de partida para o uso da ferramenta. É nesta tela que o usuário pode escolher as funções principais da ferramenta, cada opção principal do sistema será descrita a seguir :

a) Projeto, permite que o usuário escolha entre abrir ou criar um novo projeto;
b) Migração, permite que o usuário acesse o módulo da ferramenta onde é gerada a migração dos dados do sistema *COBOL* para o *ACCESS;*
c) Fontes Delphi, permite acesso ao usuário para o módulo da ferramenta que efetua a geração dos novos fontes do sistema (fontes *Delphi 4*);
d) Sobre, apresenta ao usuário uma tela que contém informações sobre a ferramenta;
e) Sair, permite ao usuário o encerramento da execução do protótipo.

Figura 5.4 – Tela principal da ferramenta.



5.2.1 PROJETOS

Quando iniciar a utilização da ferramenta, o usuário deverá criar ou escolher o projeto no qual deseja trabalhar. Para que o mesmo possa realizar o gerenciamento da migração de um ou mais sistemas, a ferramenta possibilita a criação e o uso de vários projetos.

Com a criação de vários projetos a ferramenta permite que, quando for necessário, o usuário possa :

a) gerar a partir de um sistema COBOL, um ou mais sistemas em ambiente *DELPHI*;
b) gerar um sistema em ambiente *DELPHI*, a partir de vários sistemas em COBOL.

Para que o usuário possa criar um novo projeto, deve primeiro acessar a tela principal do sistema, mostrada na figura 5.4, selecionar o item de menu ‘Projeto’, e depois selecionar a opção ‘Novo’. A ferramenta então exibirá a tela mostrada na figura 5.5 e permitirá a digitação do nome do novo projeto a ser criado.

Para efetuar a inclusão do projeto, o botão ‘Confirmar’ deve ser acionado, indicando à ferramenta que o projeto deve ser consistido para que não seja permitido a inclusão de um já existente. Caso existir, a ferramenta retorna uma mensagem informando que o projeto não pode ser criado, e o usuário deve escolher um novo nome para o projeto. Caso o nome de projeto seja válido, a ferramenta gera um novo registro de projeto, que será gravado na tabela CAAPRO, descrita no capítulo anterior, e ativa a tela de edição de informações do projeto, que é mostrada na figura 5.7.

Se o usuário não pretender cadastrar um novo projeto, pode acionar o botão ‘Cancelar’, e a ferramenta fechará a tela ativa, retornando o controle para a sua tela principal.

Figura 5.5 – Tela de inclusão de projeto.

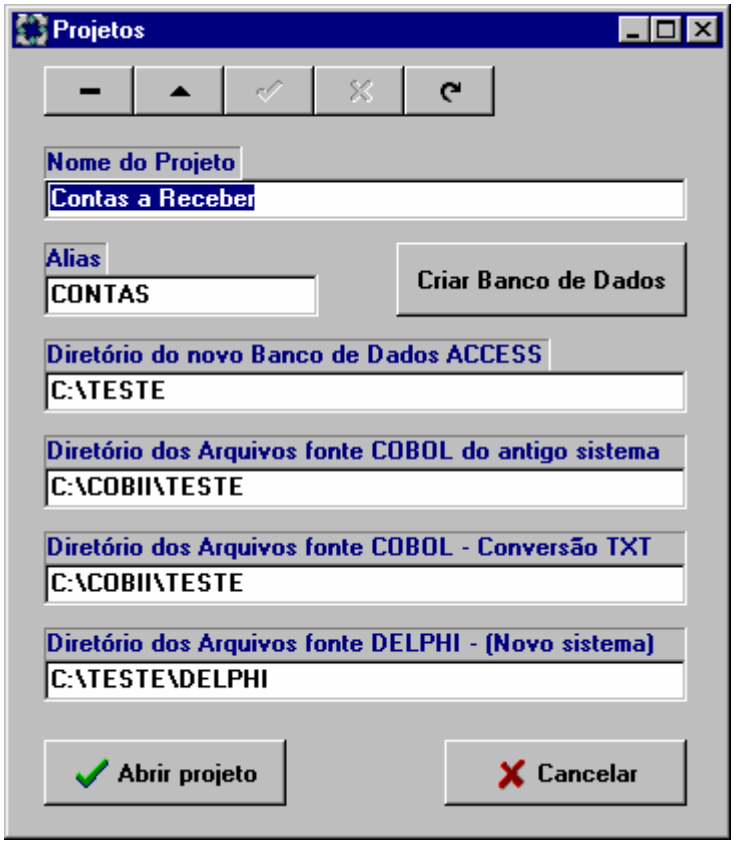
Se a ferramenta já possuir projetos cadastrados, então, o usuário poderá escolher um deles para utilização. Para fazer esta escolha, deverá acessar a tela principal do sistema, mostrada na figura 5.4, selecionar o item de menu ‘Projeto’, e depois selecionar a opção ‘Abrir’. A ferramenta então exibirá a tela mostrada na figura 5.6. e permitirá a digitação do nome do projeto a ser aberto.

O usuário poderá abrir um projeto de duas maneiras. Na primeira, deve escrever o nome do projeto que deseja abrir, dentro do componente de edição ‘Posicionar’, até que tal projeto esteja selecionado na grade de Projeto, e então acionar o botão ‘Confirmar’. Na segunda, deve efetuar um duplo clique do mouse sobre o projeto que se encontra na grade. Se não pretender abrir um projeto, pode acionar o botão ‘Cancelar’, que a ferramenta fechará a tela ativa, retornando o controle do sistema para a sua tela principal.

Figura 5.6 – Tela de abertura de projetos.

Logo após ter aberto ou criado um projeto, o usuário deve cadastrar ou verificar se os dados do projeto estão corretos, para isso, a ferramenta ativa a tela apresentada na figura 5.7, e permite que os dados do projeto sejam editados.

Figura 5.7 – Tela de edição de dados de projeto.



Nesta tela da ferramenta o usuário pode manter as informações referentes ao projeto que se deseja utilizar. A seguir será dada uma explicação sobre cada componente desta tela, descrevendo sua funcionalidade.

No canto superior da tela, o primeiro componente é um navegador de arquivos padrão do *Delphi*, este componente permite a exclusão ou a alteração dos registros de projeto. No componente de edição abaixo do navegador, pode-se alterar o nome do projeto. No componente de edição entitulado ‘Alias’ o usuário informa qual o nome do novo alias que será cadastrado para que o *Delphi* possa fazer acesso ao banco de dados que se deseja criar.

O componente de edição com o título ‘Diretório do novo banco de dados *ACCESS*’, permite que o usuário informe em que diretório será criado o banco de dados em *ACCESS*, que será utilizado para se efetuar a gravação dos dados convertidos do antigo sistema.

O componente de edição entitulado ‘Diretório dos arquivos fonte *COBOL* do antigo sistema’, permite ao usuário informar à ferramenta qual o diretório onde se encontram os arquivos fonte *COBOL* que pertencem ao sistema que se deseja converter.

O componente de edição com o título ‘Diretório dos arquivos fonte *COBOL* – Conversão TXT’, permite ao usuário informar à ferramenta qual o diretório onde devem ser gravados os arquivos gerados em *COBOL* para fazer a conversão dos dados do formato *COBOL* para formato TXT .

O componente de edição entitulado ‘Diretório dos arquivos fonte *Delphi* – (Novo sistema)’, permite que o usuário informe à ferramenta em qual diretório devem ser gravados os arquivos fontes *Delphi*, que serão gerados para formar o novo sistema.

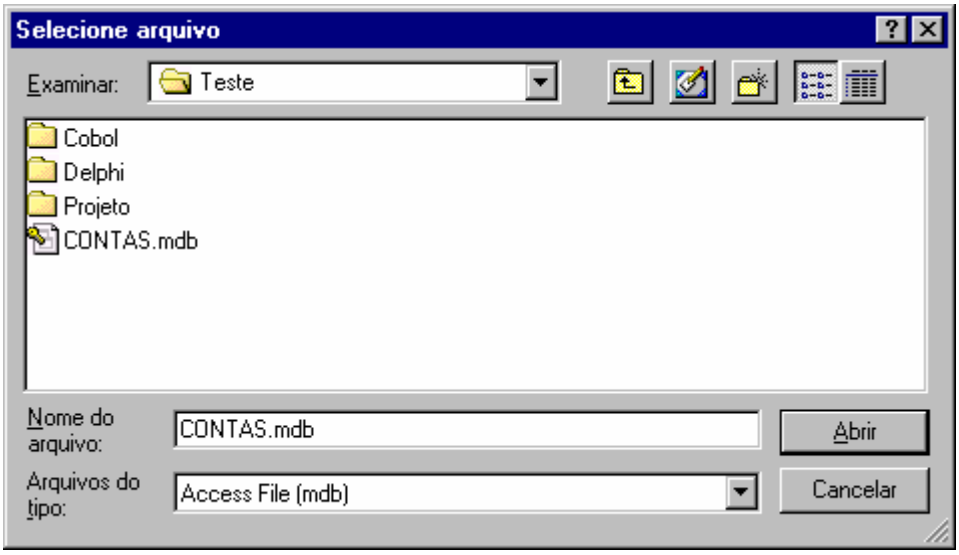
O botão ‘Abrir Projeto’, quando acionado, indica à ferramenta que o projeto escolhido pode ser aberto para utilização.

O botão ‘Cancelar’, desativa a tela de projetos e retorna o controle do sistema para a tela principal da ferramenta.

Finalmente, será explicado o botão ‘Criar Banco de Dados’, que é um caso especial do cadastro de projetos, e deve ser executado sempre quando um novo projeto é cadastrado. A função deste botão é criar um Banco de Dados para que o projeto possa gerar as tabelas do novo sistema dentro dele, ou seja, sem isto a ferramenta não é capaz de gravar e nem de ler os dados do novo sistema. Para que o botão possa ser acionado, o usuário deve informar primeiro o nome do alias que será gerado, e o diretório onde se deseja criar o arquivo de banco de dados do *MS-ACCESS 97* referente ao projeto. Após cadastrar estas informações e acionar o botão de criar banco de dados o sistema apresenta a tela mostrada na figura 5.8, onde o usuário informa o nome do banco de dados a ser criado e confirma acionando o botão ‘Abrir’.

A partir deste ponto a ferramenta gera um arquivo de banco de dados do *MS-ACCES 97*, e também cria na configuração do BDE um alias relacionado com o arquivo gerado, permitindo assim que se possa começar a migração de dados no projeto.

Figura 5.8 – Tela para criação do banco de dados *MS-ACCESS* 97.

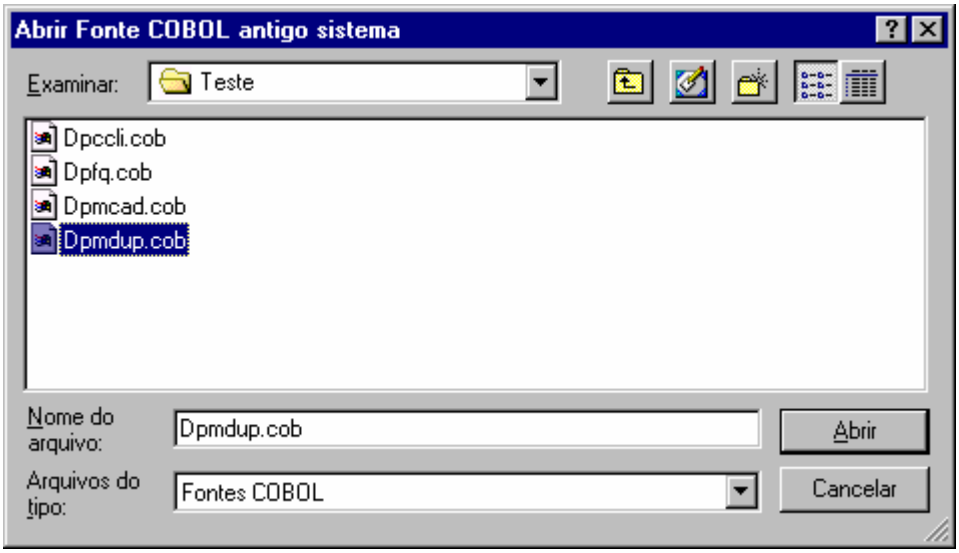


5.2.2 MIGRAÇÃO

O usuário depois de ter aberto um projeto da ferramenta, pode escolher por trabalhar com a opção de migração da ferramenta, que é a parte da mesma onde se analisa os arquivos do antigo sistema, e se faz a conversão dos dados do *COBOL* para formato TXT, e em seguida de TXT para uma tabela equivalente em *MS-ACCESS 97*.

Para acessar esta opção da ferramenta o usuário deve ter anteriormente aberto um projeto e então acessar o item de menu ‘Migração’ e depois acessar o item de menu ‘Gerar conversão de dados’. Após a execução desses passos, a ferramenta apresenta a figura 5.9, mostrando os arquivos *COBOL* do antigo sistema existente no diretório de arquivos fonte *COBOL* referente ao projeto.

Figura 5.9 – Tela de abertura de arquivo fonte *COBOL* do antigo sistema.

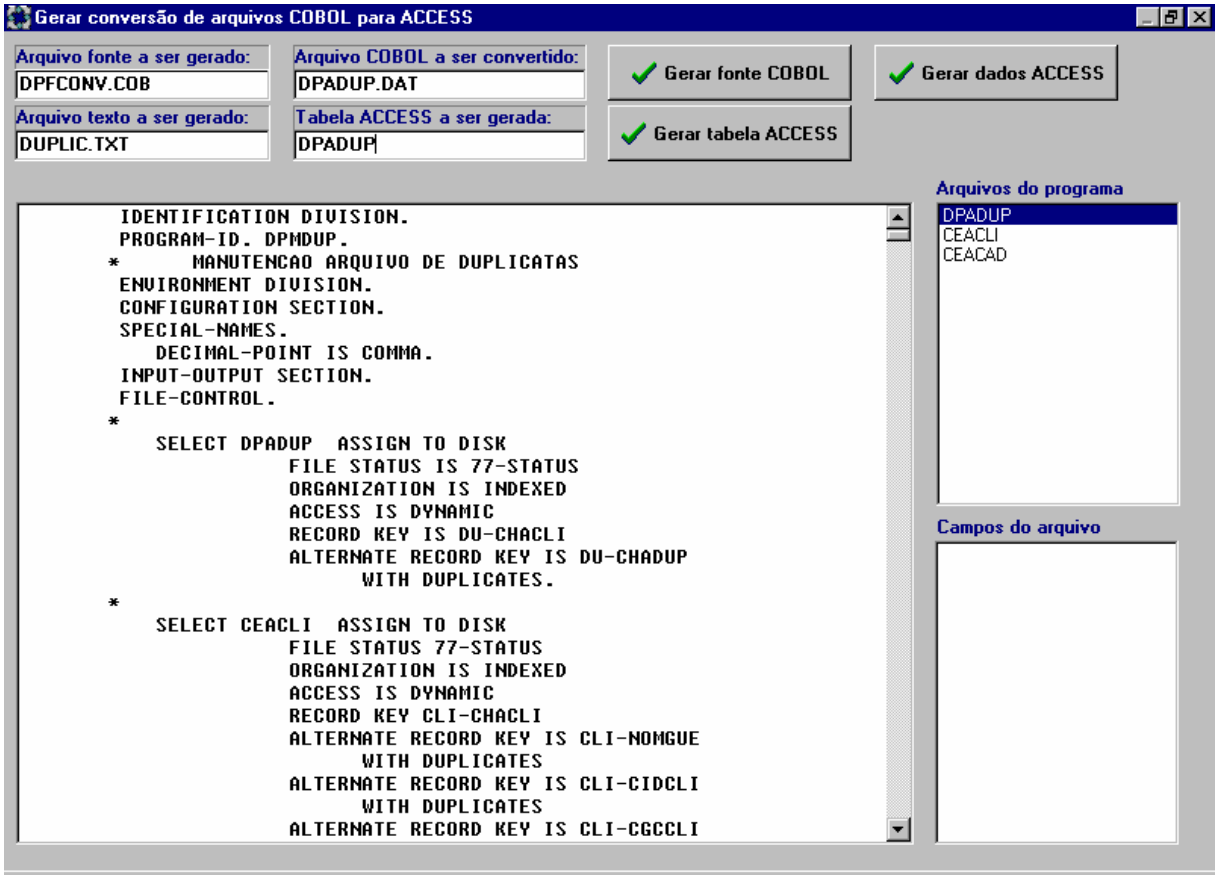


Para um melhor entendimento desta parte de migração da ferramenta, escolheu-se para servir de exemplo, um programa *COBOL* que faz parte de um sistema de contas a receber. Este fonte, em sua função original, serve para manter um cadastro de duplicatas, e seu nome como arquivo é DPMDUP.COB.

Continuando com a migração, depois do usuário ter escolhido o fonte *COBOL* que deseja analisar, a ferramenta ativa a tela 5.10, e carrega o arquivo escolhido na caixa de texto no canto inferior esquerdo da tela, eliminando caracteres de controle do arquivo, e inserindo no fonte arquivos apontados pelas cláusulas *COPY* do *COBOL*.

Enquanto a ferramenta carrega o arquivo na tela, já o analisa, encontrando os arquivos disponíveis para conversão dentro do programa, e os lista na caixa de texto entitulada ‘Arquivos do programa’. Ao final da carga do programa, a ferramenta retorna uma mensagem avisando que a carga foi completada.

Figura 5.10 – Tela de migração de dados da ferramenta.



Para prosseguir com a migração o usuário deve escolher qual o arquivo do programa que deseja converter, e então, informar os dados necessários para a criação da conversão dos dados. A seguir, serão explicados os componentes que devem ser utilizados para executar a migração de um arquivo, e também, os passos que devem ser tomados.

Os componentes que devem ser usados para dar entradas nas informações, antes da migração dos dados, são os que se encontram nas quatro caixas de edição no canto superior esquerdo da tela.

A caixa de edição ‘Arquivo fonte a ser gerado’, deve ser utilizada para informar o nome do arquivo fonte do *COBOL* a ser gerado para que sejam convertidos os dados para formato TXT. O nome do arquivo deve conter a extensão .COB, e será gravado no diretório de arquivos fonte *COBOL* de conversão, conforme informado no projeto em uso.

Na caixa de edição ‘Arquivo texto a ser gerado’, o usuário informa qual deve ser o nome do arquivo de dados em formato TXT que será gerado. A ferramenta utiliza esta informação para gravar no diretório de arquivos fonte *COBOL* de conversão, o arquivo de dados em formato TXT, conforme informado no projeto que se está utilizando.

Na caixa de edição ‘Arquivo *COBOL* a ser convertido’, o usuário informa qual o arquivo onde o programa de conversão *COBOL* deve fazer a leitura dos dados, para gerar o arquivo de dados em formato TXT.

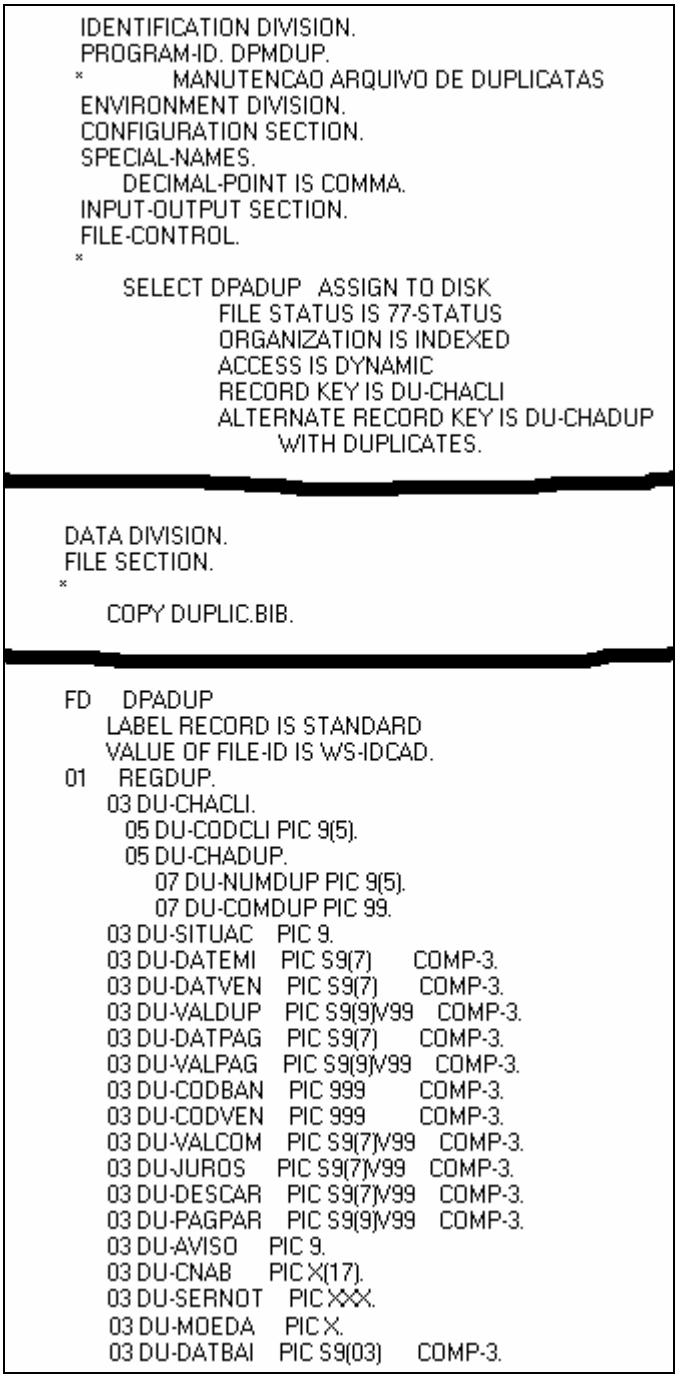
Na caixa de edição ‘Tabela *ACCESS* a ser gerada’, o usuário informa qual o nome da tabela que será gerada no banco de dados *ACCESS* do projeto. É nesta tabela que a ferramenta grava as definições dos campos, índices e também dados convertidos.

Depois de dar entrada nos dados supracitados, como primeiro passo para a migração de dados, o usuário deve acionar o botão ‘Gerar fonte *COBOL*’. A ferramenta então analisa o fonte antigo para descobrir e armazenar as informações do arquivo, e em seguida, gera o fonte de conversão baseado nestas informações.

Para melhor entendimento deste processo, é apresentada a figura 5.11, que descreve a seção FILE CONTROL do programa de exemplo, que segundo [BAS88], é a seção onde tem-se a definição do modo de acesso do arquivo, a organização das informações, a definição das chaves primária e secundária e o campo de *status* do arquivo.

A ferramenta também lista os campos e suas propriedades de nível, nome, tipo e tamanho, no canto inferior direito da tela, na caixa de texto ‘Campos do arquivo’, após analisar a FD (FILE DESCRIPTION) do programa *COBOL*. Com relação aos campos que armazenam datas no arquivo *COBOL*, ou seja, para campos que possuem *PICTURE* dos tipos (------9, 9(06), S9(7) e 999999), a ferramenta permite ao usuário escolher se deseja tratar o campo como um número ou como data.

Figura 5.11 – Descrição do arquivo no programa.



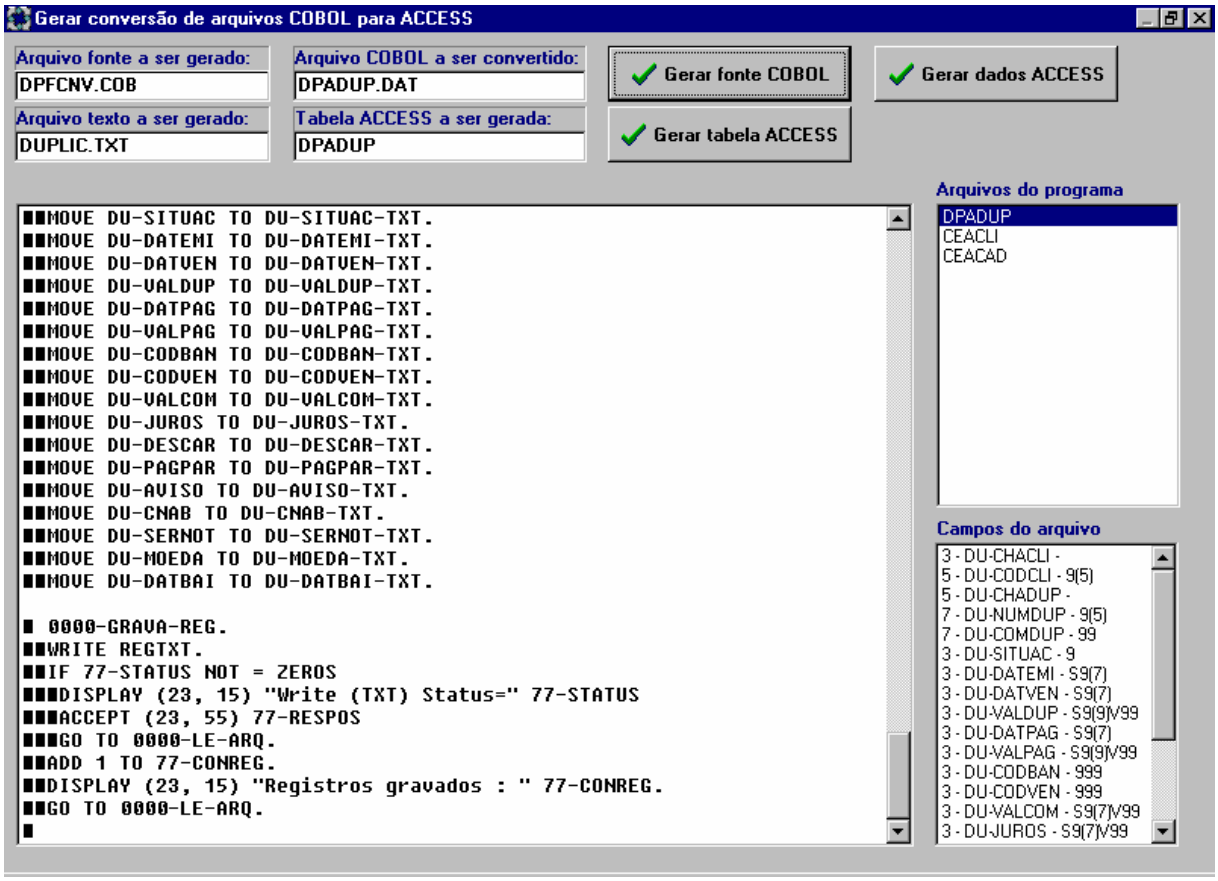
```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. DPMDUP.
      *          MANUTENCAO ARQUIVO DE DUPLICATAS
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SPECIAL-NAMES.
           DECIMAL-POINT IS COMMA.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
      *
           SELECT DPADUP  ASSIGN TO DISK
                     FILE STATUS IS 77-STATUS
                     ORGANIZATION IS INDEXED
                     ACCESS IS DYNAMIC
                     RECORD KEY IS DU-CHACLI
                     ALTERNATE RECORD KEY IS DU-CHADUP
                           WITH DUPLICATES.

       DATA DIVISION.
       FILE SECTION.
      *
           COPY DUPLIC.BIB.

       FD    DPADUP
           LABEL RECORD IS STANDARD
           VALUE OF FILE-ID IS WS-IDCAD.
       01    REGDUP.
           03 DU-CHACLI.
             05 DU-CODCLI PIC 9(5).
             05 DU-CHADUP.
                07 DU-NUMDUP PIC 9(5).
                07 DU-COMDUP PIC 99.
           03 DU-SITUAC    PIC 9.
           03 DU-DATEMI    PIC S9(7)      COMP-3.
           03 DU-DATVEN    PIC S9(7)      COMP-3.
           03 DU-VALDUP    PIC S9(9)V99   COMP-3.
           03 DU-DATPAG    PIC S9(7)      COMP-3.
           03 DU-VALPAG    PIC S9(9)V99   COMP-3.
           03 DU-CODBAN    PIC 999        COMP-3.
           03 DU-CODVEN    PIC 999        COMP-3.
           03 DU-VALCOM    PIC S9(7)V99   COMP-3.
           03 DU-JUROS     PIC S9(7)V99   COMP-3.
           03 DU-DESCAR    PIC S9(7)V99   COMP-3.
           03 DU-PAGPAR    PIC S9(9)V99   COMP-3.
           03 DU-AVISO     PIC 9.
           03 DU-CNAB      PIC X(17).
           03 DU-SERNOT    PIC XXX.
           03 DU-MOEDA     PIC X.
           03 DU-DATBAI    PIC S9(03)     COMP-3.
```

Ainda com relação à geração do arquivo fonte *COBOL* , deve se esclarecer que este arquivo é gerado a partir de um programa base, encontrado no diretório C:\Delphi\TCC\COBOL, e chama-se CNFEXP.COB. Este arquivo de apoio, permite que a ferramenta só precise adicionar código necessário para a conversão dos dados, sem precisar

gerar todo o código-fonte do programa de conversão. No final da geração do fonte de conversão, o programa é mostrado na tela para conferência do usuário.

Na figura 5.12, é apresentada a tela de conversão de dados após a geração do arquivo fonte de conversão.

Figura 5.12 – Tela de conversão após primeiro passo da conversão de dados.



O segundo passo para se converter os dados, deve ser o acionamento do botão ‘Gerar tabela *ACCESS*’. Este procedimento faz com que a ferramenta crie uma tabela no banco de dados do projeto e na tabela gerada, crie a definição dos campos e dos índices dela, conforme as informações extraídas do fonte *COBOL* analisado anteriormente.

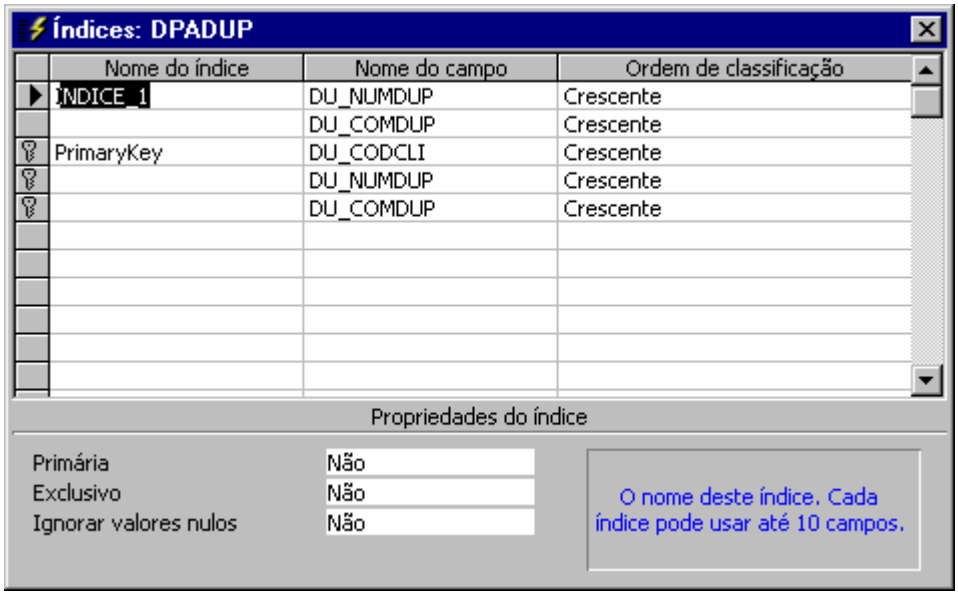
Na tabela 1 é mostrado como os campos do arquivo de teste DPADUP, foram convertidos para o formato *ACCESS*. Os campos com asterisco após a descrição formam a chave primária da tabela.

Tabela 1 – Campos gerados no *ACCESS*.

Nome da tabela: DPADUP		
Nome do campo	**Tipo**	**Tamanho**
DU-CODCLI *	Inteiro longo	
DU-NUMDUP *	Inteiro longo	
DU-COMDUP *	Inteiro longo	
DU-SITUAC	Inteiro longo	
DU-DATEMI	Data/Hora	
DU-DATVEN	Data/Hora	
DU-VALDUP	Duplo	
DU-DATPAG	Data/Hora	
DU-VALPAG	Duplo	
DU-CODBAN	Inteiro longo	
DU-CODVEN	Inteiro longo	
DU-VALCOM	Duplo	
DU-JUROS	Duplo	
DU-DESCAR	Duplo	
DU-PAGPAR	Duplo	
DU-AVISO	Inteiro longo	
DU-CNAB	Texto	17
DU-SERNOT	Texto	3
DU-MOEDA	Texto	1
DU-DATBAI	Inteiro longo	

Na figura 5.13 são mostrados os índices gerados na tabela convertida para o *ACCESS*.

Figura 5.13 – Índices gerados na tabela em *MS-ACCESS 97*.



Para finalizar a conversão dos dados, deve ser efetuado o terceiro e último passo da conversão. Para isto, o usuário deve acionar o botão ‘Gerar dados *ACCESS*’. A ferramenta então executa o compilador *COBOL*, para que o arquivo fonte de conversão seja compilado e linkeditado, gerando assim o arquivo executável de conversão. A figura 5.14, mostra o comportamento da tela de conversão de dados, durante a compilação do programa.

Depois de efetuar a geração do executável de conversão, a ferramenta dispara a execução do mesmo, para que assim possa ser gerado o arquivo de conversão TXT. A figura 5.15, mostra o comportamento da tela de conversão de dados, durante a execução do programa de conversão.

Figura 5.14 – Tela de conversão de dados durante a compilação do arquivo fonte *COBOL*.

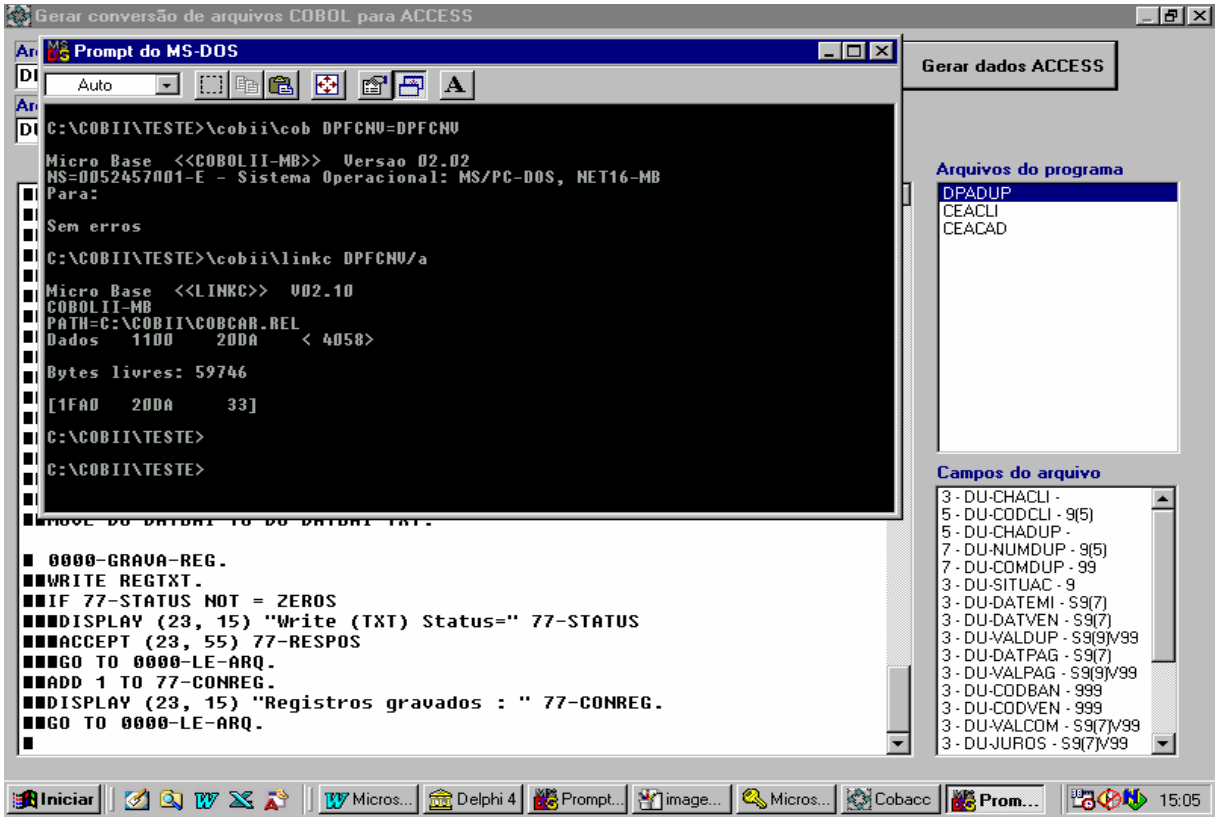
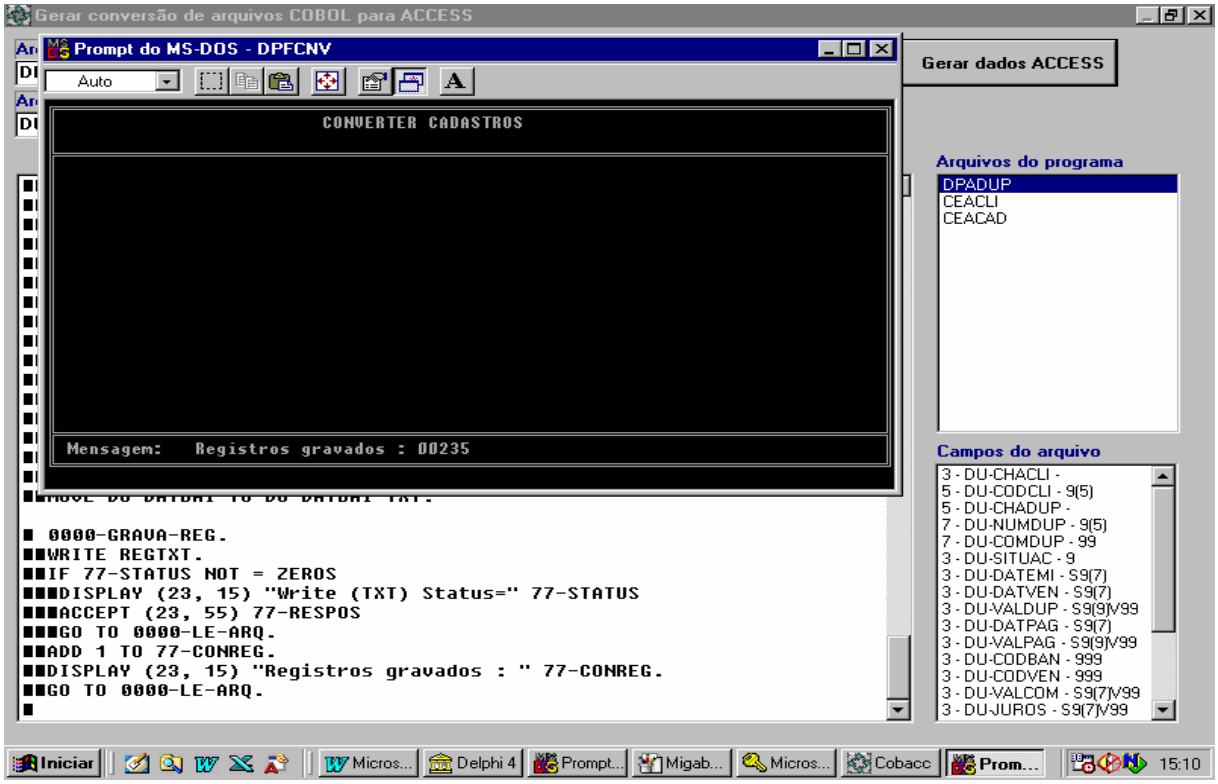
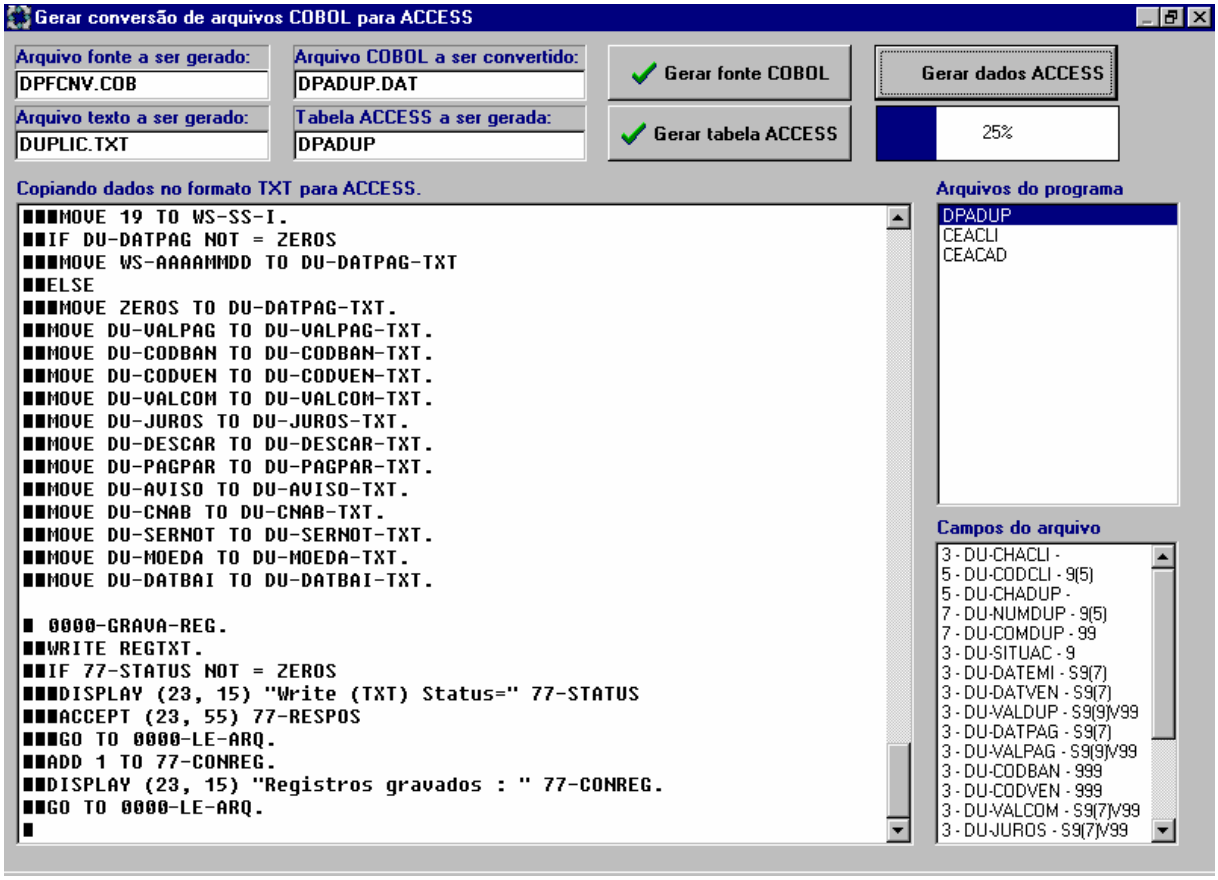


Figura 5.15 – Tela de conversão de dados durante a execução do arquivo *COBOL*.



Depois de executar o arquivo de conversão e gerar o arquivo TXT, a ferramenta então começa a gerar a gravação dos dados na tabela *ACCESS*, lendo o arquivo TXT e gravando cada campo de cada registro, de acordo com as informações recuperadas do arquivo fonte original. Enquanto faz a gravação dos dados a ferramenta apresenta uma barra de progresso, indicando qual a porcentagem já efetuada da conversão dos dados, conforme pode ser visto na figura 5.16.

Figura 5.16 – Tela de conversão durante a gravação dos dados em tabela *MS-ACCESS 97*.




```
■■■MOVE 19 TO WS-SS-I.
■■IF DU-DATPAG NOT = ZEROS
■■■MOVE WS-AAAAMMDD TO DU-DATPAG-TXT
■■ELSE
■■■MOVE ZEROS TO DU-DATPAG-TXT.
■■MOVE DU-VALPAG TO DU-VALPAG-TXT.
■■MOVE DU-CODBAN TO DU-CODBAN-TXT.
■■MOVE DU-CODVEN TO DU-CODVEN-TXT.
■■MOVE DU-VALCOM TO DU-VALCOM-TXT.
■■MOVE DU-JUROS TO DU-JUROS-TXT.
■■MOVE DU-DESCAR TO DU-DESCAR-TXT.
■■MOVE DU-PAGPAR TO DU-PAGPAR-TXT.
■■MOVE DU-AVISO TO DU-AVISO-TXT.
■■MOVE DU-CNAB TO DU-CNAB-TXT.
■■MOVE DU-SERNOT TO DU-SERNOT-TXT.
■■MOVE DU-MOEDA TO DU-MOEDA-TXT.
■■MOVE DU-DATBAI TO DU-DATBAI-TXT.

■ 0000-GRAVA-REG.
■■WRITE REGTXT.
■■IF 77-STATUS NOT = ZEROS
■■■DISPLAY (23, 15) "Write (TXT) Status=" 77-STATUS
■■■ACCEPT (23, 55) 77-RESPOS
■■■GO TO 0000-LE-ARQ.
■■ADD 1 TO 77-CONREG.
■■DISPLAY (23, 15) "Registros gravados : " 77-CONREG.
■■GO TO 0000-LE-ARQ.
■
```

Após o término da conversão o usuário deve então fechar a tela ativa e retornar à tela principal, para então poder utilizar novamente a ferramenta. Neste momento o usuário já possui os recursos para efetuar a geração do arquivo fonte em *Delphi*, para poder obter um programa que acesse os dados gerados pela ferramenta.

5.2.3 FONTES DELPHI

Esta é a parte da ferramenta onde o usuário pode gerar automaticamente um novo sistema ou programa em *Delphi*, para fazer acesso aos dados que foram convertidos. Para obter acesso à esta função da ferramenta, o usuário deve ter já aberto um projeto da ferramenta, e este por conseguinte, deve possuir ao menos uma tabela em seu banco de dados. A seguir serão descritos os procedimentos que devem ser executados para tornar possível a geração dos arquivos fontes do novo sistema.

Como primeiro passo, o usuário deve utilizar o menu da tela principal da ferramenta, utilizando a opção do menu 'Fontes *Delphi*', e em seguida, a opção 'Gerar fontes *Delphi*'. Então a ferramenta conecta-se ao banco de dados criado pelo projeto, mostrando ao usuário a tela descrita na figura 5.18, para que o mesmo digite sua senha de acesso, caso esta exista. Posteriormente, a ferramenta ativa a tela de escolha de tabela, conforme é mostrado na figura 5.17.

Figura 5.17 – Tela de escolha de tabela para geração de novo fonte em *Delphi*.

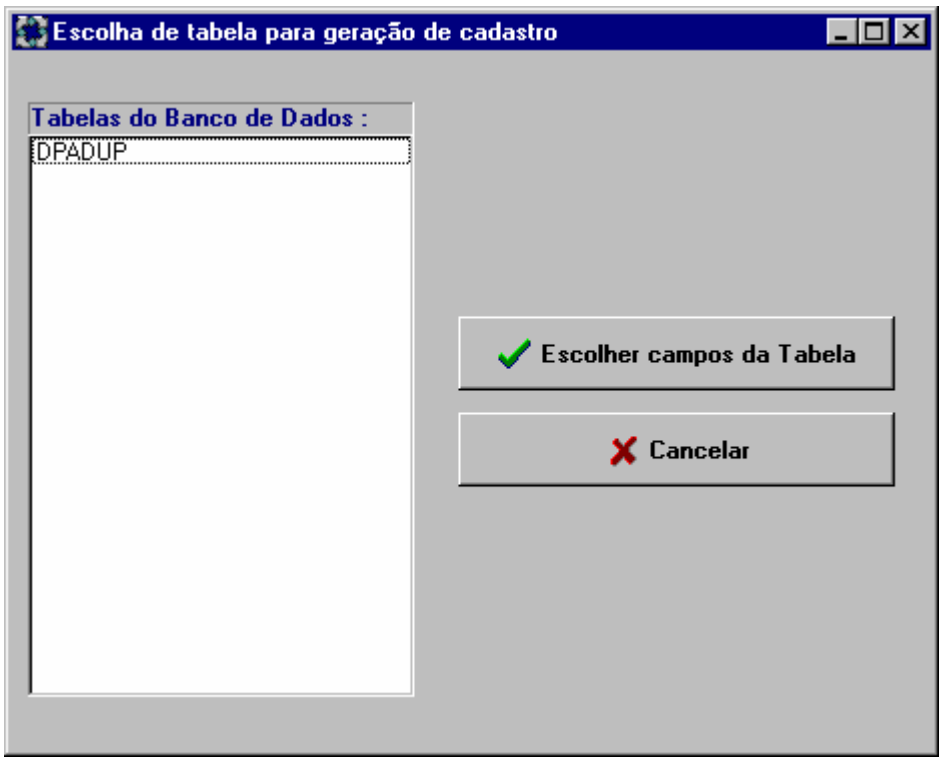


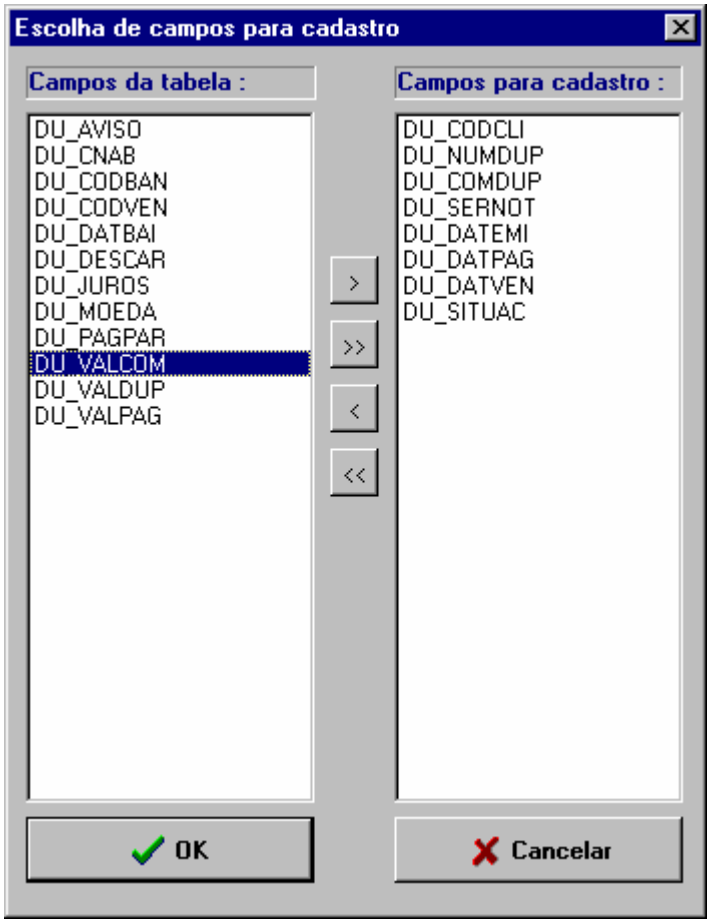
Figura 5.18 – Tela de digitação de senha para acesso a banco de dados *MS-ACCESS 97*.

O usuário então deve selecionar na caixa de texto ‘Tabelas do Banco de Dados’, a tabela que deseja utilizar para gerar o fonte do cadastro no novo sistema, o usuário deve acionar o botão ‘Escolher campos da Tabela’, e assim então a ferramenta ativa a tela de escolha de campos para cadastro conforme apresentado na figura 5.19. Para abortar esta operação o usuário pode acionar o botão ‘Cancelar’, e a ferramenta então fecha a tela ativa e retorna para a tela principal.

Na tela de escolha de campos para cadastro, o usuário pode visualizar uma lista dos campos da tabela, que pode utilizar para formar o cadastro em *Delphi*. Na caixa de texto ‘Campos da tabela’, estão os campos que podem ser utilizados, e na caixa de texto ‘Campos para cadastro’, se encontram os campos que já foram selecionados para fazer parte do cadastro a ser gerado. Para transportar os campos de uma caixa de texto para outra, o usuário deve fazer uso dos quatro botões que se encontram entre as mesmas.

É importante salientar que, para que o novo cadastro realmente funcione, o usuário deve conferir se os campos que fazem parte dos índices da tabela foram selecionados. Caso contrário, o novo sistema não estará apto a cadastrar novos dados, e também, não conseguirá navegar entre os dados existentes.

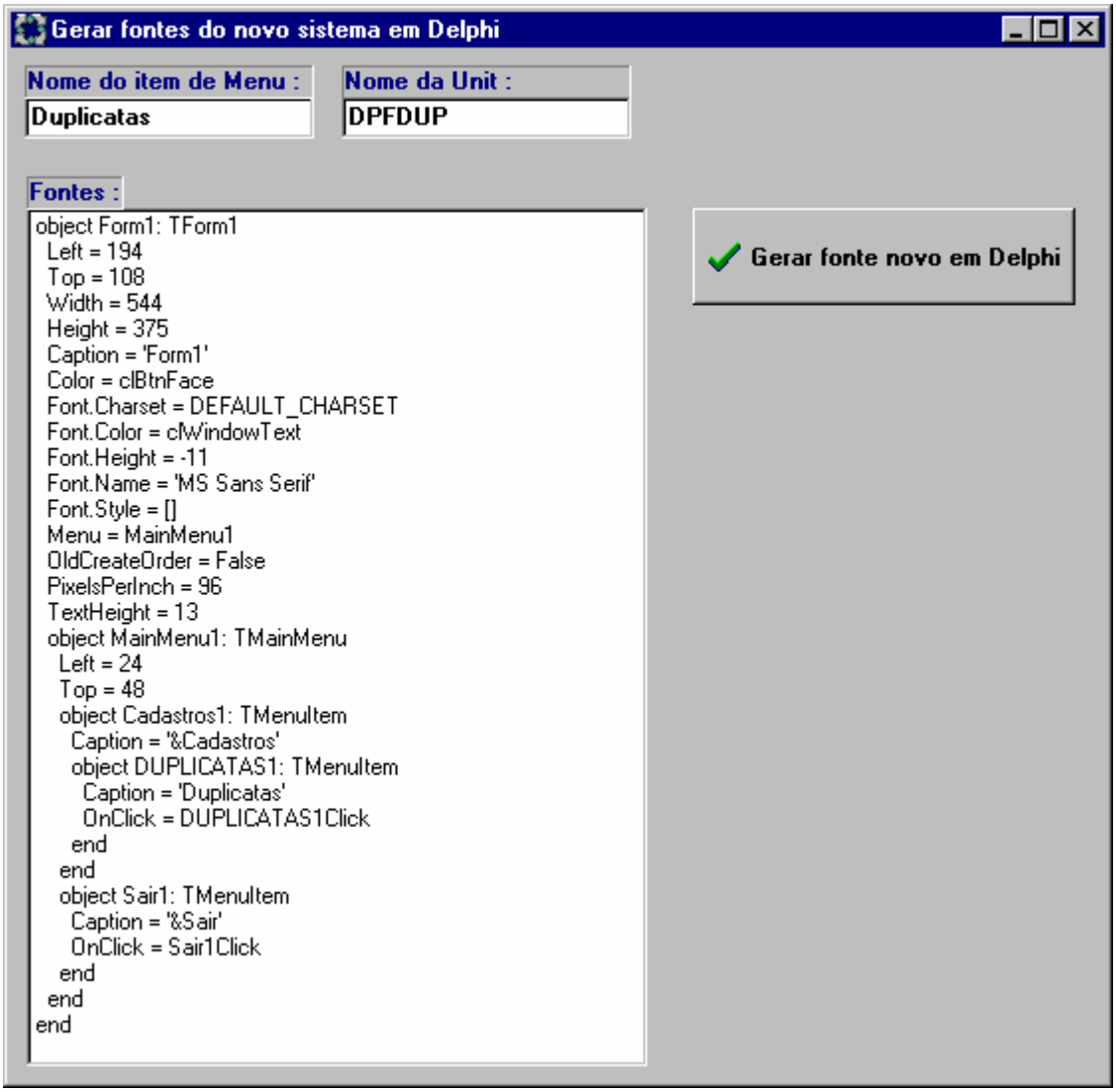
Figura 5.19 – Tela de escolha de campos para cadastro.



Depois de acabar de escolher os campos, o usuário deve acionar o botão ‘OK’ para que a ferramenta continue na geração dos arquivos fontes do novo sistema. Caso queira abortar esta operação o usuário pode acionar o botão ‘Cancelar’, e a ferramenta então fecha a tela ativa e retorna para a tela principal.

Na seqüência da geração dos fontes em *Delphi*, a ferramenta ativa a tela de geração dos fontes do novo sistema em *Delphi*. É esta a tela que o usuário deve utilizar para informar à ferramenta, qual o nome do novo fonte do sistema, e também como será feito o acesso a ele pelo menu do novo sistema. A tela supracitada é mostrada na figura 5.20.

Figura 5.20 – Tela de geração de fonte do novo sistema em *Delphi*.



Na tela da figura 5.20, o usuário utiliza as duas caixas de edição existentes para informar qual o nome do item de menu no novo sistema, ou seja, qual a opção que será gerada no menu para acessar o cadastro, e também, informar qual o nome do novo programa no novo sistema. Para efetivar a geração do novo fonte, o usuário deve acionar o botão ‘Gerar fonte novo em *Delphi*’.

A ferramenta então trata de verificar se já existe um fonte de projeto *Delphi*, no diretório de fontes *Delphi*, que está sendo utilizado pelo projeto aberto pela própria ferramenta. Se não for encontrado o arquivo, então a ferramenta procede executando a cópia

dos arquivos que servem de base para um novo sistema, que estão encontrados no diretório C:\Delphi\TCC\Delphi, para o diretório de fontes *Delphi* do projeto.

Em seguida, abre os arquivos padrão de cadastro, encontrados no diretório C:\Delphi\TCC\CADASTRO, e adiciona o código necessário para permitir o acesso aos campos da tabela que foram escolhidos na tela da figura 5.19. Estes arquivos fontes são gravados juntamente no diretório de fontes *Delphi* do projeto, com o nome que o usuário informou na caixa de edição 'Nome da Unit'. Assim, então, termina a geração do novo fonte em *Delphi*, e o novo sistema está pronto para ser compilado e executado.

5.2.4 RESULTADOS

Os resultados obtidos com a geração dos novos fontes em *Delphi* e com a conversão dos dados, podem ser conferidos neste tópico do trabalho.

A figura 5.21 apresenta a tela principal do novo sistema gerado pela ferramenta de auxílio à migração.

Figura 5.21 – Tela principal do novo sistema gerado pela ferramenta.

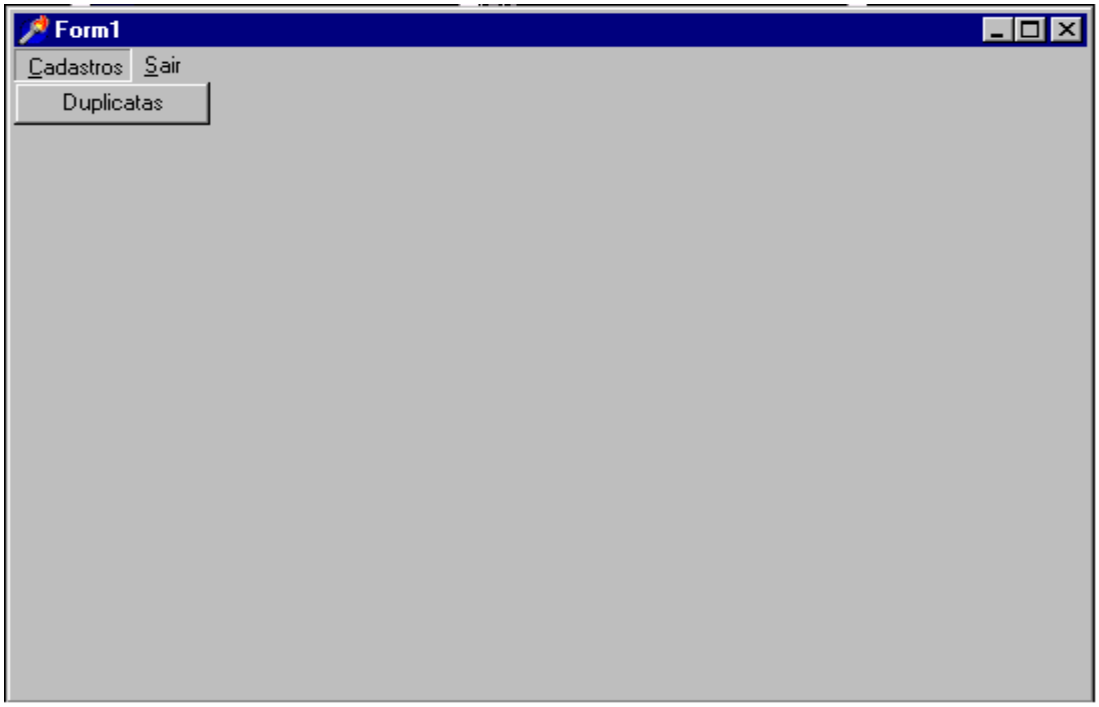
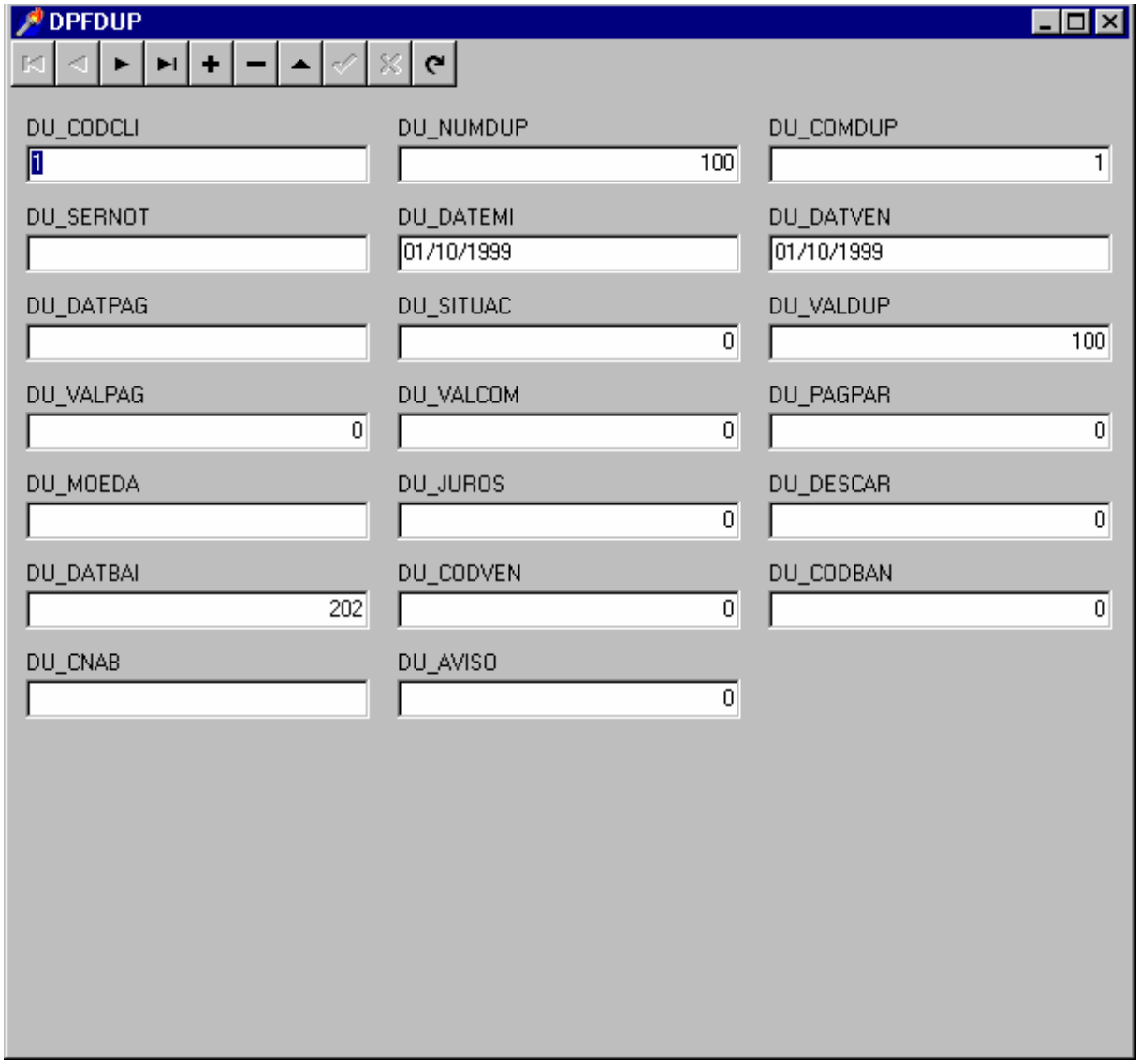


Figura 5.22 – Tela de cadastro gerada no novo sistema.



A tela da figura 5.21 mostra como foi gerado o menu do novo sistema, de acordo com o nome do item de menu que o usuário havia escolhido.

A tela da figura 5.22, mostra como é formada a tela de cadastros do novo sistema, apresentando na parte superior esquerda um navegador da tabela DPADUP convertida, e as caixas de edição utilizadas para fazer acesso aos campos da mesma tabela.

5.2.5 INSTALAÇÃO

Para a instalação da ferramenta, o usuário deve dispor do CD referente à este trabalho, pois é nele que se encontram os arquivos necessários para execução da mesma. É necessário que os softwares *Delphi 4* e o *COBOL Microbase* estejam instalados na máquina, e que se siga os passos de instalação descritos a seguir.

Deve ser criado um diretório chamado C:\Delphi\TCC, onde deve ser copiado o arquivo COBACC.EXE.

Deve ser criado o diretório C:\Delphi\TCC\COB, onde deve ser copiado o arquivo CNFEXP.COB.

Deve ser criado o diretório C:\Delphi\TCC\DADOS, onde deve ser copiado o banco de dados da ferramenta, que se chama COBACC.MDB.

O diretório C:\Delphi\TCC\DELPHI deve ser gerado, e os seguintes arquivos devem ser copiados para ele:

a) arquivos de projeto – Project1.CFG, Project1.DOF, Project1.DPR e Project.RES;
b) arquivos de *forms* e *units* – Unit1.PAS, Unit1.DFM, Unit2.PAS e Unit2.DFM;

O diretório C:\Delphi\TCC\CADASTRO deve ser gerado e os arquivos Cadastro.PAS e Cadastro.DFM devem ser copiados para ele;

Deve ser gerado um diretório público, e os arquivos do compilador *COBOL* devem ser copiados para o mesmo, copiando também o arquivo COBOL.BAT,

Para finalizar, o usuário deve criar no *BDE Administrator*, um alias com o nome ‘CA’, que aponte para o banco de dados COBACC.MDB. Assim, a ferramenta está pronta para o início da sua utilização.

6 CONCLUSÕES E SUGESTÕES

6.1 CONCLUSÕES

A reengenharia de sistemas é uma área que deve ser amplamente explorada atualmente, pois com a criação de novas tecnologias, cresce também o número de programas e a probabilidade de se efetuar uma migração de sistemas antigos para os novos ambientes.

O emprego de uma ferramenta de auxílio à migração, possibilita à uma organização uma diminuição considerável dos seus gastos com manutenção, quando esta necessita migrar seus sistemas para um novo ambiente. Isto ocorre porque o tempo e o desgaste usados para promover uma migração de sistema podem ser reduzidos com a automatização, e estes recursos podem ser redirecionados para o aprendizado e desenvolvimento de novos sistemas no novo ambiente que será utilizado.

A ferramenta apresentada neste trabalho, por ser um protótipo e ter sido desenvolvida em um curto período de tempo, não permite automatizar a migração de arquivos que contenham mais de um tipo de registro definido em seu layout, bem como, também não permite a conversão de registros que façam uso da cláusula *REDEFINES* e *OCCURS* em sua estrutura. A ferramenta, enquanto estiver em sua fase de protótipo, se destina somente à migração de registros que contenham estruturas de dados simples.

O estudo teve como objetivo a migração de dados e de telas, portanto, a ferramenta também não contempla a migração de código. Esta ferramenta serve para auxiliar na migração, e não para substituí-la, portanto, quaisquer alterações que necessitem ser executadas no novo sistema, podem e devem ser efetuadas após a geração do cadastro no novo sistema.

Devido ao tempo gasto no desenvolvimento, não se efetuou uma documentação sobre os testes com a ferramenta, porém, os arquivos que foram usados para estes testes comprovaram o funcionamento da mesma.

6.2 SUGESTÕES

A seguir são apresentadas algumas sugestões para o aperfeiçoamento do trabalho que foi realizado :

a) adicionar a migração de código e relatórios, juntamente com o uso de uma metodologia para reengenharia;
b) promover uma análise aprofundada da relação custo/benefício/prazo de uma migração de sistemas apoiada por uma ferramenta;
c) promover um estudo sobre a migração de sistemas para ambientes orientados a objetos;

REFERÊNCIAS BIBLIOGRÁFICAS

[AHO95] AHO, Alfred V.; SETHI, Ravi; ULLMAN, Jeffrey D.. **Compiladores – princípios, técnicas e ferramentas.** Rio de Janeiro : Livros Técnicos e Científicos, 1995.

[BAS88] BASTOS, Alex C.. **Programação COBOL.** São Paulo : Livros Técnicos e Científicos, 1988.

[BAU97] BAUM, Moisés. **Um estudo de caso de migração de um programa COBOL.** Blumenau, Jul., 1997. Monografia (Bacharelado em Ciências da Computação) – Departamento de Sistemas e Computação, Fundação Universidade Regional de Blumenau.

[BER98] BERGEY, John. **A Reengineering Process Framework, Proceedings of the Fifth Systems Reengineering Technology.** http://www.sei.cmu.edu/~reengineering//pubs/srtw95/BHL95, 1998.

[CAN96] CANTÚ, Marco. **Dominando o Delphi.** São Paulo : Makron Books, 1996.

[CAR93] CARVALHO, José E.M.. **Microsoft COBOL 4.5 – Programação avançada.** São Paulo : Makron Books, 1993.

[CAV98] CAVALCANTE, Mônica Miranda. **Roteiro de migração de ambiente de banco de dados.** Blumenau, 1998. Monografia (Pós-Graduação em Nível de Especialização em Tecnologia de Desenvolvimento de Sistemas) – Fundação Universidade Regional de Blumenau.

[FUR95] FURLAN, José Davi. **Reengenharia da informação – Do mito à realidade.** São Paulo : Makron Books, 1995.

[GAN91] GANE, Chris; SARSON, Trish. **Análise estruturada de sistemas.** Rio de Janeiro : Livros Técnicos e Científicos, 1991.

[MAR91] MARTIN, James; McCLURE, Carma. **Técnicas estruturadas e CASE.** São Paulo : McGraw-Hill, 1991.

[PER97] PERSPECTION, Inc.. **Microsoft ACCESS 97 – Sem mistério.** São Paulo : Berkeley, 1997.

[POP99] POPKIN Software. Products, http://www.popkin.com/products/sa2001/product.htm, 1999.

[PRE95] PRESSMAN, Roger S.. **Engenharia de software.** São Paulo : Makron Books, 1995.

[YOU90] YOURDON, Edward. **Análise estruturada moderna.** Rio de Janeiro : Campus, 1990.