Universidade Federal de São Carlos

Programa de Pós-Graduação em Ciência da Computação

Departamento de Computação

Uma Ferramenta de Apoio à Engenharia Reversa Orientada a Objetos de Legados - FAROOL

Milene Prado

Dissertação de Mestrado

"Uma **F**erramenta de **A**poio à Engenharia **R**eversa **O**rientada a **O**bjetos de **L**egados - **FAROOL**"

ORIENTADORA: Dra. *Rosângela Aparecida Delloso Penteado*
ALUNA: *Milene Prado*

São Carlos
2003

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

Prado, Milene.
P896fa Uma ferramenta de apoio à engenharia reversa orientada a objetos de legados - FAROOL / Milene Prado. -- São Carlos : UFSCar, 2003.
165 p.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2003.

1. Engenharia de software. 2. Engenharia reversa. 3. Heurística. 4. Sistema legado Cobol. 5. Metamodelo. I. Título.

CDD: 005.1 (20ª)

À minha família e

Ao meu noivo

Agradecimentos:

Primeiramente a DEUS pela oportunidade e capacidade de aprender.

Aos meus pais e ao meu noivo que não pouparam esforços para me incentivar, e cujo apoio foi de suma importância para a realização deste projeto.

À minha orientadora Dra. Rosângela Aparecida Delloso Penteado pela orientação, dedicação e amizade.

Aos meus amigos do DC em especial ao grupo de engenharia de software pela troca de experiências.

A todos os docentes pela dedicação e apoio técnico prestado neste período, em especial à professora Dra. Júnia Coutinho Anacleto Silva.

A CAPES pelo apoio financeiro ao projeto.

Aos funcionários do DC pelo carinho.

Finalmente, a todos aqueles que, de alguma forma, colaboraram, direta ou indiretamente, na realização deste projeto.

Meus sinceros agradecimentos

Sumário:

Siglas:

CASE	**C**omputer **A**ided **S**oftware **E**ngineering
CMM	**C**apability **M**aturity **M**odel
.COB	Extensão para arquivos COBOL
COBOL	**C**ommom **B**usiness **O**riented **L**anguage
DCL	**D**ata **C**ontrol **L**anguage
DLL	**D**ynamic **L**ink **L**ibraries
DML	**D**ata **M**anipulation **L**anguage
FaPRE/OO	**Fa**mília de **P**adrões para **RE**engenharia **O**rientada a **O**bjetos
FAROOL	**F**erramenta de **A**poio à engenharia **R**eversa **O**rientada a **O**bjetos de **L**egados
FD	**F**ile **D**ata
MAS	**M**odelo de **A**nálise do **S**istema
MASA	**M**odelo de **A**nálise do **S**istema **A**tual
RAD	**R**apid **A**pplication **D**evelopment
SGBD	**S**istema **G**erenciador de **B**anco de **D**ados
SQL	**S**tructured **Q**uery **L**anguage
UML	**U**nified **M**odeling **L**anguage

Lista de Figuras:

Lista de Tabelas:

Lista de Quadros:

Resumo:

Este projeto objetiva o desenvolvimento de uma ferramenta de apoio ao processo de engenharia reversa orientada a objetos a partir de sistemas legados procedimentais implementados em *COBOL*, de modo a facilitar a recuperação desses. A **F**erramenta de **A**poio à engenharia **R**eversa **O**rientada a **O**bjetos de **L**egados, denominada **FAROOL**, é um guia aos engenheiros de software durante o processo de engenharia reversa baseando-se em heurísticas pré-determinadas para que modelos orientados a objetos sejam obtidos a partir de sistemas implementados em COBOL. **FAROOL** se preocupa com duas fases, em especial, no processo de engenharia reversa orientada a objetos. A primeira fase é a de Elaboração do Modelo de Análise do Sistema Atual (MASA), quando um modelo pseudo-orientado a objetos é criado a partir do sistema legado. A segunda fase é a de Elaboração do Modelo de Análise do Sistema (MAS), quando um modelo totalmente orientado a objetos é criado, abstraindo-se o modelo anterior (MASA). Através da FAROOL, os engenheiros de software são guiados para que o processo de engenharia reversa se concretize. Primeiramente, é feita a preparação do sistema, com auxílio da ferramenta *CASE Legacy Aid*, integrada à **FAROOL** através de macros. Seguem a determinação das classes, dos atributos e dos métodos candidatos do modelo MASA, que podem se tornar classes, atributos e métodos, respectivamente, do modelo MAS. Heurísticas especiais são seguidas para a confecção dos relacionamentos entre as classes no modelo orientado a objetos. Todas as informações são armazenadas em um banco de dados relacional, facilitando a tarefa do engenheiro de software no processo de engenharia reversa. O ambiente *Delphi*; o banco de dados relacional *Interbase* e a ferramenta *Legacy Aid* foram utilizadas na elaboração da FAROOL.

Abstract:

This project objectives the development of a support tool to the reverse engineering process for procedimental legacy systems implemented in *COBOL,* in order to be facilitating their recovery. The **FAROOL** (**F**erramenta de **A**poio à engenharia **R**eversa **O**rientada a **O**bjetos de **L**egados) is a guide to software engineers in the reverse engineering process that is based on pre-defined heuristics, so that guided models to the objects are obtained from systems implemented in COBOL. FAROOL addresses two phases, in special, in the process of object guided reverse engineering. The first phase is the Elaboration of the MASA (Modelo de Análise do Sistema Atual, when a legacy systems model of pseudo-guided objects is created. The second phase is of MAS Elaboration (Modelo de Análise do Sistema), when a totally model of guided objects is create, being based on the previous model (MASA). By applying the FAROOL, the software engineers are guided so that the reverse engineering process is materialized. First, the preparation of the system is made, aid of the Legacy Aid CASE tool, integrated to the FAROOL tool through macros.They follow the determination of the candidates classes, of the candidates attributes and of the candidates methods of the MASA model, that can become candidates to the class, to the attributes and to the methods, respectively, of the MAS model. Special heuristics are followed for the confection of the relationships between the candidate class in the model of guided objects. All the information are stored in a relational database, facilitating to the task of the software engineer in the reverse engineering process. The Delphi environment; the relational database Interbase; and the Legacy Aid tool were used in the elaboration of the FAROOL.

Capítulo 1

Introdução

1.1 Considerações Iniciais

Muitas organizações fazem uso de softwares legados, os quais desempenham tarefas essenciais para o andamento dessas. Porém, são softwares normalmente implementados em linguagens ultrapassadas, mal documentados e que exigem manutenção constante e dispendiosa em tempo e em dinheiro. O desenvolvimento de um novo software para esses casos nem sempre é recomendado, pois não é possível recuperar as regras de negócio embutidas nele. Assim, a engenharia reversa surge como uma possibilidade de recuperar um modelo em alto nível de abstração para facilitar a manutenção desses sistemas. Pode-se, também, optar por esse processo como a primeira fase de um processo de reengenharia em que o sistema pode ser implementado em um novo paradigma, mais atual, com ou sem mudança de linguagem. Os gastos com manutenção de software, inclusive melhorias e adaptações, correspondem de 50 a 90 por cento dos gastos totais durante o ciclo de desenvolvimento do sistema (Vilanova 1997).

Devido ao rápido avanço da tecnologia, a vida útil dos softwares vem diminuindo cada vez mais, necessitando de manutenção, com redução de tempo e de dinheiro, porém com qualidade (Bennet 1995). Assim, o processo de reengenharia de sistemas permite recuperar o modelo de análise/projeto do sistema existente, de forma a não desperdiçar recursos.

A engenharia de sistemas deve suportar a transição de um sistema legado para um sistema mais atualizado, através do mapeamento das suas necessidades, cuidando da coexistência entre esses sistemas. Antes de iniciar um processo de engenharia reversa deve-se analisar alguns pontos: quais os esforços que serão despendidos para recuperar o sistema, o porquê de se aplicar a engenharia reversa, se existem ferramentas automatizadas

disponíveis no mercado, e assim por diante. Atualmente, existem poucas ferramentas que apóiam o processo de engenharia reversa orientada a objetos. A maioria delas apóia a re-construção do banco de dados e não a do sistema.

Este capítulo apresenta na seção 1.2 os objetivos desta dissertação. A motivação e relevância do trabalho são discutidas, respectivamente, nas seções 1.3 e 1.4. A forma como a dissertação está organizada é apresentada na seção 1.5.

1.2 Objetivos

A pesquisa refere-se à elaboração de uma ferramenta de apoio computacional ao processo de engenharia reversa orientada a objetos a partir de sistemas legados implementados em *COBOL*, gerando modelos orientados a objetos. Posteriormente, esses sistemas podem passar por reengenharia sendo implementados em linguagens orientadas a objetos. Ferramentas *CASE,* já existentes, como a *LegacyAid*® (Legacy, 2001), as heurísticas pré-determinadas por Camargo (2001a) para engenharia reversa orientada a objetos de sistemas COBOL, o ambiente *Delphi*® e o Sistema Gerenciador de Banco de Dados (SGBD) *Interbase*® são os elementos usados na elaboração da ferramenta proposta.

A funcionalidade do sistema legado é recuperada com o auxílio da ferramenta *CASE LegacyAid*®. O engenheiro de software, então, inicia o processo de composição do modelo orientado a objetos selecionando os candidatos a classes, atributos e métodos a partir do código procedimental. A ferramenta proposta, denominada FAROOL (**F**erramenta de **A**poio à Engenharia **R**eversa **O**rientada a **O**bjetos de **L**egados) consiste em armazenar esses pseudo-dados passo a passo, através da interação do engenheiro de software, por meio da interface auto explicativa implementada em *Delphi*® (Sonnino, 2002). Essas informações são armazenadas em um banco de dados, no *Interbase*® (Mecenas, 2002*)*, para posterior análise e recuperação de dados. Demais detalhes, pertinentes à inserção, à manipulação e à consulta de dados, tornam-se transparentes ao usuário.

1.3 Motivação

A motivação para a realização deste trabalho fundamentou-se na quantidade de sistemas legados de difícil manutenção, decorrente de uma documentação deficiente e da

necessidade de sua evolução para outras linguagens/ambientes. Os sistemas de informação implementados em *COBOL* exemplificam bem esse quadro. Basicamente, esses sistemas possuem alguns problemas, em particular, manutenção dispendiosa em tempo e em dinheiro, além da necessidade de se ter muitas pessoas destinadas ao entendimento dos mesmos. Aliado a isso, há o interesse em construir um apoio computacional, que integrado a uma ferramenta *CASE*, seja útil na recuperação de sistemas legados, apoiando a etapa da engenharia reversa orientada a objetos.

Um outro aspecto, não menos importante que o anterior, é a falta de ferramentas para apoiar a fase de engenharia reversa e reengenharia de sistemas. Algumas ferramentas existentes foram estudadas e pesquisadas bem como diversos artigos que tratam sobre elas, sendo consenso que não há uma que seja melhor do que outra. Nesse sentido, a primeira versão da FAROOL foi elaborada utilizando materiais já conhecidos e testados pela autora e demais membros do grupo de pesquisa ao qual está inserida.

1.4 Relevância

A relevância deste trabalho está no fato comentado na seção anterior, na escassez de ferramentas disponíveis no mercado que apóiam o processo de engenharia reversa orientada a objetos. Assim, este trabalho utiliza ferramentas e ambientes de desenvolvimento tradicionais e eficientes, com versões de demonstração que podem ser utilizadas, sem custos, para desenvolver a ferramenta proposta. O engenheiro de software necessita de poucos recursos computacionais para instalar e usar a FAROOL. Há necessidade que ele estude as heurísticas que conduzem o processo de engenharia reversa orientada a objetos. Pensando nisso, um guia acompanha a ferramenta visando facilitar a condução do processo pelo engenheiro de software menos habituado a ele. Assim, espera-se que os resultados obtidos sejam mais confiáveis em um menor espaço de tempo.

1.5 Organização da Dissertação

Esta dissertação está organizada da seguinte forma:

O **Capítulo 2** apresenta a revisão dos principais artigos técnicos aqui usados, abordando: engenharia reversa, reengenharia, linguagem *SQL*, ferramentas de engenharia

reversa e reengenharia, comparação de diversas ferramentas *CASE* que apóiam as etapas de engenharia reversa e reengenharia de sistemas implementados em *COBOL*, breve comentário sobre o ambiente de trabalho do *Delphi*® e o SGBD *Intrebase*®. No **Capítulo 3**, a arquitetura geral da ferramenta proposta, FAROOL, é descrita, evidenciando os componentes utilizados para sua construção. O **Capítulo 4** cuida da apresentação geral da FAROOL e apresenta as atividades que devem ser realizadas para que o engenheiro de software prepare um sistema que deve ser submetido ao processo de engenharia reversa orientada a objetos. No **Capítulo 5** são apresentadas as atividades para determinação do modelo de analise do sistema legado, início da modelagem orientada a objetos. No **Capítulo 6,** o engenheiro de software é orientado para que um modelo orientado a objetos seja obtido. No **Capítulo 7**, um estudo de caso, totalmente legado, pois não se conhecem os seus desenvolvedores, é parcialmente conduzido para exemplificar a utilização da FAROOL. No **Capítulo 8**, encontram-se os aspectos conclusivos do trabalho realizado e sugestões para trabalhos futuros. No **Apêndice I**, é apresentado o mapeamento entre os modelos MASA e MAS.

Capítulo 2

Revisão Bibliográfica

2.1 Considerações Iniciais

Os processos de engenharia reversa e reengenharia de software são de extrema importância visando, dentre outros aspectos, melhorar a qualidade e reduzir custos de manutenção de sistemas. São apresentadas neste capítulo diversas ferramentas que oferecem suporte à engenharia reversa e à reengenharia. Demais softwares usados para apoio à implementação de ferramentas de reengenharia orientada a objetos de sistemas legados, como o *Delphi*® (Sonnino, 2002), para implementação da interface, e o *Interbase*® (Mecenas, 2002), usado para especificar o banco de dados, também são apresentados neste capítulo, bem como a linguagem *SQL* (Oliveira, 2002) para criação do banco de dados. Esses softwares serão utilizados na ferramenta proposta nesta dissertação.

Este capítulo está organizado da seguinte forma: na seção 2.2. são feitas breves descrições a respeito de sistemas legados, reengenharia e engenharia reversa; na seção 2.3 são apresentadas abordagens para engenharia reversa; na seção 2.4 são apresentados quadros que sintetizam as principais características de algumas ferramentas que apóiam os processos de engenharia reversa e reengenharia de sistemas; na seção 2.5 o software *Delphi*® é apresentado; a seção 2.6 trata da linguagem SQL para armazenamento e manipulação das informações obtidas do processo de engenharia reversa orientada a objetos e a seção 2.7 do SGBD *Interbase*® e ferramentas auxiliares como o *IBConsole* para registro e criação do banco do projeto FAROOL. Na seção 2.8 são tecidas as considerações finais sobre o estudo realizado.

2.2 Engenharia Reversa e Reengenharia

Tanto a engenharia reversa quanto a reengenharia visam, normalmente, a melhoria da qualidade de sistemas desenvolvidos sem as técnicas existentes na engenharia de software bem como a recuperação de um modelo com nível mais alto de abstração. Esses sistemas atendem às necessidades dos usuários, mas suas manutenções são difíceis, sendo o código fonte, na maioria das vezes, a única documentação disponível. Podem estar implementados tanto em linguagens procedimentais como orientadas a objetos. Esses sistemas possuem um alto custo de manutenção, pois, além de não possuírem documentação, são suportados por equipes de desenvolvimento de software que não participaram do desenvolvimento dos mesmos (Pressman 2001).

A reengenharia é um processo de análise alterando a linguagem de programação e parcialmente a funcionalidade de um sistema. Inclui técnicas de engenharia reversa que se preocupam com a revitalização do sistema legado e de engenharia avante (Chicofsky 1990). A análise e/ou a alteração de sistemas são indicadas para softwares que ainda têm utilidade, mas devido à difícil manutenção, são necessários cuidados que proporcionem maior qualidade aos mesmos através da re-documentação, adaptação às novas tecnologias existentes no mercado, entre outras melhorias.

Segundo Jacobson (1991), a reengenharia é vista como um conjunto de cenários responsável pelas transformações de softwares procedimentais em softwares orientados a objetos devido às facilidades que o paradigma orientado a objetos apresenta. Segmentação é um processo de reengenharia somente com a mudança de paradigma, de procedimental para orientado a objetos, preservando-se as funcionalidades do software e a linguagem de programação já existente (Penteado 1999). Ou seja, documenta-se o sistema com modelos orientados a objetos e algumas alterações são realizadas no código fonte para que esse possa ficar com algumas características de orientação a objetos. Esse tipo de reengenharia permite que os mantenedores continuem utilizando a linguagem de programação que estão habituados, além de facilitar a manutenção, pois o código fonte fica organizado de acordo com os critérios de coesão e acoplamento e segue os princípios da orientação a objetos.

Em Sommerville (1995), de acordo com a abrangência do processo de reengenharia em relação a um determinado software, ele pode ser categorizado como: (a) r**eestruturação do programa -** necessária quando a estrutura do software é desprovida de uma organização ou está corrompida, como por exemplo, funções duplicadas no código, ou até mesmo, funções que nunca são chamadas; (b) **conversão do código fonte -** quando o código de um sistema escrito em uma determinada linguagem é convertido para outra. No processo de reengenharia considera-se a evolução do ambiente de programação, a arquitetura dos computadores, os recursos disponíveis e a mudança do paradigma de desenvolvimento. Uma das principais contribuições do processo de reengenharia orientada a objetos em sistemas legados procedimentais é a revitalização de sistemas para a orientação a objetos. Existem na literatura várias categorizações para reengenharia orientada a objetos, dentre elas destaca-se a de Jacobson (1991):

- Reengenharia Parcial Sem Alteração da Funcionalidade: a modelagem de parte do sistema é orientada a objetos e outra continua procedimental, porém não se altera a funcionalidade do mesmo. Nesse caso, é importante que as linguagens de programação das duas partes se comuniquem, por exemplo, C e C++.

- Reengenharia Completa Sem Alteração da Funcionalidade: trata-se da preparação do modelo de análise, passando pelo mapeamento de cada objeto de análise para a implementação do software antigo até o re-projeto do software, utilizando-se uma técnica orientada a objetos e linguagem orientada a objetos, sem que se altere a funcionalidade original.

Jacobson classifica **Reengenharia Com Total Alteração da Funcionalidade** como sendo o processo normal de engenharia avante. Neste trabalho esse conceito é utilizado somente como de engenharia avante e não reengenharia.

O processo de engenharia reversa consta da análise de um sistema para identificar os seus componentes e inter-relacionamentos, com o intuito de criar uma outra forma de representação, em um nível mais alto de abstração (Pressman 2000). Esse processo tem como objetivo a recuperação da documentação e dos modelos inerentes às etapas de análise

e projeto do ciclo de vida de um sistema. Através dela, pode-se ter uma melhor manutenção do sistema.

2.3 Abordagens para Engenharia Reversa

2.3.1 Método *Fusion/RE*

Em Penteado (1996b), *Fusion/RE* é uma proposta de engenharia reversa orientada a objetos, que contém um conjunto de passos para que a partir de sistemas legados procedimentais obtenha-se um modelo de análise orientado a objetos, sendo a funcionalidade do sistema mantida. Inicialmente, quatro passos foram elaborados para *Fusion/RE* e os modelos do método *Fusion* (Colleman, 1994) foram utilizados para a modelagem orientada a objetos.

- *Revitalização da Arquitetura do Sistema*: análise do código legado para que se obtenha completo entendimento do sistema que está sendo submetido à engenharia reversa.

- *MASA (Modelo de Análise do Sistema Atual)*: elaboração de um modelo com pseudo-classes, visando a abstração das estruturas de dados do legado para objetos e os procedimentos para métodos.

- *MAS (Modelo de Análise de Sistemas)*: criação de um modelo de análise orientado a objetos, abstraído a partir do *MASA*, dando ênfase ao domínio da aplicação e não à implementação.

- *Mapeamento MASA x MAS*: equivalência entre o modelo de análise do sistema orientado a objetos e o modelo de análise do sistema legado atual, mostrando como os elementos que devem estar presentes no *MAS* estão atualmente implementados no sistema legado.

Posteriormente, dois passos foram incorporados ao *Fusion/RE* para que a segmentação de sistemas fosse realizada (Penteado, 1998). Ou seja, para que o código continuasse implementado na mesma linguagem, porém com características de encapsulamento de dados e métodos bem organizados, consultando e/ou alterando somente a estrutura de dados.

2.3.2 Processo PRE/OO

Em Lemos (2002), é apresentada uma proposta de evolução do método *Fusion/RE* (Penteado 1996a), (Penteado 1996b) e (Penteado 1998). Consiste em um Processo de REengenharia Orientada a Objetos (PRE/OO) que utiliza a UML (OMG, 2003) para representar os modelos orientados a objetos. O modelo de processo utilizado foi alterado de seqüencial linear para evolutivo. Além disso, as *KPAs (Key Process Area)* do Nível 2 de maturidade do *SW-CMM (Capability Maturity Model for Software)* (Curtis et al., 1995) são utilizadas visando o controle de qualidade durante o processo realizado.

O processo é descrito através de padrões (soluções pesquisadas, testadas e que visam resolver problemas que normalmente ocorrem em determinadas situações), sendo utilizada a Família de Padrões para REengenharia Orientada a Objetos (FaPRE/OO), proposta em Recchia (2002a). Os padrões seguem o formato proposto na Linguagem de padrões de Engenharia Reversa e Reengenharia (Demeyer et al., 2000).

O processo PRE/OO está organizado em sete clusters de padrões divididos em dois grupos principais: padrões para melhoria da qualidade da reengenharia e padrões para a realização da reengenharia orientada a objetos.

PRE/OO é um processo genérico de reengenharia de sistemas legados procedimentais para orientados a objetos, sendo instanciado sistemas que usam *Delphi*® sem as características orientadas a objetos para sistemas com essas características (Lemos 2002).

2.3.3 Aplicações do Método *Fusion/RE* em Sistemas *COBOL*

O método *Fusion/RE* (Penteado 1996a), (Penteado 1996b) e (Penteado 1998), pode ser aplicado a sistemas implementados em qualquer linguagem (Lemos 2002). Porém, para que a sua utilização seja otimizada, são necessárias algumas diretrizes.

Em Camargo (2001a) foram criadas algumas diretrizes específicas para conduzir o processo de reengenharia de sistemas procedimentais implementados em *COBOL* para sistemas orientados a objetos, implementados em *JAVA* que podem ser disponibilizados na *Web* utilizando *servelets*. Camargo et al (2002) mostram que os padrões da FaPRE/OO podem ser utilizados para condução de sistemas legados *COBOL* para *JAVA*. Porém, nesta

dissertação são utilizadas as heurísticas propostas para engenharia reversa (Camargo, 2001), apresentadas nas tabelas 1 e 2, por esta já está em andamento quando os padrões foram apresentados.

O Quadro 1 apresenta as heurísticas para a fase de Elaboração do MASA (Modelo de Análise do Sistema Atual).

Quadro 1 - Diretrizes para a Engenharia Reversa de Programas em COBOL - Fase 1 (Camargo, 2001)

Engenharia Reversa Orientada a Objetos de Programas COBOL Usando FUSION/RE (Fase 1)			
FASE	PASSO		
	No.	Nome	Descrição do Passo
1. Elaboração do MASA	1.1	Identificação dos principais FD's	Esses FD's são utilizados na elaboração do MASA
	1.2	Mapeamento dos FD's para pseudo-classes	Os FD's especificados geram pseudo-classes em UML
	1.3	Identificação de itens elementares e de grupo	Itens elementares: pseudo-atributos; itens de grupo: agregações
		1.3.1 Redefinição de item elementar em item de grupo (caso 1)	Será criada uma nova classe com a mesma semântica do atributo da classe todo
		1.3.2 Redefinição de item de grupo em outro (caso 2)	Um item de grupo (candidato à classe parte) com atributos em comum em, relação à classe todo
OBS: FD (File Description).		1.3.3 Redefinição de item de grupo geral em outro (caso 3)	Um item de grupo de âmbito geral, geralmente de nível 1, é redefinido em outro, gerando dois registros distintos
	1.4	Identificação de relacionamentos de associação	Dependência estrutural entre FD's resulta em associação
	1.5	Identificação dos pseudo-métodos e suas anomalias	Parágrafos originam pseudo-métodos e tem suas anomalias analisadas

O Quadro 2 apresenta as heurísticas utilizadas na fase de Elaboração do MAS (Modelo de Análise do Sistema).

Quadro 2 - Diretrizes para a Engenharia Reversa de Programas em COBOL - Fase 2 (Camargo, 2001)

Engenharia Reversa Orientada a Objetos de Programas COBOL Usando FUSION/RE			
2. Elaboração do MAS	2.1	Eliminação das pseudo-classes que representam datas	Eliminam-se as pseudo-classes que representam datas
	2.2	Identificação de relacionamentos de generalização/especialização	Analisam-se as agregações do MASA que se originam dos 3 casos da utilização do REDEFINES. Para cada caso, dá-se um tratamento diferente
		2.2.1- caso 1 – (Item 1.3.1)	Elimina-se o atributo que tem a mesma função da classe parte
		2.2.2- caso 2 – (Item 1.3.2)	Relacionamentos de agregação são substituídos pelos de generalização/especialização
		2.2.3- caso 3 – (Item 1.3.3)	Elimina-se a classe parte que é dependente de implementação
	2.3	Conversão de nomes	Classes, atributos e métodos têm seus nomes alterados para mnemônicos mais significativos
	2.4	Identificação de métodos	Cada pseudo-método tem seu nome alterado para mnemônicos mais significativos
	2.5	Refinamentos adicionais	São feitas alterações que podem facilitar a leitura e a compreensão do modelo

2.3.4 Técnicas de Engenharia Reversa Orientada a Objetos

Em Breuer (1991), são descritas técnicas de Engenharia reversa que produzem notações orientadas a objetos para sistemas de processamento de dados implementados em *COBOL*. Inicia-se a compreensão do sistema *COBOL* pela análise das declarações de entrada e saída (I/O), que devem ser relacionadas aos diagramas de estrutura de dados gerados pela *DATA DIVISION*. Três estágios devem ser cumpridos: limpeza ou reestruturação do código (*clean*), especificação (*specify*) e simplificação (*simplify*).

Limpeza ou reestruturação do código é o estágio no qual o programa *COBOL* é reestruturado em uma linguagem intermediária, denominada *Uniform*. Especificação é o estágio no qual são produzidas especificações relativas ao código fonte que podem ser facilitadas por meio de interações humanas no caso de dúvidas. Finalmente, a simplificação é o estágio no qual são aplicadas transformações automáticas sobre as especificações obtidas no segundo estágio, com o objetivo de torná-las mais legíveis. O Quadro 03 sumariza esses três estágios.

Quadro 3 - Estágios para Engenharia Reversa Orientada a Objetos de Sistemas COBOL

Estágio	*Características*
Limpeza ou *Ree* Programa.cob *(Clean)* Processo de reestruturação Linguagem Intermediária UNIFORM	
E. Código Fonte *(Specify)* Abstrair especificações Interação do Usuário *Simplificaç* Espec *(Simplify)* Especificações mais leg´ Transformações	

2.4 Ferramenta para Apoio à Engenharia Reversa e à Reengenharia

Ferramentas de engenharia reversa e reengenharia contribuem desde o entendimento de sistemas complexos até o auxílio da re-documentação do sistema, com relatórios que auxiliam a exclusão de código repetido, a avaliação e a recuperação do sistema legado.

Diversas ferramentas que apóiam as fases de engenharia reversa e reengenharia, total ou parcialmente, foram pesquisadas: Rigi® (2001), WithClass® (2001), Gupro® (2001), ANALSoftSpec® (2001), Bauhaus® (2001), CIAO® (2001), Acacia® (2001), Grasp® (2001), Refine-C® (2001), TKSee® (2001), DBMain® (2001), COSMOS® (2001), Dpute® (2001), RoadMap® (2001) e Legacy Aid® (2001). Os quadros de 04 a 09

apresentam as principais características das ferramentas, consideradas para esta dissertação, por cuidarem de código em COBOL. Detalhes das outras ferramentas podem ser encontrados em (Prado, 2002).

Quadro 4 - Ferramenta RIGI

Ferramenta	Características
Rigi®	**Linguagem:** Cobol, C e C++; **Apoio:** engenharia reversa e manutenção de sistemas legados; **Criação:** desenvolvida por um grupo de pesquisadores do Departamento de Ciência da Computação da Universidade de Victoria no Canadá; **Características:** é multi-plataforma: Sun SPARC, IBM RISC System 6000 e PC com windows 95 e NT; gratuita e gráfica; **Utilização:** divide-se em duas partes: uma realizada no DOS, responsável pela geração do arquivo.rsf, a partir do código fonte do sistema, para posterior leitura no editor gráfico da ferramenta; e outra no editor gráfico (RIGIEDIT) que carrega e manipula o modelo gráfico (arquivo.rsf); **Oferece:** relatórios estatísticos do modelo gráfico que ajudam no entendimento do programa e opções de navegação pelo modelo gráfico; **Vantagens:** fácil aquisição; distribuição gratuita; ocupa relativamente pouco espaço (aproximadamente 5Mb); dá suporte a manutenção de sistemas (Engenharia Reversa de sistemas legados), **Desvantagens:** modelo gráfico confuso para sistemas complexos; *help* não embutido na ferramenta; manual *online* com poucas informações a respeito das configurações internas da ferramenta e das variáveis de ambiente; uso dificultado quando se trabalha com sistemas grandes e multi-arquivos; ainda em desenvolvimento, portanto, constantes *uploads* devem ser feitos para atualizar a versão em uso. **Site:** http://www.rigi.csc.uvic.ca/

Quadro 5 - Ferramenta Legacy Aid

Ferramenta	Características

LegacyAid®	**Linguagem:** Cobol; **Apoio:** criação de diagramas de fluxo de controle e soluções para o bug do milênio; **Criação:** a CASEMaker Inc./Corporate eadquarters **Características:** ferramenta de análise, reengenharia e re-documentação para o entendimento de sistemas legados; **Utilização:** na recuperação de sistemas legados implementados especificamente em COBOL; **Oferece:** análise de impacto das modificações para o sistema; criação de diagramas lógicos e estruturais do programa; geração de métricas como o Diagrama de Kiviat e o relatório de complexidade; geração automática e impressão de mais de trinta tipos de documentos; editor inteligente para o código; revitalização do código através da reengenharia e outros; **Vantagens:** alta qualidade dos gráficos gerados para os códigos, incluindo os sistemas complexos; simulação passo-a-passo do programa via modelo gráfico; relatórios importantes para melhor análise do código; *help* bem estruturado, **Desvantagens:** aceita somente programas COBOL; uso relativamente dificultado devido à grande quantidade de recursos. **Site:** http://www.casemaker.com

Quadro 6 - Ferramenta Dpute

Ferramenta	Características
Dpute®	**Linguagem:** COBOL ; **Apoio:** foco nos dados e relacionamentos entre dados e no entendimento de sistemas legados cuja manutenção é dispendiosa dando suporte à engenharia reversa dos mesmos; **Criação:** desenvolvida por um laboratório de Engenharia de Software sob supervisão e orientação do professor Dr. W.T. Tsai; **Características:** gera um modelo gráfico chamado ***Module-level COBOL Dependence Graphs (MCDGs)*** a partir do código fonte de um sistema em COBOL; **Utilização:** no suporte à manutenção de sistemas legados em COBOL; **Oferece:** classificação automática de variáveis, laços do programa e análise de efeitos; **Vantagens:** incorporação de modelos, conceitos e algoritmos para aproximações de programas centrados em dados; trabalha com sistemas pequenos (poucas linhas de código) ou com sistemas mais complexos (multi-arquivos), **Desvantagens:** não divulgada em artigos científicos como uma ferramenta de destaque, e portanto, pouco se pode comparar com outras ferramentas que dão o mesmo suporte. **Site:** http://www.cs.umn.edu/Research/softeng/sm/dpute/dpute.tar.gz

Quadro 7 - Ferramenta DBMain

Ferramenta	Características
DBMain®	**Linguagem:** SQL, COBOL, CODASYL, IMS, RPG e XML; **Apoio:** engenharia reversa e engenharia avante de sistemas com aplicações de Banco de Dados; **Criação:** departamento de Ciência da Computação da Universidade de Namur; **Características:** representação de modelos: conceitual, lógico e físico de objetos; múltiplas visões (4 hipertextos e 3 visões lógicas); **Utilização:** em sistemas que possuem aplicações de Banco de Dados e fazem uso de representações em SQL; **Oferece:** um modelo genérico que descreve os componentes procedimentais dos sistemas de informação; anotações técnicas e semânticas de cada objeto especificado; caixa de ferramenta para normalização do modelo conceitual; geração de código com SQL e relatórios técnicos; **Vantagens:** voltado para dar suporte à engenharia reversa e à engenharia avante, melhorando manutenção de sistemas legados; apóia sistemas com aplicações de banco de dados, oferecendo recursos para geração de modelos conceitual, lógico e físico, **Desvantagens:** muito voltado para sistemas que contém aplicações de Banco de Dados; pouco divulgado em artigos científicos brasileiros (mais conhecido nas Universidades canadenses). **Sites:** http://www.fundp.ca.be/

Quadro 8 - Ferramenta RoadMap

Ferramenta	Características
RoadMap®	**Linguagem:** COBOL ; **Apoio:** re-documentação de sistemas; **Criação:** pela **RainCode Corporation**; **Características:** gera uma documento em *html* com cores para melhor enfatizar o código (*SQL* e comentários) e as *tags* para uma boa navegação pelo código; multi-usuários (permite a definição de grupos de usuários específicos para cada projeto); **Utilização:** para sistemas em COBOL e cuja manutenção é um fator de risco para permanência do sistema no mercado; **Oferece:** referências cruzadas em relação a tabelas *SQL*, arquivos e mapas; **Vantagens:** apóia o processo de re-documentação de sistemas em COBOL; oferece, como resultado, um documento em *html*, **Desvantagens:** não gera um modelo gráfico. **Site:** http://www.raincode.com/cobolroad/

Quadro 9 - Ferramenta COSMOS

Ferramenta	Características
Gupro®	**Linguagem:** COBOL **Apoio:** entendimento de diferentes tipos de aplicações; **Criação:** desenvolvida por Emendo Software Group; **Características:** disponível para *download* para plataforma Windows 95, 98 e NT; ocupa aproximadamente 15Mb; podem ser observados dois níveis principais na ferramenta: um referente ao sistema e outro referente aos componentes do sistema; **Utilização:** normalmente para testar software e analisar a qualidade de sistemas; **Oferece:** uma visão geral dos componentes do sistema, bem como os relacionamentos entre eles; uma visão estrutural do sistema a ser analisado; diagrama de fluxo, estrutura de dados, referências e métricas de qualidade e traçado ponto a ponto do programa COBOL; **Vantagens:** voltada para reengenharia e engenharia reversa de sistemas legados; o *help* é bem estruturado e de fácil navegação; possui qualidade nas métricas voltadas para análise e teste de sistemas legados, **Desvantagens:** apesar de ser divulgada no meio acadêmico, visa testes de software e não propriamente aplicações de engenharia reversa. **Site:** http://www.emendo.com

Os autores Bellay e Gall (Belley, 1998) testaram e avaliaram ferramentas de engenharia reversa, tais como: *Refine-C® versão1.1a, Imagix® 4D Release 2.7, Rigi® e SniFF® for C/C++ versão 2.3.1*, segundo os seguintes critérios: capacidade de análise do código fonte, representação gráfica (gráfica e textual), edição, navegação, plataforma e multi-usuários. Concluíram que não há uma que possa ser considerada a melhor dentre as quatro. Cada uma possui características peculiares importantes que as tornam capazes de auxiliar na engenharia reversa de um sistema, embora estejam restritas ao contexto que se aplicam.

Após os estudos feitos optou-se, neste trabalho, pelo uso da ferramenta *Legacy Aid*® por: permitir perfeita interação com o SGBD *Interbase*®, possuir versão *DEMO* disponível e trabalhar com a linguagem *COBOL* que é a linguagem alvo. Assim, informações complementares da ferramenta *Legacy Aid*® são apresentadas a seguir.

A *Legacy Aid*® (Legacy Aid, 2001) é uma ferramenta de analise, de reengenharia e de re-documentação para o entendimento de sistemas legados. Desenvolvida por *CASEMaker Inc./Corporate Headquarters,* trabalha com sistemas em *COBOL*, conhecidos

como “sistemas caixas-pretas”. Reduz os efeitos e custos de manutenção de sistemas em aplicações *COBOL* utilizando: ***Acucobol, ANSI COBOL, IBM COBOL II, Microfocus COBOL, e Tandem COBOL***, incluindo arquivos ***BM JCL, CICS/MVS, BMS, DB2, e Tandem SQL Obey.***

O projeto gerenciador da *LegacyAid*® permite a manipulação/visão, a análise, a reengenharia, a re-documentação de um sistema e o gerenciamento de tarefas através da orientação a objetos, com a definição de classes e objetos, Figura 01.

A área de trabalho permite visão do sistema que gerencia objetos (usuários, grupos e projetos) e visão de projeto mostrando os objetos relacionados (Candidatas, Dicionário de Dados, Arquivos e Dicionário das Classes de Operações do Projeto). Cada projeto tem seu proprietário (usuário que o definiu) e os membros (equipe de desenvolvimento).

A ferramenta proporciona aos desenvolvedores de sistemas em *COBOL*, através de interface gráfica, outros benefícios como: rápido entendimento da lógica do programa; localização eficiente de objetos no código; impacto das modificações; custos para correção dos erros e como os dados caminham pelo sistema.

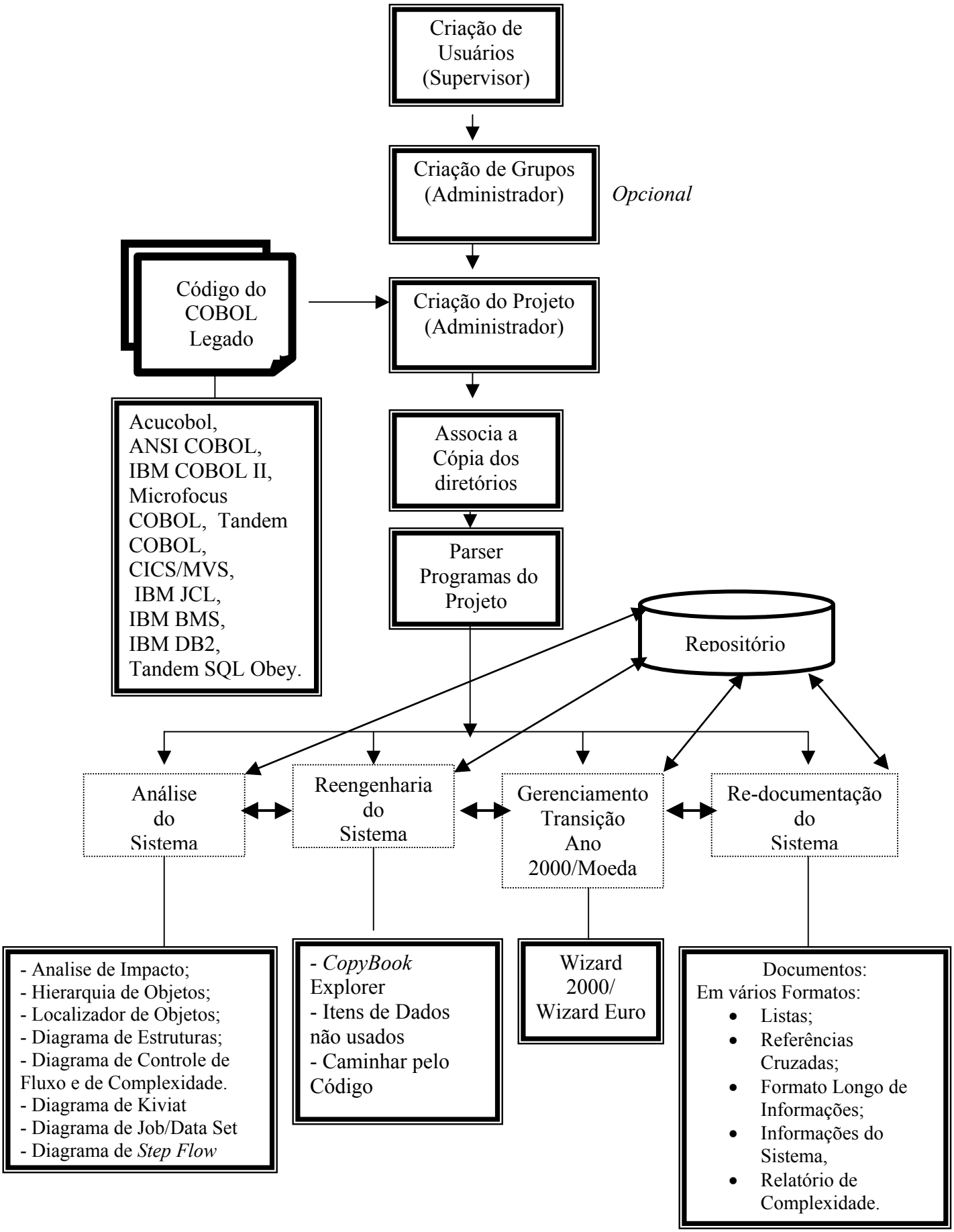

Criação de Usuários (Supervisor)
Criação de Grupos (Administrador)
Opcional
Código do COBOL Legado
Criação do Projeto (Administrador)
Acucobol, ANSI COBOL, IBM COBOL II, Microfocus COBOL, Tandem COBOL, CICS/MVS, IBM JCL, IBM BMS, IBM DB2, Tandem SQL Obey.
Associa a Cópia dos diretórios
Parser Programas do Projeto
Repositório
Análise do Sistema
Reengenharia do Sistema
Gerenciamento Transição Ano 2000/Moeda
Re-documentação do Sistema
- Analise de Impacto;
- Hierarquia de Objetos;
- Localizador de Objetos;
- Diagrama de Estruturas;
- Diagrama de Controle de Fluxo e de Complexidade.
- Diagrama de Kiviat
- Diagrama de Job/Data Set
- Diagrama de Step Flow
- CopyBook Explorer
- Itens de Dados não usados
- Caminhar pelo Código
Wizard 2000/ Wizard Euro
Documentos:
Em vários Formatos:
Listas;
Referências Cruzadas;
Formato Longo de Informações;
Informações do Sistema,
Relatório de Complexidade.

Figura 1 - Visão Geral da LegacyAid®

2.5 Delphi® 6.0

Delphi® (Sonnino, 2002) é um produto de alta performance da ***Borland*** (Borland, 2003) para plataforma *Windows*, único em sua categoria, combinando códigos totalmente compiláveis, ferramentas visuais e tecnologia para a composição de bases de dados escaláveis, além de possuir facilidades para um rápido desenvolvimento em plataforma *Windows* e aplicações *client/server*. Pode ser encontrados dois produtos:

- ***Delphi® Client/Server***, de alta performance e facilidade para o desenvolvimento de aplicações e suporte a bancos de dados do tipo Cliente/Servidor, e
- ***Delphi® Desktop***, de alta performance e facilidade para o desenvolvimento de aplicações e suporte a bancos de dados locais, permitindo total portabilidade à versão ***Client/Server***.

Delphi® oferece dois níveis de programação distintos: de *designer*, que utiliza os recursos de programação visual e aproveita componentes prontos, e o de *component writer*, que escreve os componentes para o designer utilizar em suas aplicações. Pode-se dizer que o *component writer* permite a programação em um nível mais baixo e o *designer* em um nível mais alto.

O ambiente *Delphi®* foi selecionado para implementação do FAROOL, pois está entre os ambientes **RAD** (Rapid Application Development) mais utilizados para desenvolvimento de sistemas de informação no Brasil, (Byte Brasil, 1997).

A linguagem base do ambiente *Delphi®* é *Object Pascal*, que permite o desenvolvimento de software tanto no paradigma procedimental (estruturado) quanto no orientado a objetos. Sua interface gráfica é orientada a eventos facilitando o desenvolvimento de interfaces interativas.

Entre as características principais do ambiente *Delphi*® e que fazem dele um dos mais utilizados, podemos citar:

- Construção de *DLLs (Dynamic Link Libraries)*
- Velocidade dos aplicativos implementados em *Delphi*® é praticamente a mesma dos desenvolvidos em C ou C++.
- Construção de objetos reutilizáveis, obedecendo ao paradigma orientado a objetos.

Os programas desenvolvidos em *Delphi*® são divididos em código fonte chamados de unit. Para cada programa, existe um cabeçalho de especificações com o seu nome, seguido de uma cláusula opcional uses e de uma estrutura de bloco de declarações e comandos.

A cláusula uses lista as units que serão associadas ao código fonte. Esse comportamento se assemelha à uma declaração include em C, ou import em JAVA. Os componentes estruturais do *Delphi*® são apresentados no Quadro 10.

Do ponto de vista do ambiente *Delphi*®, a partir da versão 5, o acesso e a manipulação dos dados são realizados através do mecanismo *BDEAdministrator,* para acesso a banco de dados relacionais locais ou em máquinas remotas. Esse mecanismo permite a abstração da forma como os dados serão armazenados fisicamente, isto é, em tabelas locais ou em um gerenciador de banco de dados. Na aplicação, esses dados são controlados pelo componente *Dataset*, (tabela lógica).

Utilizando o *BDEadministrator*, os seguintes componentes poderão ser usados na aplicação:

- TTable: corresponde às tabelas (*Tables*) armazenadas em um banco de dados remoto ou em arquivos locais.
- TQuerys: permitem a manipulação dos dados através da definição de *querys* e da linguagem *SQL* para consulta, inserção e/ou adaptação no banco.

- TStoredProc: correspondente ao *Stored procedures* que reúne instruções *SQL* declaradas e armazenadas em um servidor de banco de dados.
- TNestedTable: corresponde ao *Nestes datasets* que são registros no SGBD *Oracle8* em forma de conjuntos aninhados.

Quadro 10 - Principais Arquivos de um Projeto desenvolvido em Delphi®

Tipo de Arquivo Gerado	Definição - Componente	**Caracterísitcas - Projeto**
<<.pas>>	Cada unit está associada a um arquivo, que contém o código fonte do projeto implementado em Pascal Orientado a Objetos.	**Permite o compartilhamento entre programas (entre outras units), a distribuição de componentes entre desenvolvedores, sem necessitar do código fonte, apenas o objeto (.DCU).**
<<.drp>>	É o principal do projeto, contém a lista com todas as units usadas e é responsável pela inicialização da aplicação.	**Representa o programa principal da aplicação, o main, que referencia às units.**
<<.dfm>>	Contém as descrições dos controles do Form (interface do aplicativo) com suas respectivas propriedades.	**Contém as informações do formulário vinculado a uma unit.**
<<.dcu>>	Corresponde a uma unit compilada.	**Resultado da compilação do arquivo <<.pas>>.**
<<.dof>>	Contém as opções do projeto.	**Contém as opções como diretivas, diretórios e demais opções para compilação.**
<<.res>>	**Contém os recursos do projeto.**	**Representam ícones e bitmaps utilizados na aplicação.**

Dentre os principais componentes visuais oferecidos pela ferramenta, destacam-se o DataControls, como: DBText, DBGrid, DBEdit, DBNavigator, DBListBox, DBCombobox, entre outros. Para que esses componentes atualizem ou exibam dados, eles precisam estar em um TForm. Eles são os meios para se construir uma interface, permitindo o estabelecimento de uma comunicação com o usuário da aplicação desenvolvida em *Delphi®*.

O formulário pode ser visto como um objeto que contém outros objetos. Uma aplicação ficará localizada em um formulário principal, que interage com outros formulários criados.

Os controles visuais que atualizam dados estão divididos em controles que representam: um campo simples, como o DBText e o DBEdit; um conjunto de registros, como o DBGrid e controles de navegação, como o TDBNavegator.

A ferramenta *Delphi*® trabalha com três camadas lógicas: apresentação, lógica e de acesso a dados, estabelecendo três categorias de componentes: visuais, de acesso à base de dados e de ligação.

- Componentes Visuais – são responsáveis pela interface com o usuário e estão disponíveis em Data Controls (paleta de componentes).
- Componentes de Acesso à Base de Dados – são responsáveis por criar toda a estrutura necessária para acessar e manipular o banco de dados e estão disponíveis em Data Access (paleta de componentes).
- Componentes de Ligação – são responsáveis pela interface entre as duas camadas acima. Torna os componentes visuais independentes dos componentes de acesso. O principal componente de ligação é o TDataSource, disponível em Data Access (paleta de componentes).

2.6 Linguagem SQL

A linguagem *SQL* (Oliveira, 2002) apresenta uma série de comandos que permitem a definição dos dados, a chamada de *DDL* (*Data Definition Language*), composta, entre outros, pelos comandos Create, que é destinado à criação do banco de dados, das tabelas que o compõem, além das relações existentes entre as tabelas. Como exemplo de comandos da classe *DDL*, têm-se os comandos Create, Alter e Drop.

Os comandos da série *DML* (*Data Manipulation Language*) são destinados a consultas, inserções, exclusões e alterações em um ou mais registros de uma ou mais tabelas de maneira simultânea. Como exemplo de comandos da classe *DML,* têm-se os comandos Select, Insert, Update e Delete. Uma subclasse de comandos *DML*, a *DCL* (*Data Control Language*), dispõe de comandos de controle como Grant e Revoke.

A Linguagem *SQL* tem como características importantes sua capacidade de: (1) gerenciar índices, sem a necessidade de controle individualizado de índice corrente, algo muito comum nas linguagens de manipulação de dados do tipo registro a registro; (2) construção de visões, que são formas de visualizar os dados como listagens independente das tabelas e da organização lógica dos dados; (3) cancelar uma série de atualizações ou gravá-las, depois do início de uma seqüência de atualizações. Os comandos Commit e Rollback são responsáveis por estas facilidades.

A linguagem *SQL* consegue implementar estas soluções, somente pelo fato de estar baseada em banco de dados, que garantem por si mesmo a integridade das relações existentes entre as tabelas e seus índices.

2.7 Sistema Gerenciador de Banco de Dados (SGBD) *Interbase*®

O *Interbase*® (Mecenas, 2002) é um Sistema de Gerenciamento de Banco de Dados (SGBD) Cliente/Servidor que é compatível com *SQL-ANSI-92* e foi desenvolvido para ser um banco de dado independente de plataforma e de sistemas operacionais. Várias ferramentas de auxílio compõem o *Interbase*®, dentre elas, destacam-se:

- *Documentation* – funciona como um guia para quem ainda não conhece o *Interbase*®.
- ***IBConsole*** – ferramenta onde o banco pode ser criado, consultado e manipulado. Oferece suporte à linguagem *SQL* para esses feitos em um recurso chamado *Interactive SQL*.
- ***Interbase® Replication Manager*** – gerencia as replicações de um banco de dados.
- ***Interbase® Replication Server*** – é um servidor de replicações.

O SGBD *Interbase*® dispensa o uso de grandes servidores, utiliza pouco espaço em disco para sua instalação e pouca memória em situações normais de uso. Por isso, a plataforma necessária para a sua instalação e utilização pode ser reduzida, diminuindo, consideravelmente, os custos de um projeto.

Ao longo de seu desenvolvimento, foram introduzidas muitas características, entre elas: acesso nativo a *driver* **JDBC**; commit automático de duas fases; sombreamento do banco de dados; replicação; tratamento de *Blob´s*, e Sistema de eventos. Outra grande vantagem do *Interbase*® é o fato dele ser multiplataforma, ou seja, funciona em vários sistemas operacionais, dentre eles destacam-se: *Windows 9x; Windows NT; Windows 2000; Windows XP; Linux, e Solaris.*

O *IBConsole* é o gerenciador de dados que acompanha o *InterBase*®. Toda e qualquer criação, relacionamento, manutenção, é feito no *ISQL*, via linha de comando. No *IBConsole* o usuário master é "*SYSDBA*" e a sua senha é "*masterkey*", que permite criar um novo banco de dados ou registrar um já existente.

2.8 Considerações Finais

Este capítulo teceu considerações sobre reengenharia e, em especial, sobre engenharia reversa, bem como apresentou abordagens que possibilitam a engenharia reversa orientada a objetos de sistemas legados procedimentais: *Fusion/RE*, *PRE/OO* e as heurísticas para sistemas legados *COBOL*. Foi apresentado um panorama de algumas ferramentas existentes que apóiam a engenharia reversa e reengenharia, sendo o enfoque principal para as ferramentas que apóiam sistemas legados implementados ma linguagem *COBOL*, que são os sistemas alvo neste trabalho.

As ferramentas *Rigi, Legacy Aid®, Dpute, DBMain, Gupro* e *COSMOS* foram as principais *CASE*(s) pesquisadas para a determinação da melhor ferramenta de engenharia reversa que atendesse às necessidades deste trabalho. Merece especial atenção à ferramenta *Legacy Aid*®, que oferece diversos recursos para análise do código fonte de sistemas legados em *COBOL* e permite visões que ajudam no entendimento do sistema legado bem

como na sua recuperação. A *Legacy Aid*® foi escolhida para ser integrada à FAROOL para a realização de parte do processo de engenharia reversa devido aos recursos que oferece como: referências cruzadas, grafo de fluxo de dados e vários relatórios que auxiliam na documentação.

O ambiente *Delphi*® é um produto da *Borland* para implementação de aplicações visuais com armazenamento de informações, bem como consultas e inserções em banco de dados. Neste trabalho, foi utilizada para a implementação de uma interface interativa, com vários objetos visuais para melhor satisfazer o usuário e para a associação dessa interface com o SGBD *Interbase*®.

O SGBD *Interbase*®, de distribuição gratuita e de fácil integração com a ferramenta *Delphi*® que tem o *IBConsole* como ferramenta auxiliar, também foram utilizados neste projeto. A linguagem para criação, manipulação, consulta e inserção no banco de dados é a *SQL*.

O próximo capítulo apresenta a arquitetura da ferramenta FAROOL e os produtos utilizados para o desenvolvimento da mesma.

Capítulo 3

Arquitetura da Ferramenta FAROOL

3.1 Considerações Iniciais

A ausência de ferramentas que auxiliam no processo de engenharia reversa orientada a objetos motivou o tema desta dissertação. Dessa forma, foi investigada a possibilidade de integrar ferramentas existentes para apoiar a realização da etapa de engenharia reversa no processo de reengenharia orientada a objetos a partir de sistemas legados implementados em *COBOL*. O resultado dos estudos realizados propiciou a confecção de uma **F**erramenta de **A**poio à engenharia **R**eversa **O**rientada a **O**bjetos de **L**egados (FAROOL), considerando heurísticas pré-determinadas por Camargo (2001a) para sistemas implementados em *COBOL*. A sigla FAROOL será utilizada a partir desse momento e no restante desta dissertação sempre que necessário referenciar a ferramenta de apoio à engenharia reversa orientada a objetos.

Este capítulo apresenta na seção 3.2 a arquitetura geral da FAROOL, considerando: sua arquitetura externa; sua estrutura em camadas e sua visão geral. Na seção 3.3 é apresentada como foi a construção da FAROOL, sua integração com a ferramenta *CASE Legacy Aid*®; o metamodelo definido para a geração do banco de dados; o estudo e o uso das heurísticas pré-determinadas para automação parcial do processo de engenharia reversa. Na seção 3.4 são apresentadas as considerações finais.

3.2 Arquitetura da Ferramenta FAROOL

A FAROOL foi desenvolvida utilizando os conceitos e fundamentos definidos em heurísticas pré-determinadas em Camargo (2001a) que foram apresentadas no Capítulo 2. A Figura 2 mostra a arquitetura geral da FAROOL e a Figura 3 suas entradas e saídas.

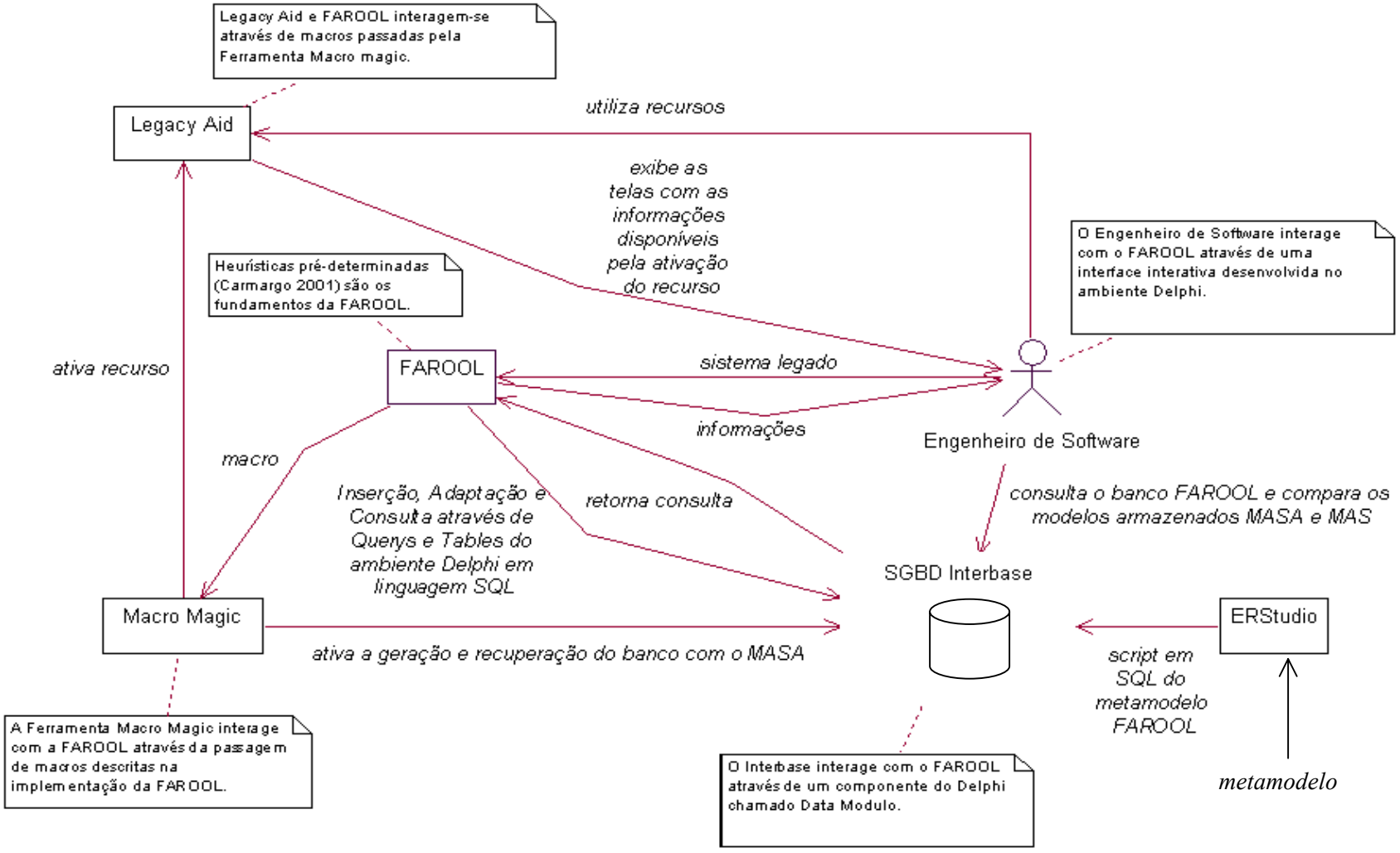


Figura 2 - Arquitetura Geral da FAROOL

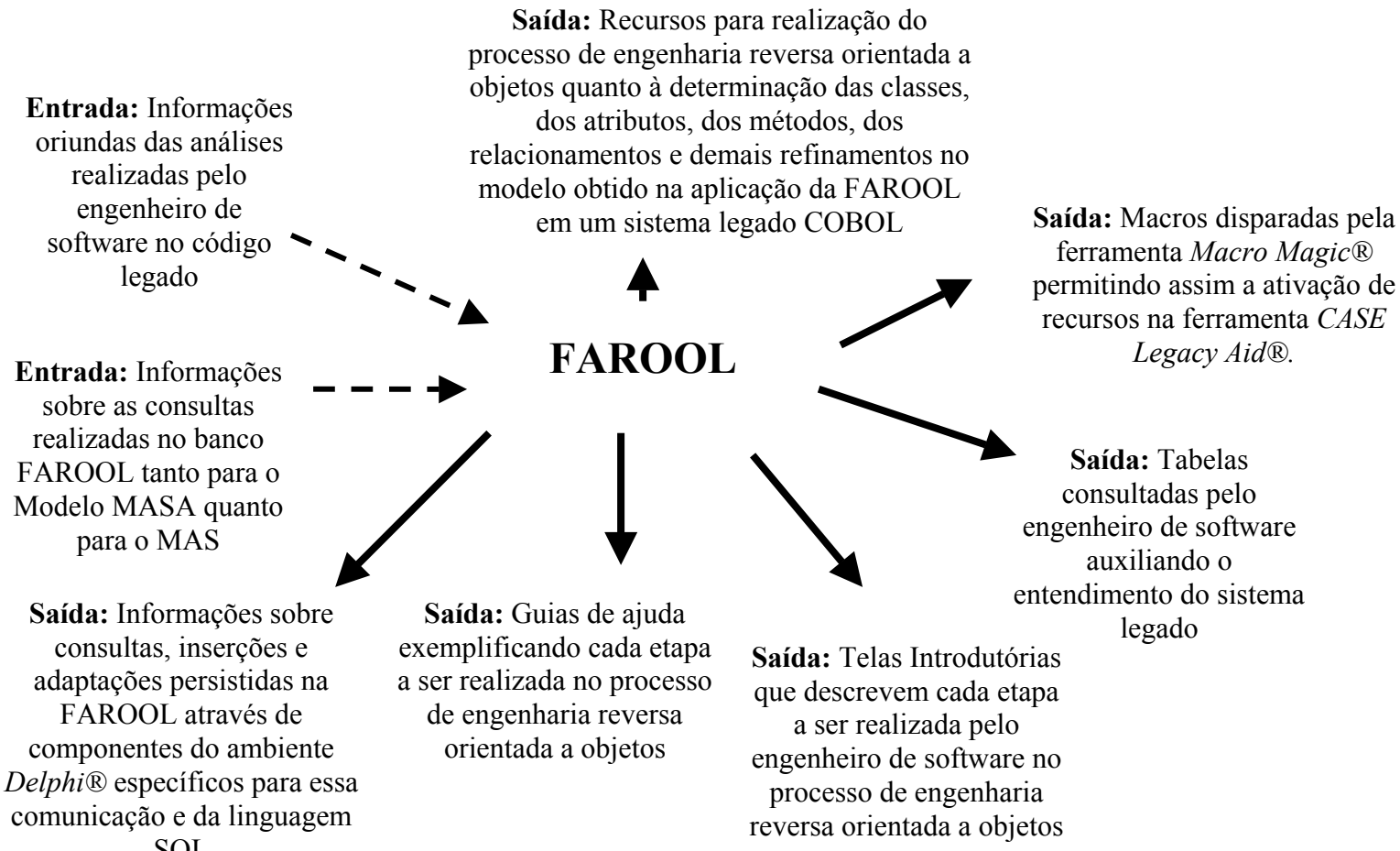


Figura 3 - Entradas e Saídas da FAROOL

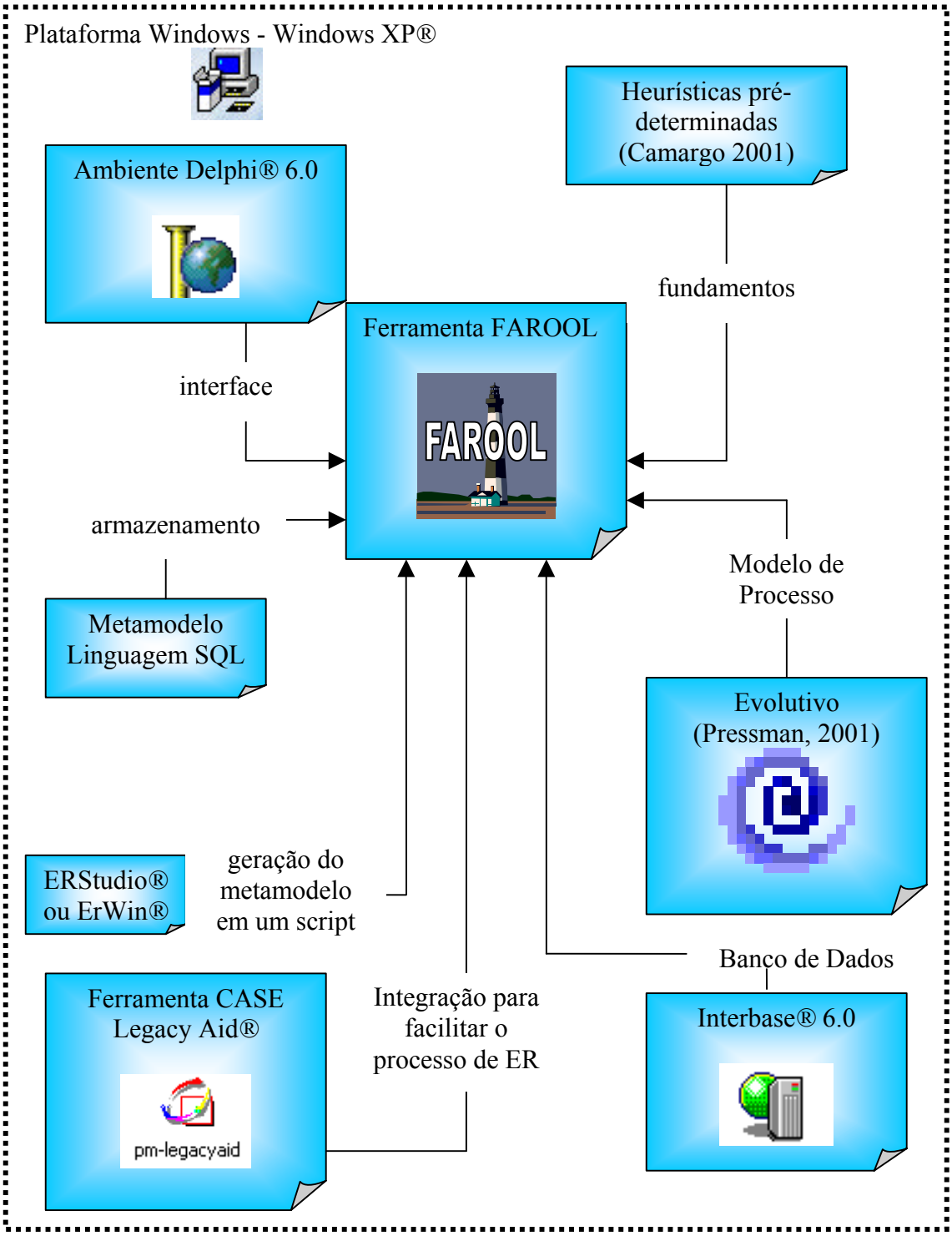


Figura 4 – Componentes da Ferramenta FAROOL

A Figura 4 apresenta os componentes da FAROOL e suas contribuições no processo de Engenharia Reversa.

Para o armazenamento das informações obtidas ao longo do processo de engenharia reversa foi desenvolvido um metamodelo, no qual são considerados todas as tabelas e campos necessários para criação do banco de dados da FAROOL. A partir desse metamodelo foi gerado o *script* na linguagem *SQL* com auxílio da *ERStudio*® (ERStudio, 2003). Esse *script* foi carregado na ferramenta de apoio *IBConsole* do SGBD *Interbase*® *6.0*, gerando o banco de dados da FAROOL.

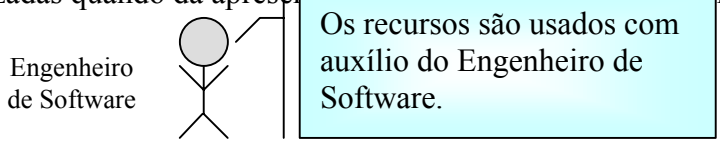
O modelo de processo considerado para esse desenvolvimento é o evolutivo. Dessa forma, a primeira versão da FAROOL é apresentada possibilitando a automatização parcial do processo de engenharia reversa orientada a objetos. Assim, a interação com o engenheiro de software é muito importante para a realização das tarefas envolvidas no processo de recuperação dos sistemas legados implementados em *COBOL*, existindo um guia interativo nela inserido para esse fim.

Algumas tarefas para elaboração de um modelo pseudo orientado a objetos são feitas com a ferramenta *CASE Legacy Aid*® integrada à FAROOL. Isso ocorreu para auxiliar, principalmente, na determinação das classes, dos atributos e dos métodos candidatos para definição do modelo de análise orientada a objetos do sistema legado. A forma como essa integração foi realizada é apresentada nas próximas seções desse capítulo.

Para o desenvolvimento da interface, bem como da integração do banco da FAROOL, foi utilizado o ambiente *Delphi*®, versão 6.0, já apresentado no capítulo 2. O processo de desenvolvimento da interface da FAROOL será detalhado nas próximas seções desse capítulo.

Toda essa estrutura foi definida para a plataforma *Windows*, utilizando o *Windows* XP (Windows, 2002), mas com testes feitos e resultados satisfatórios obtidos também para os sistemas operacionais *Windows* 98, NT e 2000.

A arquitetura interna da ferramenta FAROOL com os principais recursos oferecidos ao engenheiro de software é apresentada na Figura 5. As letras entre parênteses serão utilizadas quando da apresentação de cada um dos componentes/atividades.



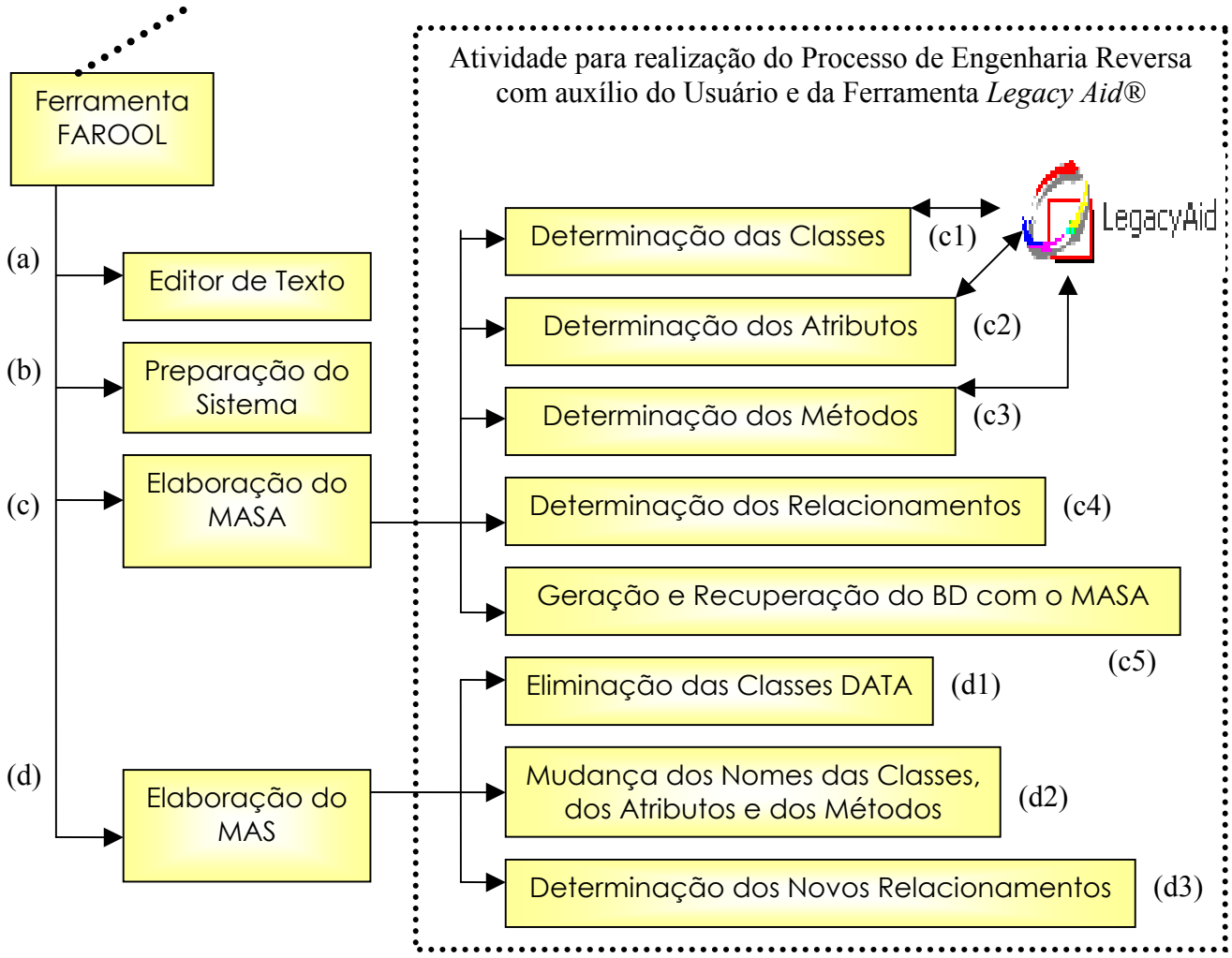


Figura 5 - Arquitetura Interna da Ferramenta FAROOL

A ferramenta FAROOL oferece um Editor de Texto simples (a), para que o engenheiro de software possa abrir um programa COBOL, possa ler e visualizar seu código, bem como permite copiar, substituir, imprimir ou colar trechos desse código e pesquisar por palavras chaves que facilitem o entendimento geral do sistema.

A Preparação do Sistema (b) permite que os arquivos de programas <<.COB>> sejam carregados, os arquivos de estrutura mapeados e que seja criado um novo projeto para referenciar o sistema legado nas fases que se seguem. O processo de engenharia reversa é dividido em duas fases principais: MASA (Modelo de Análise do Sistema Atual) e MAS (Modelo de Análise do Sistema).

A Elaboração do MASA (c) é o começo do processo de engenharia reversa quando algumas características do sistema são determinadas e armazenadas no banco de dados para futuras análises e consultas. A primeira atividade é a Determinação das Classes Candidatas (c1), que são os FDs do sistema legado. Na Determinação dos Atributos (c2), de forma simplificada, pode-se dizer que são tratados os itens de dados de cada FD determinado anteriormente, do sistema legado. A Determinação dos Métodos (c3), consiste da análise dos procedimentos do código legado, que no paradigma da orientação a objetos, podem ser entendidos como sendo os candidatos a métodos das classes definidas anteriormente. O engenheiro de software, com auxílio da FAROOL, deve proceder a Determinação dos Relacionamentos entre as classes determinadas (c4), sendo necessária uma análise minuciosa do código, conforme apresentado no próximo capítulo desta dissertação. Pode ser gerado um backup do banco de dados (BD) com o MASA para, posteriormente, ser recuperado e comparado com os dados finais armazenados pela FAROOL (MAS). Detalhes a respeito do mapeamento entre os modelos MASA e MAS encontram-se no Apêndice I dessa dissertação, portanto não serão discutidos nesse capítulo.

A Elaboração do MAS (d), permite: a criação do modelo orientado a objetos; a mudança dos nomes das classes, dos atributos e dos métodos, substituindo-os por mnemônicos mais significativos para facilitar futuras manutenções; e a definição de alguns relacionamentos especiais, que dependem de análises minuciosas do código do sistema legado por parte do engenheiro de software. Esses refinamentos estão representados por (d1), (d2) e (d3). A estrutura em camadas da FAROOL é mostrada na Figura 6.

Camada 3: Camada de Interface, onde a FAROOL e a *Legacy Aid*® atuam. Essa camada está mais próxima do engenheiro de software e é com ela que o mesmo interage para realização de todo o processo de engenharia reversa. A interface da FAROOL foi desenvolvida utilizando o ambiente *Delphi® 6.0*.

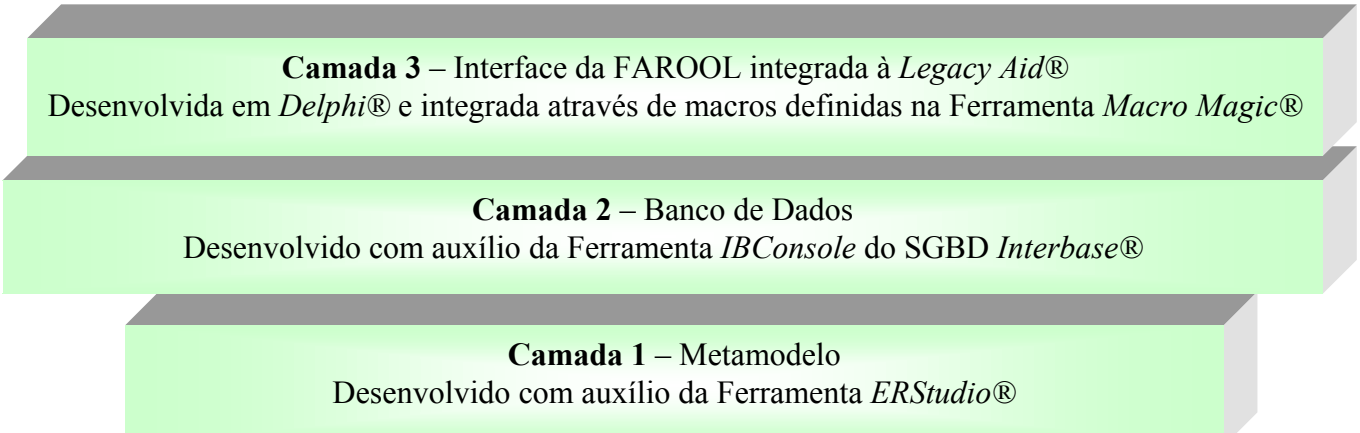


Figura 6 - Estrutura em Camadas da FAROOL

Camada 2: Camada de Banco de Dados da FAROOL que armazena as informações recuperadas ao longo do processo de engenharia reversa. Esse banco foi criado utilizando uma ferramenta auxiliar *IBConsole* do SGBD *Interbase*® 6.0, que é integrada ao *Delphi*® através de componentes específicos oferecidos pela mesma para tratamento de banco do dados.

Camada 1: Camada de Metamodelo, onde o banco foi definido. Contém o esquema com as principais tabelas e campos para armazenamento dos dados recuperados.

O ambiente da ferramenta FAROOL pode ser melhor entendido analisando-se a Figura 7 que apresenta uma visão geral em termos: do papel do engenheiro de software no processo de engenharia reversa; da forma como a ferramenta FAROOL e ferramenta *Legacy Aid*® foram integradas e do armazenamento das informações obtidas no processo de engenharia reversa no banco de dados relacional. Para facilitar o entendimento do leitor foram colocadas letras entre parênteses na Figura 7 representando ações que são realizadas e descritas a seguir.

A FAROOL passa a macro, específica do recurso a ser ativado na *Legacy Aid*®, para a *Macro Magic* (a). Essa por sua vez, ativa o recurso da ferramenta *Legacy Aid*® solicitado (b). O Engenheiro de Software consulta, quando necessário, o recurso disponibilizado na *Legacy Aid*® (c) para pesquisar a informação solicitada pela FAROOL (d), retornando-a (e). A maioria das vezes, FAROOL solicita informações que não dependem da *Legacy Aid*® e o usuário, baseado nos seus próprios conhecimentos e nos retornos da própria FAROOL, informa o que está sendo solicitado. As informações passadas para a FAROOL, bem como as obtidas por elas com os recursos da própria análise do sistema legado, são inseridas no banco de dados (f). Essas informações são, posteriormente, consultadas e adaptadas (g).

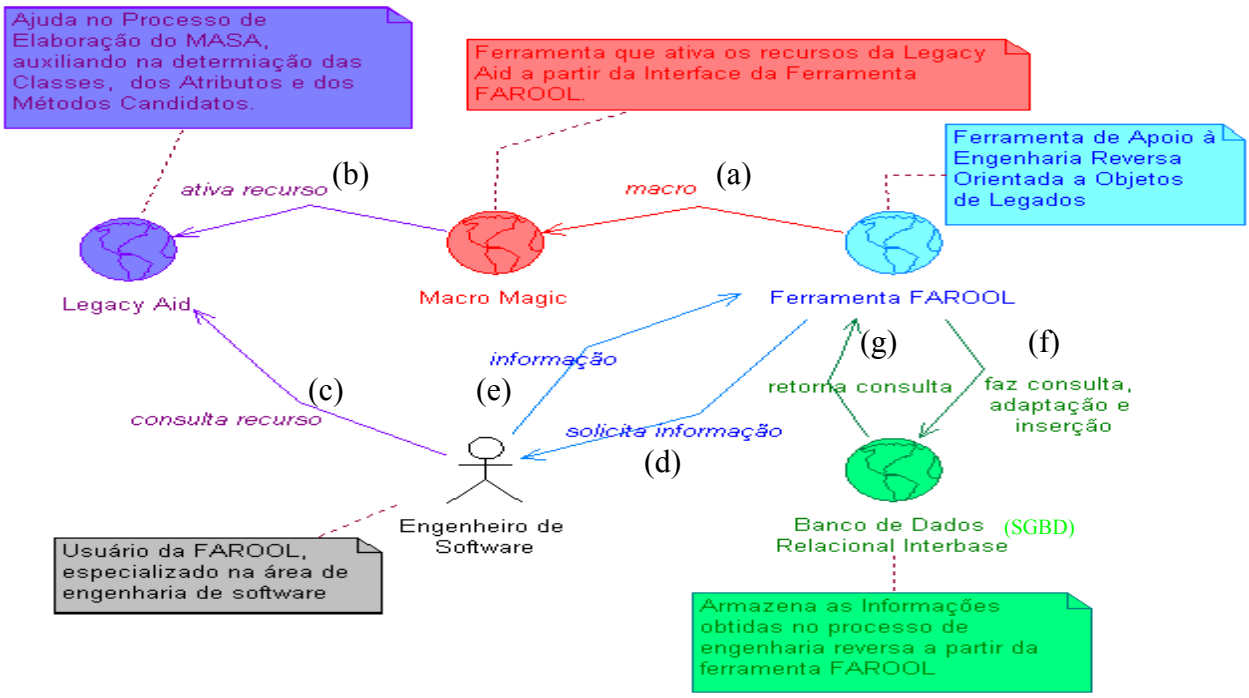


Figura 7 - Visão geral do ambiente da Ferramenta FAROOL para realização do Processo de Engenharia Reversa

3.3 Construção da Ferramenta FAROOL

FAROOL utiliza a abordagem *Fusion/RE* de engenharia reversa, sendo os modelos de classes expressos em *UML* com as heurísticas para sistemas legado implementados em linguagem *COBOL* propostas por Camargo (2001a) e apresentadas no capítulo 2.

Os Quadros 11 e 12 apresentam as duas principais fases para o processo de engenharia reversa orientada a objetos usando em FAROOL. A coluna 1 apresenta as fases; na coluna 2 são apresentadas as heurísticas propostas por Camargo e na coluna 3 tem-se a descrição de cada uma delas de acordo com a sua realização na ferramenta proposta, FAROOL.

Quadro 11 - Principais heurísticas utilizadas na implementação do FAROOL (MASA)

FASE	Heurística	Descrição

(*) REDEFINES: comando utilizado na linguagem COBOL para redefinir um determinado trecho de código e reusar parte do código já definido no sistema. **Elaboração do MASA**	**Identificação do FD** **"Todo FD é candidato à pseudo-classe"**	**Com integração da CASE *Legacy Aid*®, FAROOL permite a identificação das pseudo-classes, além de armazenar essas informações no banco de dados.**
	Identificação dos Itens de Dados **"Todo Item de Dado é candidato a pseudo-atributo"** **(Itens Elementares e de Grupo)**	**Com a integração da ferramenta CASE *Legacy Aid*®, FAROOL permite a identificação dos pseudo-atributos, baseados na classificação em itens elementares e de grupo, além de armazenar essas informações no banco de dados do apoio.**
	Identificação dos procedimentos **"Todo procedimento é candidato a pseudo-método"** **(Anomalias)**	**Com integração da ferramenta CASE *Legacy Aid*®, FAROOL permite a identificação dos pseudo-métodos e de suas anomalias, além de armazenar essas informações no banco de dados do apoio.**
	Determinação dos Relacionamentos	**FAROOL auxilia o usuário na determinação dos principais relacionamentos, com ênfase nos de associação, bem como cardinalidades , navegabilidade e definição dos papéis (Roles), além de armazenar essas informações no banco de dados do apoio.**
	Determinação dos Relacionamentos de Associação (*) Casos de Redefinição (REDEFINES)	**Baseado em redefinições, FAROOL faz avaliações de possíveis relacionamentos e auxilia o engenheiro de software na determinação dos mesmos.**

Quadro 12 - Principais heurísticas utilizadas na implementação do FAROOL (MAS)

PASSO

FASE	Nome	Descrição
(*) REDEFINES: comando utilizado na linguagem COBOL para redefinir um determinado trecho de código e reusar parte do código já definido no sistema **Elaboração do MAS**	**Remoção das Pseudo-Classes DATA**	**FAROOL permite a remoção de pseudo-classes DATA, bem como a definição de atributos para substituição dessas pseudo-classes.**
	Mudança dos Nomes das Classes	**FAROOL permite a mudança dos nomes das classes por outros mais significativos, facilitando a manutenção futura e atendendo as expectativas dos engenheiros de software.**
	Mudança dos Nomes dos Atributos	**FAROOL permite a mudança dos nomes dos atributos por outros mais significativos, facilitando a manutenção futura e atendendo as expectativas dos engenheiros de software.**
	Refinar Métodos	**FAROOL permite a mudança dos nomes dos métodos observadores e construtores por outros mais significativos**
	Determinação de Novos Relacionamentos (*) Casos de Redefinição (REDEFINES)	**FAROOL permite a determinação de novos relacionamentos, de generalização principalmente, baseado nos casos de redefinição e na interação com o engenheiro de software.**

A ferramenta *Legacy Aid*® foi integrada à ferramenta deste projeto para a realização da fase inicial de engenharia reversa, para que se obtenha possíveis classes, atributos e métodos candidatos. Dessa forma, possibilitou-se a melhor utilização desses recursos pelos usuários do FAROOL, ou seja, através de macros desenvolvidas utilizando a ferramenta *Macro Magic*®, descritas na seção 3.3.1. Os sistemas legados são submetidos à *Legacy*

Aid® e com a interação do engenheiro de software, as informações solicitadas pelas interfaces da FAROOL são identificadas e armazenadas no banco de dados da mesma.

A seguir são apresentadas as macros desenvolvidas e utilizadas para essa integração.

3.3.1 Ferramenta Macro Magic® e Definição das Macros

A ferramenta *Macro Magic*® permite o desenvolvimento de macros, que são formas de acessar recursos e janelas de outras ferramentas. Esse acesso pode ser feito através do clique de um botão na interface, modo implementado na ferramenta FAROOL. A macro é disparada e executa um trecho de código que permite disponibilizar recursos internos de outras ferramentas, como os da ferramenta *Legacy Aid*®, Figura 8.

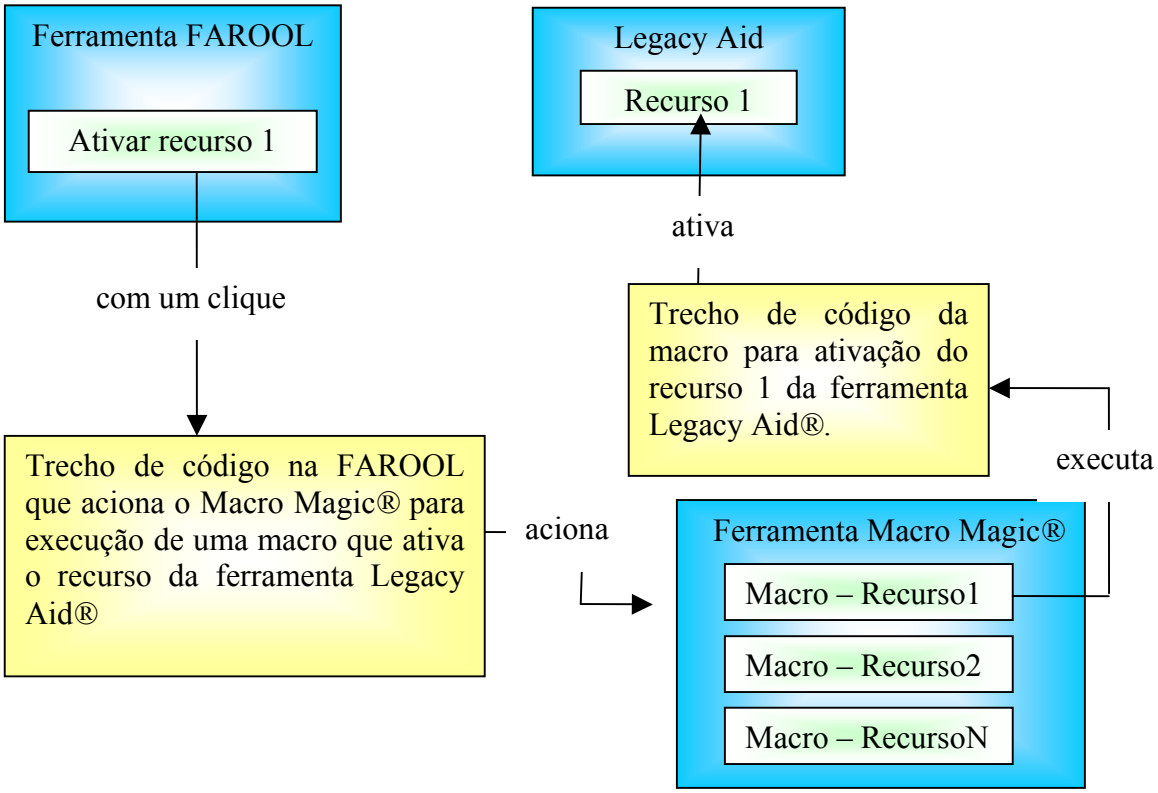


Figura 8 - Definição das Macros utilizando o Macro Magic

Diversas macros foram desenvolvidas neste trabalho. Elas ativam recursos da ferramenta *Legacy Aid*® para realizar atividades específicas descritas a seguir:

- para criação de um novo projeto, necessária para carregar os programas do sistema legado a ser recuperado.

- para aplicação do *parser* nos programas COBOL, criada para reconhecer o sistema legado possibilitando o uso da *Legacy Aid*® a partir da FAROOL.

- para mapeamento dos arquivos de estrutura do sistema, para que o sistema legado, submetido à FAROOL seja recuperado de forma completa, considerando todo o código fonte e os arquivos de estrutura chamados de *Copy Files* na programação *COBOL*.

- para determinação dos *FDs* candidatos à classe, no paradigma da orientação a objetos.

- para determinação das chaves dos programas, necessárias para futuras análises no código e determinação de associações entre classes.

- para classificação dos *FDs* de leitura, quando somente consultam dados do sistema sem alterá-los; ou de escrita e reescrita, quando os alteram. Essa informação é necessária porque são consideradas classes *válidas* ao processo de engenharia reversa, somente as classes oriundas de *FDs* classificados como de escrita ou reescrita (WRITE ou REWRITE)

- para determinação dos atributos candidatos que são os itens de dados definidos em cada FD.

- para determinação dos métodos candidatos que são os procedimentos do sistema legado.

- para mapeamento entre os Modelos de Análise MASA e MAS, ativando o *IBConsole* para geração de um backup do MASA e, posteriormente, para recuperação do mesmo no momento de comparação (mapeamento) entre esse modelo e o modelo final gerado (MAS).Ainda para auxiliar o mapeamento é ativado o Note Pad para armazenar o nome do programa, o antigo nome da pseudo-classe no MASA e o seu novo nome no MAS.

O metamodelo utilizado para construção da FAROOL é apresentado na seção seguinte.

3.3.2 - Metamodelo

Um metamodelo é um modelo com classes, atributos e métodos, que após ser implementado, serve como um repositório de dados, no qual dados são inseridos, criados, manipulados e consultados por usuários de uma determinada aplicação e/ou ferramenta.

O metamodelo usado para a implementação da FAROOL pode ser observado na Figura 9. Cada classe do metamodelo é uma tabela do banco de dados, sendo o *Interbase*® utilizado neste projeto. Cada atributo de uma determinada classe é um campo da tabela, e cada conjunto composto por todos os atributos de uma classe é um registro da tabela. O banco é composto de dezoito classes, cada uma delas com uma lista de atributos. Isto significa que o banco de dados relacional, criado a partir desse metamodelo, é um banco composto por dezoito tabelas, cujos campos são os atributos especificados para cada classe no metamodelo. As principais classes do metamodelo definido para FAROOL, bem como uma breve descrição da mesma, são apresentadas nas tabelas 1, 2 e 3.

A Tabela 1 apresenta as classes do metamodelo , com suas respectivas descrições, utilizadas no armazenamento dos programas (Program), classes (Class) e (IntermediateTable), atributos (Attribute), métodos (Operation), (ParamOperation), (IsObservation) e (IsConstruction).

A Tabela 2 apresenta as classes do metamodelo, com suas respectivas descrições, utilizadas para armazenamento dos relacionamentos, denominadas: Relationship, Dependence, Association, Realize e Generalization.

A Tabela 3 apresenta as classes: EntityRole, DependenceRole, AssociationRole, RealizeRole e GeneralizationRole, que são utilizadas para definição dos papéis (roles) dos relacionamentos armazenados na Tabela 2.

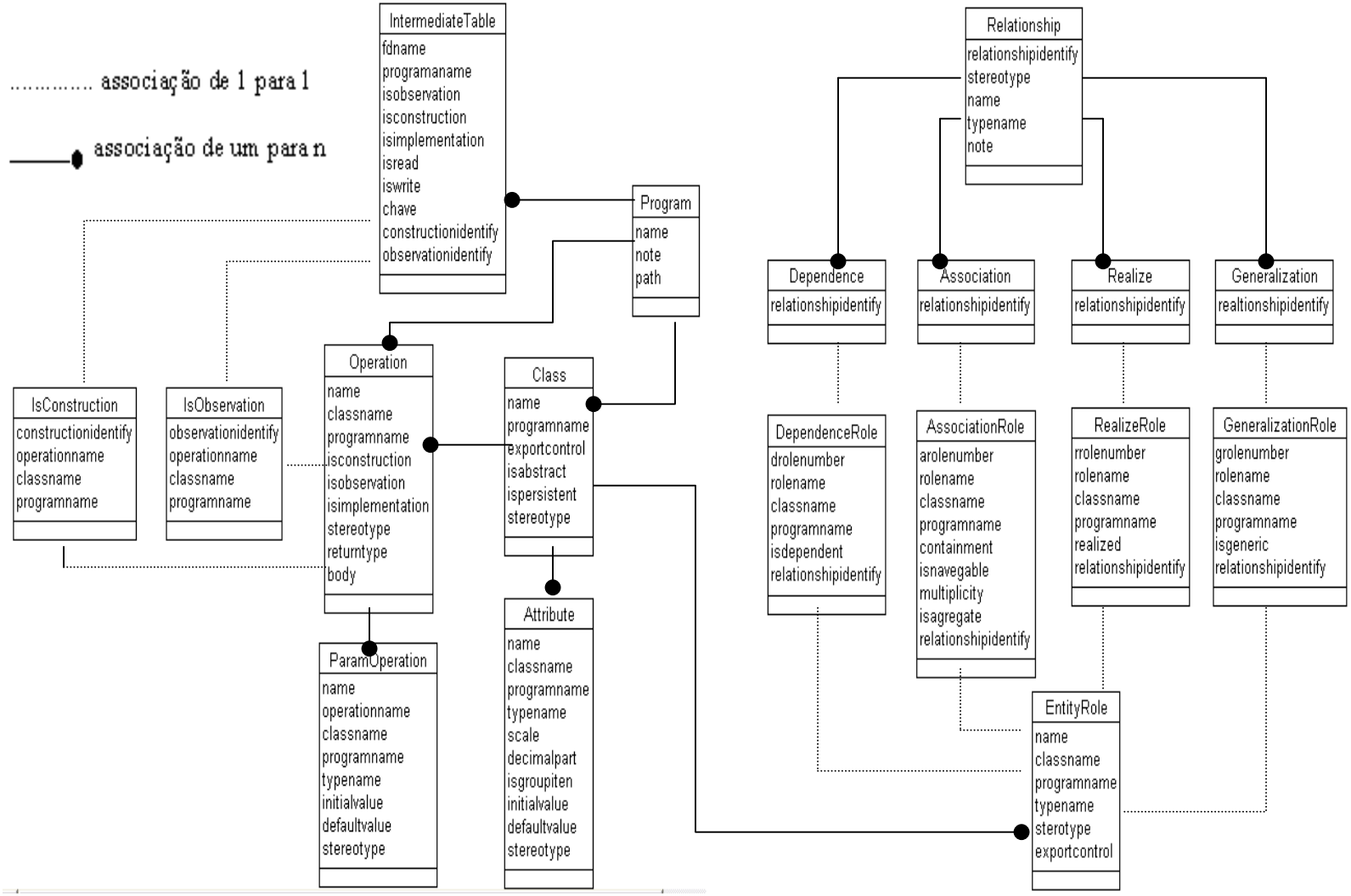


Figura 9 - Metamodelo do Projeto FAROOL

Tabela 1 - Classes do Metamodelo para armazenamento dos Programas, Classes, Atributos e Métodos

Nome da Classe	Descrição da Classe
Program	Responsável pelo armazenamento dos programas, guardando o nome, o caminho e uma nota com a sua descrição. É a primeira classe a ser chamada pela FAROOL para armazenamento das informações da fase de Elaboração do MASA.
IntermediateTable	Responsável pelo armazenamento das informações obtidas no processo de identificação dos FDs. Armazena o nome, a chave e o programa do FD, bem como se é de escrita e/ou leitura, construtor, observador ou de implementação. É a segunda classe chamada pela FAROOL para armazenamento de informações também obtidas na fase de Elaboração do MASA.
Class	Responsável pelo armazenamento das classes candidatas válidas no processo de engenharia reversa, guardando seus nomes, programas correspondentes e demais informações caso seja necessário.
Attribute	Responsável pelo armazenamento dos atributos candidatos, guardando seus nomes, classes e programas correspondentes, tipos, tamanhos e demais informações caso seja necessário.
Operation	Responsável pelo armazenamento dos métodos candidatos, guardando seus nomes, classes e programas correspondentes, sua classificação como sendo construtor, observador ou de implementação, seu corpo e demais informações caso seja necessário
ParamOperation	Responsável por armazenar, quando necessário os parâmetros dos métodos armazenados em Operation. Como não é proposta nesse trabalho a mudança completa da linguagem procedimental para a orientada a objetos, essa classe ainda não é utilizada, sendo considerada apenas para trabalhos futuros.
IsConstruction	Responsável pelo armazenamento dos métodos candidatos classificados no processo de engenharia reversa como construtores. São armazenados, para facilitar consultas no banco por parte da FAROOL, seus nomes, programas e classes correspondentes.
IsObservation	Responsável pelo armazenamento dos métodos candidatos classificados no processo de engenharia reversa como observadores. São armazenados, para facilitar consultas no banco por parte da FAROOL, seus nomes, programas e classes correspondentes.

Tabela 2 - Classes do Metamodelo para armazenamento dos Relacionamentos

Nome da Classe	Descrição da Classe

Relationship	armazena os identificadores dos relacionamentos entre classes encontrados no sistema.
Dependence	armazena os identificadores dos relacionamentos de dependência entre classes.
Association	armazena os identificadores dos relacionamentos de associação entre classes.
Realize	armazena os identificadores dos relacionamentos de interface entre classes.
Generalization	armazena os identificadores dos relacionamentos de generalização entre classes.

Tabela 3 -Classes do Metamodelo para armazenamento dos Papéis (roles)

Nome da Classe	Descrição da Classe
EntityRole	armazena os papéis (roles) definidos para cada relacionamento armazenado em Relationship, guardando seus nomes, classes e programas correspondentes e demais informações pertinentes.
DependenceRole	armazena as informações específicas dos papéis dos relacionamentos de dependência.
AssociationRole	armazena as informações específicas dos papéis dos relacionamentos de associação.
RealizeRole	armazena as informações específicas dos papéis dos relacionamentos de interface.
GeneralizationRole	armazena as informações específicas dos papéis dos relacionamentos de generalização.

Uma vez definido o metamodelo, com classes e seus relacionamentos, deve-se gerar um *script* em linguagem *SQL* para que o banco seja criado. A *ERStudio*® (ERStudio, 2003) e a *ErWin* (ErWin, 2002) são exemplos de ferramentas que geram automaticamente esses *script* em *SQL*.

Na Figura 10, é apresentado parte do *script* gerado pela *ERStudio*®, correspondente ao metamodelo da Figura 9, que deu origem ao banco de dados relacional da ferramenta FAROOL no SGBD *Interbase*®. Esse *script* é então carregado pela ferramenta auxiliar do SGBD *Interbase*®, o *IBConsole*. Assim, as tabelas e os campos são criados, permitindo que os usuários manipulem, insiram, modifiquem, apaguem ou mesmo consultem esses dados, Figura 11.

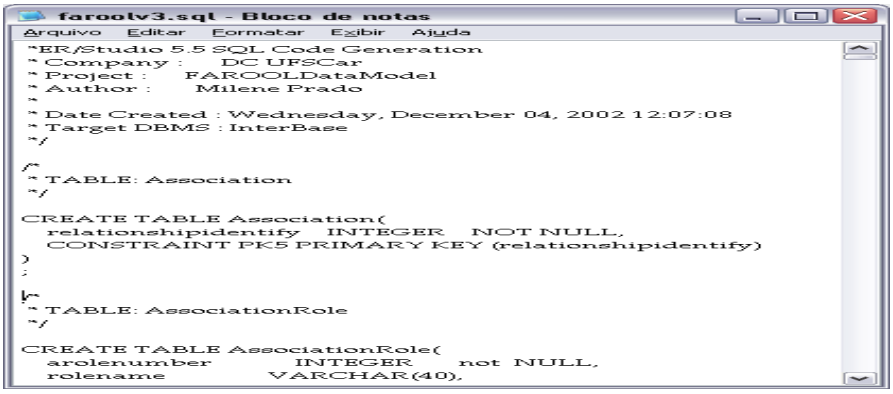


Figura 10 - Script em SQL do metamodelo do Projeto FAROOL

Especificado o banco de dados na ferramenta *IBConsole* e selecionando o recurso Interactive SQL na opção Tools..., o usuário pode carregar o *script* para geração do banco.

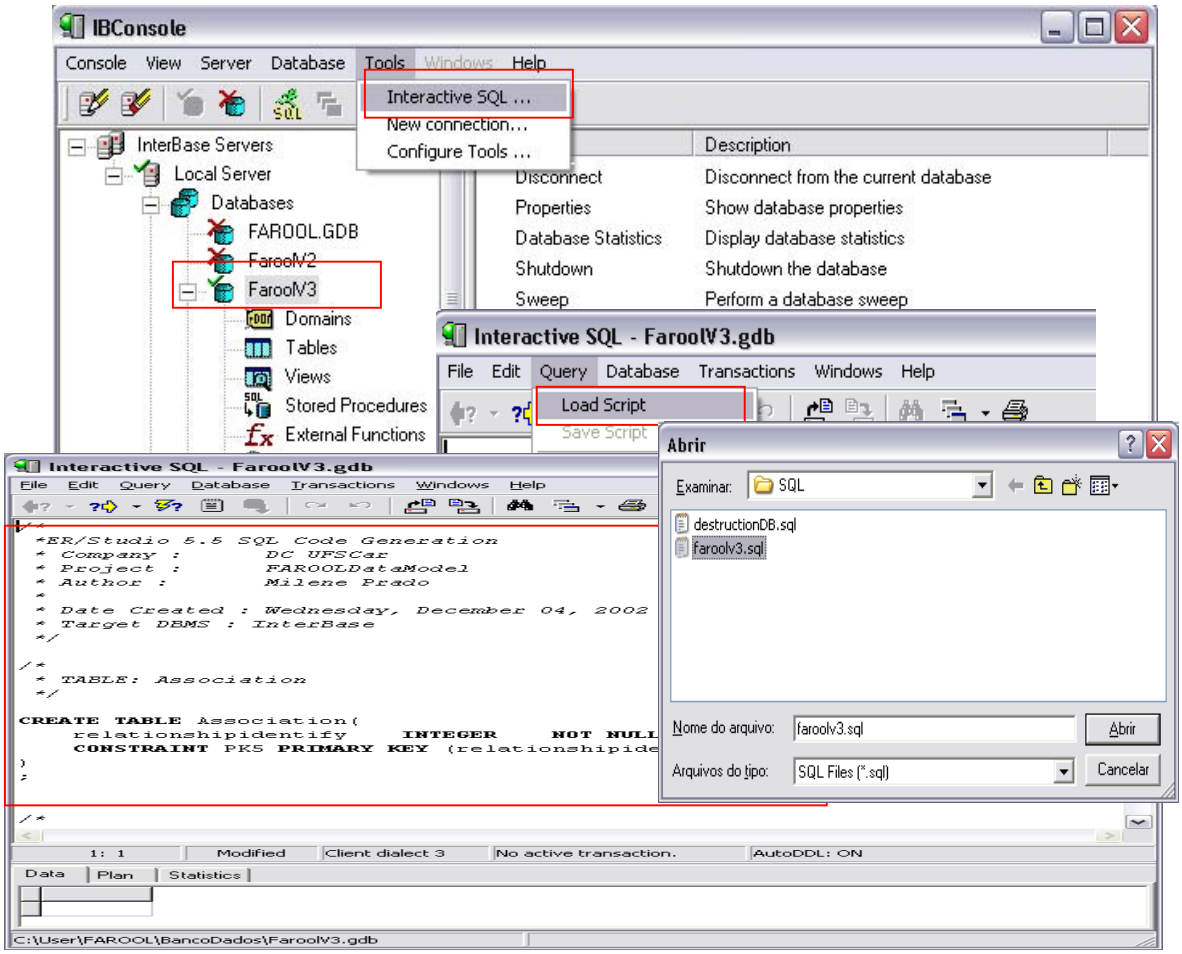


Figura 11 - IBConsole – Recurso Interactive SQL

Como FAROOL foi desenvolvida no ambiente *Delphi*® e utiliza o SGBD *Interbase*®, deve existir uma integração entre o SGBD *Interbase*® e a ferramenta *Delphi*®, como será tratado a seguir.

3.3.3 Utilização do Ambiente Delphi® 6.0

A ferramenta FAROOL foi desenvolvida no ambiente *Delphi*® versão 6.0 da *Borland*. As linguagens utilizadas no *Delphi*® para implementar a interface, bem como para comunicá-la com o banco de dados definido no *Interbase*® foram a Linguagem *Pascal* orientada a objetos e a Linguagem *SQL*. A linguagem *SQL* foi utilizada para definir *querys* que são disparadas quando o engenheiro de software utiliza um recurso que consulta, insere ou manipula dados do banco de dados da FAROOL. As *querys* são componentes do *Delphi*® preparados para definição de uma consulta, seleção, adaptação e/ou inserção de dados no banco. Foi implementado um editor de textos para a ferramenta FAROOL que permite a abertura, impressão, visualização, pesquisa por palavras, edição dos códigos dos programas, e alteração das fontes dos programas *COBOL*. Vários recursos foram desenvolvidos e implementados, como os apresentados na Figura 5 deste capítulo. Esses recursos dependem da interação com o engenheiro de software para serem realizados e para que o processo de engenharia reversa se concretize. A interface possui muitas janelas de ajuda, passo a passo para o engenheiro de software.

A interação entre o banco de dados e a interface da FAROOL, é feita pelo componente **DATABASE**. Para definir o banco de dados, como o seu nome, o caminho onde o mesmo se encontra na máquina, entre outras particularidades do banco, o componente disponibiliza um quadro para seleção do banco de dados, definido previamente no *Interbase*®.

O nome do banco de dados é colocado no campo Name da Figura 12, nesse caso FaroolInt O Campo Alias name contém o nome do banco de dados cadastrado anteriormente no *Delphi*®, e pode ou não ser o mesmo do campo Name. Na Figura 12 é colocado FAROOLV5. As informações padrões do campo Parameter overrides, Figura 12, devem ser selecionadas e completadas de acordo com as necessidades específicas do banco cadastrado. No caso do banco FAROOL, **SYSDBA** e **masterkey** são respectivamente, o identificador e a senha padrão definidos pelo *Interbase*®.

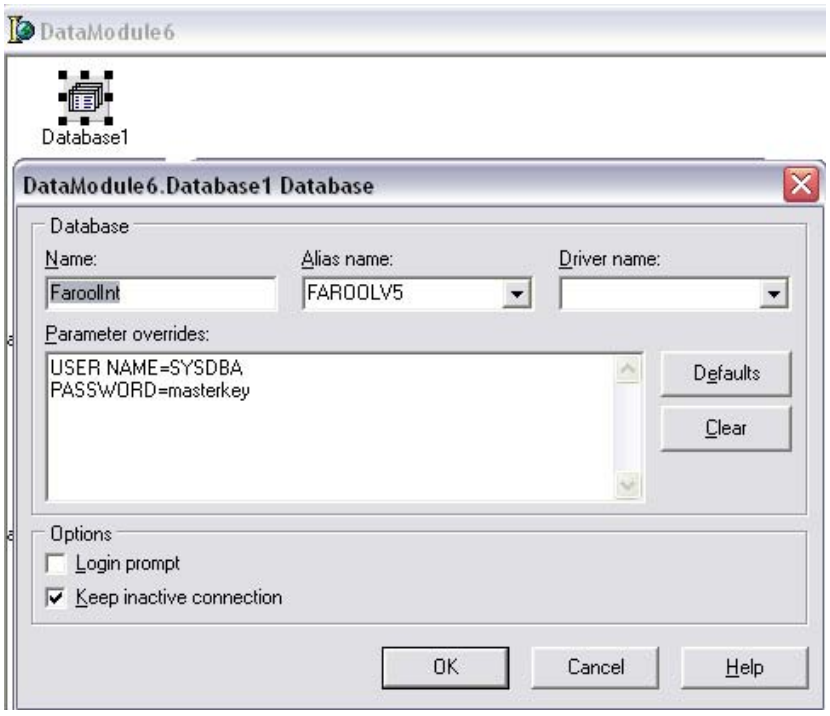


Figura 12 - Definição do banco de dados da ferramenta FAROOL no Componente DATABASE da ferramenta Delphi®

Para acesso ao banco de dados e realização de uma inserção, uma consulta ou uma adaptação, a ferramenta *Delphi*® permite a definição de *Query*(s) e *Table*(s), através dos comandos INSERT, SELECT e UPDATE da linguagem *SQL*, respectivamente.

Ao usuário da FAROOL, esses conceitos são transparentes, pois já estão definidos quando de seu uso.

3.4 Considerações Finais

Esse capítulo teceu algumas considerações a respeito da estrutura geral da FAROOL, dentre elas:

- A arquitetura externa da ferramenta, sendo definidos os principais componentes utilizados para o desenvolvimento da Ferramenta FAROOL;

- A arquitetura interna da ferramenta, sendo considerados os recursos oferecidos pela mesma ao engenheiro de software para a realização do processo de engenharia reversa;

- A estrutura em camadas da ferramenta, na qual foram identificadas três camadas principais: a do metamodelo que é a mais distante do engenheiro de software, usuário da FAROOL; a do banco de dados da FAROOL com as tabelas e os campos baseados no metamodelo definido na camada anterior e a camada de interface que é a mais próxima do engenheiro de software, que permite através de interações o uso de recursos para a realização do processo de engenharia reversa;

- A apresentação e descrição dos componentes utilizados para a construção e desenvolvimento da FAROOL, entre eles: o conjunto de heurísticas adaptado; a ferramenta *Macro Magic* que através de macros definidas previamente, integra a ferramenta *Legacy Aid*® à FAROOL; a ferramenta *Legacy Aid*®; o metamodelo definido para a geração do banco de dados da ferramenta FAROOL; a ferramenta de apoio *IBConsole* e o SGBD *Interbase*®, responsáveis pela criação do banco de dados da ferramenta FAROOL; a Linguagem *SQL* para geração do script com as tabelas definidas no metamodelo; e a ferramenta *Delphi*® para desenvolvimento da interface da ferramenta FAROOL e integração do banco através de componentes específicos para esse tratamento.

Capítulo 4

FAROOL: Apresentação Geral e Fase de Preparação do Sistema

4.1 Considerações Iniciais

Neste capítulo serão apresentados os principais recursos da FAROOL para auxiliar no processo de engenharia reversa orientada a objetos de sistemas legados implementados em *COBOL*. Com o uso dessa ferramenta pretende-se também auxiliar o engenheiro de software na manutenção do sistema legado para que sejam identificadas as partes do sistema que abrigam determinadas funcionalidades.

Este capítulo está organizado da seguinte forma: na seção 4.2, é feita a apresentação geral da ferramenta, com os menus de entrada, recursos, oferecidos pela mesma para a realização do processo de engenharia reversa. Na seção 4.3 são apresentadas as principais atividades envolvidas no processo de engenharia reversa e os respectivos recursos da FAROOL utilizados para realização das mesmas. Na seção 4.4 é apresentada a Fase de Preparação do Sistema, a primeira do processo de engenharia reversa orientada a objetos e as considerações finais são comentadas na seção 4.5.

4.2 Apresentação Geral da Ferramenta FAROOL

A ferramenta FAROOL deve ser iniciada a partir de clique-duplo em um arquivo executável que pode ser representado por um ícone na tela, ou mesmo pelo DOS, a partir do caminho que o executa. Essas formas de acesso à FAROOL são apresentadas nas Figuras 13 e 14, respectivamente. Na inicialização pelo DOS, deve ser determinado o diretório no qual o arquivo executável <<farool.exe>> está localizado. No caso do exemplo apresentado, essa localização é determinada pelo caminho c:\User\FAROOL\farool versao1.0\faroolv5.exe.

Figura 13 - Acesso à ferramenta FAROOL a partir de um ícone

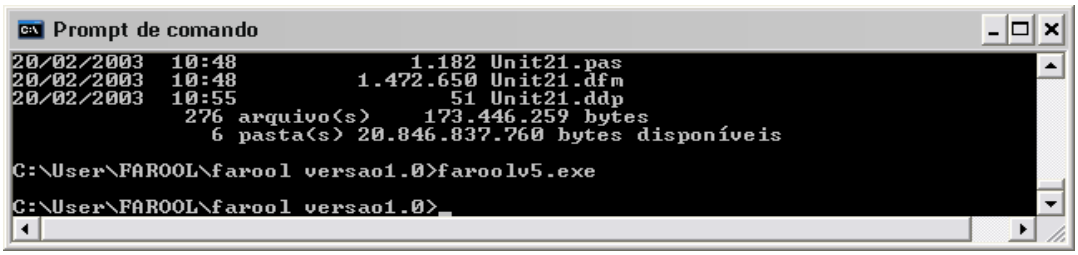
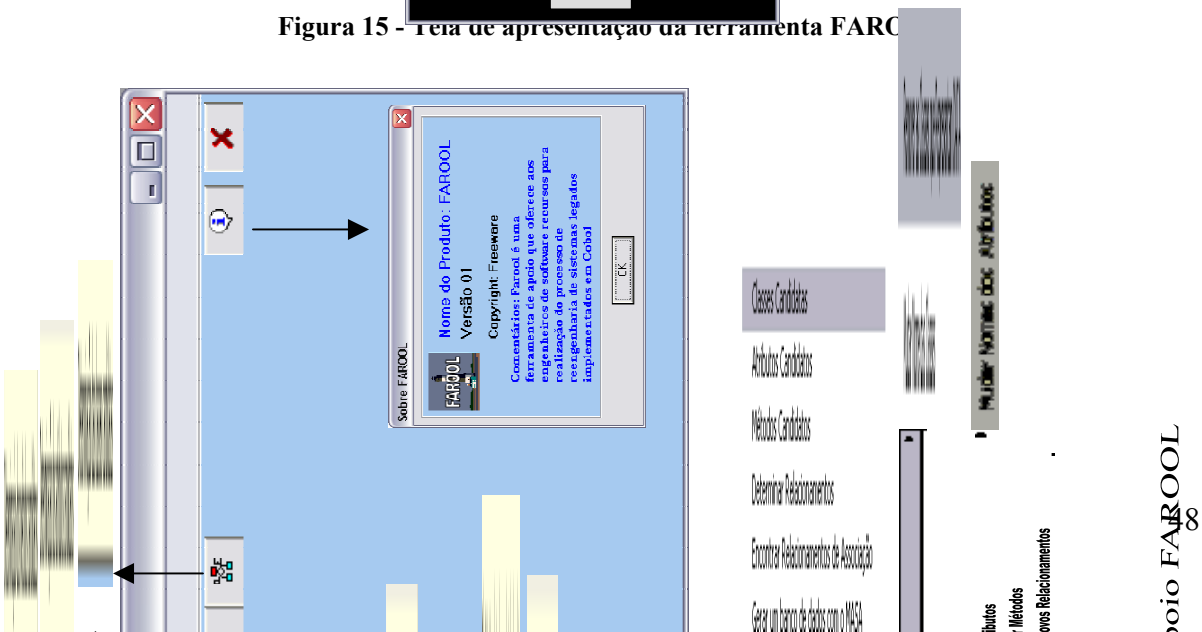


Figura 14 - Acesso à ferramenta FAROOL a partir do DOS

A tela Bem Vindo à Ferramenta FAROOL, exibida após a inicialização da ferramenta, é apresentada na Figura 15. Quando o botão OK dessa tela for escolhido, o menu principal da ferramenta FAROOL é exibido, Figura 16.

Figura 15 - Tela de apresentação da ferramenta FARO



Todos os recursos existentes na ferramenta FAROOL estão apresentados na Figura 16. Os ícones apresentam *hints*, mensagens que auxiliam a identificação dos botões.

FAROOL oferece um Editor de Textos Simples para visualização do código do Sistema em (a), permitindo: abrir programas; salvar, limpar e imprimir o conteúdo do editor; copiar e colar trechos do programa; substituir e localizar palavras no código e retornar a uma ação feita no editor. O Editor é importante para registrar análises no código feitas pelo engenheiro de software, no processo de entendimento do sistema legado. O acesso a esses recursos são a partir das opções Arquivo e Editar do menu principal FAROOL.

O processo de Engenharia Reversa Orientada a Objetos (EROO) realizado pela FAROOL está disponível na opção EROO, bem como as atividades para a sua realização. Dentre as principais atividades, destacam-se: Preparação do Sistema (Fase 1), Elaboração do MASA (Fase 2), Elaboração do MAS (Fase 3). A Fase 1 será detalhada neste capítulo, na seção 4.4; enquanto que as Fases 2 e 3 serão detalhadas nos capítulos 5 e 6, respectivamente.

A título de apresentação da FAROOL é oferecida a opção Sobre FAROOL. Em uma tela é exibido um breve comentário a respeito de sua funcionalidade, a versão utilizada e demais considerações pertinentes ao engenheiro de software.

4.3 Atividades envolvidas no Processo de Engenharia Reversa e os Recursos oferecidos pela Ferramenta FAROOL para realização das mesmas

Para realização do processo de engenharia reversa utilizando a FAROOL algumas atividades devem ser realizadas pelo engenheiro de software, conforme apresentado nos Quadro 13 e 14. Essas atividades, bem como a aplicação dos recursos para a sua realização, serão apresentadas nos capítulos referentes a cada fase do processo de engenharia reversa orientada a objetos.

Quadro 13 - Atividades para realização das Fases de Preparação do Sistema e de Elaboração do MASA

Atividade	Fase	Recurso	Descrição da Funcionalidade
1	**Preparação do Sistema (Fase 1)**	EROO ► Preparação do Sistema	Esse recurso permite que o sistema seja preparado para uso dos demais recursos da ferramenta FAROOL e da ferramenta integrada *Legacy Aid*®.
2	**Elaboração do MASA (Fase 2)**	EROO ► Elaboração do MASA ► Classes Candidatas	Esse recurso permite a determinação dos programas e suas respectivas classes candidatas, iniciando o processo de engenharia reversa do sistema legado.
3		EROO ► Elaboração do MASA ► Atributos Candidatos	Esse recurso permite a determinação dos itens de dados do sistema legado que são candidatos a atributos das classes determinadas no passo anterior.
4		EROO ► Elaboração do MASA ► Métodos Candidatos	Esse recurso permite a determinação dos procedimentos do sistema legado que são candidatos a métodos dos programas e das classes determinadas no passo 2.
5		EROO ► Elaboração do MASA ► Determinar Relacionamentos	Esse recurso permite a determinação dos relacionamentos encontrados pelo engenheiro de software a partir de análises minuciosas no código do sistema legado ao longo do processo de engenharia reversa
6		EROO ► Elaboração do MASA ► Encontrar Relacionamentos de Associação	Esse recurso permite a determinação dos relacionamentos de associação, quando encontrados casos especiais de redefinição no código do sistema legado.

7		EROO ► Elaboração do MASA ► Gerar um Banco de Dados com o MASA	Esse recurso permite a geração de um backup do banco de dados com o modelo pseudo-orientado (MASA) definido até o momento. (**Apêndice I**)
8		EROO ► Elaboração do MASA ► Recuperar o Banco de Dados com o MASA	Esse recurso permite a recuperação do backup do banco de dados criado em 7 para comparação com o banco definido ao final do MAS. (**Apêndice I**)

Quadro 14 - Atividades para realização da Fase Elaboração do MAS

Atividade	Fase	Recurso	Descrição da Funcionalidade
1	**Elaboração do MAS (Fase 3)**	EROO ► Elaboração do MAS ► Classes ► Remover Classes que Representam "DATA"	Esse recurso permite que sejam removidas as classes candidatas que representem "data" no sistema legado.
2		EROO ►Elaboração do MAS ►Classes ►Mudar Nomes das Classes	Esse recurso permite a mudança dos nomes das classes para mnemônicos mais significativos.
3		EROO ► Elaboração do MAS ► Atributos ► Mudar Nomes dos Atributos	Esse recurso permite a mudança dos nomes dos atributos para mnemônicos mais significativos.
4		EROO ► Elaboração do MAS ► Atributos ► Refinar Métodos	Esse recurso permite a mudança dos nomes dos métodos construtores e observadores para mnemônicos mais significativos.

5		EROO ► Elaboração do MAS ► Determinar Novos Relacionamentos	Esse recurso permite a determinação de relacionamentos especiais, como generalização e associações com cardinalidade diferenciada de 1 para 1 e/ou de 1 para 0..1, a partir da análise dos casos de redefinição encontrados no sistema.

4.4 Fase de Preparação do Sistema

Nesta seção é tratada a fase de Preparação do Sistema, necessária para criação um novo projeto, no qual os programas de extensão <<.COB>> do sistema legado são carregados e uma descrição geral do sistema é realizada.

Para iniciar o processo de recuperação de um sistema legado procedimental utilizando a FAROOL, o engenheiro de software deve Preparar o Sistema a ser analisado, utilizando o recurso EROO -> Preparação do Sistema, oferecido no menu principal da ferramenta, Figura 17. Esse recurso instrui o engenheiro de software na preparação do sistema legado para sua posterior análise, realização do processo de engenharia reversa e aplicação de vários recursos oferecidos pela ferramenta *Legacy Aid*® integrada à FAROOL. A tela introdutória para Preparação do Sistema é apresentada na Figura 18.

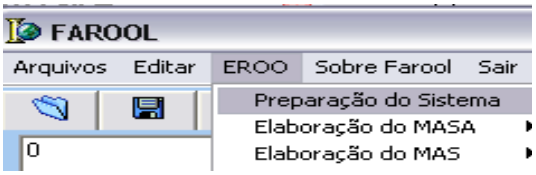
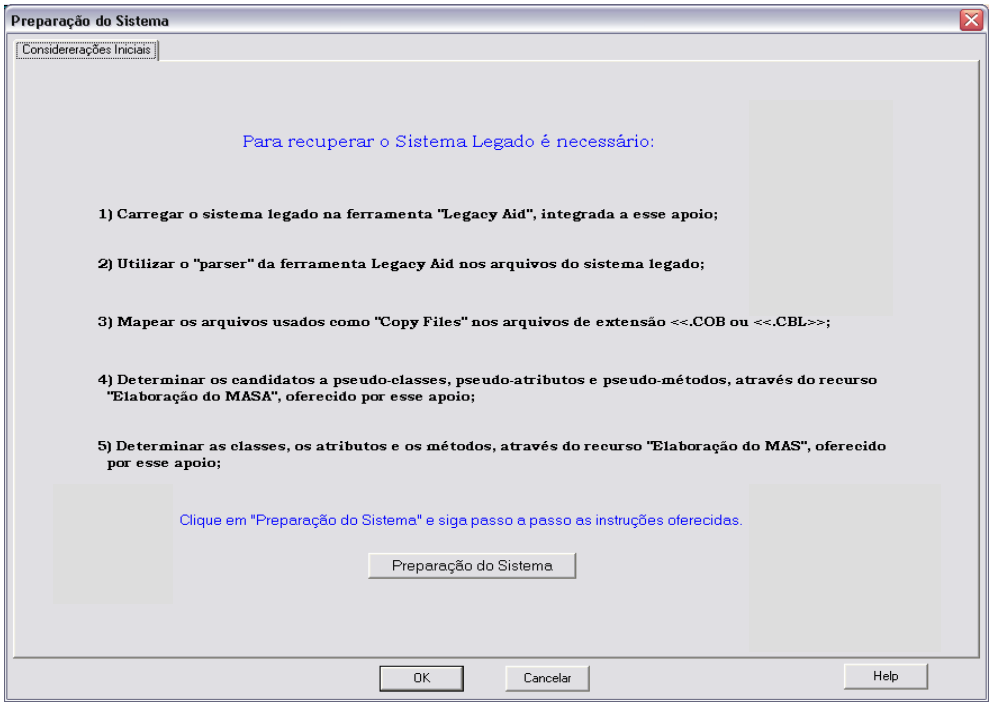


Figura 17 - Opção Preparação do Sistema oferecida pela FAROOL



O processo de engenharia reversa a ser realizado com auxílio da FAROOL é descrito passo a passo quando se clica em Preparação do Sistema, Figura 19. Para cada atividade a ser realizada, de acordo com os Quadros 13 e 14 apresentados anteriormente, existem guias auxiliares, rotulados de Ajuda.

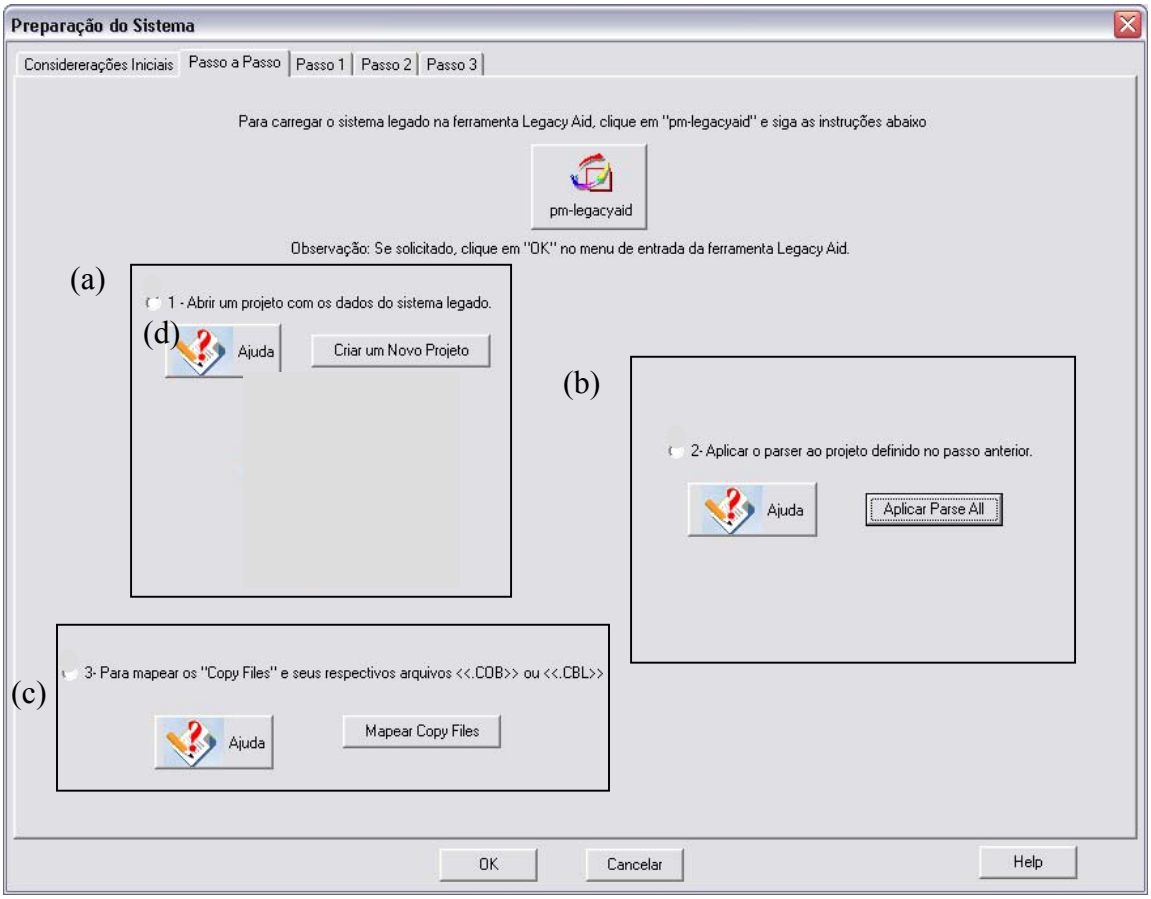


Figura 19 - Tela Passo a Passo para Preparação do Sistema

A atividade Preparação do Sistema está dividida em três passos principais, que devem ser realizados na seguinte ordem: Criação de um Novo Projeto (a), Aplicação do Parser (b) e Mapeamento dos Arquivos Copy Files (c), que podem ser vistos na Figura 19.

O guia de ajuda para Criar um Novo Projeto é obtido pelo engenheiro de software ao clicar no botão Ajuda, Figura 19 (d). Dessa forma, uma nova tela será exibida com o guia específico para esse passo, Figura 20. Para cada passo, há telas semelhantes à apresentada e

que serão omitidas neste capítulo por esse motivo. A documentação completa da ferramenta pode ser encontrada em (Prado, 2003).

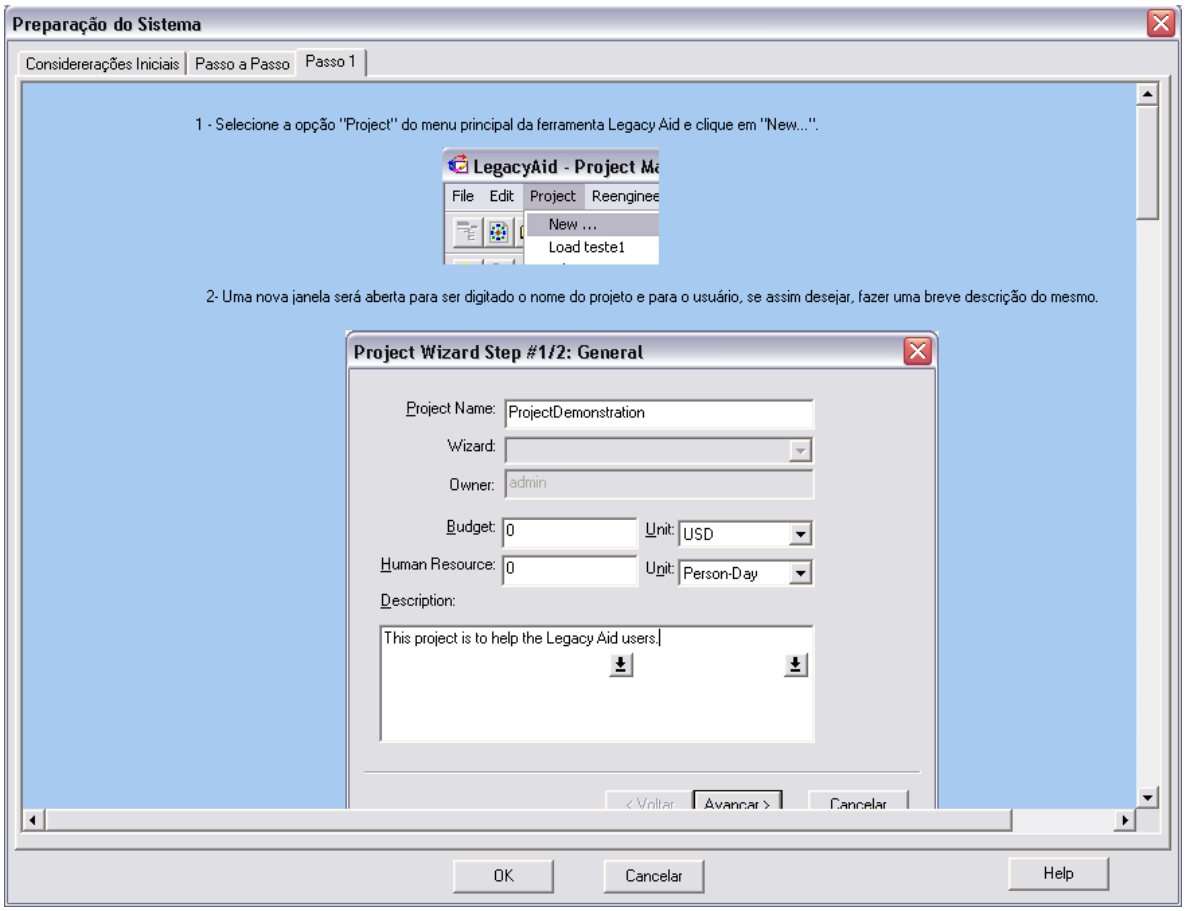
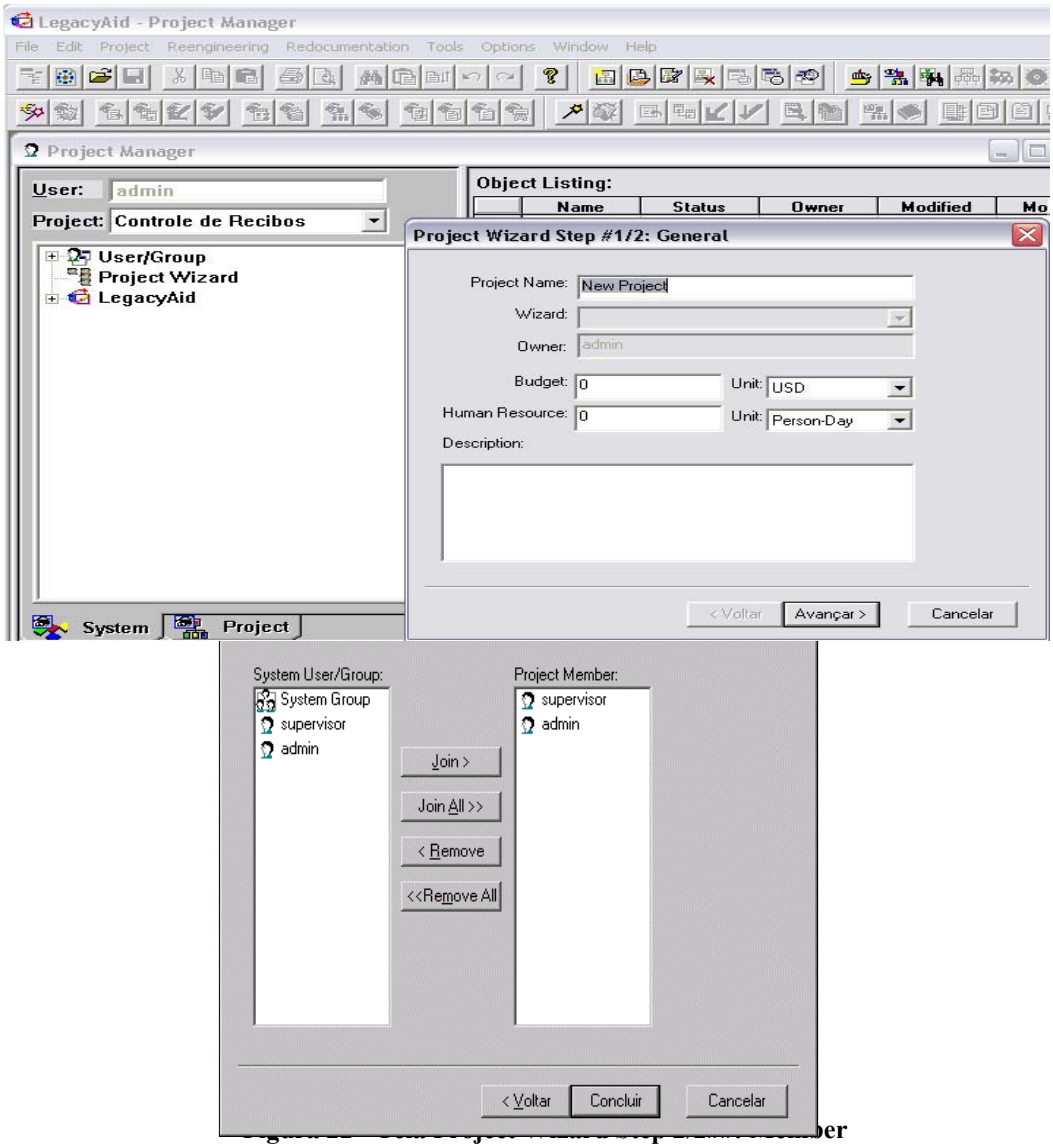


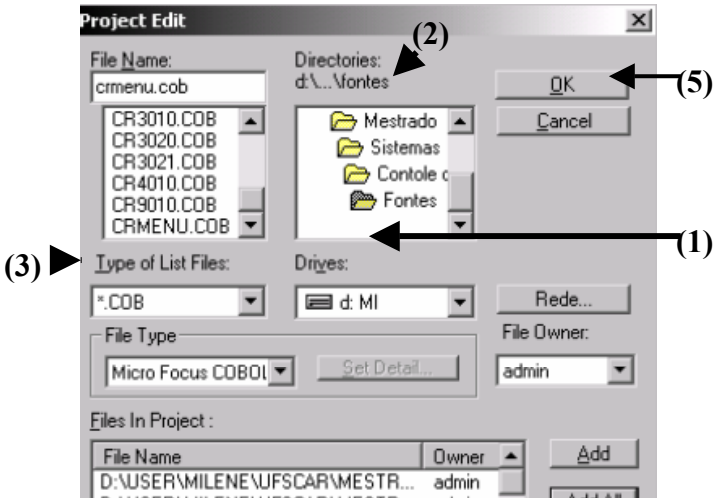
Figura 20 - Guia para a realização do passo Criação de um Novo Projeto

Quando a opção Criar um Novo Projeto for escolhida, Figura 20(a), é apresentada a tela Project Wizard Step #1/2: General, Figura 21. O engenheiro de software deve escolher um nome para o projeto e colocá-lo em Project Name, bem como é possível e recomendado, que faça uma descrição geral do sistema em Description.

Ao clicar em Avançar é exibida a tela Project Wizard Step #2/2: Member, Figura 22, onde o engenheiro de software seleciona em ProjectMember os usuários que terão acesso ao sistema durante o processo de engenharia reversa. Normalmente, são considerados os usuários supervisor e admin, já definidos na tela. Após essa escolha deve-se clicar em Concluir, Figura 22, quando aparece a tela Project Edit, Figura 23, na qual

são definidos os programas que compõem o sistema legado, a partir da seleção do driver em Drivers (1), do diretório em Directories (2) e da extensão dos arquivos em Type of list files (3), que podem ser <<.COB>> e/ou <<CBL>>. O engenheiro de software deve clicar em ADD ALL (4), para carregar todos os arquivos <<.COB>> definidos e em OK (5), para confirmar e finalizar a Criação do Novo Projeto.





Legenda:
1-Seleção do drive onde está o diretório que contém o código fonte do sistema.
2-Seleção do diretório que contém o código fonte do sistema.
3-Seleção da extensão dos arquivos a serem listados e carregados na ferramenta.
4-Adição dos arquivos do sistema ao projeto.
5-Confirmação das

(4)

Figura 23 - Tela Project Edit

Concluída a Criação de um Novo Projeto deve-se proceder a Aplicação do Parser, ativando o botão Aplicar Parser ALL, Figura 19(b). A tela Parsing, Figura 24, é apresentada ao engenheiro de software para que os programas do sistema legado sejam reconhecidos pela ferramenta *Legacy Aid*®.

Figura 24 - Tela Parsing

A próxima atividade consiste do mapeamento dos arquivos *Copy Files* em arquivos <<.COB>>, realizada ao ativar a opção Mapear Copy Files, Figura 19(c). A Tela Mapping é exibida como mostra a Figura 25. O engenheiro de software deve selecionar em Program, o programa, para o qual será feito o mapeamento, optando pela opção Unresolved Copy Files em Mapping Type. Caso seja apresentado no quadro Unresolved Copy Files algum arquivo de estrutura não associado a um diretório válido, o engenheiro de software deve clicar em Browser e selecionar o diretório e o arquivo desejado. Clicando duas vezes no nome do arquivo o mapeamento é realizado. Esse processo deve se repetir quantas vezes forem necessárias para todo o sistema, isto é, para todos os arquivos de estrutura de cada programa do sistema legado.

Se o quadro Unresolved Copy Files não apresentar arquivos pendentes de mapeamento, o engenheiro de software pode selecionar um outro programa em Program e continuar a análise. Os arquivos *Copy Files* são, normalmente, arquivos que descrevem a estrutura de um determinado trecho de código de um programa na linguagem *COBOL*, muitas vezes, esses arquivos estão disponíveis no próprio diretório de arquivos fonte <<.COB>>, quando isso não ocorrer, devem ser mapeados conforme especificado no passo Mapear Copy Files.

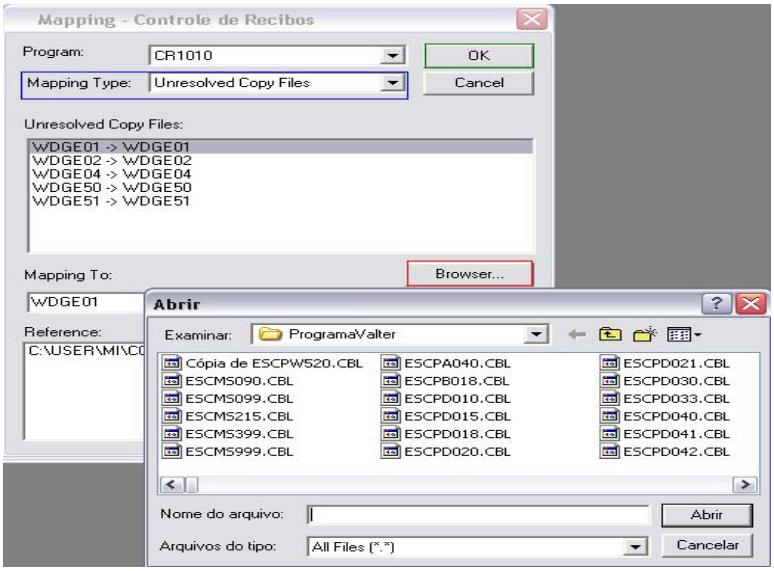


Figura 25 - Tela Mapping para mapeamento dos Copy Files

4.5 Considerações Finais

Neste capítulo fez-se, primeiramente, a apresentação geral da ferramenta, considerando-se os principais recursos por ela oferecidos. Dentre esse recursos destaca-se um editor de textos com as seguintes operações: abrir código, limpar, colar trechos de código, copiar trechos de código, imprimir código, substituir palavras e/ou trechos de códigos, localizar palavras e mudar sua fonte. Há também opções para realização do processo de engenharia reversa dividido em três fases importantes: a de preparação do sistema, a de elaboração do MASA, para determinação das classes, atributos e métodos candidatos, bem como para

determinação dos relacionamentos entre classes e a de elaboração do MAS, fornecendo a modelagem orientada a objetos.

Em seguida, a fase de Preparação do Sistema foi apresentada, na qual são realizadas as seguintes atividades:

- Criação de um Novo Projeto, os arquivos <<.COB>>, do sistema legado, são selecionados e carregados e as considerações gerais do sistema são determinadas, tais como: a definição de um projeto específico para o sistema legado, uma breve descrição do mesmo e a seleção dos usuários que participam da sua recuperação.

- Aplicação do Parser, necessária para que os arquivos <<.COB>> do sistema, carregados no passo anterior, sejam reconhecidos para aplicação dos recursos disponíveis na FAROOL.

- Mapeamento dos Arquivos Copy Files, necessário quando os arquivos *Copy Files*, isto é, arquivos de estrutura do sistema legado, não estão disponíveis no mesmo diretório dos arquivos fontes <<.COB>> do sistema.

Para realização dessa fase, sem uma ferramenta de suporte como a FAROOL, o engenheiro de software teria que analisar cada arquivo fonte <<.COB>> do sistema legado, utilizando um editor de textos, inserir, quando requeridos, os trechos de código específicos dos arquivos de estrutura para todos os programas do sistema, usando os comandos *copy* e *paste*. Para isso, despenderia-se de muito tempo, e conseqüentemente, dinheiro, desestimulando a recuperação e reuso de códigos legados, e, incentivando a criação de novos sistemas de forma desorganizada.

FAROOL permite a otimização das atividades, oferecendo automação de alguns passos para realização das mesmas, e conseqüentemente, favorecendo a recuperação de sistemas legados, cujas funcionalidades ainda atendem as expectativas da empresa.

No próximo capítulo será apresentada a Fase de Elaboração do MASA (Modelo de Análise do Sistema Atual).

Capítulo 5

FAROOL: Fase de Elaboração do MASA

5.1 Considerações Iniciais

Uma vez feita a Preparação do Sistema legado, envolvendo a Criação de um Novo Projeto, a Aplicação do Parse nos arquivos do sistema e o Mapeamento dos Arquivos Copy Files em arquivos <<.COB>>, o engenheiro de software pode prosseguir com processo de engenharia reversa orientada a objetos, preocupando-se em modelar um sistema com características orientadas a objetos.

Este capítulo cuida da elaboração preliminar dos modelos orientados a objetos a partir do código legado. Está organizado da seguinte forma: na seção 5.2 é apresentada cada uma das atividades necessárias para a Fase de Elaboração do MASA e na seção 5.3, são tecidas considerações finais.

5.2 Fase de Elaboração do MASA

Como apresentado anteriormente, no Quadro 14, Capítulo 4, diversas atividades devem ser realizadas para que um modelo pseudo orientado a objetos seja construído. Assim deve determinar: as Classes Candidatas; os Atributos Candidatos; os Métodos Candidatos e os Relacionamentos entre Classes.

5.2.1 Determinação das Classes Candidatas

Na opção EROO deve-se escolher, primeiramente, Elaboração do MASA, e a partir daí Determinação das Classes Candidatas, como pode ser visto na Figura 26. A tela Classes Candidatas da Figura 27 é exibida apresentando na aba Heurística o procedimento que deve ser realizado para que as classes candidatas sejam obtidas.

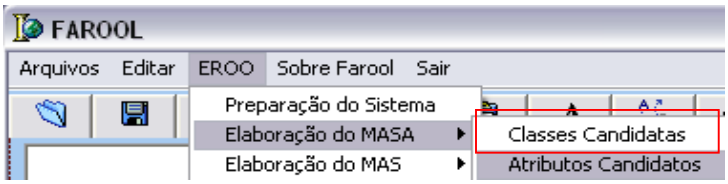


Figura 26 - Opção para Determinação das Classes Candidatas

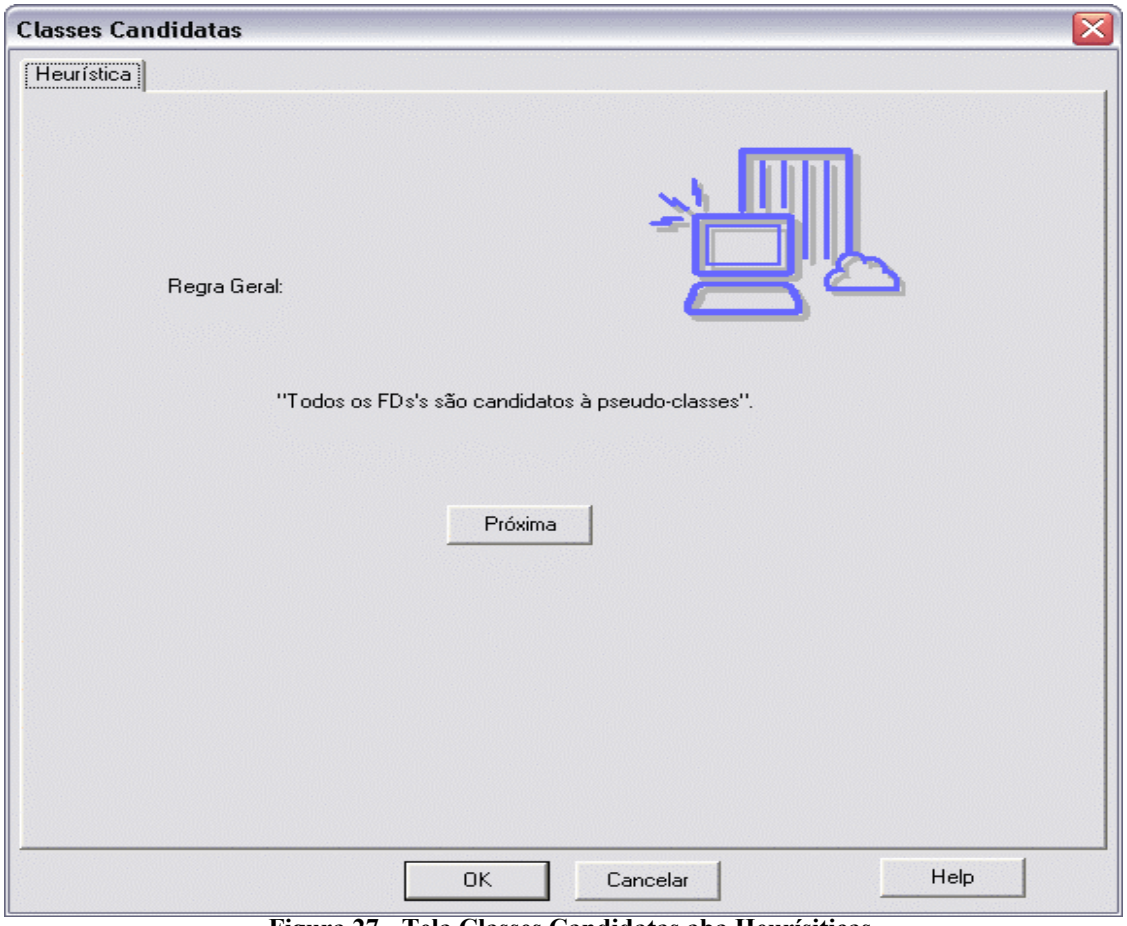


Figura 27 - Tela Classes Candidatas aba Heurísiticas

Quando a opção Próxima for escolhida, é exibida a tela com a aba FDs, Figura 28. Primeiramente, caso a ferramenta *LegacyAid*® não esteja aberta, o engenheiro de software deve escolher a opção pm-legacyaid (a) para abrir essa ferramenta.

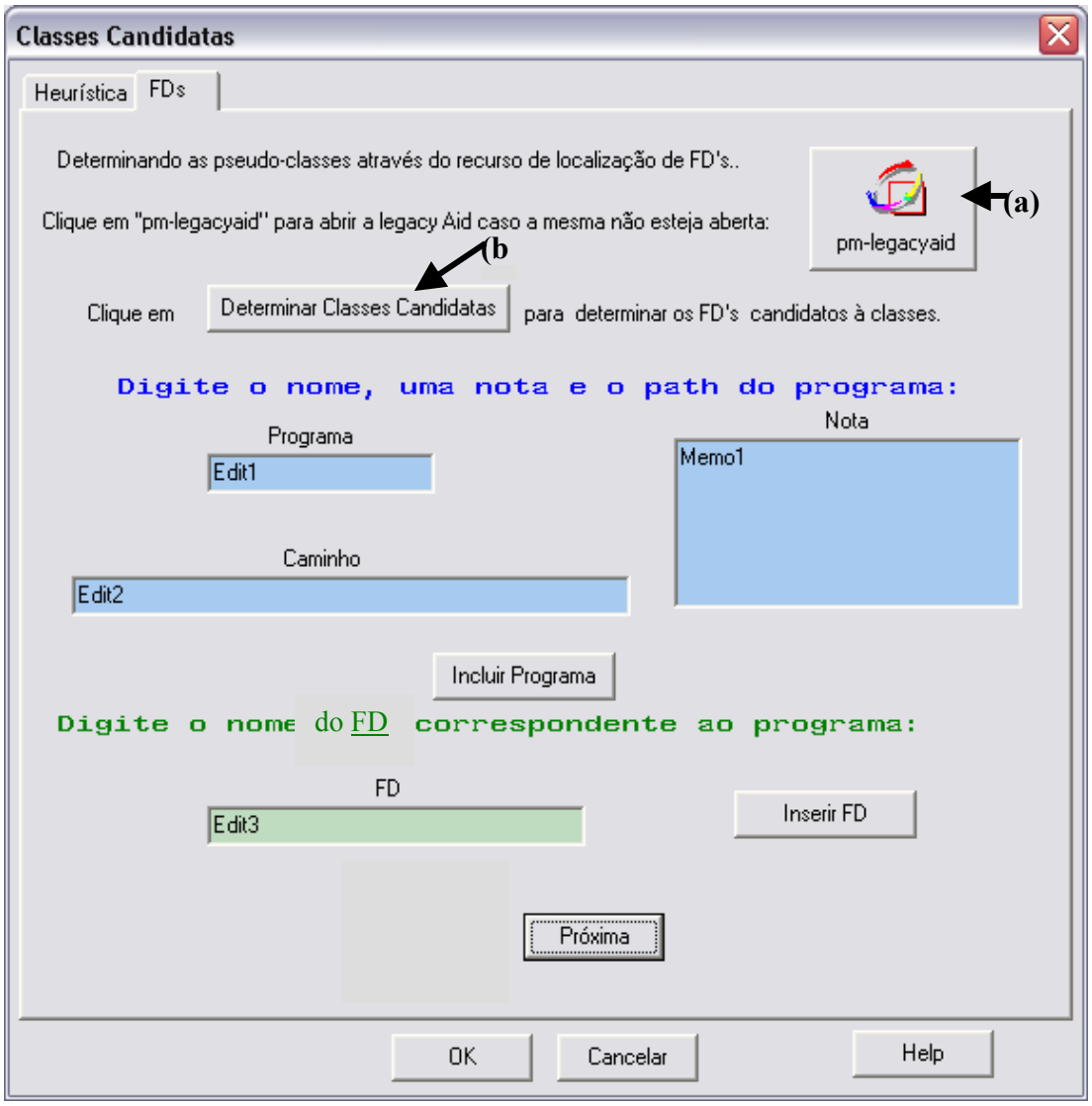
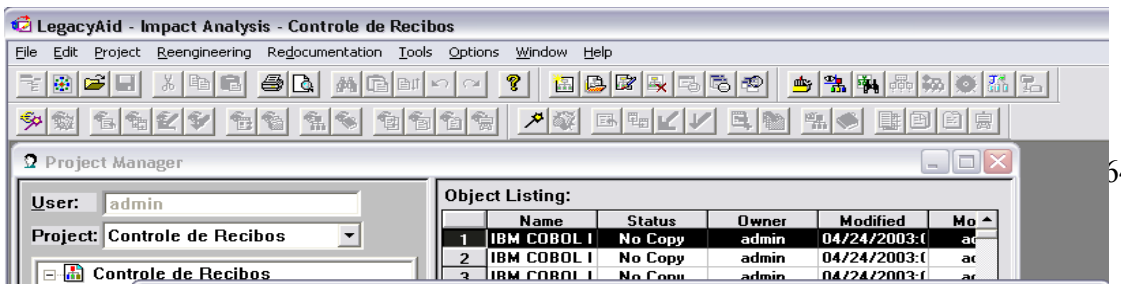


Figura 28 - Tela Classes Candidatas aba FDs

Ao escolher a opção Determinar Classes Candidatas (b), a tela Impact Analysis é exibida como na Figura 29. O engenheiro de software deve então selecionar a opção Program em Primary Class e FD em Secondary Class e clicar em Display Result.



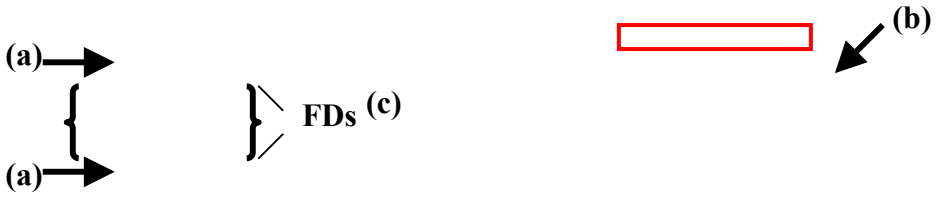


Figura 29 - Tela Impact Analysis

O engenheiro de software deve identificar os programas apresentados em (a) da Figura 29 e o caminho exibido em Definition, apresentado em (b). Com base nessas informações devem ser preenchidos os campos Programa, Caminho e Nota, requeridos pela FAROOL e disponíveis na aba FD, da tela Classes Candidatas, Figura 28. Para confirmar a inserção dessas informações o engenheiro de software deve escolher a opção Incluir Programa, Figura 28. Esse processo deve ser feito para todos os programas exibidos em (a) da Figura 29. Para cada programa incluído, o engenheiro de software deve informar os respectivos FDs, disponíveis na Figura 29 em (c), e preencher o campo FD requerido na Figura 28. Para confirmar a inserção do FD correspondente ao programa inserido, o engenheiro de software deve escolher a opção Inserir FD na Figura 28.

Para futuras análises, ao longo do processo de engenharia reversa do sistema legado, é necessário determinar as chaves dos programas em *COBOL*. Nos novos relacionamentos, por exemplo, as chaves são utilizadas para determinar se existem associações entre as classes e a cardinalidade desses relacionamentos. Esse processo será explicado detalhadamente, posteriormente, neste capítulo.

Ao selecionar a opção Próxima da Figura 28, a tela da Figura 30 é apresentada para tratar a Determinação das Chaves dos Programas quando a aba com esse nome for

selecionada. Optando-se por Determinar as Chaves dos Programas a tela Find In Files é exibida, os programas são apresentados em Look in e suas respectivas chaves podem ser identificadas em Statement, Figura 31.

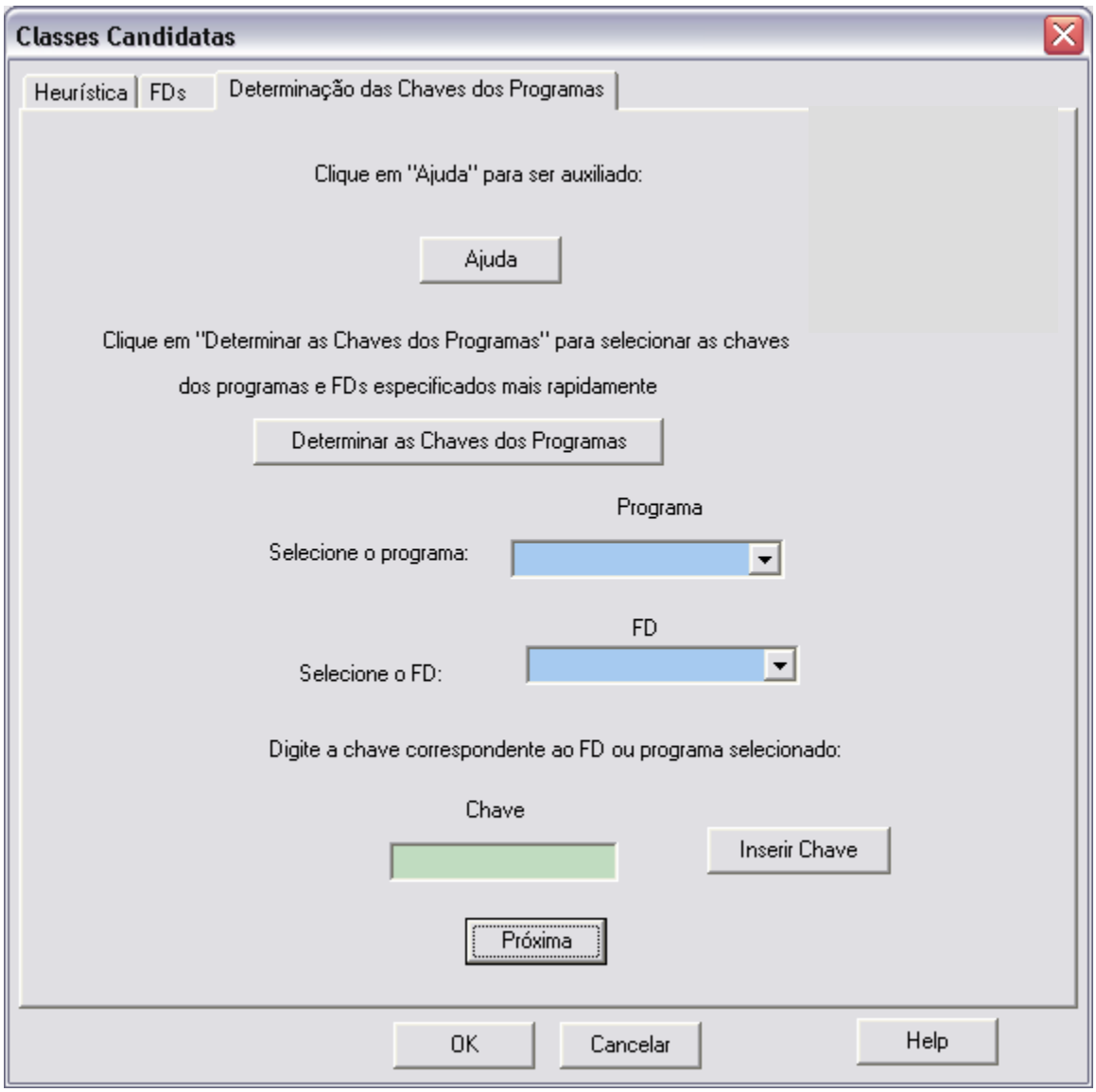


Figura 30 - Tela Classes Candidatas aba Determinação das Chaves dos Programas

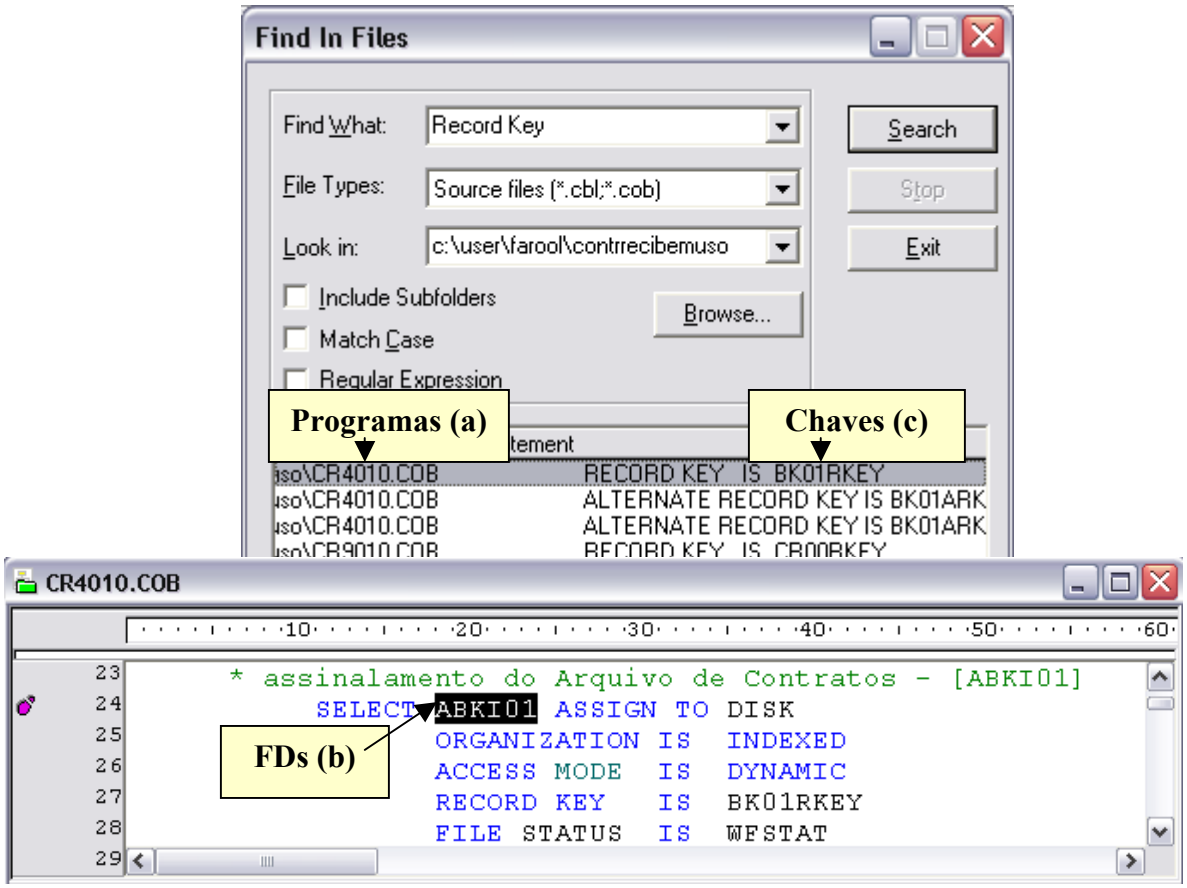


Figura 31 - Tela Find In Files para Determinação das Chaves

Considerando a Figura 30, o engenheiro de software deve selecionar o programa em Selecione o Programa, o FD em Selecione o FD, e preencher manualmente, com o nome da chave, o campo Chave. Os programas, FDs e respectivas chaves, tidas como entradas nesses campos, podem ser obtidas considerando as informações rotuladas por (a), (b) e (c) na Figura 31. Para exibição da tela do programa CR4010.COB, que oferece o FD correspondente a ele, é necessário um clique-duplo no nome do programa exibido na Figura 31 (a). Para confirmar a inserção da chave, correspondente ao programa e FD selecionados, o engenheiro de software deve escolher a opção Inserir Chave, Figura 30. O processo de Determinação das Chaves dos Programas descrito deve ser repetido para todos os programas exibidos na Figura 31 (a).

Quando a opção Próxima da tela da Figura 30 for ativada, a tela representada pela aba Classificação dos Programas e FDs associados aos procedimentos de leitura, escrita ou reescrita, Figura 32, é exibida

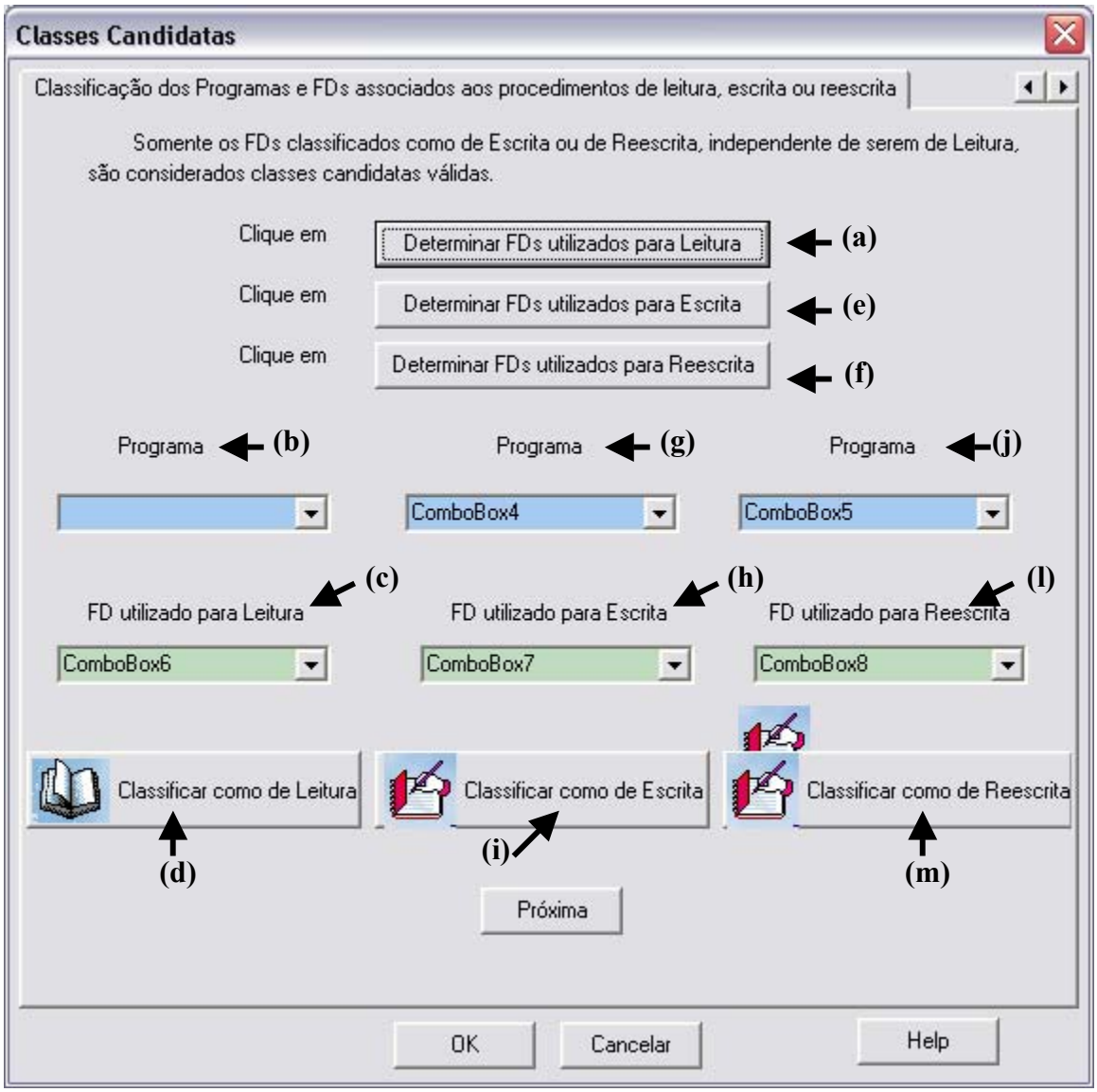


Figura 32 - Tela Classes Candidatas aba Classificação dos FDs

A classificação dos programas é necessária para determinar quais são as classes válidas ao processo de engenharia reversa orientada a objetos. Tendo como base heurísticas pré-determinadas, são consideradas válidas as classes candidatas oriundas de FDs associados aos procedimentos de escrita e/ou reescrita e os utilizados em procedimentos de leitura. Esses procedimentos bem como os programas a eles associados são chamados de construtores. Alguns procedimentos são próprios da linguagem de implementação utilizada sendo então denominados como de implementação. Outros estão associados aos procedimentos que fazem leitura de dados que são chamados de observadores.

Primeiramente, devem ser classificados os programas e FDs associados aos procedimentos de leitura (observadores), optando-se por Determinar FDs utilizados para Leitura, Figura 32 (a), é exibida a tela Find In Files, Figura 33.

Para garantia de que o nome associado ao comando READ refere-se aos dados corretos para análise, o engenheiro de software pode visualizar o código fonte da *procedure division*, Figura 34, a partir de um clique-duplo no caminho especificado em Look in, Figura 33(a). Com essa certeza, a tela da Figura 32 pode ser completada selecionando a informação a partir dos *combo(s)*: Programa, o programa desejado, no exemplo CR1010; FD utilizado para Leitura, o FD desejado, no exemplo ACRI00 e ativar o botão Classificar como de Leitura, Figura 32 (b), (c) e (d), respectivamente.

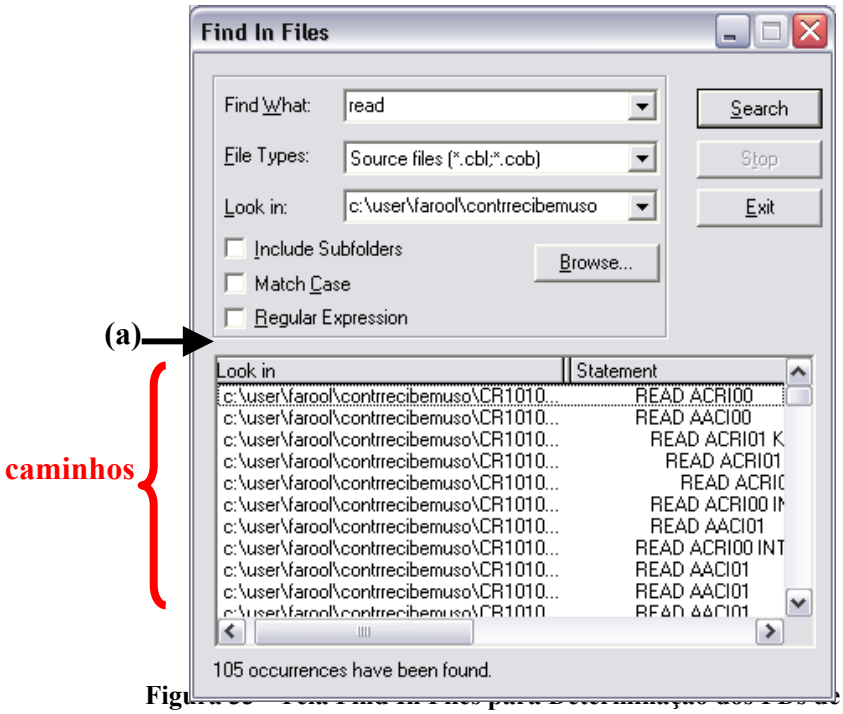


Figura 33 – Tela Find In Files para Determinação dos FDs de Leitura

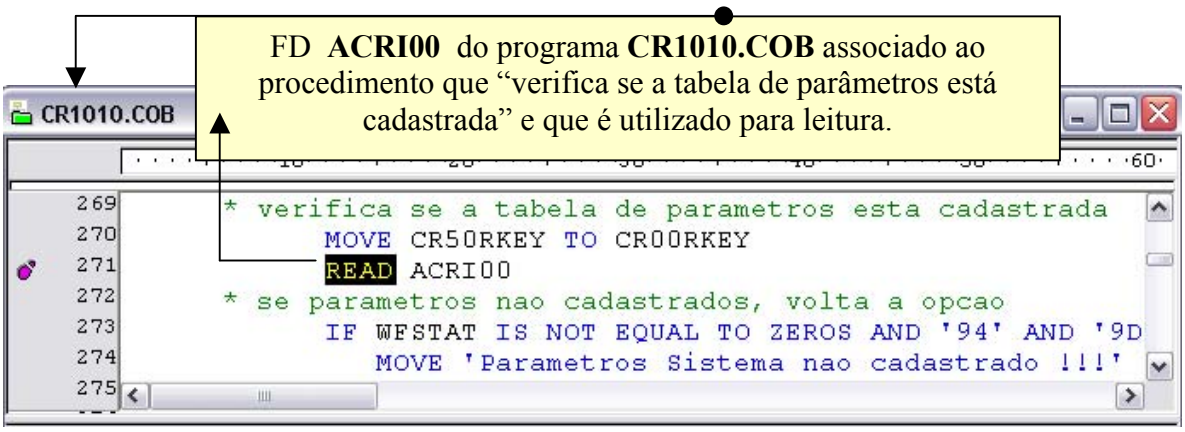


Figura 34 - Telas Find In para FDs de Escrita para FDs de Reescrita

O mesmo procedimento, descrito anteriormente para os observadores, é usado para classificar os FDs associados aos procedimentos de Escrita/Reescrita. Escolhendo-se a opção Determinar FDs utilizados para Escrita/Reescrita, Figura 32 (e) e (f) respectivamente, selecionando-se aqueles FDs apresentados em Look in, com WRITE ou REWRITE, respectivamente, no campo Statement (semelhante ao da tela exibida na Figura 33).

Escolhendo a opção Próxima na tela apresentada na Figura 32, a tela Resultado da Classificação é exibida, conforme apresentado na Figura 35.

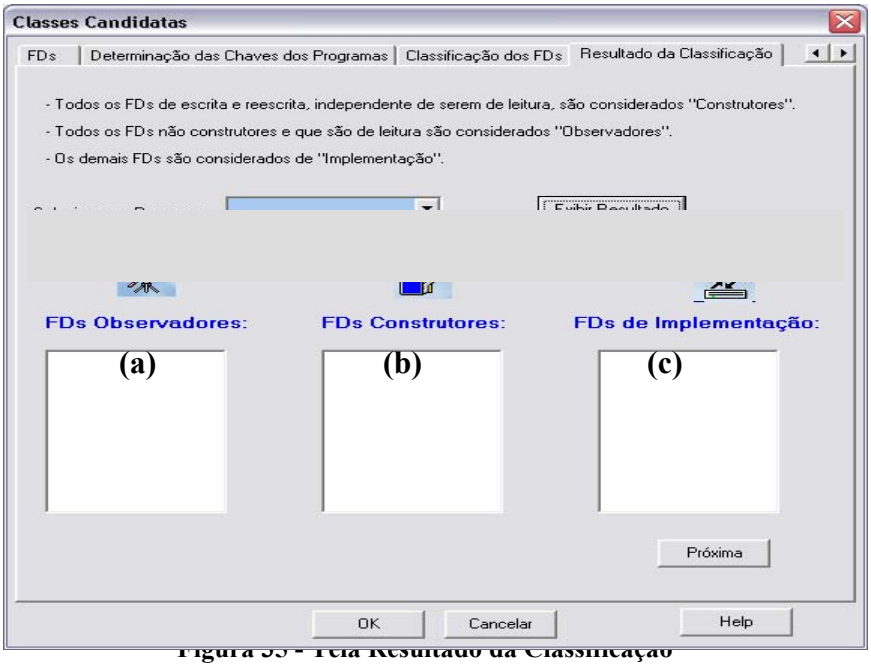


Figura 35 - Tela Resultado da Classificação

Selecionando um programa no *combo* Selecione um Programa e a opção Exibir Resultado, os quadros FDs Observadores (a), FDs Construtores (b) e FDs de Implementação (c) exibirão os FDs associados ao procedimentos observadores, construtores e de implementação, respectivamente, classificados como anteriormente descrito.

Se a opção Próxima, Figura 35, for escolhida a tela Resultado Final da Análise é exibida como mostra a Figura 36, sendo que o engenheiro de software determinou as classes candidatas e pode passar à análise dos resultados obtidos.

As classes candidatas consideradas são aquelas cujos FDs estão associados aos métodos que as modificam. Para visualizar as classes, o engenheiro de software deve escolher a opção Classes Candidatas Válidas e/ou a opção Mostrar Classes Candidatas. A primeira opção permite somente a visualização das classes válidas não permitindo a inclusão e/ou exclusão de novas classes, Figura 36 (a). Já a segunda opção, oferece um quadro em forma de tabela, Figura 37, no qual o engenheiro de software pode inserir, excluir e/ou navegar pelas classes válidas.

Considerando Figura 36, uma outra forma de remover alguma classe do modelo em análise é selecionando o programa em Selecione um programa, a classe candidata a ser removida em Selecione uma classe Candidata e escolhendo a opção Remover Classe Candidata. Essa opção dá ao engenheiro de software a possibilidade de simplificar e adaptar o modelo de análise do sistema legado, de acordo com o conhecimento que possui sobre sistemas e sobre desenvolvimento orientado a objetos.

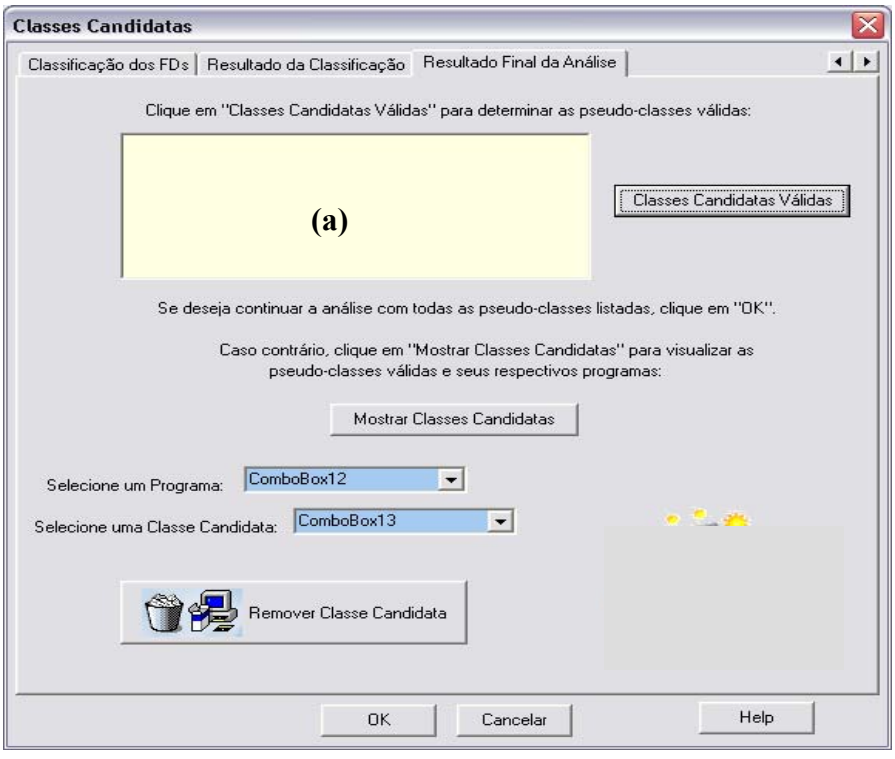


Figura 36 - Tela Resultado Final da Análise

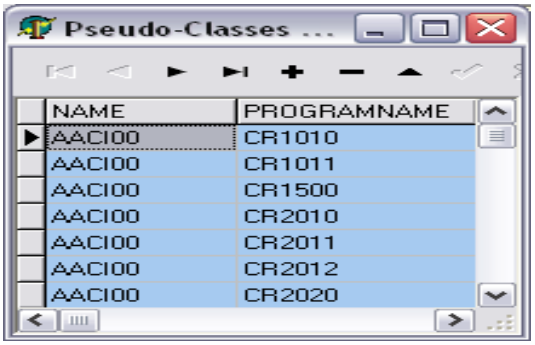


Figura 37 - Resultado Final da Análise trazido do banco FAROOL

5.2.2 Determinação dos Atributos Candidatos

Dando continuidade ao processo de Elaboração do MASA, após a determinação e inserção das classes candidatas, o engenheiro de software deve determinar e inserir os atributos candidatos. Isso é feito escolhendo-se a opção Atributos Candidatos, no menu EROO da FAROOL, Figura 38.

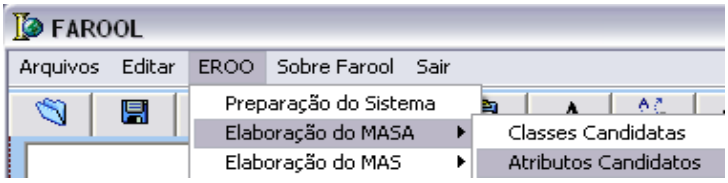


Figura 38 - Opção para Determinação dos Atributos Candidatos

A tela Atributos Candidatos tem o *layout* da Figura 39 e assim como para a Determinação das Classes Candidatas, a primeira aba apresenta a heurística utilizada para essa atividade.

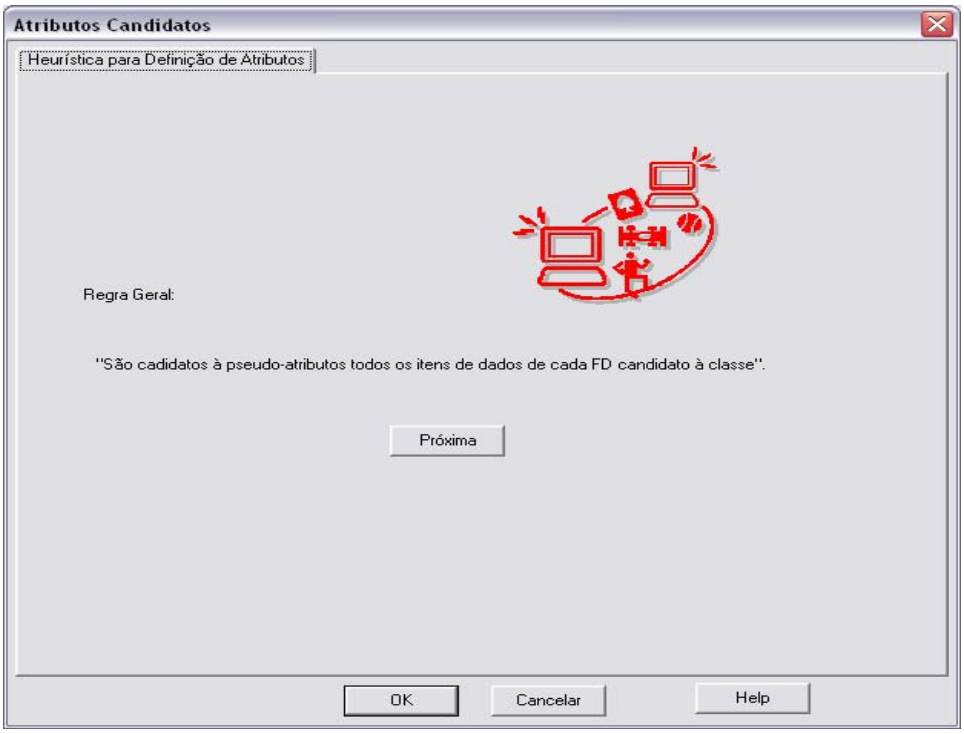
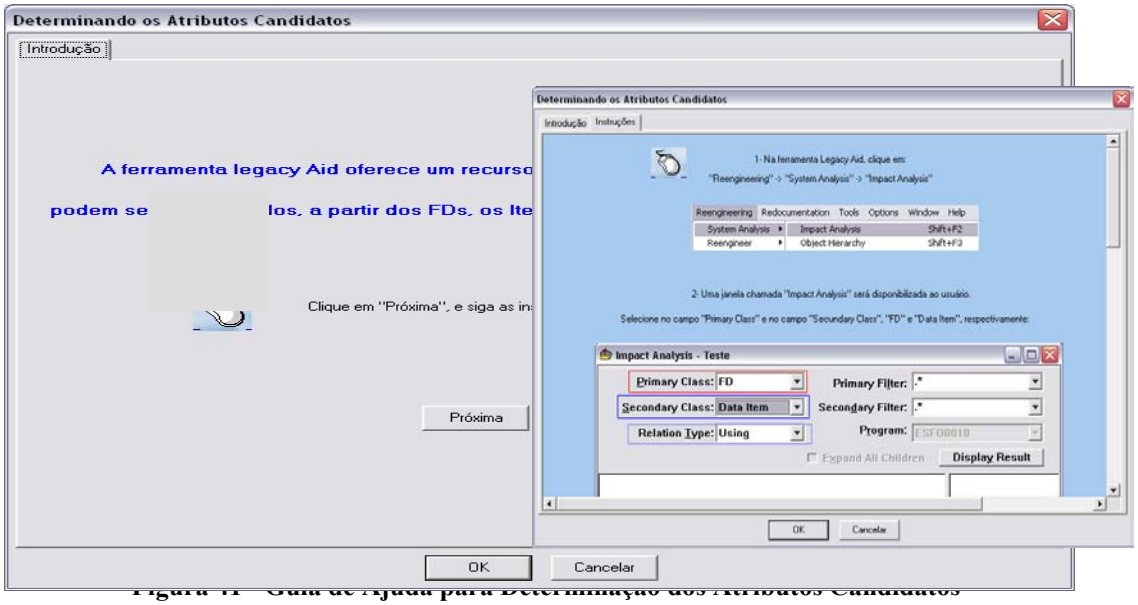


Figura 39 - Tela Heurística para Definição dos Atributos

Clicando em Próxima, a tela Determinação dos Atributos é exibida, Figura 40, sendo que o engenheiro de software pode optar por Ajuda ou Determinação dos Atributos Candidatos. Se a opção for por Ajuda, um guia auxiliará o processo como mostrado na Figura 41. Como citado anteriormente, esse tipo de guia é oferecido em toda a ferramenta por um botão rotulado de Ajuda.

Figura 40 - Tela Atributos Candidatos aba Determinação dos Atributos



Considerando a tela da Figura 40, se a opção Determinação dos Atributos Candidatos for escolhida, a tela Impact Analysis é exibida, Figura 42.

Figura 42 - Tela Heurística para Definição dos Atributos

Ao selecionar a opção FD em Primary Class e Data Item em Secondary Class, e clicar em Display Result, da Figura 42, será fornecida uma lista com os FDs e seus respectivos atributos; em Definition é exibido o caminho em que o FD foi definido. Com

um clique-duplo nesse caminho é exibida a tela correspondente ao trecho de código do FD escolhido, Figura 43.

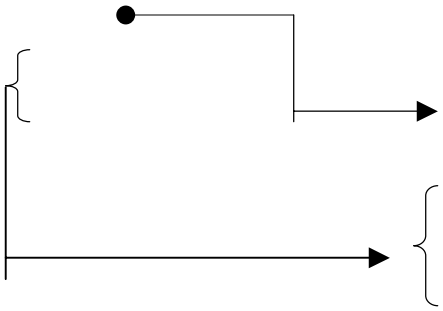


Figura 43 - Tela correspondente ao trecho de código do FD AACI00

Juntamente com a tela Impact Analysis é oferecida ao engenheiro de software a aba Determinação dos Atributos na tela Atributos Candidatos, Figura 44.

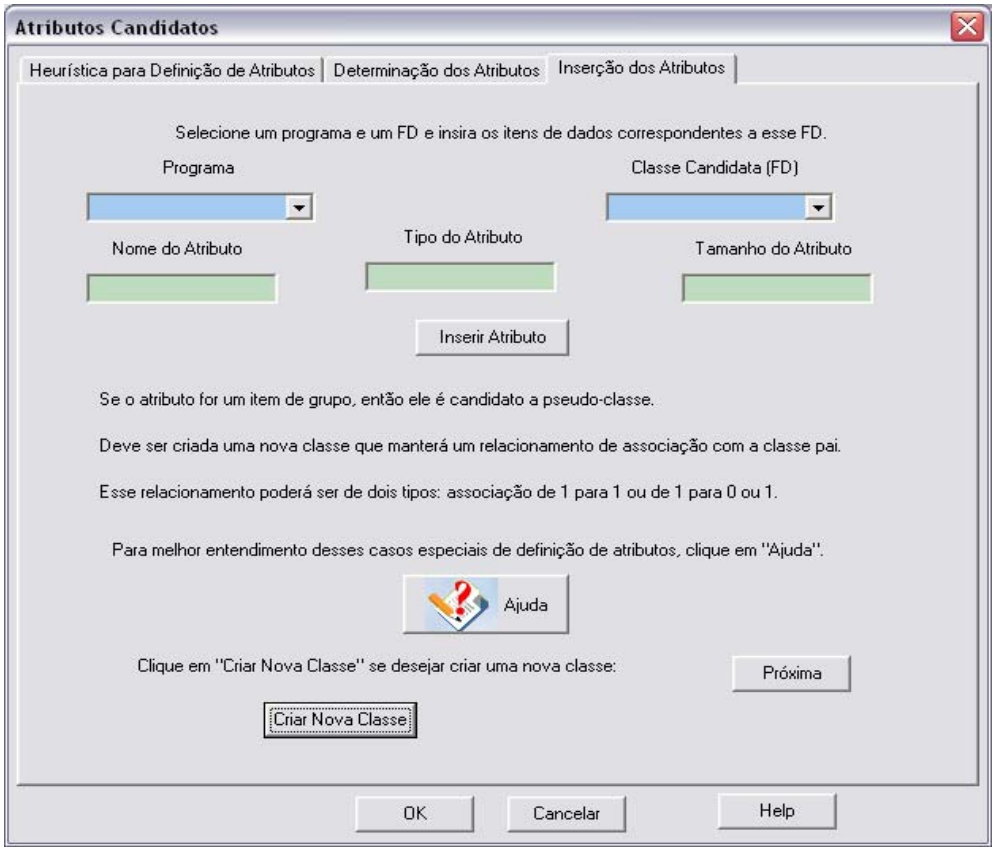


Figura 44 - Tela Atributos Candidatos aba Inserção dos Atributos

O engenheiro de software deve selecionar um programa e uma classe candidata nos *combo(s)* com esses rótulos, além de preencher os campos Nome do Atributo, Tipo do Atributo e Tamanho do Atributo. Essas informações são obtidas através do código fonte do FD, conforme apresentado na Figura 43. Para confirmar a inserção dessas informações, a opção Inserir Atributo deve ser escolhida. As informações são, então, armazenadas no banco de dados da FAROOL para posteriores consultas e análises.

Dentro da linguagem *COBOL* existem números de níveis que permitem a estruturação de um registro lógico com números que variam de 01 a 49. Uma vez que uma subdivisão tenha sido especificada, tem-se um item de grupo que pode ser ainda mais subdividido, dando origem aos itens elementares que não são mais subdivididos. Um registro pode ser constituído de uma seqüência de itens elementares ou pode ser somente um item elementar. Um item de grupo é uma seqüência de um ou mais itens elementares, ou também, de um ou mais itens de grupo. Quando isso ocorre, ao selecionar Ajuda encontra-se o procedimento a seguir. Uma nova classe deve ser criada, selecionando-se a opção Criar Nova Classe na Figura 44, sendo então apresentada a tela da Figura 45.

Ao selecionar o Nome do Programa, o Nome da Classe Pai e o Nome do Atributo, requeridos na Figura 45 e indicados por (a), (b) e (c), uma nova classe é criada clicando-se em Criar Classe.

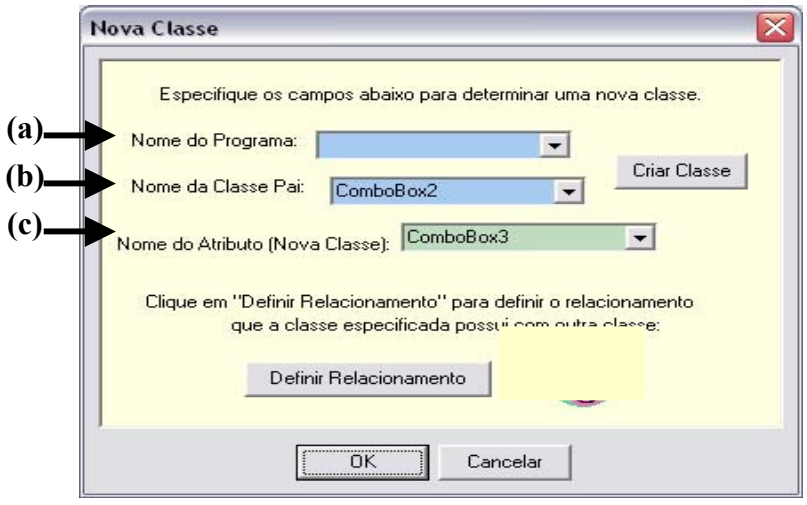


Figura 45 - Tela Nova Classe para Criar Novas Classes

Baseado em heurísticas, a classe pai e a nova classe mantém, normalmente, um relacionamento de associação com cardinalidade 1 para 1. Clicando em Definir Relacionamento, o engenheiro de software pode definir esse relacionamento passo a passo, Figura 46. Esses relacionamentos são os encontrados na *UML* e podem ser de Dependência (D), de Associação (A), de Interface/Realize (R) ou de Generalização (G), indicados pelas letras (a), (b), (c) e (d), respectivamente na Figura 46.

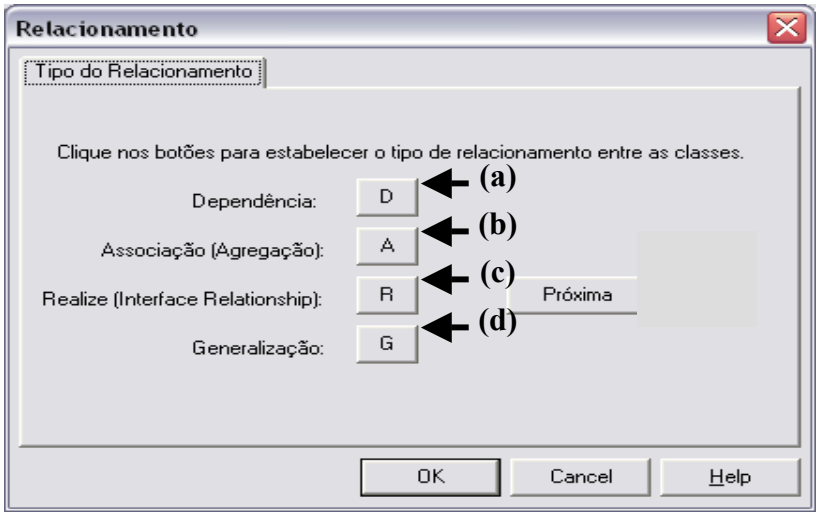
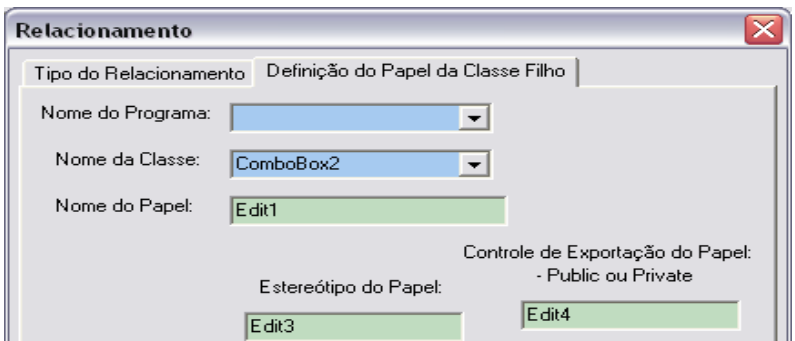


Figura 46 - Tela Relacionamento aba Tipo do Relacionamento

Clicando-se em Próxima na tela apresentada, Figura 46, a tela Relacionamento aba Definição do Papel da Classe Filho é exibida. O engenheiro de software deve, então, especificar o Nome do Programa, o Nome da Classe e definir o Nome do Papel para a classe filho, conforme Figura 47 referências (a), (b) e (c), respectivamente. O campo Controle de Exportação do Papel pode ser desconsiderado pelo engenheiro de software, a menos que deseje registrar o papel do relacionamento, em relação à classe filho, quanto a Public ou Private. Para confirmar a definição do papel da classe filho deve-se clicar em Definir Papel do Filho, sendo que a ferramenta armazena as informações no banco de dados para posteriores análises.



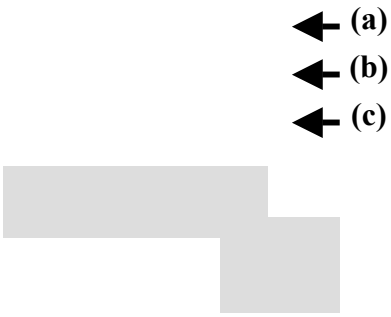


Figura 47 - Tela Relacionamento aba Definição do Papel da Classe Filha

Escolhendo-se a opção Próxima, a tela Relacionamento aba Definição do Papel da Classe Pai é exibida, Figura 48. O engenheiro de software deve proceder da mesma forma que para a Definição do Papel do Filho.

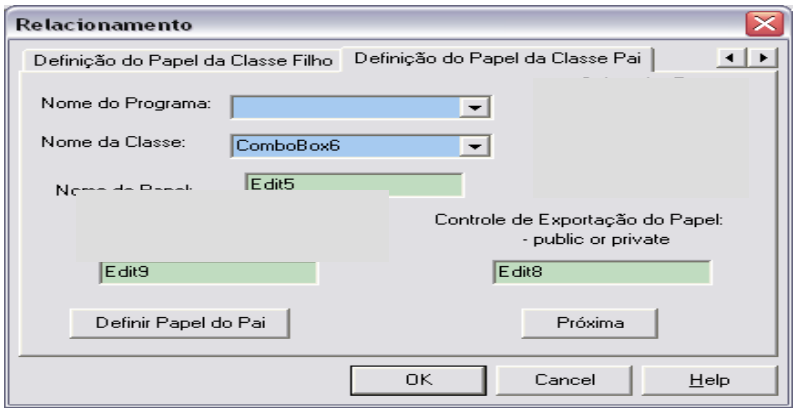


Figura 48 - Tela Relacionamento aba Definição do Papel da Classe Pai

Clicando-se em Próxima, Figura 48, a tela Especificações do Relacionamento da Classe Filho é exibida para detalhar esse relacionamento. Se o relacionamento entre classes for de Dependência, de Interface/Realize ou de Generalização, essa tela apresenta-se como uma notificação de que o processo de determinação do relacionamento entre classes foi concluído, nenhuma especificação é requerida e o usuário pode continuar o processo de determinação dos atributos, como mostra a Figura 49 para o caso de relacionamento de dependência.

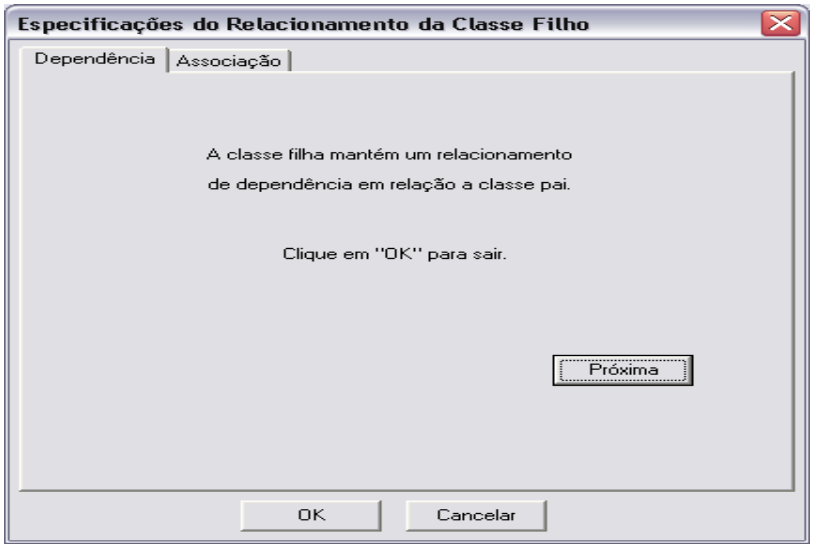


Figura 49 - Tela Especificações do Relacionamento da Classe Filho aba Dependência

Sendo o relacionamento de associação/agregação é exibida a tela Especificações do Relacionamento da Classe Filho aba Associação para que se determine a cardinalidade e o relacionamento: se É de agregação? ou É navegável?, Figura 50. Para isso, devem ser selecionados: o programa em Selecione o Programa, o nome da classe filho em Selecione a Classe e o nome do role em Selecione o Nome do Role. Para confirmar as especificações, o engenheiro de software deve clicar em Atualizar o Papel do Filho. O mesmo processo deve ser realizado para especificar o relacionamento para a classe pai. Clicando em Próxima, tela da Figura 50, é exibida a tela Especificações do Relacionamento da Classe Pai aba Associação com o mesmo *layout* que a anterior.

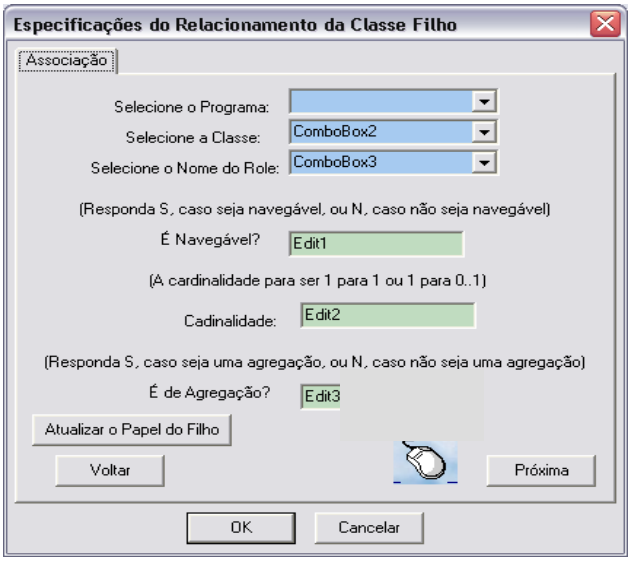


Figura 50 - Tela Especificações do Relacionamento da Classe Filha aba Associação

Após a determinação dessas especificações, o engenheiro de software deve retornar a tela apresentada na Figura 44, selecionar o programa em Programa (correspondente à nova classe criada a partir da tela da Figura 45); a nova classe em Classe e especificar o Nome do Atributo, o Tipo do Atributo e o Tamanho do Atributo. Esses atributos são os itens elementares dessa nova classe e podem ser obtidos na tela correspondente ao trecho de código do FD apresentada na Figura 43.

Ao clicar em Próxima na tela da Figura 44, a tela Atributos Candidatos aba Exibir Resultado é aberta, oferecendo recursos para visualização dos atributos, Figura 51. O engenheiro de software deve clicar em Resultado da Determinação dos Atributos para visualizar os atributos candidatos (Name), Figura 52, bem como suas respectivas classes (Classname), programas (Programname), tipos (Typename) e tamanhos (Scale), que foram armazenados no banco da FAROOL. São permitidas: inserções de atributos (+), remoções de atributos (-), e a navegação pelos mesmos (►, ◄ , ▲), utilizando o navegador representado por (a) na Figura 52.

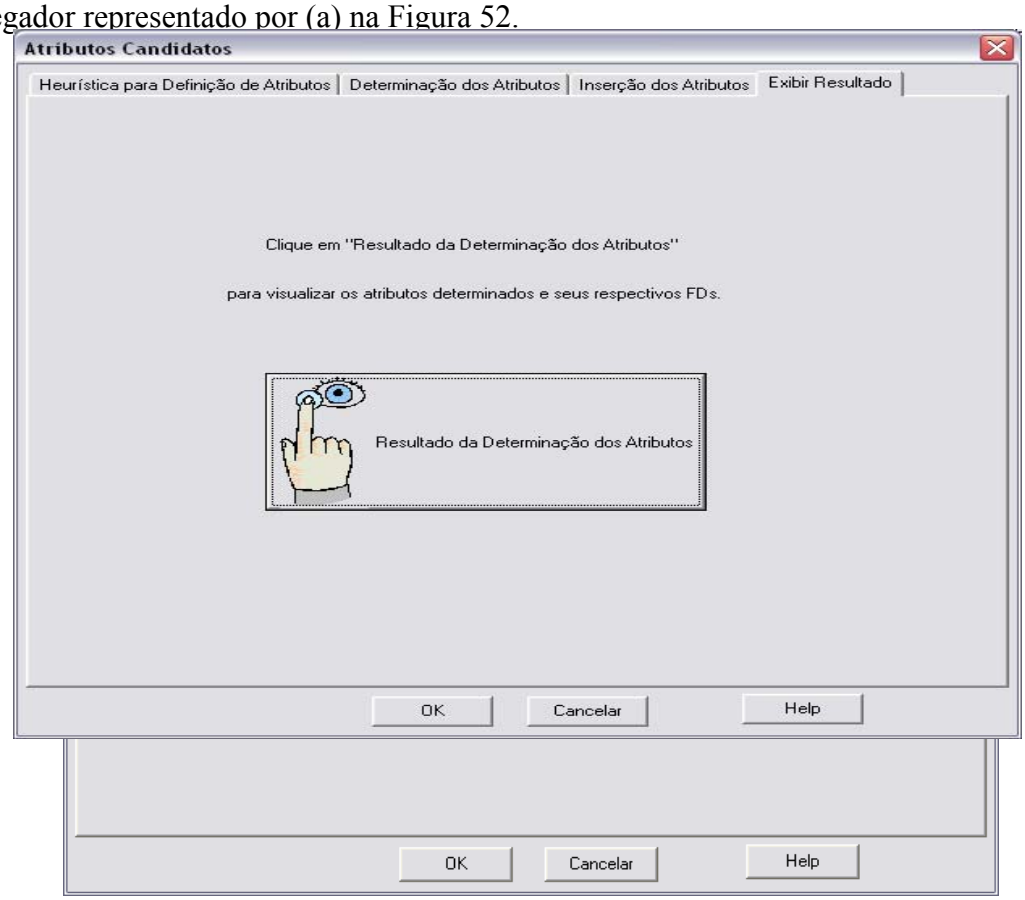
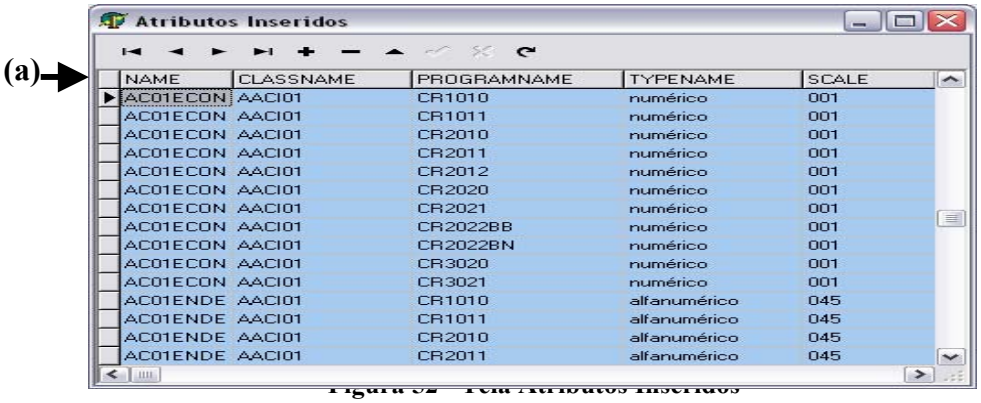


Figura 51 - Tela Atributos Candidatos aba Exibir Resultado



5.2.3 Determinação dos Métodos Candidatos

Dando continuidade ao processo de elaboração do MASA, após a determinação e inserção dos atributos candidatos, o engenheiro de software deve determinar e inserir os métodos candidatos. A determinação dos métodos candidatos começa a ser realizada optando-se por Métodos Candidatos, no menu principal da FAROOL, Figura 53.

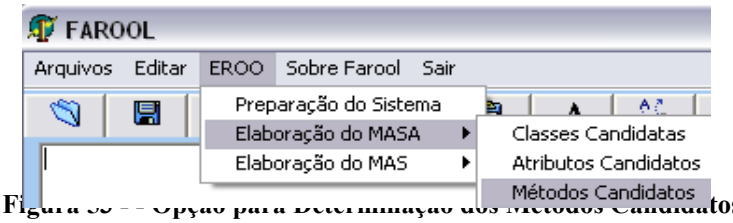


Figura 53 - - Opção para Determinação dos Métodos Candidatos

A tela rotulada de Métodos Candidatos aba Heurística para Definição dos Métodos, Figura 54, é apresentada, exibindo a heurística utilizada nessa atividade.

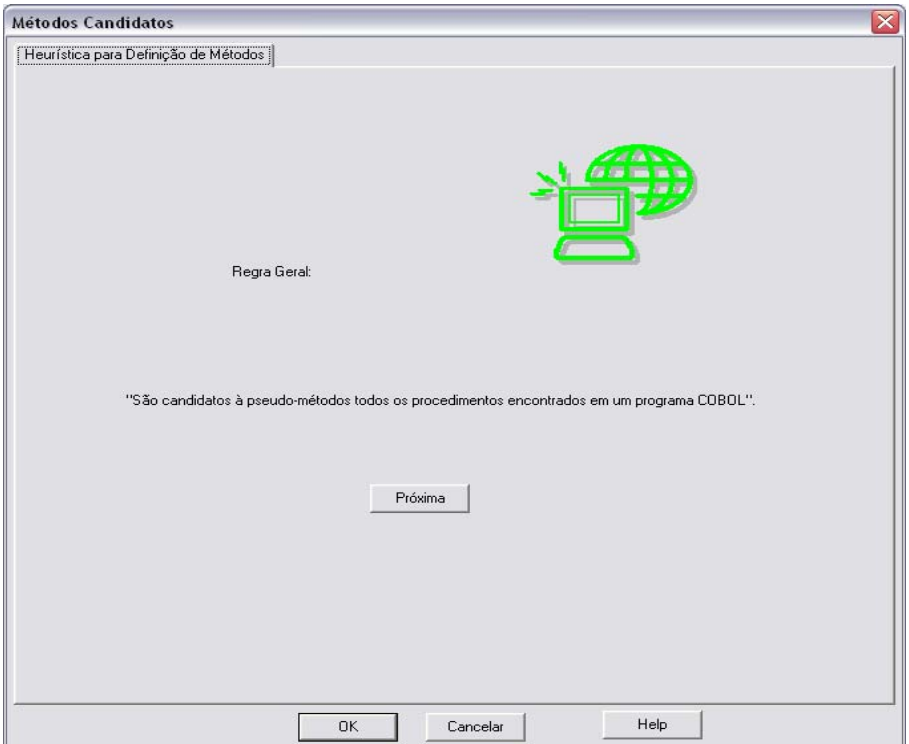


Figura 54 - Tela Métodos Candidatos aba Heurística para Definição dos Métodos

Clicando em Próxima, é exibida a tela da Figura 55. O engenheiro de software pode solicitar Ajuda, em (a), e/ou ativar diretamente o recurso Determinação dos Métodos Candidatos, em (b), com o qual a tela Impact Analysis é exibida, Figura 56.

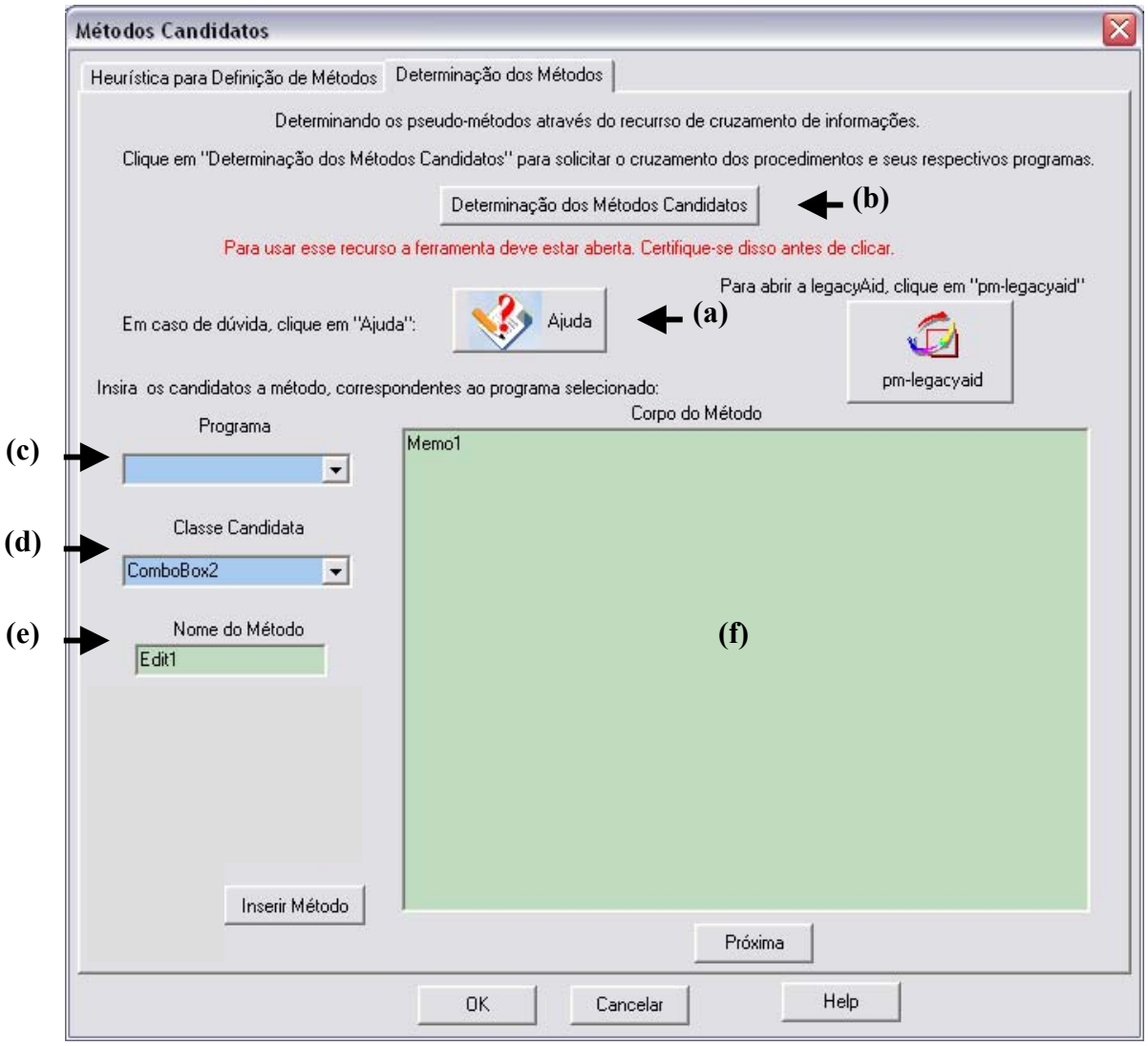


Figura 55 - Tela Métodos Candidatos aba Determinação dos Métodos

Ao selecionar a opção Program em Primary Class e Paragraph em Secundary Class e clicar em Display Result, (a), uma lista com os programas e os métodos será exibida em (b). O corpo do método pode ser identificado com um clique-duplo no caminho disponível em Definition, (c), o mesmo procedimento realizado para classes. A partir das informações obtidas os campos rotulados por Programa, Classe Candidata, Nome do Método e Corpo do Método, Figura 55 indicados, respectivamente, em (c), (d), (e), (f), são preenchidos. Normalmente, os procedimentos, aqui denominados de métodos, estão associados a programas que utilizam várias ou todas as classes desses programas. Nesse caso, o engenheiro de software deve selecionar o programa e para cada classe desse, associar o método.

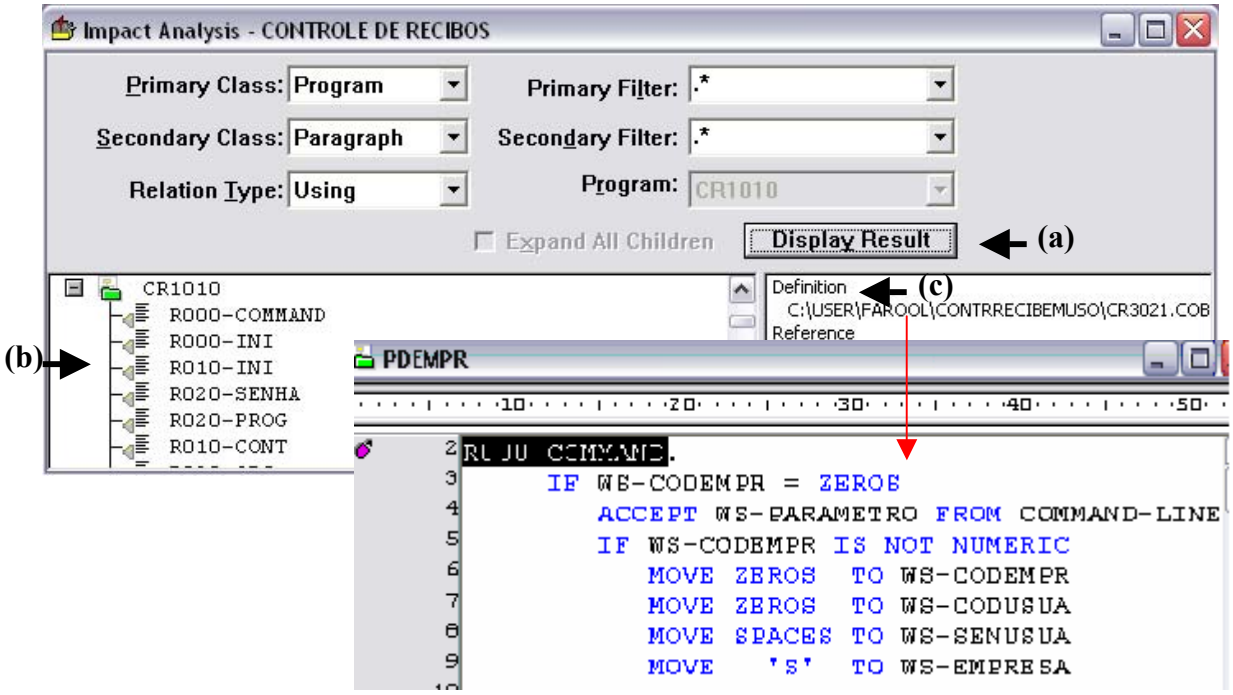


Figura 56 - Tela Impact Analysis para Determinação dos Métodos Candidatos

Clicando em Próxima na tela apresentada na Figura 55, é exibida a tela Métodos Candidatos aba Processo de Classificação dos Métodos, Figura 57. Para melhor documentação e posteriores análises, os métodos são classificados em observadores, construtores e de implementação. Os métodos observadores são os métodos de leitura que não modificam dados do programa. Os métodos construtores são métodos capazes de modificar dados através da escrita, da reescrita e/ou da regravação dos mesmos. Os

métodos de implementação são aqueles próprios ao tipo de linguagem implementação utilizada. Essas informações são armazenadas no banco de dados da FAROOL para posteriores análises pertinentes ao processo de engenharia reversa.

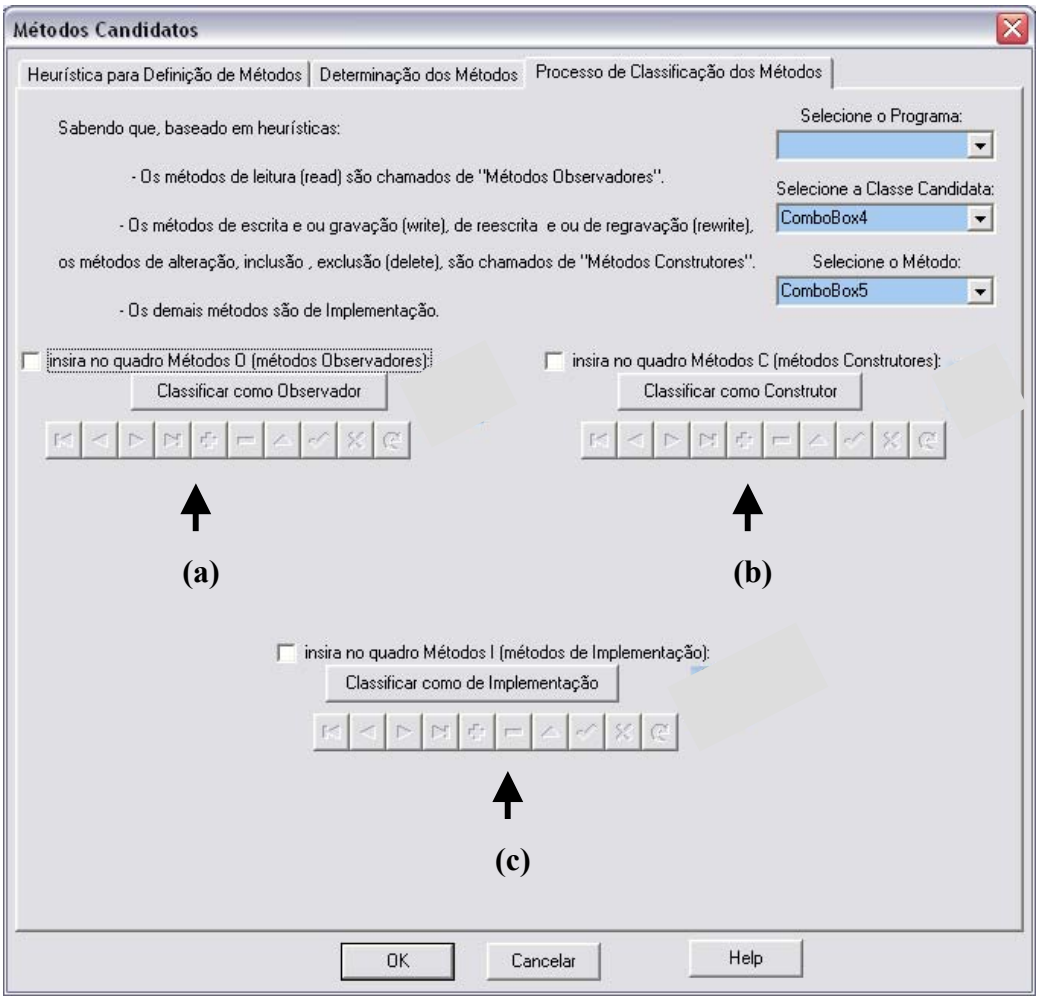


Figura 57 - Tela Métodos Candidatos aba Processo de Classificação dos Métodos

Para classificar os métodos o engenheiro de software deve fazer a leitura e a análise do trecho de código correspondente ao método, como mostrado na Figura 56(c). Posteriormente, na tela da Figura 57, ele seleciona o programa para o qual ele classificará os métodos (Selecione o Programa), a Classe para a qual ele deseja que o método atue com essa classificação (Selecione a Classe Candidata), e o próprio método a ser classificado em Observador, Construtor, ou de Implementação (Selecione o

Método). Para classificar o método selecionado em Observador/ Construtor/ de Implementação, o engenheiro de software deve clicar no botão com o rótulo Classificar como Observador/ Construtor/ de Implementação indicado em (a), (b) e (c), respectivamente, na Figura 57. A Determinação dos Métodos Candidatos está finalizada quando todos os procedimentos do sistema legado estiverem classificados.

Podem existir relacionamentos ainda não verificados, principalmente aqueles relacionados a classes formadas por itens elementares. O procedimento a ser realizado é o mesmo descrito anteriormente quando da exibição da Figura 46.

5.2.4 Encontrar Relacionamentos Especiais de Associação

Alguns relacionamentos de associação são casos verificados a partir de redefinições de itens de dados, que são considerados candidatos a atributos baseados em heurísticas pré-determinadas pela opção Encontrar Novos Relacionamentos de Associação, no menu principal da ferramenta FAROOL, Figura 58.

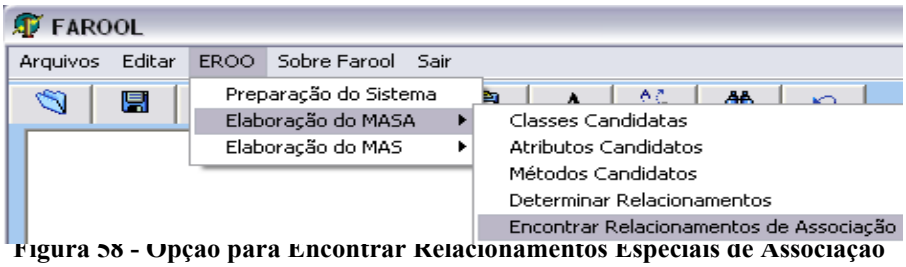


Figura 58 - Opção para Encontrar Relacionamentos Especiais de Associação

A tela Encontrar Relacionamentos de Associação aba Introdução detalha o processo que é realizado, Figura 59.

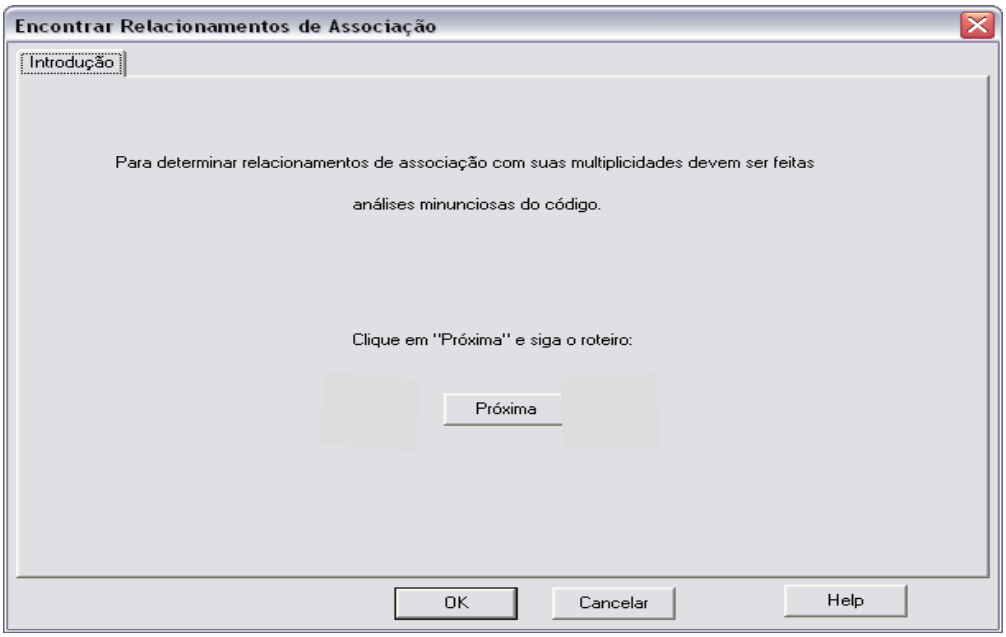


Figura 59 - Tela Encontrar Relacionamentos de Associação aba Introdução

Se a opção Próxima for escolhida, a aba Relacionamento de Associação é ativada tendo-se a tela da Figura 60 que permite a seleção, por parte do engenheiro de software, de um programa, de uma classe e, se existir, da chave correspondente a esse conjunto programa/classe, escolhendo 1-Selecione o Programa, 2-Selecione a Classe e 3-Selecione a Chave, respectivamente. A partir dessa chave são verificadas as possíveis classes do sistema que a referenciam, ou se há algum Item Elementar dessa chave em outras classes se ela for um Item de Grupo. Dessa forma, FAROOL disponibiliza no quadro (a) da Figura 60, o código do programa selecionado em 1-Selecione o Programa. O programa escolhido é visto a partir do clique no botão Visualizar Código do Programa que aparece acima do quadro (a), Figura 60.

Para que outros programas que fazem referência a essa mesma chave (3), Figura 60, sejam examinados, o engenheiro de software deve selecionar o botão Visualizar Código de Outro Programa, que aparece acima do quadro (b) dessa mesma figura. Cabe ao engenheiro de software o exame aos programas para verificar se um possível relacionamento de associação existe. Isso ocorre quando o nome, tipo e tamanho da chave (conteúdo do *combo* 3) coincidem com algum item elementar ou de grupo. Quando houver coincidência, o botão Determinar Associação, Figura 60(c), deve ser ativado e procedido

conforme descrito anteriormente quando da exibição da tela da Figura 46, mas antes disso, o engenheiro de software tem que saber a cardinalidade desse relacionamento.

A título de ilustração serão consideradas duas classes C1 e C2 que mantêm um relacionamento de associação, sendo a classe C1 correspondente à selecionada em 2-Selecione a Classe e a classe C2 do outro programa. Para identificação das cardinalidades deve-se verificar o número de vezes que o campo que representa a chave na classe C2 se repete. Esse número é a cardinalidade máxima do relacionamento da classe C2 em relação à classe C1. Para a identificação da cardinalidade mínima deve-se verificar na *procedure division* do código exibido em (b), se algum procedimento obriga a inserção mínima de dados no campo na classe C2. Caso não haja restrição, a cardinalidade será 0. Dessa forma, a cardinalidade da classe C2 em relação à classe C1 foi detectada. Deve-se agora analisar a classe C1 e fazer o mesmo tipo de análise a fim de identificar as cardinalidades para esse lado do relacionamento. Se a classe C1 não apresentar um campo que represente a chave da classe C2, significa que instâncias da classe C1 podem ser usadas por várias ou nenhuma instâncias da classe C2. Sendo assim, a cardinalidade será 0..n. A cardinalidade dos relacionamentos pode ser omitida ou solucionada posteriormente sempre que o engenheiro de software sentir dificuldade em determiná-la, seja por falhas na implementação do sistema legado ou devido à necessidade de suposição por parte do engenheiro de software.

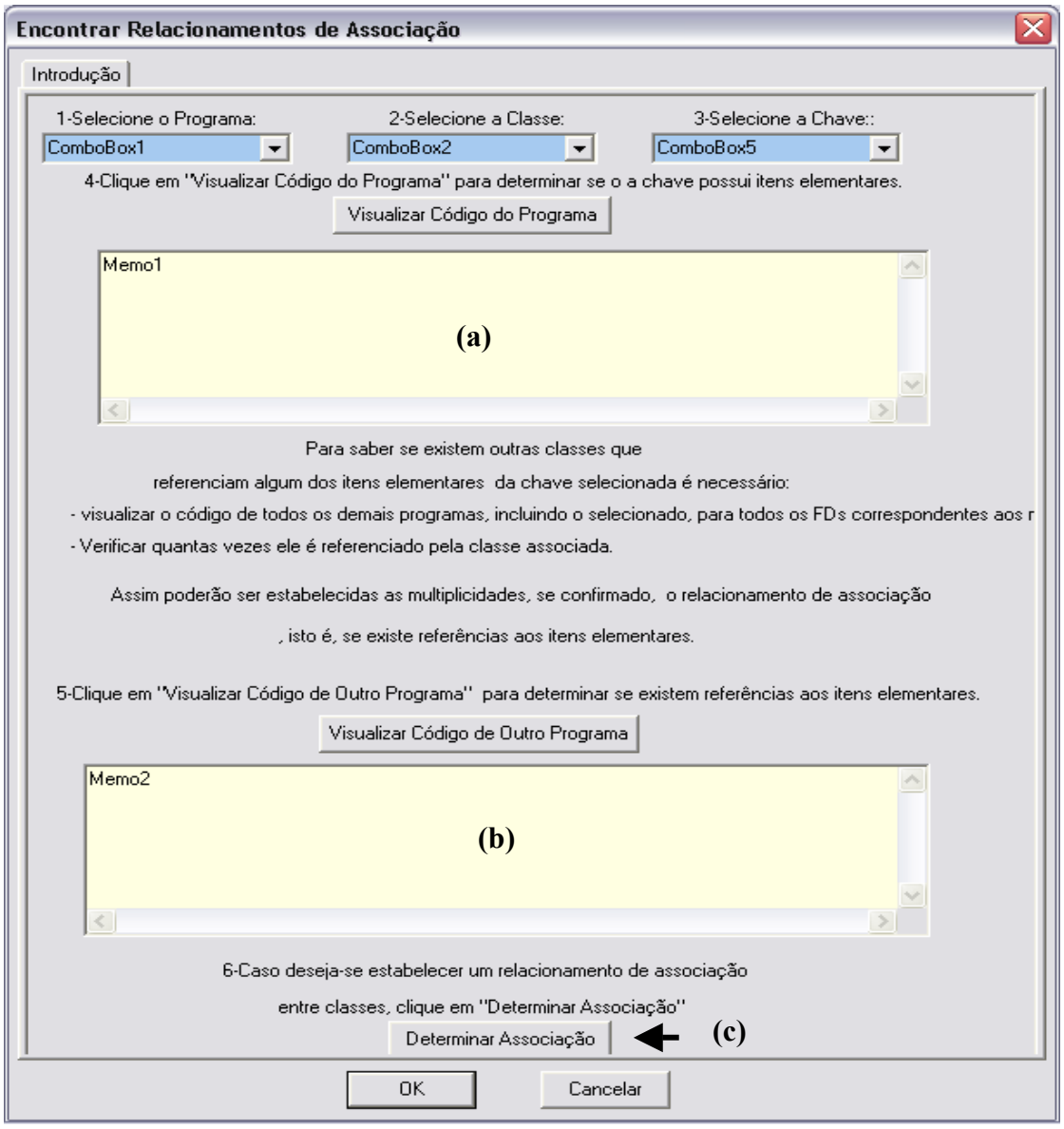


Figura 60 - Tela Encontrar Relacionamentos de Associação aba Relacionamentos de Associação

5.3. Considerações Finais

Neste capítulo foi apresentada a fase de Elaboração do MAS com a determinação das classes, dos atributos e métodos candidatos, dos relacionamentos entre as classes inseridas no banco de dados da ferramenta FAROOL e dos casos especiais de relacionamentos de associação a partir de redefinições no código do sistema.

Na Determinação das Classes são considerados os principais FDs do código do sistema legado para cada programa como classes candidatas. De acordo com o tipo de

procedimento associado a um determinado FD, há classificação em observadores (aqueles cujos itens de dados não são alterados pelos procedimentos), construtores (aqueles cujos itens de dados sofrem alterações por meio dos procedimentos) e de implementação (aqueles que têm características específicas de acordo com a linguagem de programação e ambiente utilizados). Somente os FDs cujos itens de dados são modificados pelos procedimentos são considerados classes candidatas válidas no processo de engenharia reversa orientada a objetos.

Na Determinação dos Atributos Candidatos são considerados os itens de dados de cada classe candidata válida determinada anteriormente. Se um item de dado for um item de grupo, o mesmo é candidato a uma nova classe e seus itens elementares são candidatos a atributos dessa nova classe.

Na Determinação dos Métodos Candidatos são considerados os procedimentos que compõem cada programa do sistema legado. Esses métodos podem ser classificados como sendo construtores, observadores, ou de implementação.

Na Determinação dos Relacionamentos são considerados os relacionamentos de dependência, associação, realize/interface e generalização, bem como os papéis (roles) da classe Filha e da classe Pai. Para os relacionamentos de associação são definidos ainda a cardinalidade e a navegabilidade.

Existem alguns Casos Especiais de Associação que são definidos a partir de referências entre as chaves dos FDs. Dependendo de quantas vezes a chave é referenciada por uma classe em relação à outra classe, é definida a cardinalidade da associação. Análises minuciosas no código são necessárias, dependendo do conhecimento do engenheiro de software sobre o sistema legado.

As informações obtidas ao final da fase de Elaboração do MASA são armazenadas no banco de dados da FAROOL, baseado no metamodelo definido especificamente para essa ferramenta.

No próximo capítulo será tratada a fase de Elaboração do MAS.

Capítulo 6

FAROOL: Fase de Elaboração do MAS

6.1 Considerações Iniciais

Após a fase de Elaboração do MASA, em que um primeiro modelo pseudo orientado a objeto é constituído, o engenheiro de software deve prosseguir com processo de engenharia reversa orientada a objetos, refinando mais o sistema legado em busca de classes que podem ser eliminadas, nomes de classes e de atributos que podem ser alterados para mnemônicos mais significativos, para que um modelo puramente orientado a objeto seja obtido.

Essa fase preocupa-se em refinar o modelo obtido na fase anterior. A seção 6.2 apresenta as atividades que devem ser realizadas para que isso ocorra e na seção 6.3 são abordados os pontos observados durante essa fase.

6.2 Fase de Elaboração do MAS

Para o engenheiro de software iniciar a Elaboração do MAS, ele deve primeiramente identificar e remover as classes que representem datas. Programadores da linguagem *COBOL* utilizam itens de grupo para representar datas, com três itens elementares: dia, mês e ano. Portanto, é comum a existência de FDs (classes candidatas) que representam datas em sistemas legados procedimentais. O ideal é que essas classes sejam removidas do código para melhor organização do mesmo. A ferramenta FAROOL, baseada em heurísticas pré-definidas, disponibiliza recursos para que a remoção dessas classes seja feita mais eficientemente.

6.2.1 Remoção das Classes que Representam DATA

A Elaboração do MAS, bem como a remoção das classes que representam datas, inicia-se com a escolha da opção EROO -> Elaboração do MAS -> Classes -> Remover as Classes que Representam DATA, como apresentado na Figura 61.

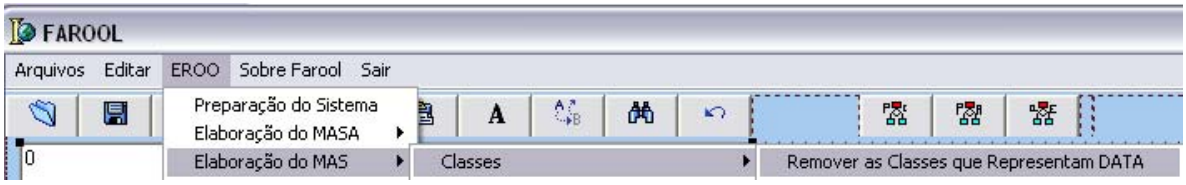


Figura 61 - Opção para remoção das classes que representam DATA

A partir da escolha da opção acima citada, a tela da Figura 62 é exibida.

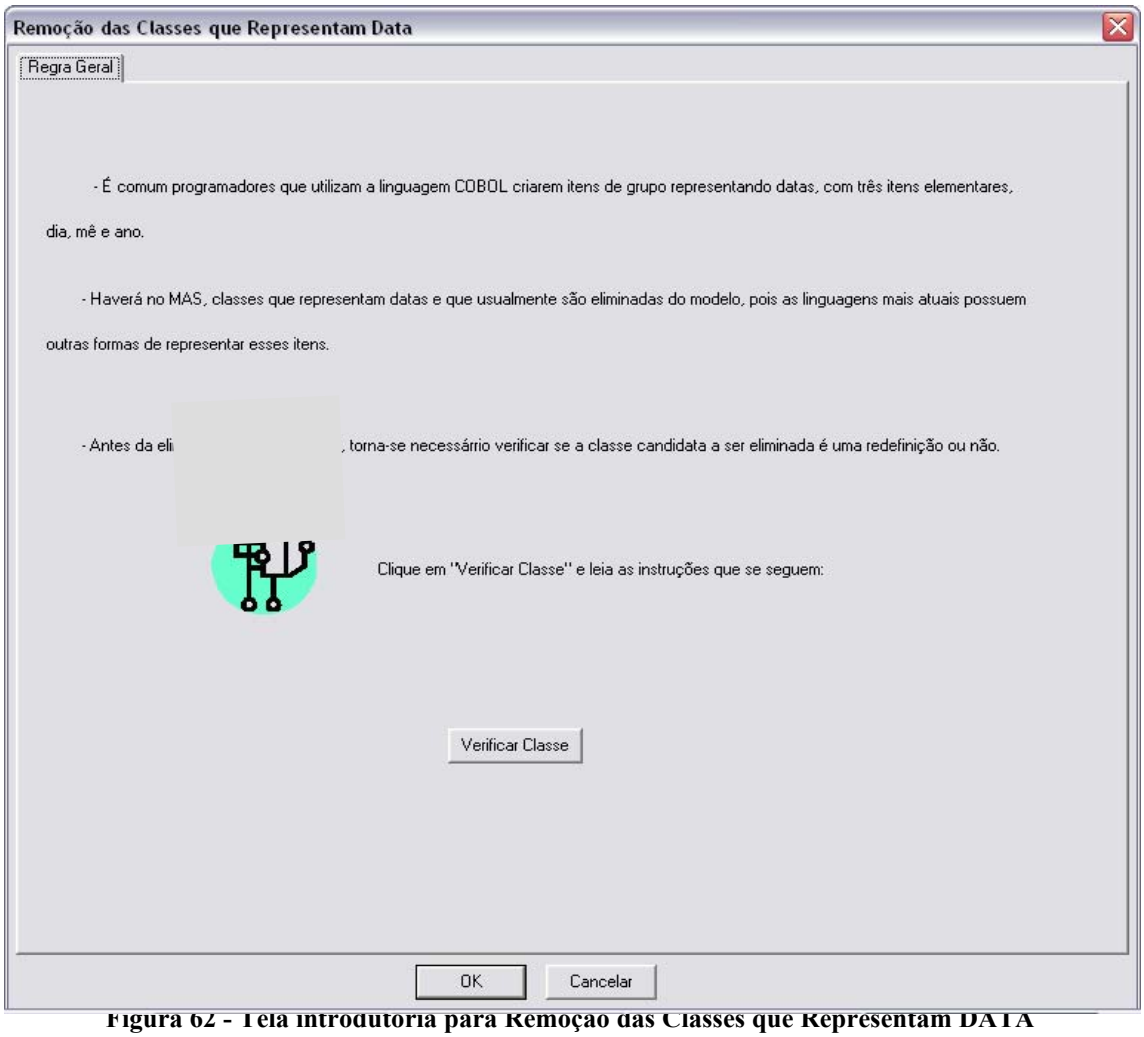
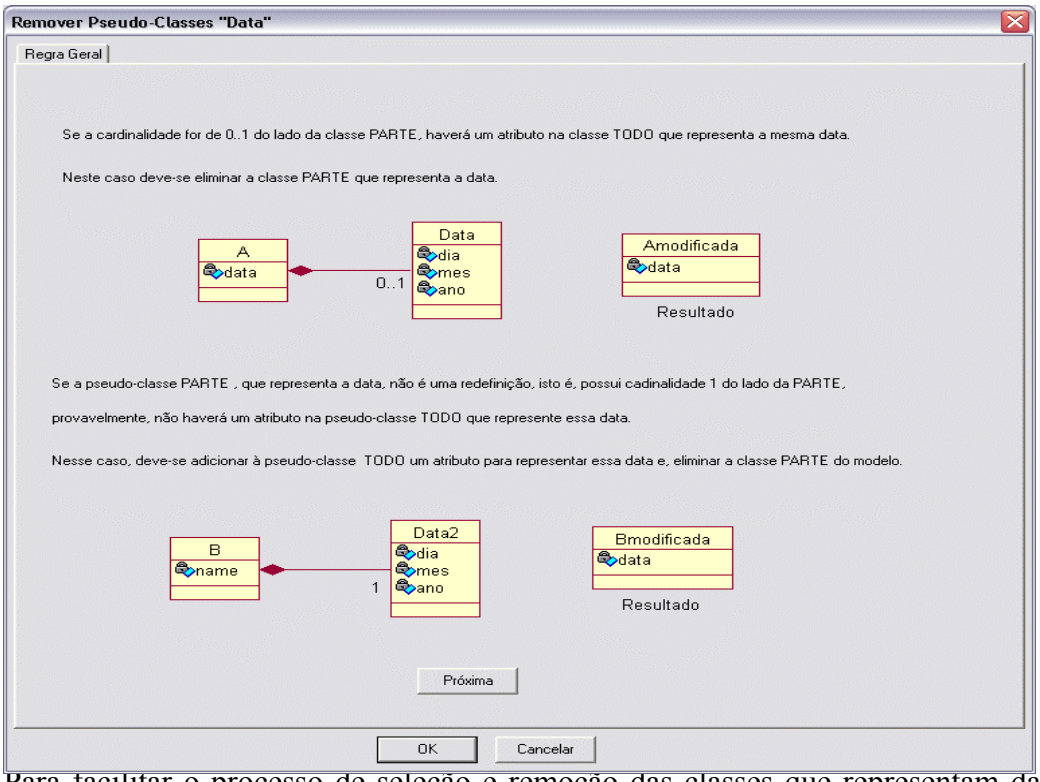


Figura 62 - Tela introdutória para Remoção das Classes que Representam DATA

Essa tela é introdutória e orienta o engenheiro de software no processo de remoção. Para que uma classe seja removida do código, algumas análises são levadas em consideração. Para verificar se uma determinada classe deve ser removida, o engenheiro de software deve escolher a opção Verificar Classe na aba Regra Geral da tela da Figura 62.

Em seguida, a tela Verificação das Classes, Figura 63, é exibida. Ela contém um guia que alerta o engenheiro de software para pontos que devem ser considerados para a remoção de classes que representam data. Selecionando-se a opção Próxima a tela da Figura 64 é exibida.



Para facilitar o processo de seleção e remoção das classes que representam datas, a FAROOL oferece diversos recursos para identificar, selecionar e, se necessário, remover essas classes. Considerando a Figura 64, o processo de seleção e remoção dessas classes inicia-se com a escolha da opção Visualizar Casos Candidatos à Remoção, com a qual é feita a seleção das classes que possuem relacionamentos de associação, cujas cardinalidades são 0..1 ou 1 nas classes parte(s). Essas classes são visualizadas no quadro (a) da Figura 64. Enquanto existirem classes candidatas à remoção, o engenheiro de software deve continuar o processo de remoção. Caso contrário, o mesmo será notificado com o aviso exibido na Figura 65.

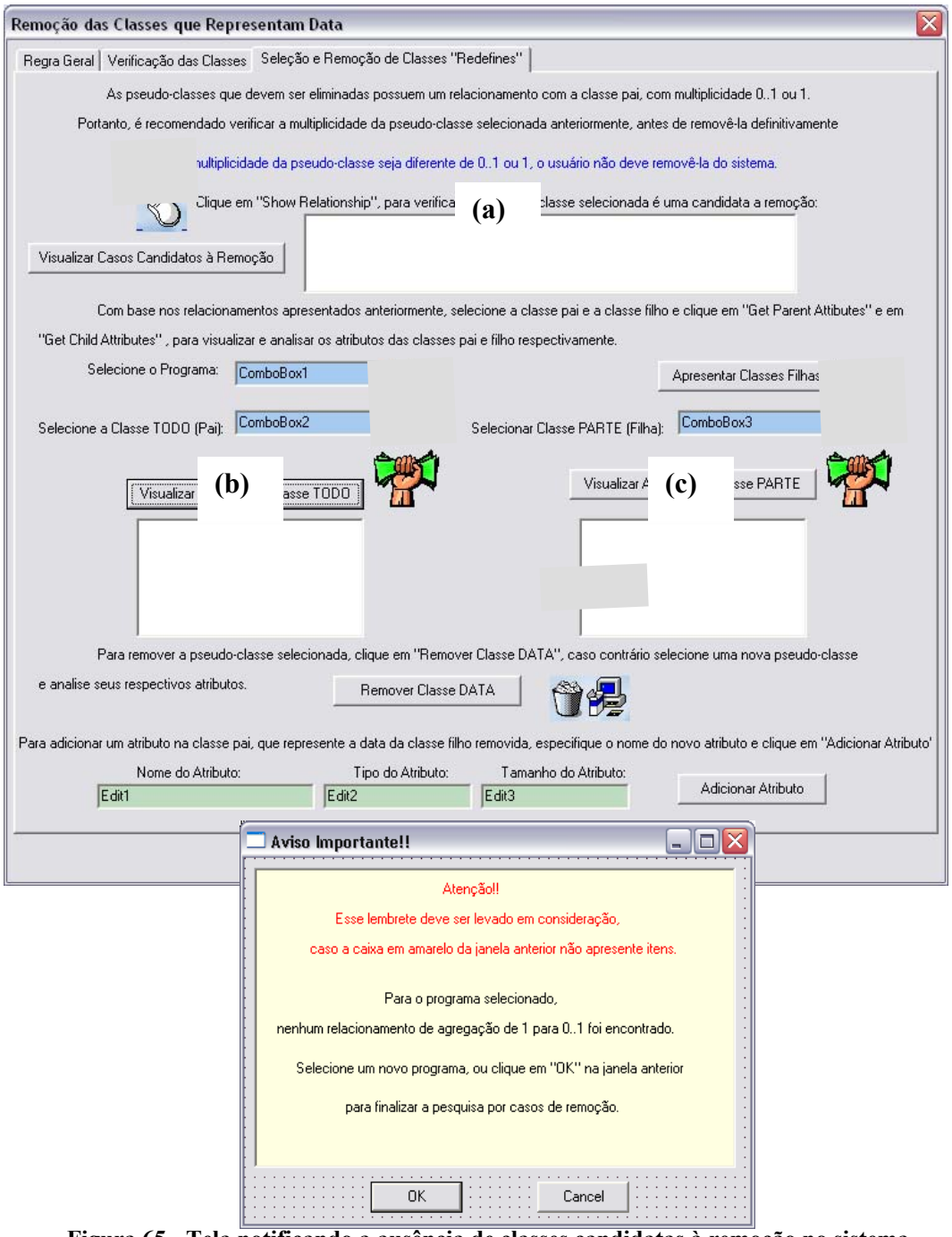


Figura 65 - Tela notificando a ausência de classes candidatas à remoção no sistema

O processo de remoção de classes, após a exibição das mesmas no quadro da Figura 64(a), prossegue com a seleção do programa, através da opção Selecione o Programa; da classe pai (todo), opção Selecione a Classe Pai (TODO), e com a visualização dos seus atributos ao escolher a opção Visualizar os Atributos da Classe Pai, quadro (b) da Figura 64.

Para saber quais são as classes que mantêm relacionamentos com a classe Pai selecionada, o engenheiro de software deve escolher a opção Apresentar Classes Filhas. As classes Filhas (PARTEs), que mantêm relacionamentos com a classe Pai selecionada, são consultadas no banco FAROOL e disponibilizadas na tela após a escolha da opção Selecione Classe PARTE (Filhas) pelo engenheiro de software . Os atributos de cada classe parte são vistos após a escolha da opção Visualizar os Atributos da Classe Filha, quadro (c) da Figura 64.

O engenheiro de software deve comparar os atributos da classe Pai com os da classe Filha, Figura 64 (b) e (c), respectivamente. Caso existam atributos que representam data em (b) e em (c) a classe filha deve ser removida, escolhendo-se a opção Remover Classe DATA. Caso não existam atributos que representam data em (b), mas existam em (c), o engenheiro de software deve remover a classe filha, escolhendo a opção Remover Classe DATA, e adicionar um atributo que represente essa data na classe pai. Isto é, realizado com o preenchimento dos campos Nome do Atributo, Tipo do Atributo e Tamanho do Atributo, e clicando em Adicionar Atributo.

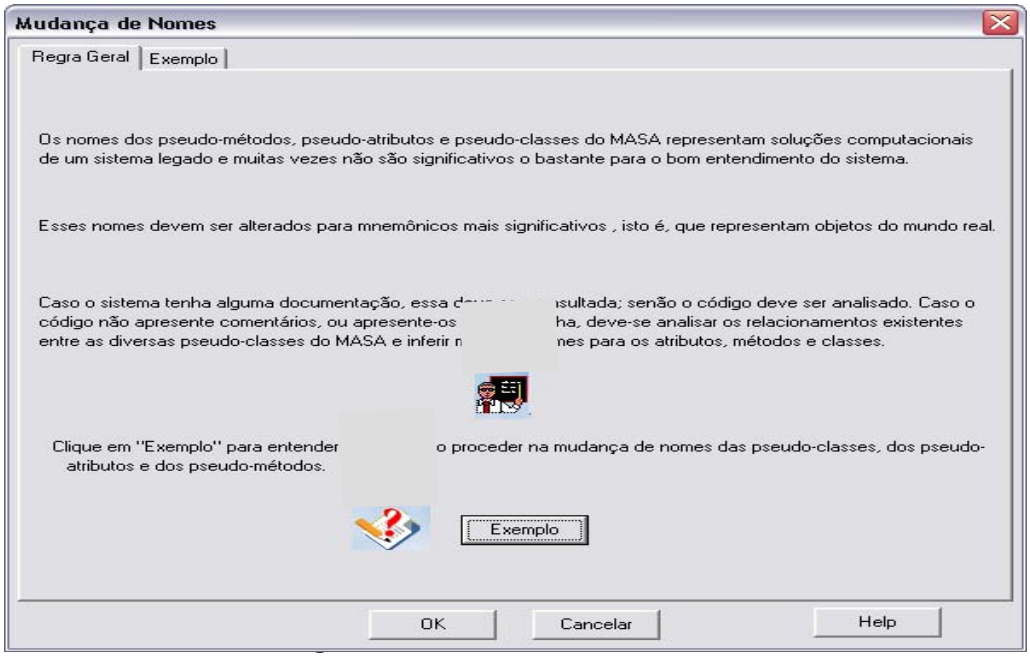
Para dar continuidade ao processo de Elaboração do MAS, os nomes das classes, dos atributos e dos métodos podem ser mudados para nomes mais significativos de acordo com a funcionalidade do sistema, como forma de documentação. O código legado e a possível documentação existente devem ser consultados para essa atividade. FAROOL auxilia a mudança de nomes apresentando um conjunto de janelas interativas, oferecendo dicas e sugestões para mudança dos nomes, e armazena essas informações em seu banco de dados.

6.2.2 Mudança dos Nomes das Classes

Mudar Nomes das Classes deve ser a opção escolhida no menu principal da FAROOL, como exemplifica a Figura 66.

Figura 66 - Opção para Mudar os Nomes das Classes

A tela Mudança de Nomes apresenta ao engenheiro de software breve introdução da necessidade de se trabalhar com nomes mais significativos no sistema legado, como mostra a Figura 67.



Quando a opção Exemplo for escolhida, a tela da Figura 68 é exibida. Nessa tela são apresentados alguns exemplos ilustrando como o engenheiro de software deve proceder para que as mudanças de nomes por mnemônicos mais significativos sejam bem sucedidas. Em seguida, a opção Mudar Nomes das Classes pode ser escolhida para que primeiramente o engenheiro de software escolha o programa que terá seu código analisado, através da opção Abrir Código do Programa, Figura 70. A tela da Figura 70 é apresentada para a escolha do arquivo a ser examinado, sendo esse então apresentado no quadro existente na Figura 69.

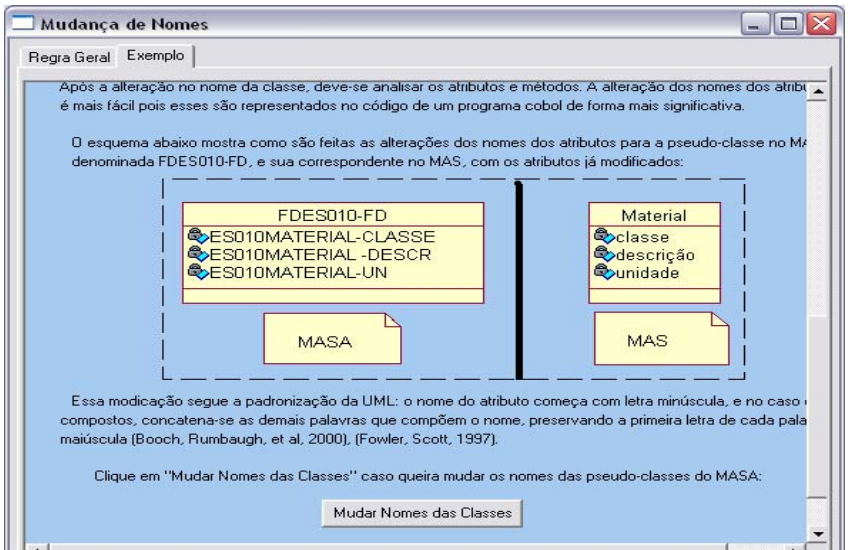


Figura 68 - Tela referente à aba Exemplos que ilustra o processo de mudança de nomes das classes

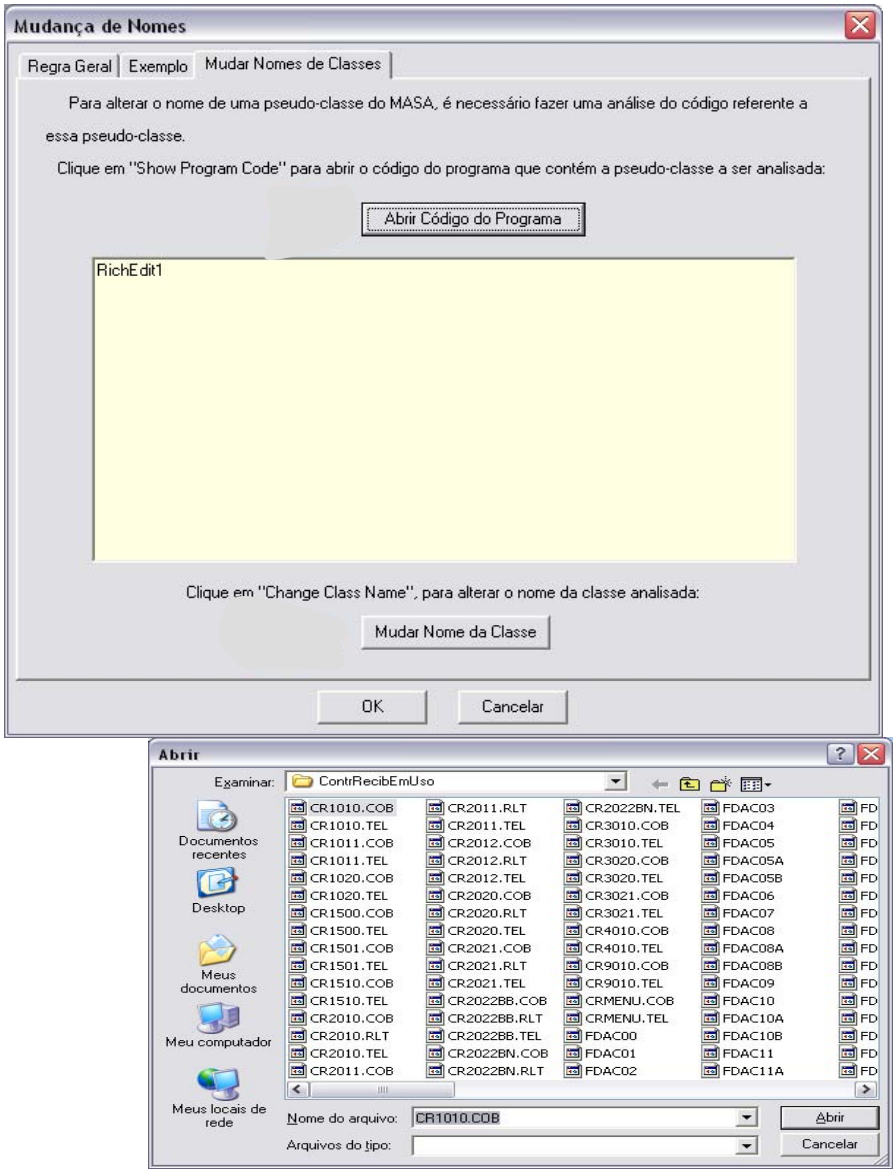
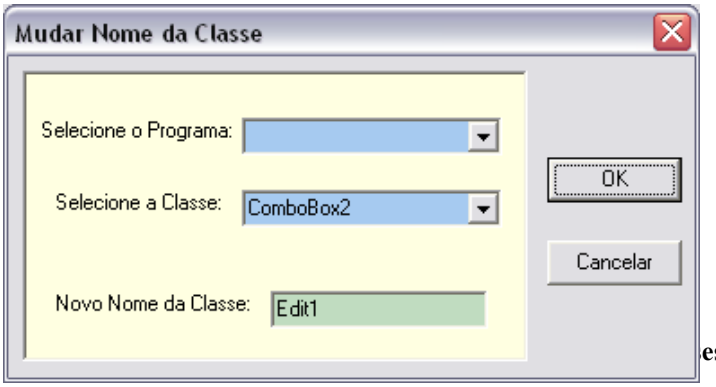


Figura 70 - Tela para seleção do programa que contém a classe candidata à mudança de nome

Ao escolher a opção Mudar Nome da Classe, da Figura 69, a tela da Figura 71 é exibida. O engenheiro de software deve selecionar o programa pela opção Selecione o Programa e a classe por Selecione a Classe, deve digitar um novo nome para a classe selecionada no campo Novo Nome da Classe e clicar em OK. O processo de Mudança do Nome da Classe, referente ao programa selecionado é concluído. Esse processo deve ser realizado para todas as classes do sistema que precisam ter seus nomes alterados.



es

6.2.3 Mudança dos Nomes dos Atributos

Para alteração dos nomes dos atributos o procedimento é análogo ao descrito para Mudança dos Nomes das Classes, somente escolhendo-se a opção Mudar Nomes dos Atributos, no menu principal da FAROOL como mostra a Figura 72.

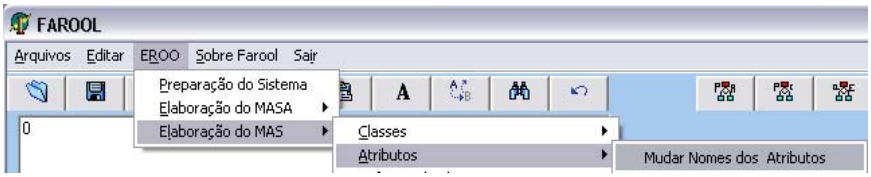


Figura 72 - Opção para Mudar os Nomes dos Atributos

Após a seleção dessa opção, uma tela com as regras gerais para a mudança de nome são apresentadas de modo semelhante ao apresentado na Figura 67. Se a opção Exemplo for escolhida a tela da Figura 73 é apresentada.

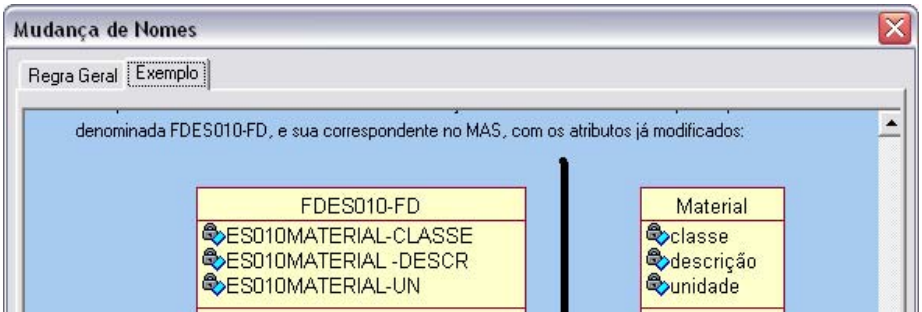


Figura 73 - Tela que ilustra o processo de mudança de nomes dos Atributos por meio de um exemplo

Escolhendo-se a opção Mudar Nomes dos Atributos, Figura 73, é exibida a tela da Figura 74. Como ocorre no caso das classes, apresentado na seção anterior, há necessidade da análise do código que é feita optando-se por Abrir Código do Programa, Figura 74. Nesse momento, a tela apresentada na Figura 70 é apresentada e após a escolha o código é exibido no quadro (a) da Figura 74.

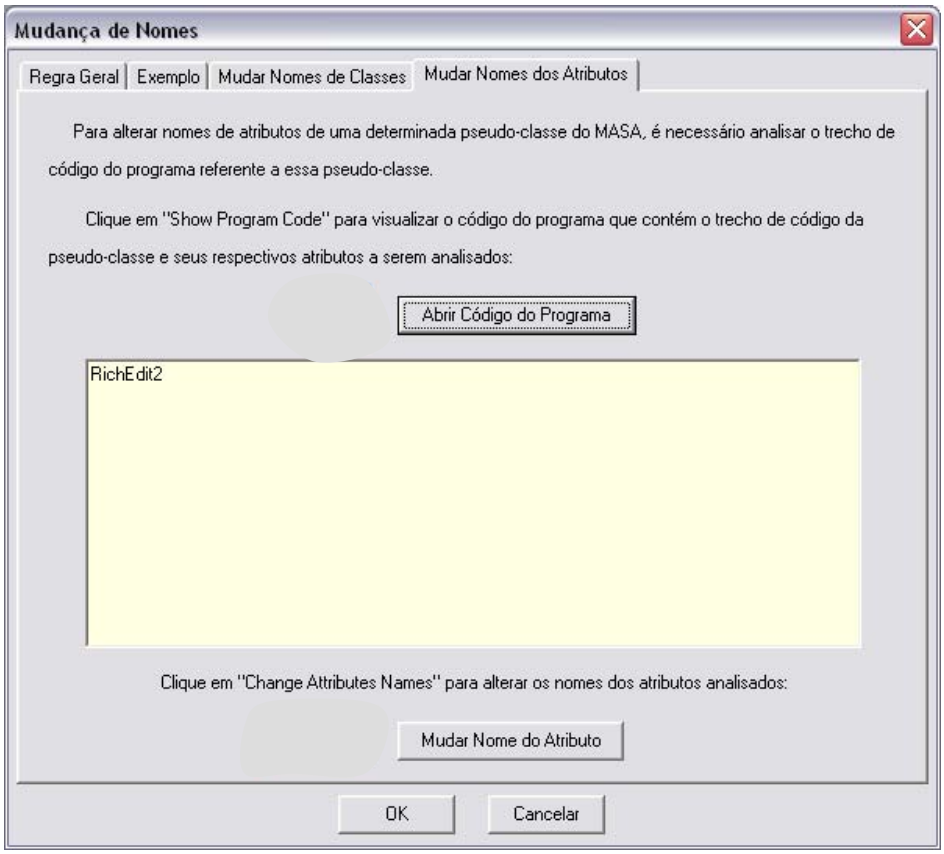


Figura 74 - Tela que apresenta a aba Mudar Nomes dos Atributos

Se a alteração de nome de um atributo for necessária a opção Mudar Nome do Atributo deve ser escolhida, sendo então exibida a tela da Figura 75. Nessa tela, o engenheiro de software deve selecionar o programa e a classe que contém o atributo candidato à substituição do nome, através de Selecione o Programa e Selecione a Classe, respectivamente. Clicando-se em Visualizar Atributos são disponibilizados os atributos relacionados à classe escolhida para que o engenheiro de software não substitua o nome de um outro atributo, caso sejam selecionados o programa errado e/ou a classe errada, é exibido em (a), os atributos referentes ao programa e sua respectiva classe. O atributo desejado deve ser relacionado pela opção Selecione o Atributo e o novo nome do atributo é fornecido no campo Novo Nome do Atributo. Para confirmar a mudança do nome, o botão Efetuar Mudança é ativado. Esse procedimento deve ser realizado para todos os atributos do sistema que precisam ter seus nomes substituídos por mnemônicos mais significativos.

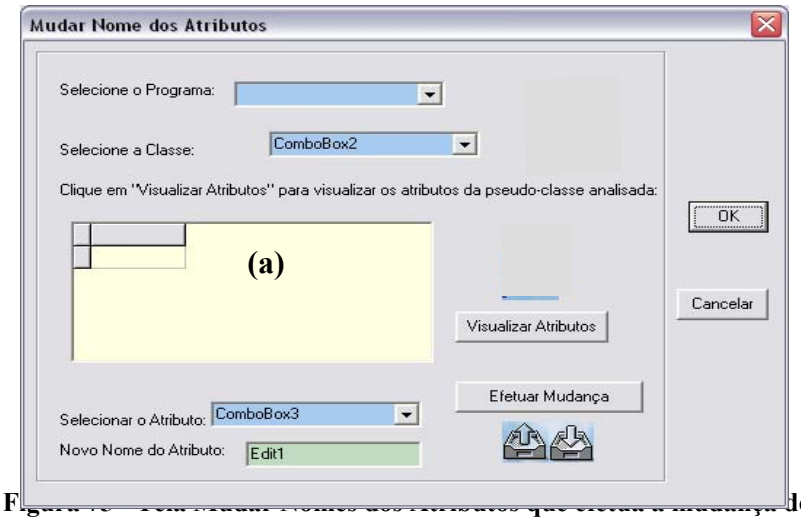


Figura 75 - Tela Mudar Nomes dos Atributos que efetua a mudança do nome

6.2.4 Refinando os Métodos

O refinamento de métodos para escolha de nomes mais significativos, inicia-se com a escolha das opções EROO-> Elaboração do MAS -> Refinar Métodos, no menu principal da FAROOL, Figura 76.

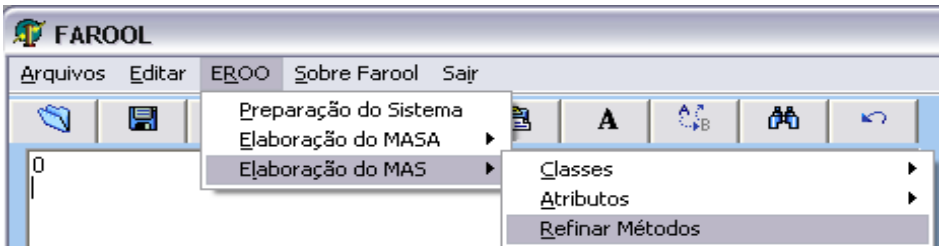


Figura 76 - Opção Refinar Métodos que efetua a mudança dos nomes dos métodos

A tela da Figura 77 é exibida contendo breve introdução ao processo de refinamento dos métodos. Ao ativar o botão Próxima nessa figura, a aba Refinamento é apresentada, Figura 78, permitindo a seleção de um programa, de uma classe e de um método nos *combos* Selecione o Programa, Selecione a Classe e Selecione o Método, respectivamente. Com base em heurísticas pré-determinadas, a mudança de nomes dos métodos é significativa em se tratando de métodos construtores e observadores. Portanto, FAROOL oferece a classificação do método selecionado, quadro (a) da Figura 78, quando o botão Verificar Classificação do Método for ativado. Uma vez verificada a classificação do método como sendo construtor ou observador, o engenheiro de software deve visualizar o código do programa que contém o método, selecionando o botão Visualizar Código do Programa, Figura 78, que será exibido no quadro (b) da Figura 78.

Para alterar o nome do método, o engenheiro de software deve preencher o campo Novo nome do método e clicar em Alterar Nome do Método. Esse procedimento descrito deve ser realizado para todos os métodos dos programas e classes do sistema.

Para finalizar a Elaboração do MAS, e conseqüentemente, o processo de engenharia reversa, a FAROOL oferece recursos para a Determinação de Novos Relacionamentos entre as classes que podem surgir devido à necessidade de que se tenha um modelo completamente orientado a objeto.

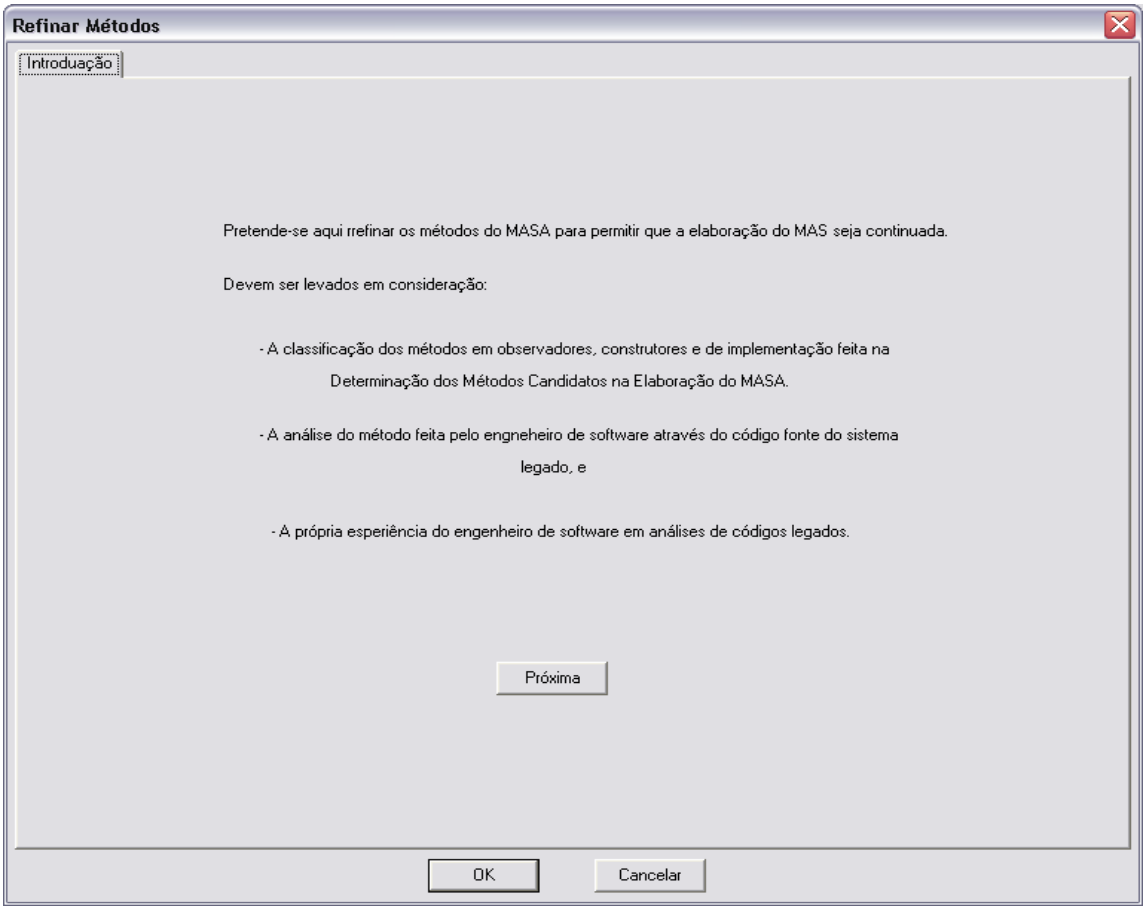


Figura 77 - Tela introdutória ao procedimento Refinar Métodos

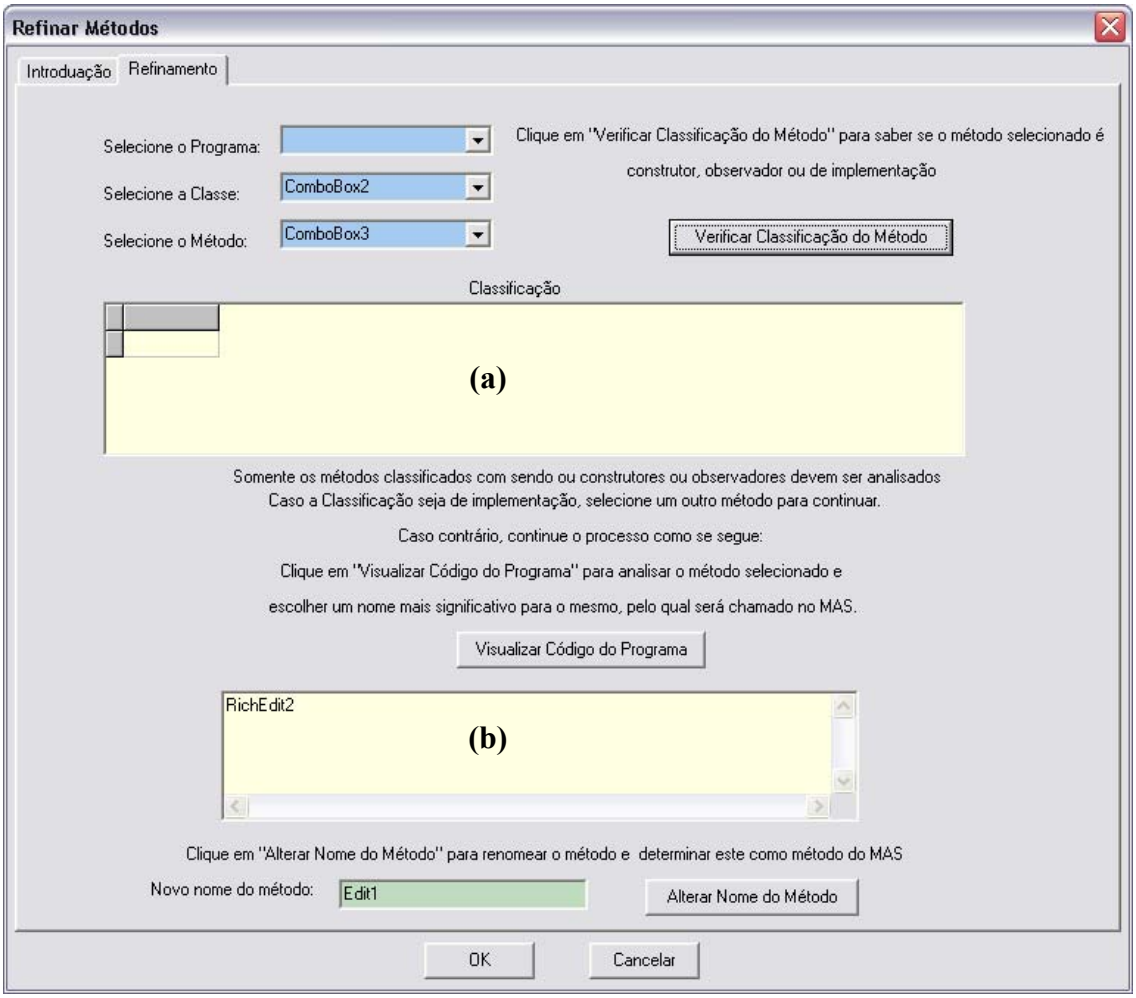


Figura 78 - Tela para refinamento dos métodos

6.2.5 Determinação de Novos Relacionamentos

A Determinação de Novos Relacionamentos inicia-se com a escolha das opções EROO -> Elaboração do MAS -> Determinar Novos Relacionamentos, no menu principal da FAROOL, como mostra a Figura 79.

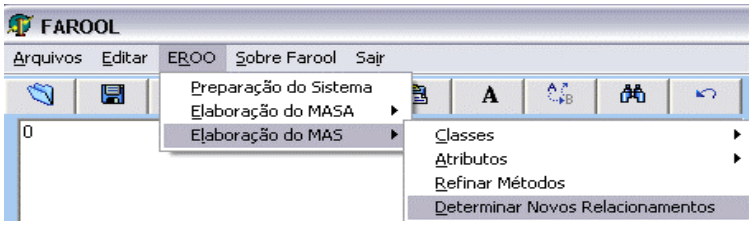
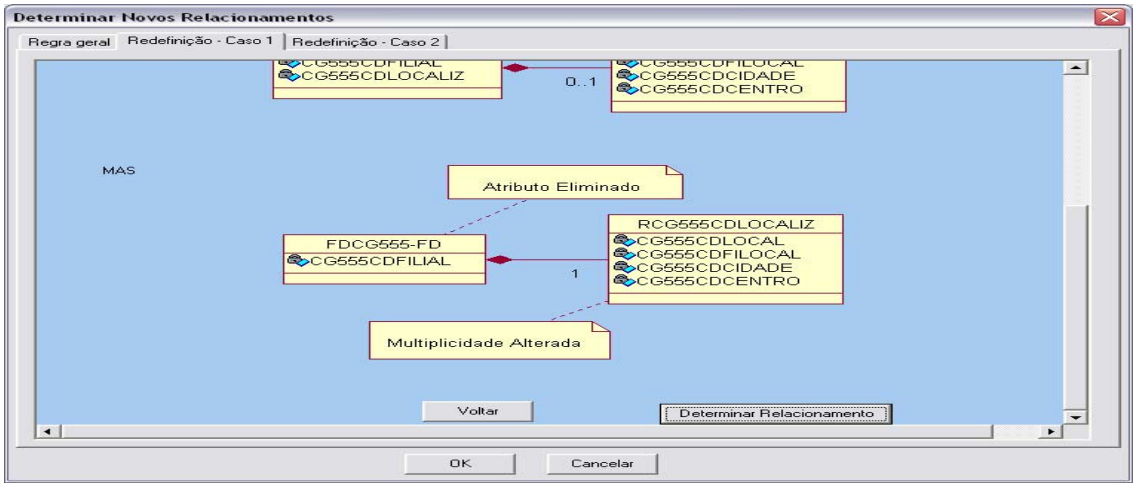


Figura 79 - Opção para Determinar Novos Relacionamentos

Após a opção ter sido escolhida a tela da Figura 80 é exibida contendo explicações (Regra Geral) de como se deve proceder para que novos relacionamentos, tanto de agregação quanto de generalização sejam determinados.

Em particular, para o caso de determinação de um novo relacionamento de agregação, o engenheiro de software deve escolher a opção Redefinição Caso 1, quando é exibido um guia, exemplificando esse caso como mostra a tela da Figura 81.



Um dos objetivos principais nesse procedimento é a análise dos relacionamentos de agregação de classes do MASA, cuja cardinalidade em relação à classe parte é 0..1. Esses relacionamentos merecem especial atenção, pois as suas classes parte(s) foram geradas por

redefinições de outros itens, isto é, com o uso do comando *redefines*. Existem dois casos principais de redefinição: redefinição de um item elementar em um item de grupo e redefinição de um item de grupo em outro.

Para o caso de redefinição de um item elementar em um item de grupo, a classe (antigo FD), que tem esse tipo de redefinição, gera no MASA um relacionamento de agregação em que a classe parte tem a mesma função de um atributo da classe todo. Para exemplificar esse caso de redefinição, considere o exemplo apresentado na Figura 82.

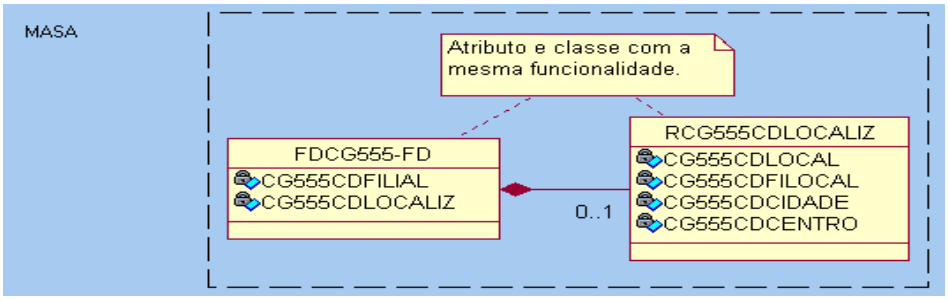


Figura 82 - Exemplo de relacionamento com caso de redefinição de um item elementar em um item de grupo no MASA

A classe RCG555CDLOCALIZ tem a mesma funcionalidade do atributo CG555CDLOCALIZ da classe FDCG555-FD. E as classes FDCG555-FD e RCG555CDLOCALIZ mantêm um relacionamento de agregação, cuja cardinalidade em relação à classe parte é de 0..1, sendo portanto o caso de redefinição de um item elementar em um item de grupo. No MAS, o engenheiro de software deve eliminar o atributo CG555CDLOCALIZ da classe todo, permanecendo com a classe parte, e alterar a cardinalidade da classe todo em relação à classe parte, que antes era de 0..1, para 1, Figura 83.

Para o caso de redefinição de um item de grupo em outro, na linguagem *COBOL*, é observado um comportamento semelhante à generalização da linguagem orientada a objetos.

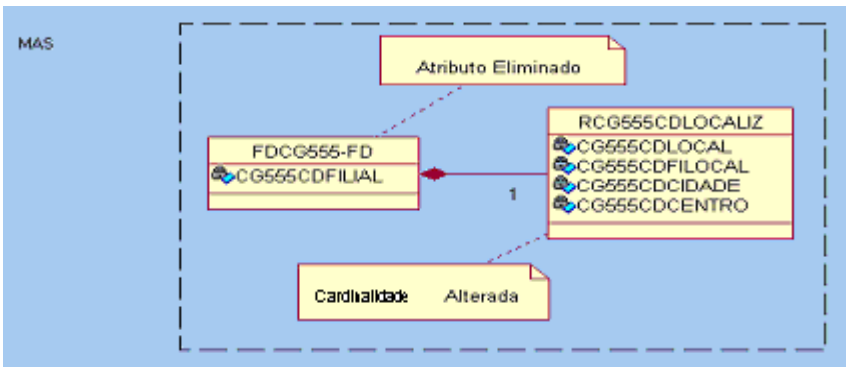
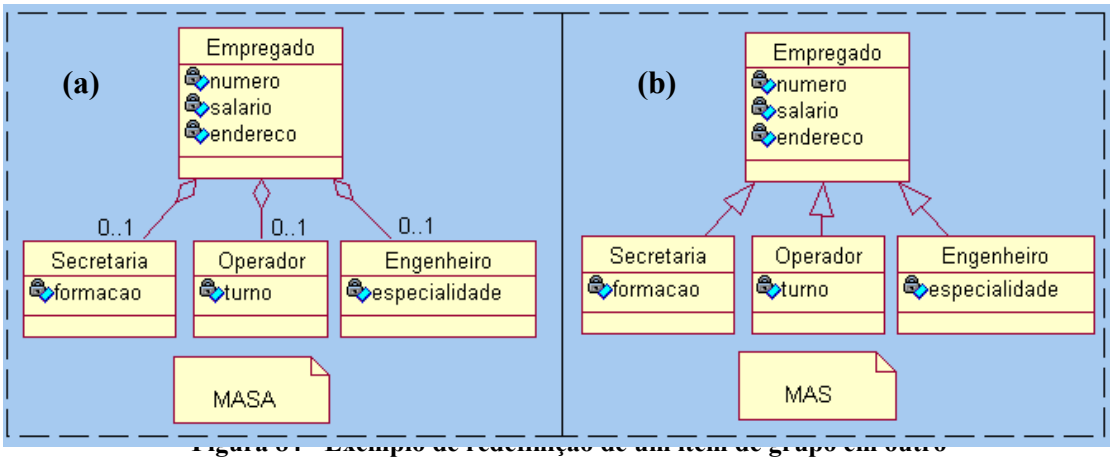


Figura 83 - Diagrama de classe do MAS após a remoção de atributo

Quando o engenheiro de software identificar no MASA relacionamentos de agregação com cardinalidade 0..1, cuja classe todo não possui um atributo com a mesma funcionalidade da classe parte, está caracterizado um caso redefinição de um item de grupo em outro, exigindo uma análise detalhada do código. No MAS, esses relacionamentos de agregação são substituídos por relacionamentos de generalização/especialização e as cardinalidades eliminadas. Um exemplo ilustrativo para esse processo é apresentado na Figura 84.



A classe Empregado, no MASA, possui agregações com as classes Secretária, Operador e Engenheiro, cujas cardinalidades desses relacionamentos são de 0..1 em relação à classe parte, Figura 84 (a). O engenheiro de software deve observar que não existem atributos na classe todo que possuem a mesma funcionalidade das classes parte(s). Essa verificação caracteriza, portanto, um caso de redefinição de um item de grupo em outro, isto é, os atributos definidos na classe todo são herdados pelas classes parte(s). As classes parte(s) se comportam como especializações da classe todo, e por isso, possuem atributos a mais que os definidos na mesma. Para representar essa herança dos atributos da classe todo e a adição de novos atributos especializados na classe parte, dentro dos programas implementados em COBOL, é utilizado o comando *Redefine*. No processo de

Elaboração do MAS, o engenheiro de software deve substituir o relacionamento de agregação pelo de generalização e eliminar as cardinalidades, Figura 84(b).

Retornando a tela da Figura 81 quando a opção Determinar Relacionamento é escolhida para que se realize o procedimento acima discutido, a tela da Figura 85 é exibida. O engenheiro de software deve fazer a seleção do programa que deseja examinar, em Selecione o Programa e visualizar se existem casos de redefinição, candidatos à agregação, relacionamentos com cardinalidade de 0..1 em relação classe parte, selecionando Determinação dos Relacionamentos Candidatos a Alterações, quadro da Figura 85 (referenciado por (a)). Se não existirem casos candidatos à agregação nesse quadro, o engenheiro de software será notificado com a tela apresentada na Figura 86. Se existirem casos candidatos à agregação, o engenheiro de software deve continuar, selecionando a classe todo de um dos relacionamentos candidatos à agregação no *combo* denominado 3-Selecione a Classe TODO (Pai).

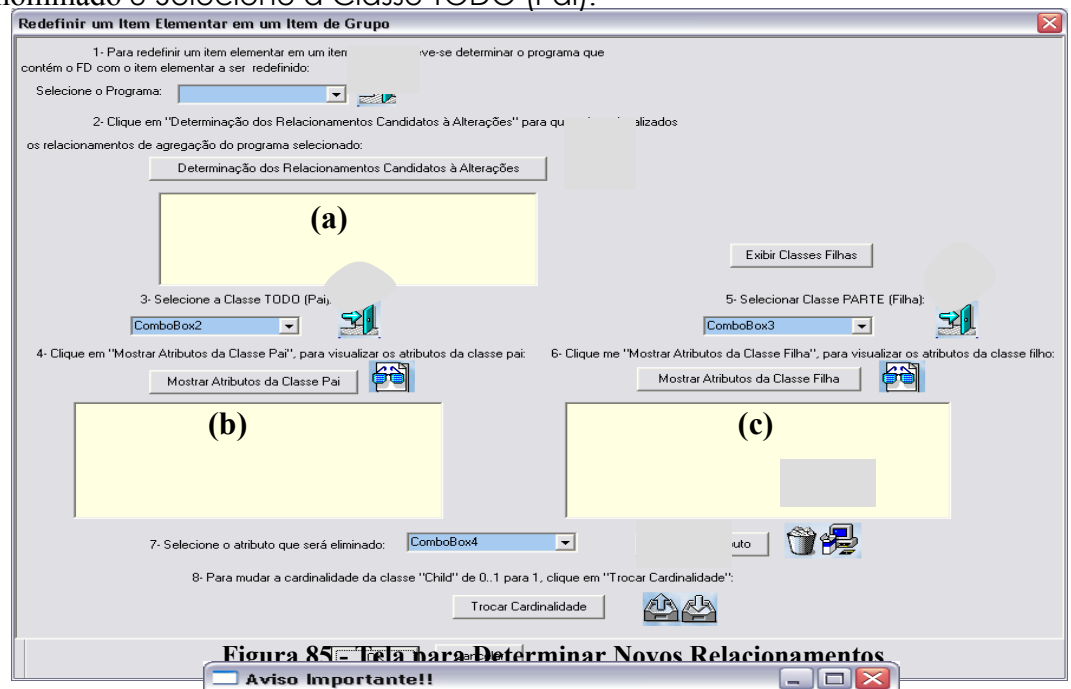


Figura 85 - Tela para Determinar Novos Relacionamentos

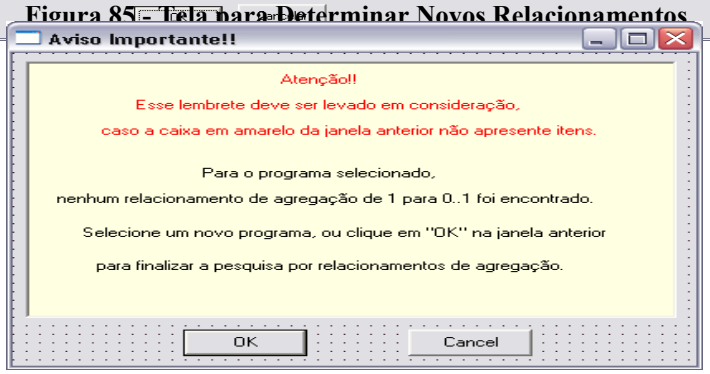


Figura 86- Tela notificando a ausência de candidatos à agregação

Em seguida, quando se escolhe a opção Mostrar Atributos da Classe Pai, os atributos, exibidos na Figura 85 (b), devem comparados com os atributos da classe parte do relacionamento. As classes parte(s) que mantém relacionamentos de agregação com a classe todo, candidatos à modificação, são obtidas a partir da escolha da opção Exibir Classes Filhas. O banco será consultado e as classes parte(s) candidatas disponibilizadas poderão ser selecionadas em Selecione a Classe PARTE (Filha).

Para visualizar os atributos da classe parte, deve ser escolhida a opção Mostrar Atributos da Classe Filha. Os atributos, exibidos em (c), são comparados com os atributos da classe todo, exibidos em (b) da Figura 85 para verificar se a funcionalidade da classe parte é igual a de um dos atributos da classe todo. Confirmada essa redundância o engenheiro de software deve selecionar o atributo a ser eliminado no *combo* rotulado 7-Selecione o atributo que será eliminado e clicar em Remover Atributo.

Para alterar a cardinalidade do relacionamento entre as classes todo e parte, em relação a essa última, é necessário escolher a opção Trocar Cardinalidade. Finalizando assim, a determinação de um novo relacionamento de associação com cardinalidade de 1 para 1 entre as classes e formalizando o caso de redefinição de um item elementar em um item de grupo.

Para o caso de determinação de novos relacionamentos de generalização, o engenheiro de software deve escolher a opção Redefinição Caso 2, disponível na tela da Figura 79. Nesse caso, é exibido o guia exemplificando o caso de redefinição de um item de grupo como mostra a Figura 87. Ao escolher a opção Determinar Relacionamento a tela da Figura 88 é apresentada ao engenheiro de software.

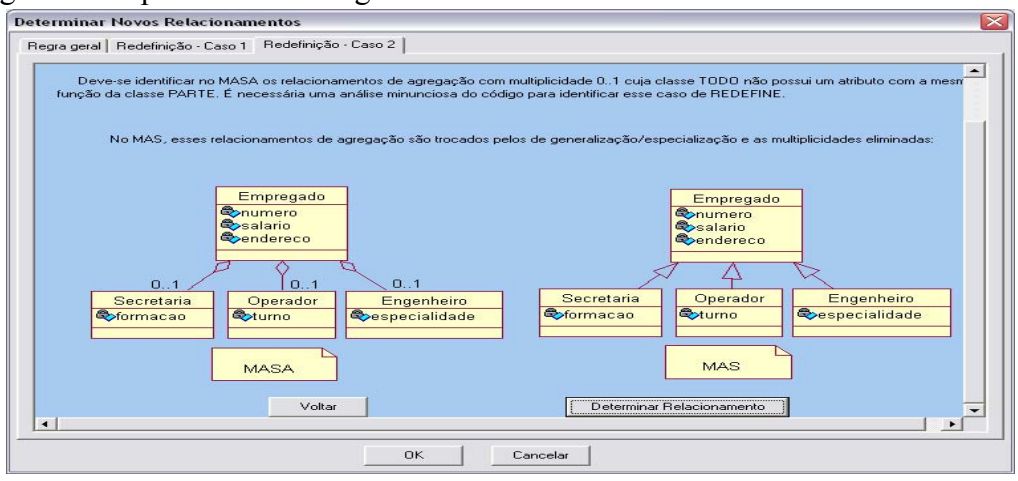


Figura 87 - Tela Redefinição Caso 2 para relacionamentos de generalização

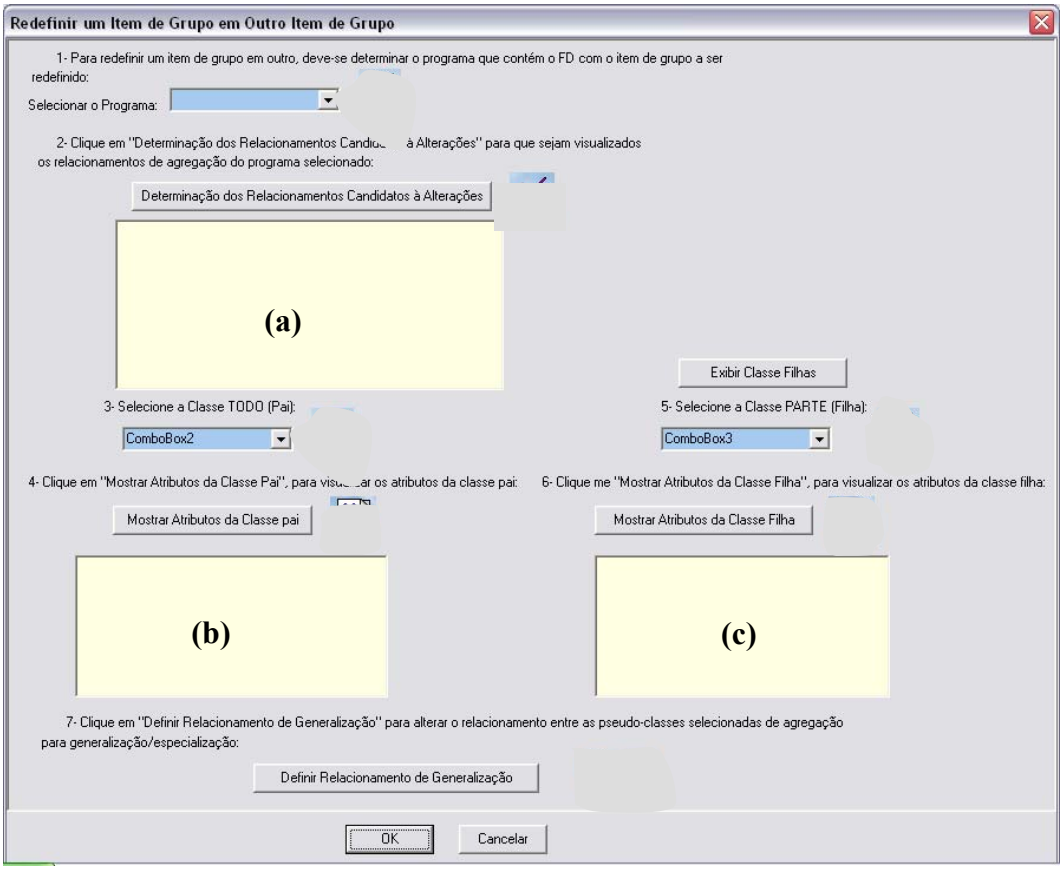


Figura 88 - Tela Redefinir um Item de Grupo em Outro Item de Grupo

Do mesmo modo que o caso anterior, há necessidade de que se escolha o programa através do *combo* rotulado por Selecione o Programa. Tendo como base heurísticas pré-determinadas, somente os relacionamentos de agregação cuja cardinalidade em relação à classe parte é de 0..1 são candidatos à generalização. Portanto, ao selecionar a opção Determinação dos Relacionamentos Candidatos a Alterações, esses candidatos são visualizados no quadro referenciados por (a) da Figura 88. Caso não existam casos candidatos à generalização nesse quadro, o engenheiro de software será notificado com a tela semelhante à apresentada na Figura 86.

Considerando a Figura 88, quando do preenchimento do quadro (a), seleciona-se a classe todo de um dos relacionamentos candidatos à generalização, através do *combo*

rotulado por 3-Selecione a Classe TODO (Pai). Selecionando-se a opção Mostrar Atributos da Classe Pai, são exibidos em (b) os atributos da classe todo que serão comparados com os da classe parte do relacionamento.

Para determinar as classes parte(s) que mantêm relacionamentos de generalização com a classe todo, candidatos à modificação, o engenheiro de software deve escolher a opção Exibir Classes Filhas. O banco será consultado e as classes parte(s) candidatas podem ser selecionadas no *combo* rotulado por 5-Selecione a Classe PARTE (Filha). A opção Mostrar Atributos da Classe Filha, quando ativada, exibe os atributos da classe parte em (c) da Figura 88 para que sejam comparados com os atributos da classe todo, exibidos em (b), para verificar se é um caso de redefinição de um item de grupo em outro, como o exemplificado na Figura 84.

Confirmada a existência desse caso de redefinição, é necessário escolher a opção Definir Relacionamento de Generalização, existente na tela da Figura 88. De forma transparente ao engenheiro de software, a ferramenta consulta o banco eliminando o relacionamento de associação, e criando um novo relacionamento de generalização entre as classes todo e parte analisadas e eliminando as cardinalidades. Finalizando assim, a determinação de um novo relacionamento de generalização entre as classes e formalizando o caso de redefinição de um item de grupo em outro item de grupo.

Após a realização desses passos encerra-se a fase de elaboração do Modelo de Análise do Sistema (MAS) com características orientadas a objetos.

6.3 Considerações Finais

Nesse capítulo, foi apresentada a Fase de Elaboração do MAS, bem como suas principais atividades que se destacam:

- A remoção das classes que representam DATA, permitindo uma otimização do código, uma vez que atributos podem representar a funcionalidade dessas classes logo eliminando-as do modelo orientado a objeto.

- A mudança dos nomes das classes e dos atributos por mnemônicos mais significativos, facilitando, posteriormente, o processo de manutenção e entendimento do sistema.

- O refinamento dos métodos, substituindo os nomes dos métodos construtores e observadores por mnemônicos mais significativos.

- A determinação de novos relacionamentos, especificamente para relacionamentos de: agregação, onde é formalizado o caso de redefinição de um item elementar em um item de grupo e de generalização, onde é formalizado o caso de redefinição de um item de grupo em outro item de grupo.

O processo de Elaboração do MAS sem uma ferramenta de apoio como a FAROOL, era realizado com análises minuciosas do código do sistema legado, sendo portanto uma tarefa que podia apresentar vários erros devido à quantidade de elementos que devem ser analisados. Além disso, os dados não eram armazenados em banco de forma transparente, dependendo de muita atenção e dedicação do engenheiro de software. Isso podia conduzir a um processo mal feito de recuperação dos sistemas legados, dificultando sua manutenção, e muitas vezes, até desestimulando o processo de engenharia reversa entre os profissionais, devido à dificuldade encontrada em aplicar esse processo. Mesmo com a FAROOL erros podem ocorrer, mas infere-se que esses sejam em número menor do que com o processo anterior, sem apoio computacional.

Os dados armazenados no banco de dados da FAROOL permitem a elaboração de um diagrama de classes completo de classes, atributos, métodos e relacionamentos que é o início para o processo de reengenharia com mudança de linguagem de programação.

O engenheiro de software pode, a partir do modelo de classes obtido, iniciar a implementação em uma linguagem orientada a objetos. Características de interface para interação homem-computador não são exploradas na FAROOL, ficando por conta dos conhecimentos do engenheiro de software a elaboração dessas.

O próximo capítulo apresenta um estudo de caso com a utilização da ferramenta desenvolvida neste projeto.

Capítulo 7

Aplicação da FAROOL em um Estudo de Caso

7.1 Considerações Iniciais

Neste capítulo será apresentada a aplicação da FAROOL em um estudo de caso para mostrar o seu funcionamento e as suas contribuições no processo de engenharia reversa de sistemas legados procedimentais para sistemas orientados a objeto.

O estudo de caso escolhido é um sistema legado *COBOL* denominado **"Controle de Recibos"** descrito na seção 7.2. Na seção 7.3, o processo de engenharia reversa realizado com a FAROOL é apresentado, e a seção 7.4 exibe as considerações finais sobre a aplicação da ferramenta FAROOL no estudo de caso.

7.2 Descrição do Sistema do Estudo de Caso

O sistema **Controle de Recibos** é composto de 19 arquivos de extensão <.cob>, 26 arquivos de extensão <.tel>. O sistema está dividido em três partes principais: estrutura que descrevem a estrutura geral de cada programa <<.COB>>, executáveis e fontes (<<.COB>>); organizadas conforme o Quadro 15.

O sistema **Controle de Recibos** foi desenvolvido para controlar os contratos de uma dada empresa, tendo sob sua responsabilidade a manutenção: do cadastro dos contratos; do arquivo mensal nos contratos; da situação dos clientes cadastrados; dos parâmetros do sistema que são passados para outros departamentos da empresa, a alteração da data do sistema quando fechados os ajustes mensais; das atividades a serem realizadas desde o cadastro do contrato até a confirmação e emissão do recibo; da impressão dos relatórios referentes aos contratos estabelecidos por representantes e clientes da empresa; da impressão dos relatórios relativos à movimentação mensal da empresa, da emissão do aviso de cobrança aos clientes e representantes; da emissão de recibos mensais aos compradores e

departamentos administrativos da empresa; do cálculo dos valores de reajuste dos contratos e a eliminação dos contratos inativos,

Quadro 15 - Organização do Sistema de Controle Recibos

	Quantidade de arquivos	Descrição
Estrutura	186 arquivos	São arquivos necessários para execução do código fonte. Eles armazenam a estrutura geral do sistema e são carregados na ferramenta *Legacy Aid*®, a partir da **FAROOL** por meio de macros, a medida que são chamados pelos programas do sistema.
Executáveis	228 arquivos entre dll(s), dtv(s) e exe(s)	Esses arquivos não são tratados pela **FAROOL,** pois servem para que a funcionalidade do sistema seja observada.
Fontes	45 arquivos, sendo: • 26 <.tel> (não usados) • 19 <.cob>	Os 19 arquivos de extensão <.cob> são usados como entrada na ferramenta *Legacy Aid*® ativada a partir da **FAROOL** para iniciar o processo de engenharia reversa orientada a objeto.

Esse sistema está integrado ao sistema **"Contas a Receber"** e ao sistema **"Controle de Estoque"**, não considerados no estudo aqui apresentado.

O **Controle de Recibos** é um sistema que possui razoável documentação no seu código fonte, porém está implementado de forma desestruturada. Como todo sistema legado não possui documentos que mostrem o seu desenvolvimento e as regras de negócio utilizadas. Ele foi submetido à ferramenta FAROOL como estudo de caso para se analisar a utilidade da mesma para o processo de engenharia reversa de sistemas legados

desenvolvidos por pessoas que não fazem parte deste projeto, e, também não são conhecidas do grupo.

Na próxima seção inicia-se a aplicação da ferramenta proposta ao estudo de caso.

7.3 Aplicação da FAROOL

O processo de engenharia reversa do sistema legado proposto foi realizado considerando três fases principais: a fase de Preparação do Sistema, a fase de Elaboração do MASA e a fase de Elaboração do MAS, tratadas separadamente nas seções que se seguem.

7.3.1 Fase de preparação do sistema

Conforme foi apresentado no capítulo 4, essa fase é realizada em três passos: Criação de um Novo Projeto, Aplicação do Parser e Mapeamento dos Arquivos de Estrutura do sistema legado.

Para o caso do sistema Controle de Recibos foi criado um projeto com o nome de CONTROLE DE RECIBOS para o campo Project Name e no campo Description consta breve descrição da funcionalidade do sistema, Figura 89.

Para diferenciar os usuários que têm acesso ao sistema foram definidas as especificações padrões, que podem ser vistas na Figura 90.

Figura 89 - Tela Inicial para Criação de um Novo Sistema

Figura 90 - Tela determinação dos usuários do sistema

Após essas definições, foi determinado o *driver* C onde se encontram os arquivos do sistema proposto e a pasta ContrRecibEmUso na qual estão os arquivos fontes do sistema. Foram selecionados os arquivos fontes com extensão <<.COB>>, disponibilizados no campo File Name. Como todos os arquivos disponíveis devem ser tratados no processo de engenharia reversa, foi selecionada a opção Add All, Figura 91.

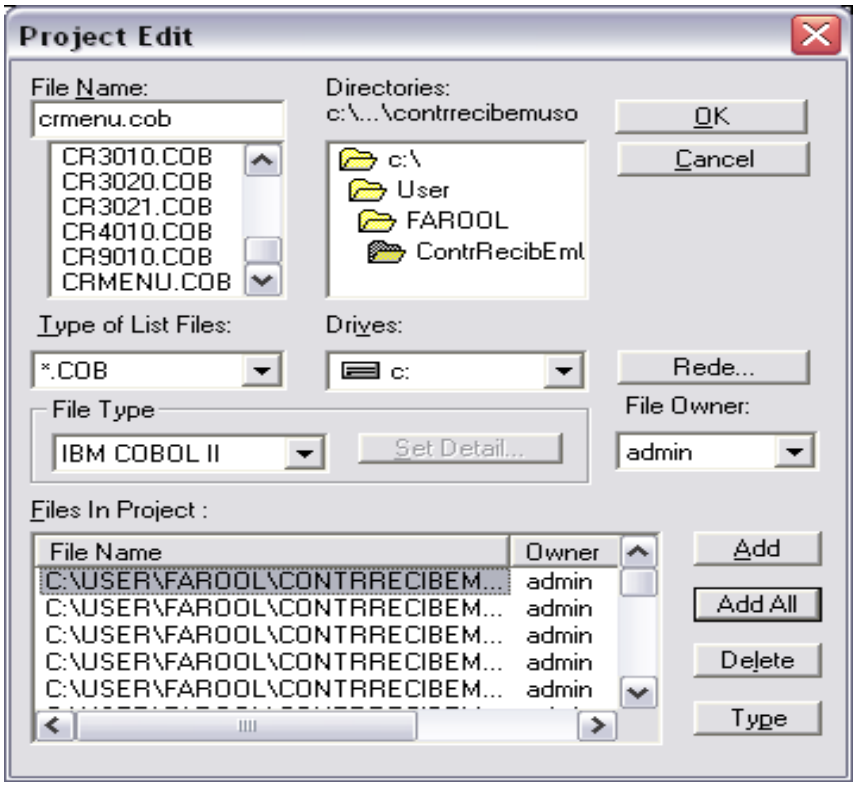


Figura 91 - Tela determinação dos arquivos fontes a serem carregados no projeto

O sistema foi então submetido à aplicação do *parser* para os arquivos <<.COB>>, como pode-se observar pelo campo Parsing File da Figura 92.

Figura 92 - Aplicação do Parser nos arquivos fontes do sistema

O resultado parcial obtido, após a aplicação do *parser* no sistema Controle de Recibos, pode ser visto na Figura 93.

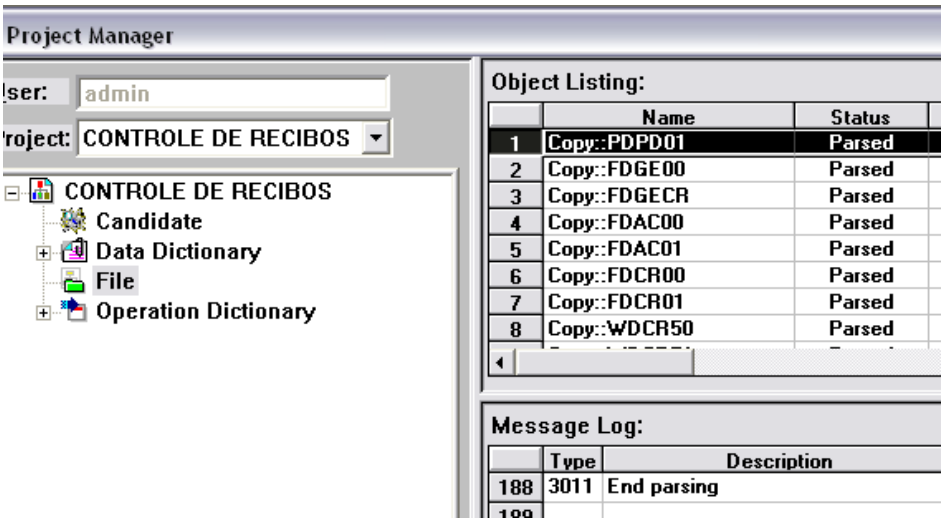


Figura 93 - Resultado da Aplicação do Parser

Para finalizar a fase de Preparação do Sistema, é realizado o mapeamento para incorporar os arquivos que definem a estrutura do sistema, chamados de *Copy Files*, conforme ilustrado na Figura 94.

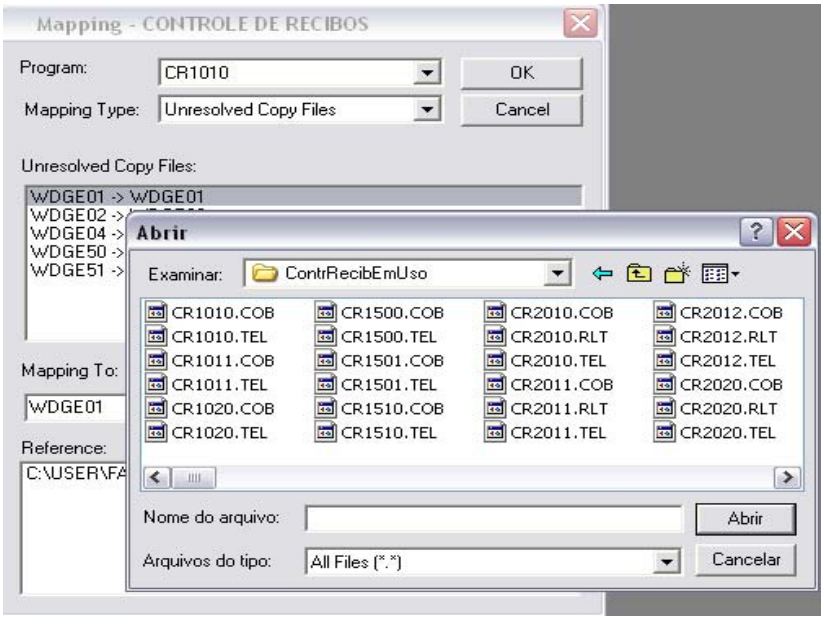


Figura 94 - Mapeando os arquivos Copy Files do Sistema Controle de Recibos

7.3.2 Fase de Elaboração do MASA

A elaboração do MASA (Modelo de Análise do Sistema) consta da determinação das classes candidatas e seus relacionamentos, dos atributos candidatos, dos métodos candidatos.

1- Determinação das Classes Candidatas – Foram analisados os FDs de cada programa do sistema Controle de Recibos, obtendo-se o resultado parcial mostrado na Figura 95. Os programas definidos e seus respectivos FDs foram inseridos nos campos da interface da ferramenta FAROOL, Figura 96 e armazenados no banco de dados da mesma na classe Class, Figura 97.

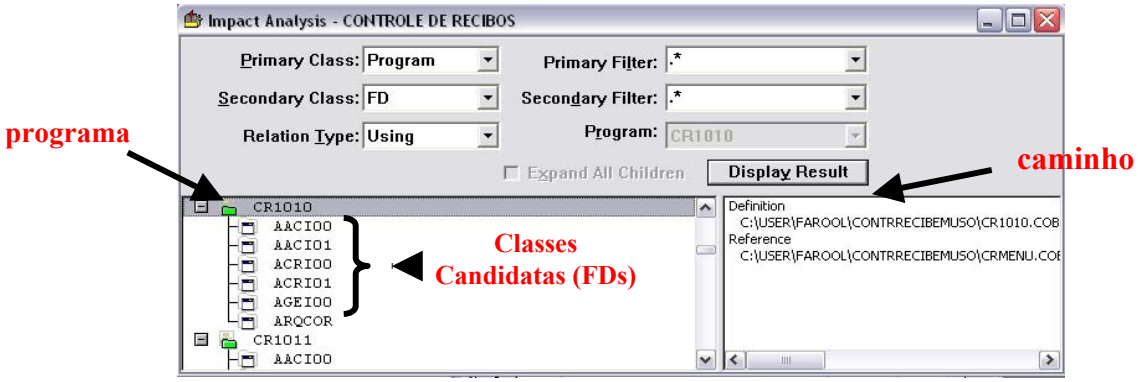


Figura 95- Tela Impact Analysis para determinação dos programas e respectivos FDs

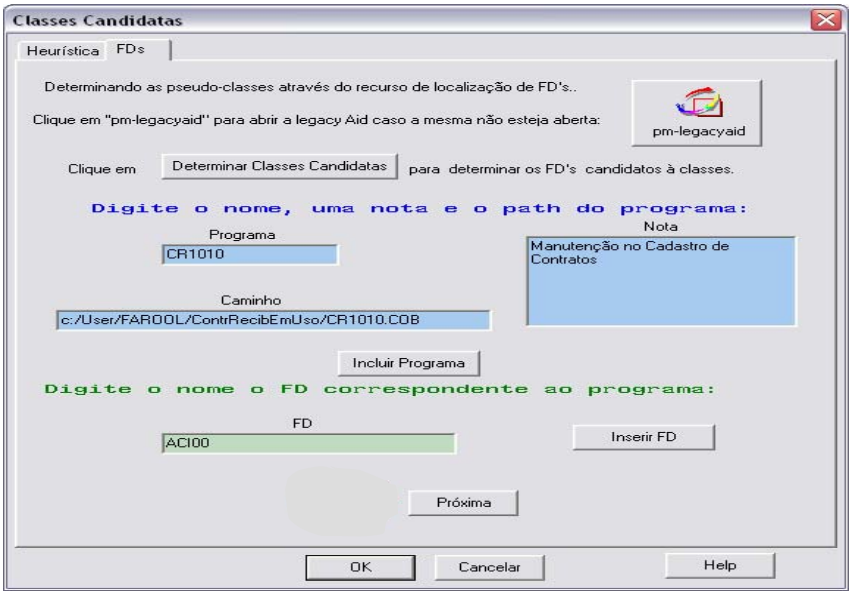


Figura 96- Tela Classes Candidatas aba FDs para inserção dos FDs

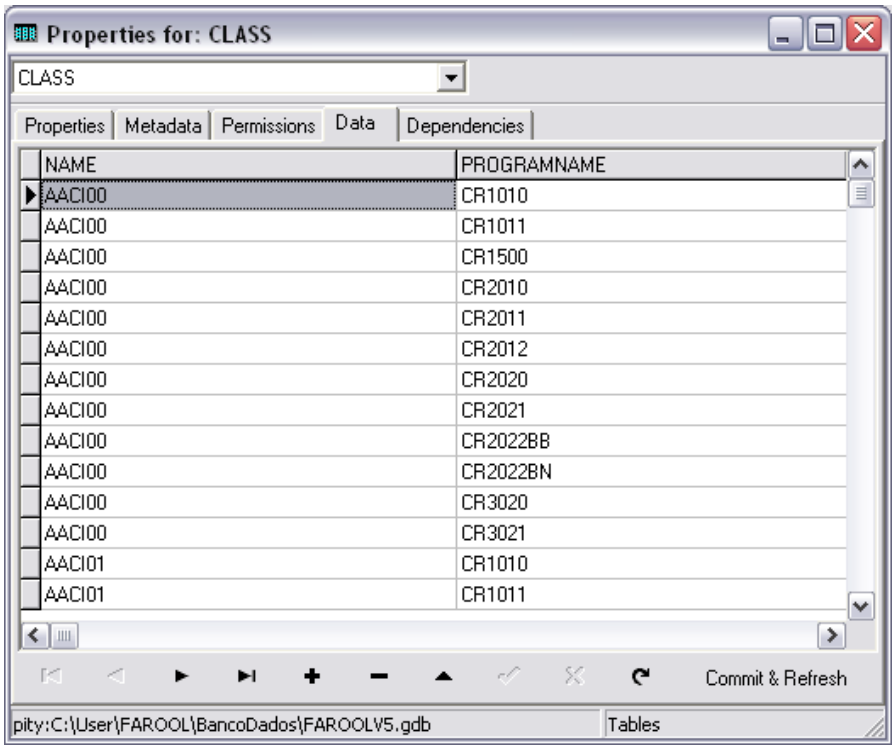


Figura 97 - Armazenamento das classes candidatas (FDs) no banco FAROOL

Para definição das chaves de cada programa foi percorrido o sistema Controle de Recibos através do recurso Find In, Figura 98. Essas informações são selecionadas como pode ser visto na tela apresentada na Figura 99 e armazenadas no banco FAROOL na classe *IntermediateTable*, conforme mostrado na Figura 100.

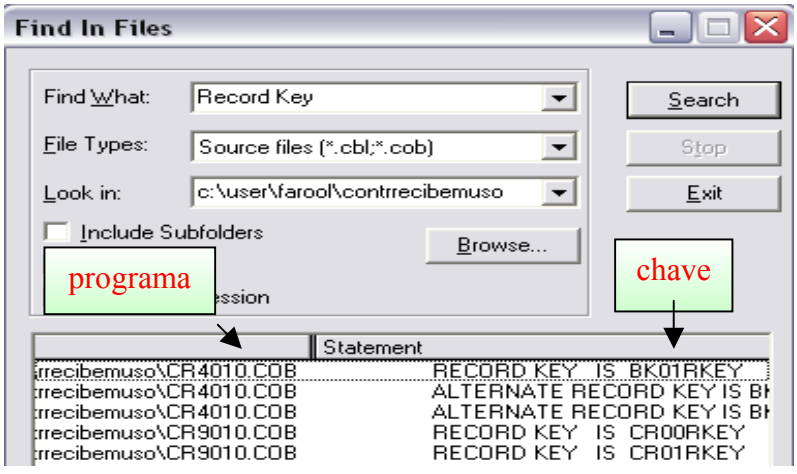


Figura 98 - Tela Find In para determinar as chaves dos programas

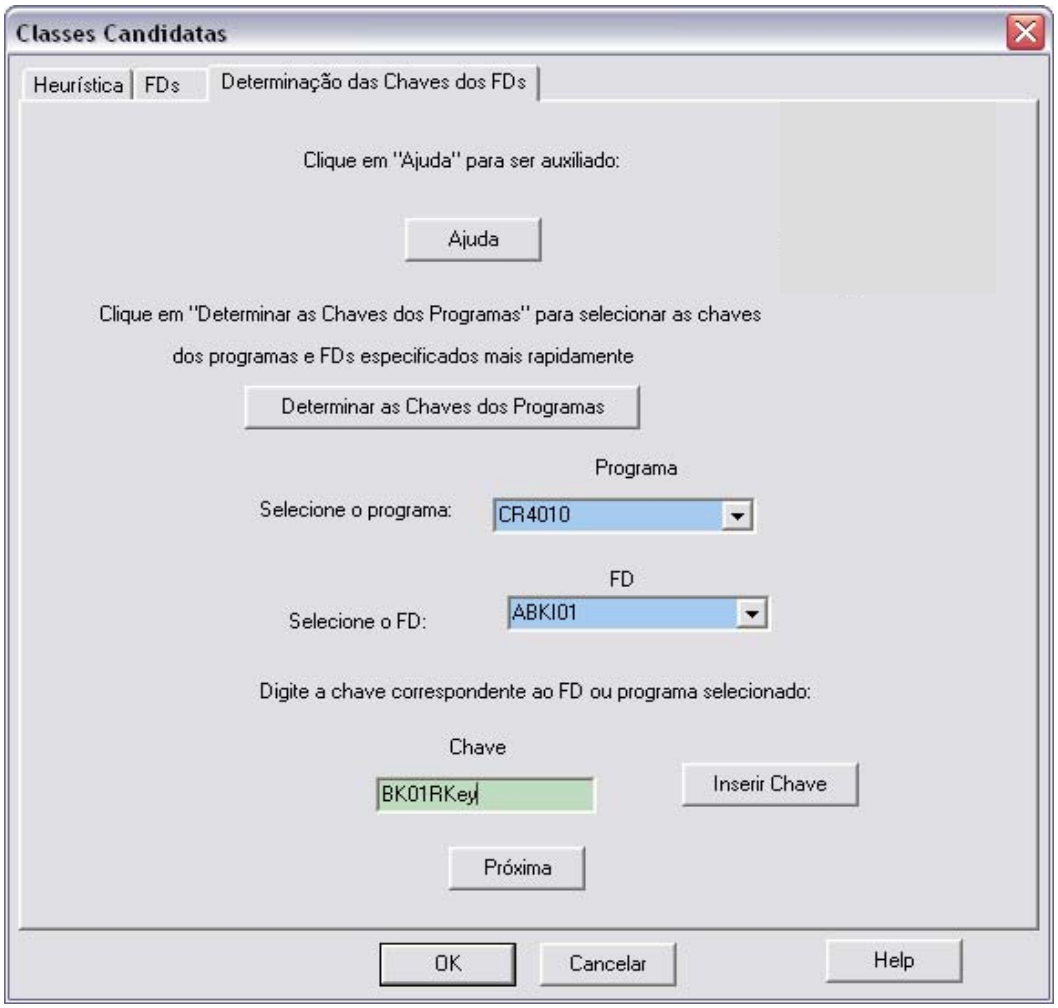


Figura 99 - Tela Classes Candidatas aba Determinação das Chaves dos FDs

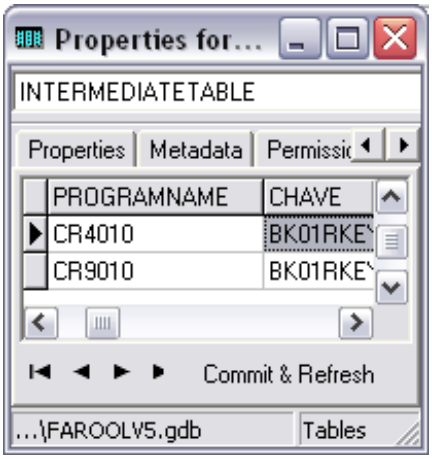


Figura 100 - Armazenamento no banco FAROOL das Chaves

A classificação das classes candidatas como sendo de leitura, escrita e reescrita, foi realizada com base nas informações obtidas a partir da análise do sistema, conforme são apresentadas na Figura 101, para os casos de leitura (*READ*), escrita e/ou reescrita (*WRITE* e/ou *REWRITE*). Essas informações são colocadas na tela Classes Candidatas aba Classificação dos FDs, e armazenadas no banco FAROOL na classe *IntermediateTable*, como mostra a Figura 102.

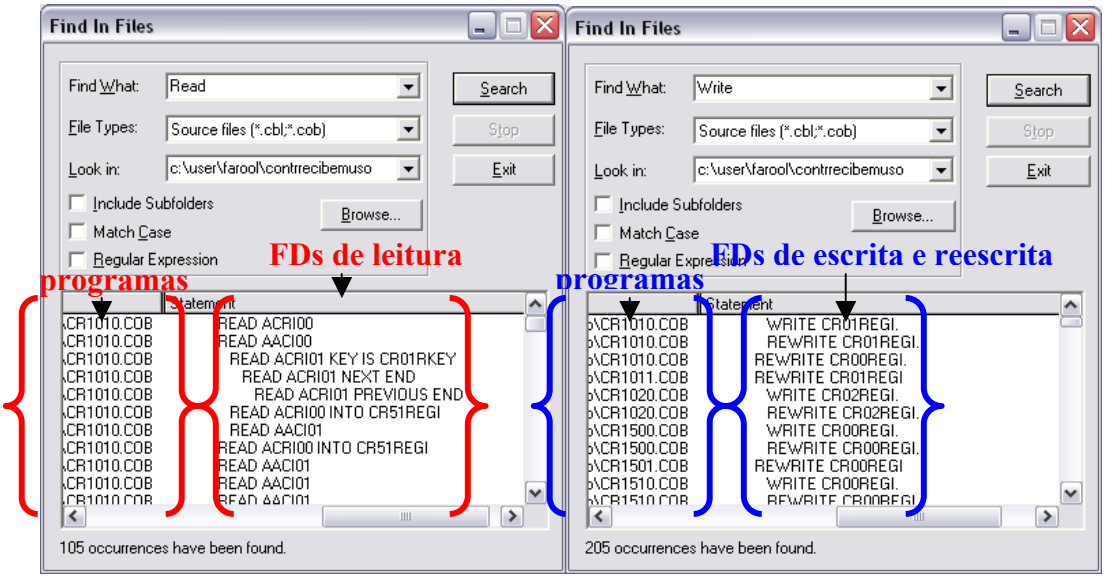


Figura 101 - Telas para classificação das Classes Candidatas em leitura, escrita e Reescrita

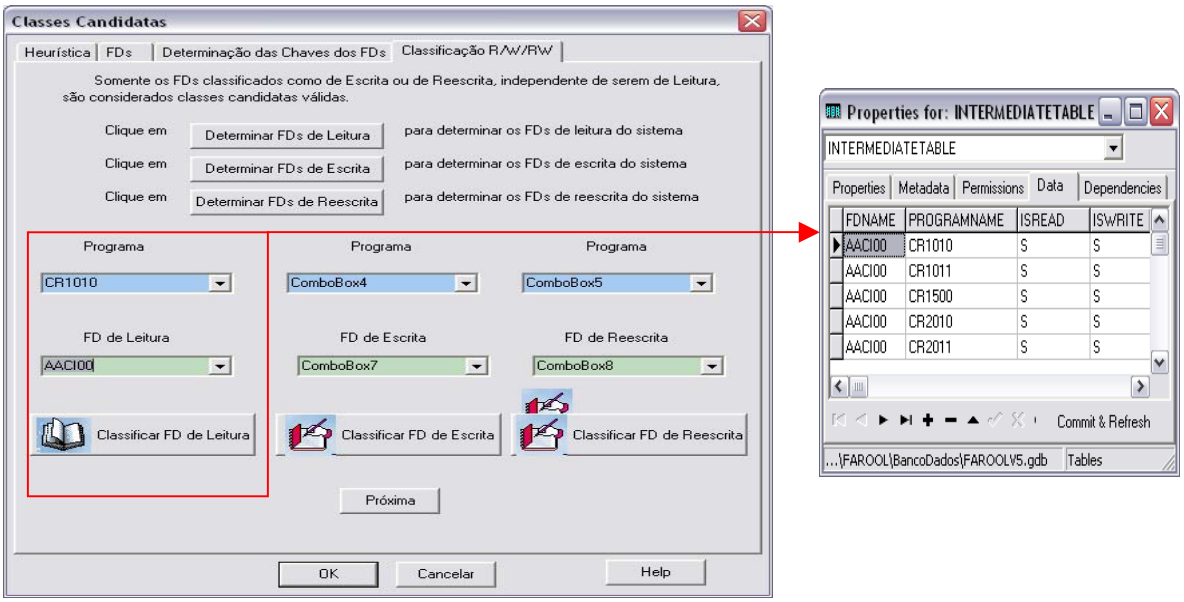


Figura 102 - Tela Classe Candidata aba Classificação dos FDs e armazenamento na tabela IntermediateTable do banco FAROOL

Os FDs existentes no sistema Controle de Recibos foram classificados em classes observadoras, construtoras ou de implementação. Os resultados parciais obtidos, para o caso específico do programa CR1010, são apresentados na Figura 103.

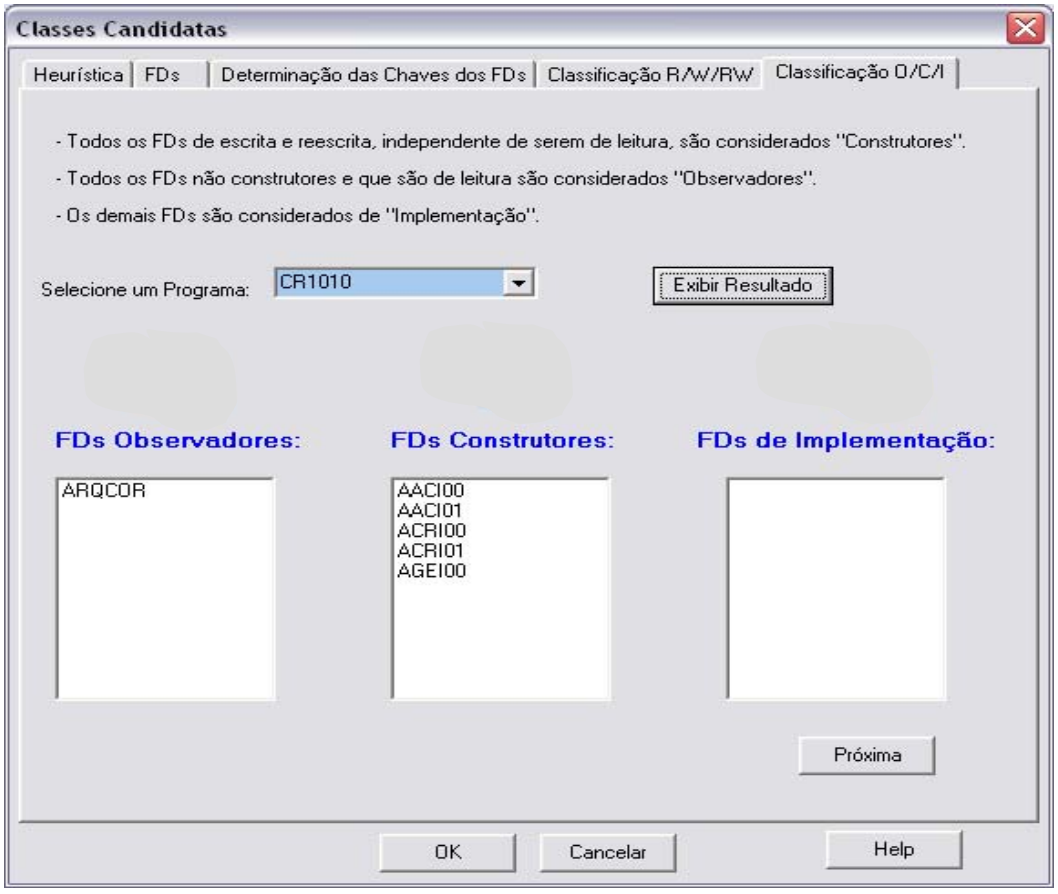


Figura 103 - Resultado da Classificação dos FDs para o programa CR1010 do sistema Controle de Recibos

As classes candidatas são escolhidas como mostra a Figura 104. O engenheiro de software, a partir de seu conhecimento do sistema legado, pode eliminar aquelas que não se tornarão classes por serem informações usadas devido ao tipo de implementação utilizada. No sistema de Controle de Recibos todas as classes geradas a partir dos FDs são consideradas.

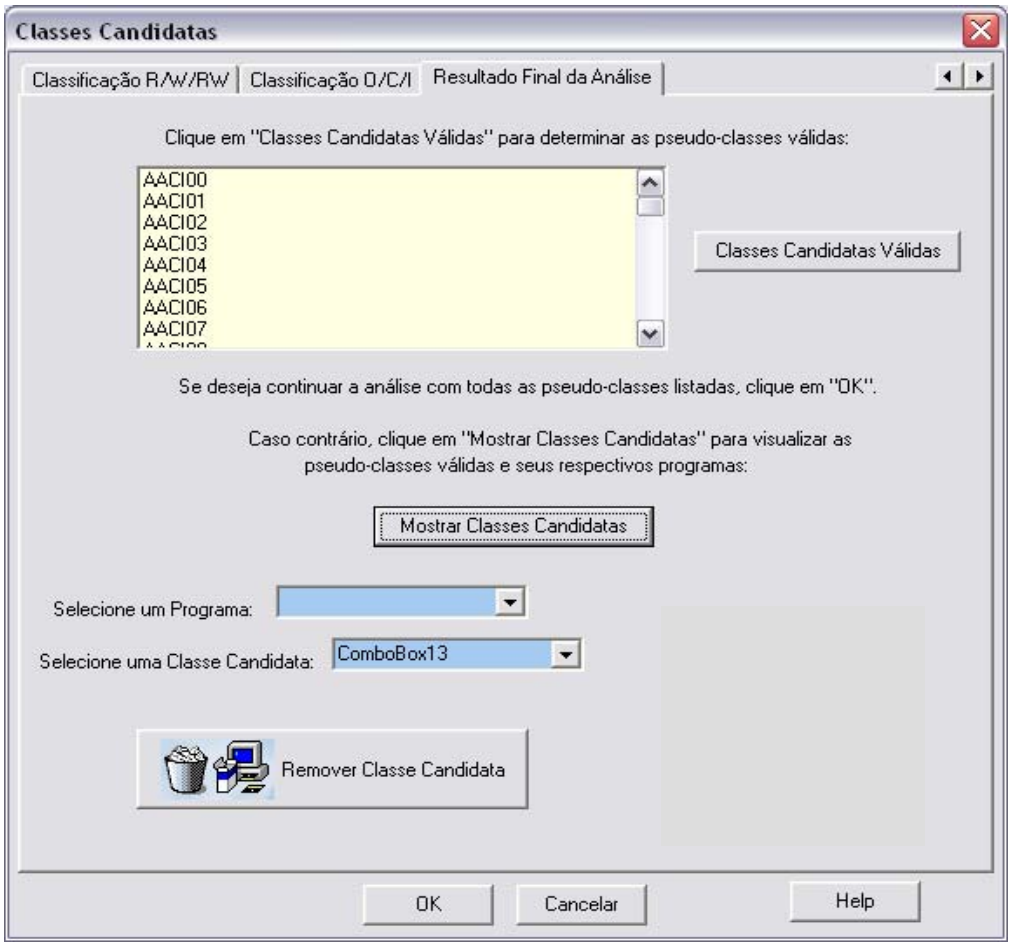


Figura 104 - Classes Candidatas Válidas do sistema Controle de Recibos para o Processo de Engenharia Reversa

A tabela 4 exibe as classes candidatas para cada programa do sistema em uso. Essas classes estão armazenadas no banco de dados FAROOL. Se o engenheiro de software desejar visualizar essas classes, ele deve optar por Mostrar Classes Candidatas, Figura 102, quando uma tabela com todas as classes candidatas, bem como com seus respectivos programas e suas chaves são oferecidos.

O próximo passo cuida do tratamento dos atributos encontrados no sistema Controle de Recibos.

Tabela 4 - Programas do sistema Controle de Recibos e suas respectivas classes candidatas:

Numeração	Programa	Classe
1	CR1010	AACI00; AACI01; ACRI00; ACRI01; AGEI00.
2	CR1011	AACI00; AACI01; ACRI00; ACRI01; AGEI00.
3	CR1020	AACI00; AACI01; ACRI00; ACRI02; AGEI00.
4	CR1500	AACI00; ACRI00; AGEI00.
5	CR1501	ACRI00; AGEI00.
6	CR1510	ACRI00; AGEI00.
7	CR2010	AACI00; AACI01; ACRI00; ACRI01; AGEI00.
8	CR2011	AACI00; AACI01; ACRI00; ACRI01; AGEI00.
9	CR2012	AACI00; AACI01; ACRI00; ACRI01; AGEI00.
10	CR2020	AACI00; AACI01; ACRI00; ACRI02; AGEI00.
11	CR2021	AACI00; AACI01; ACRI00; ACRI02; AGEI00.
12	CR2022BB	AACI00; AACI01; AACI02; ACRI00; ACRI02; AGEI00.
13	CR2022BN	AACI00; AACI01; AACI02; ACRI00; ACRI02; AGEI00.
14	CR3010	ACRI00; ACRI01; AGEI00.
15	CR3020	AACI00; AACI01; AACI00; ACRI01; ACRI02; AGEI00.
16	CR3021	AACI00; AACI01; AACI02; AACI03; AACI04; AACI05; AACI06; AACI07; AACI08; AACI10; AACI13; AACI15; AACI16; AACI19; AACI30; AACI31; AACI40; AACI85; ACRI00; ACRI01; ACRI02; AGEI00.
17	CR4010	ABKI01; ACRI00; ACRI01; AGE00.
18	CR9010	ABKI01; ACRI00; ACRI01; AGE00.
19	**CRmenu**	**AGEI00.**

2- Determinação dos Atributos Candidatos – Para realização dessa tarefa foram analisados os itens de dados existentes para cada FD do sistema Controle de Recibos. A Figura 105 mostra o resultado parcial da análise de um FD, o AACI01.

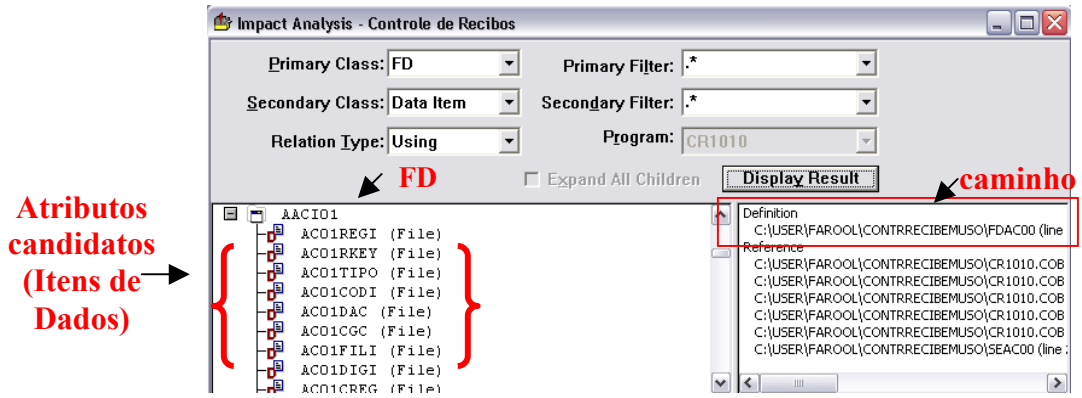


Figura 105 - Tela Impact Analysis para determinação dos Atributos Candidatos

Com um clique-duplo no caminho especificado em Definition, obtém-se o trecho de código, correspondente ao FD AACI00 com seus respectivos Itens de dados, candidatos a atributos, Figura 106.

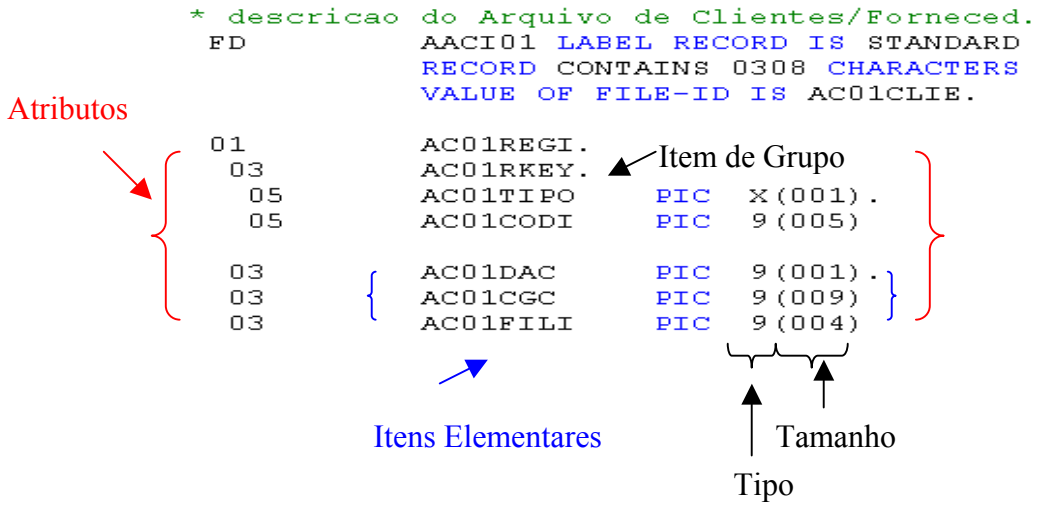


Figura 106 - Trecho de código do FD ACI00 do sistema Controle de Recibos

Essas informações são utilizadas como entrada para a FAROOL nos *combos* apresentados na Figura 107: Programa recebe o nome do programa, CR1010; Classe Candidata (FD) recebe o nome do FD, AC01REGI; Nome do Atributo recebe AC01DAC; seu tipo, numérico e seu tamanho, 001 são colocados, respectivamente, em Tipo do Atributo e Tamanho do Atributo.

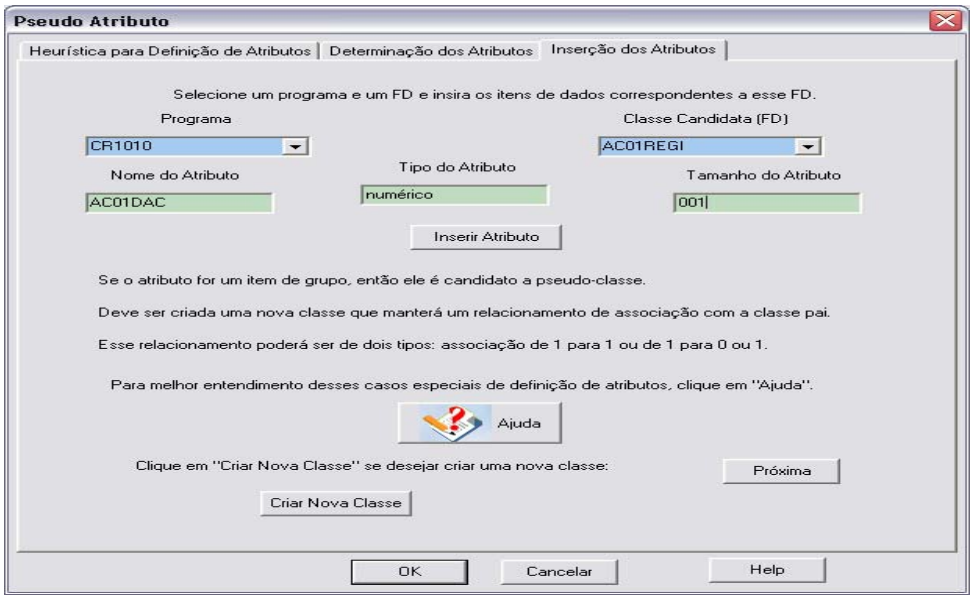


Figura 107 - Tela Atributos aba Inserção dos Atributos

AC1REGI é um item de grupo, que pelas heurísticas, favorece à criação de uma nova classe. A Figura 108 exibe a interface para esse procedimento.

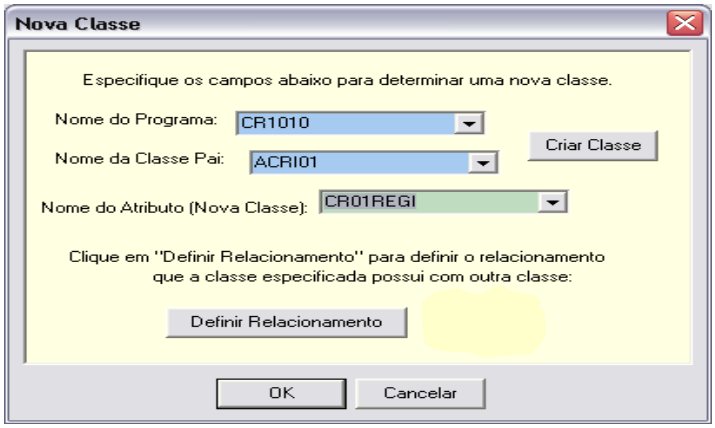


Figura 108 - Criação da Classe correspondente ao Item de Grupo AC01REGI

A definição dos relacionamentos entre as classes criadas e as classe que continham os atributos pode ser entendida analisando o caso particular que envolve a classe AACI01 do programa CR1010 e o item de grupo que deu origem a nova classe AC1REGI. Nesse caso, é definido o tipo de relacionamento de associação, pois com base em heurísticas, uma nova classe oriunda de um item de grupo, AC1REGI, mantém uma associação com a classe Pai, AACI01. Por meio da tela da Figura 109 clicando-se em A fica estabelecido tal relacionamento. Esse procedimento deve ser feito para todos os itens de grupo encontrados

no sistema legado quando da análise da Figura 105. Posteriormente, deve ocorrer a determinação dos papéis (*roles*) para a classe Filha e Pai cujo relacionamento foi definido, utilizando as telas da Figura 110.

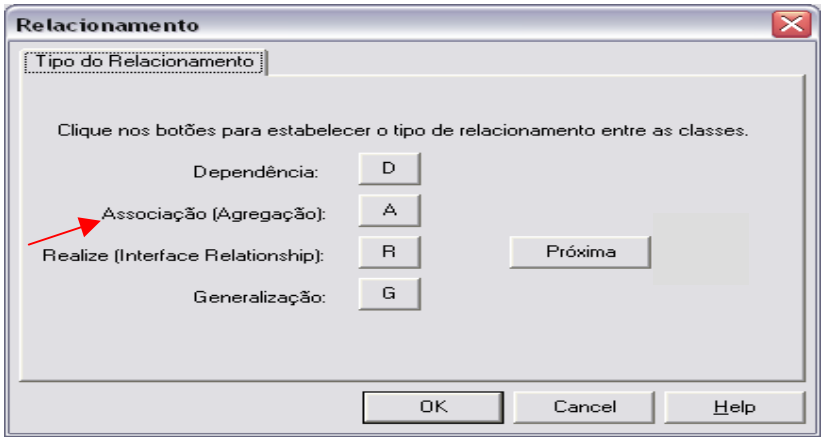


Figura 109 - Determinação do tipo de Relacionamento

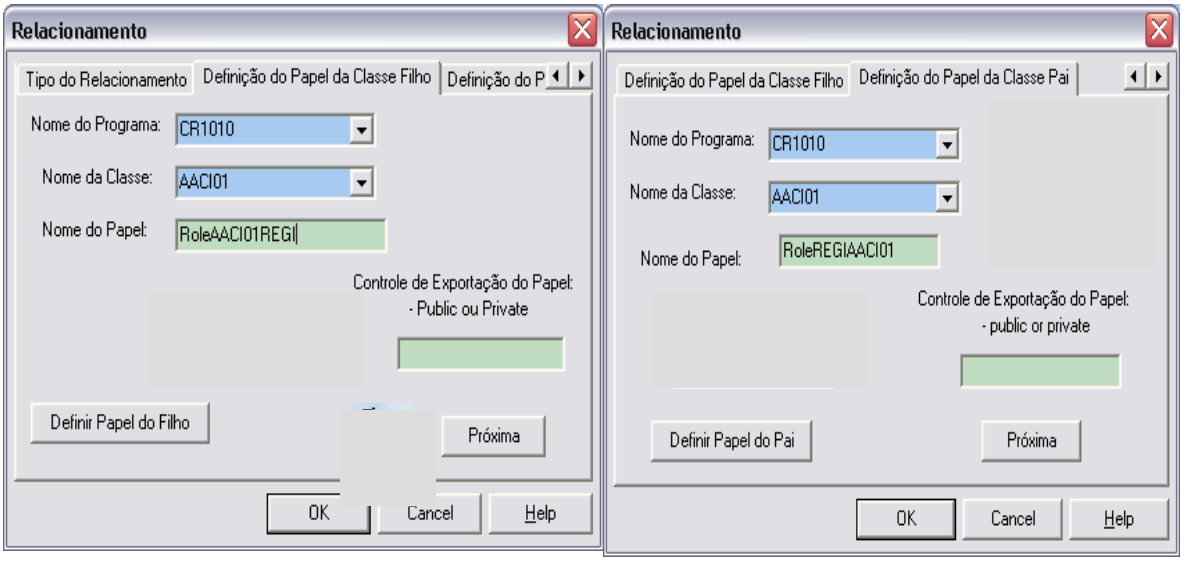


Figura 110 - Determinação dos Papeis da Classe Filho AC01REGI e da Classe Pai ACI01

Outras informações são necessárias, por exemplo, a cardinalidade, se é navegável, se é de agregação, etc., como mostram as telas da Figura 111. A cardinalidade desse relacionamento, também com base em heurísticas, pode ser de 1 para 0..1 ou de 1 para 1. Para o caso tomado como exemplo, a classe mais geral AACI01 mantém uma associação de 1 para 0..1 com a nova classe AC01REGI .

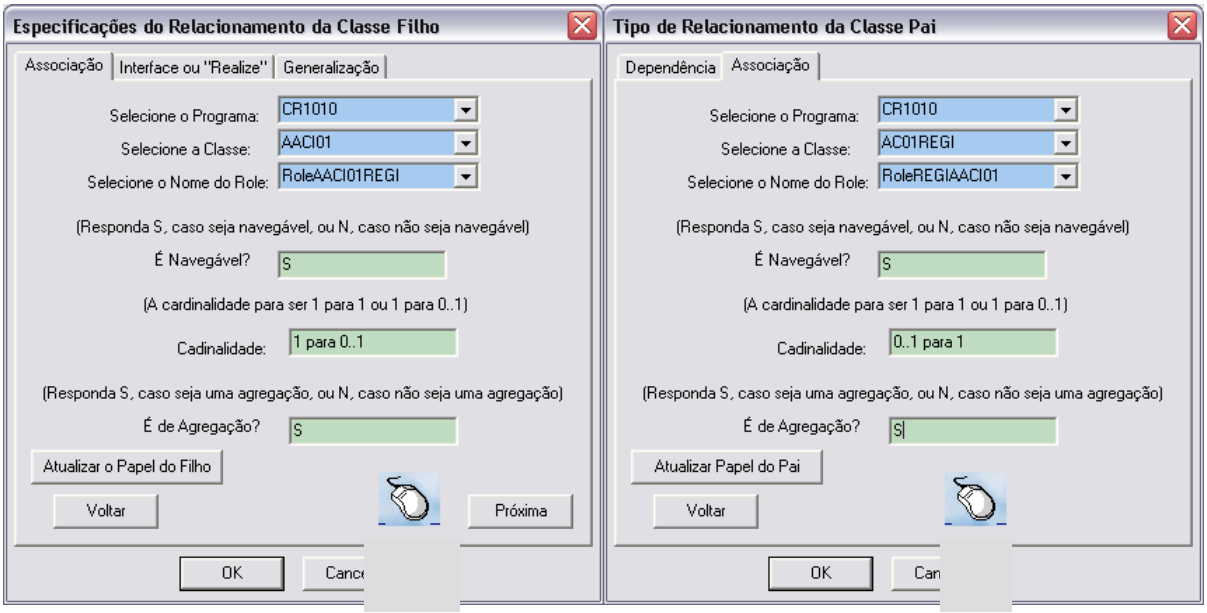
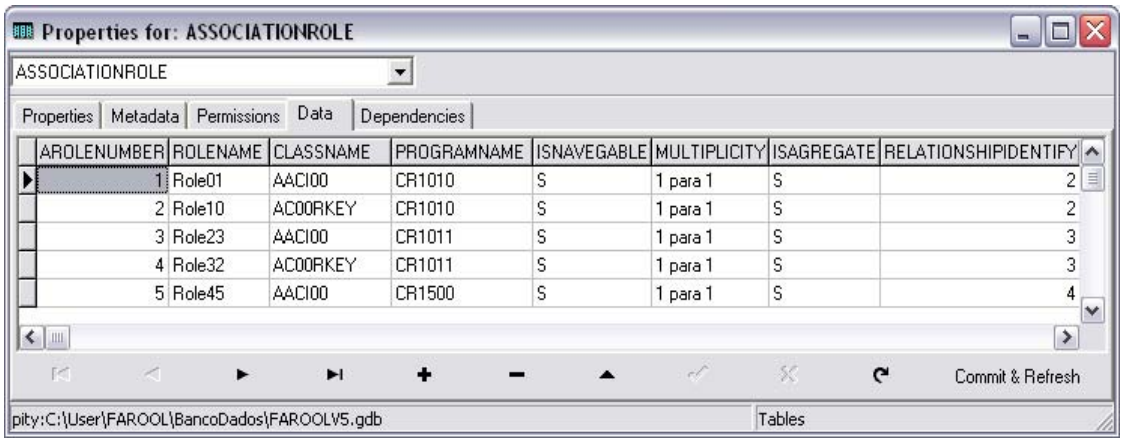


Figura 111 - Especificações da Associação entre as Classes AC01REGI e a AACI01

O banco de dados FAROOL resultante, após o armazenamento das informações, é mostrado parcialmente na Figura 112.



AROLENUMBER	ROLENAME	CLASSNAME	PROGRAMNAME	ISNAVEGABLE	MULTIPLICITY	ISAGREGATE	RELATIONSHIPIDENTIFY
1	Role01	AACI00	CR1010	S	1 para 1	S	2
2	Role10	AC00RKEY	CR1010	S	1 para 1	S	2
3	Role23	AACI00	CR1011	S	1 para 1	S	3
4	Role32	AC00RKEY	CR1011	S	1 para 1	S	3
5	Role45	AACI00	CR1500	S	1 para 1	S	4

Figura 112 - Banco de dados da FAROOL com os relacionamentos de Associação

A Figura 113 exibe a tela com os elementos de dados que foram selecionados como atributos candidatos após a finalização da atividade responsável pela determinação dos atributos.

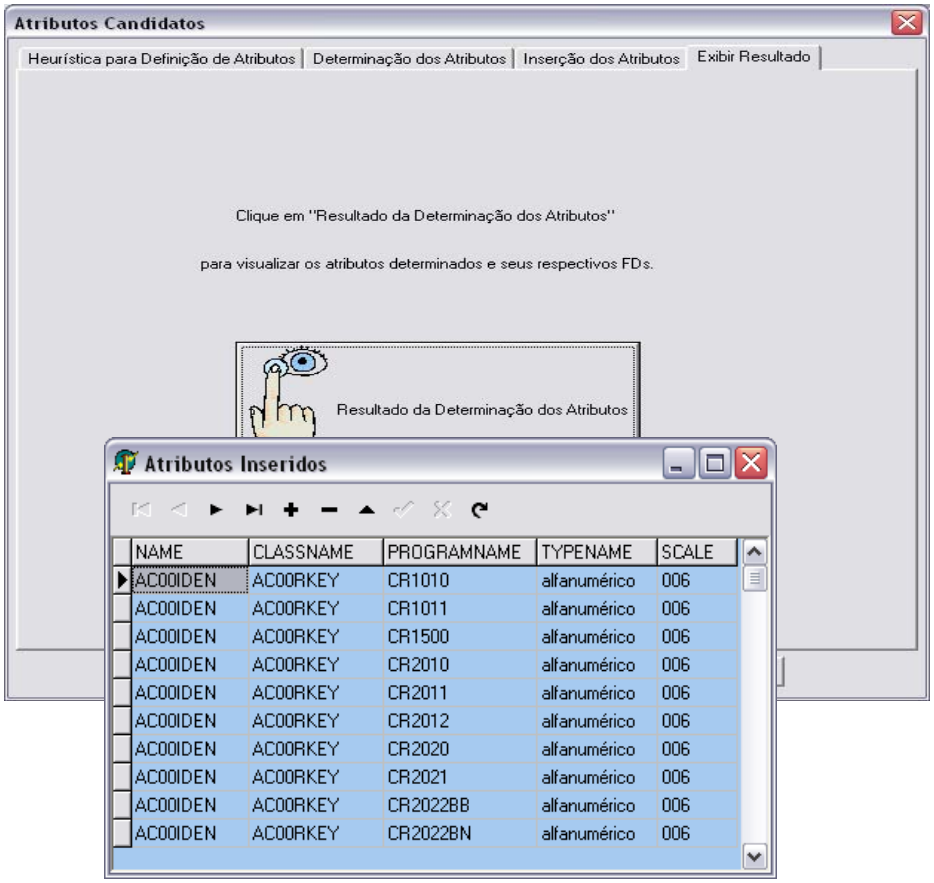
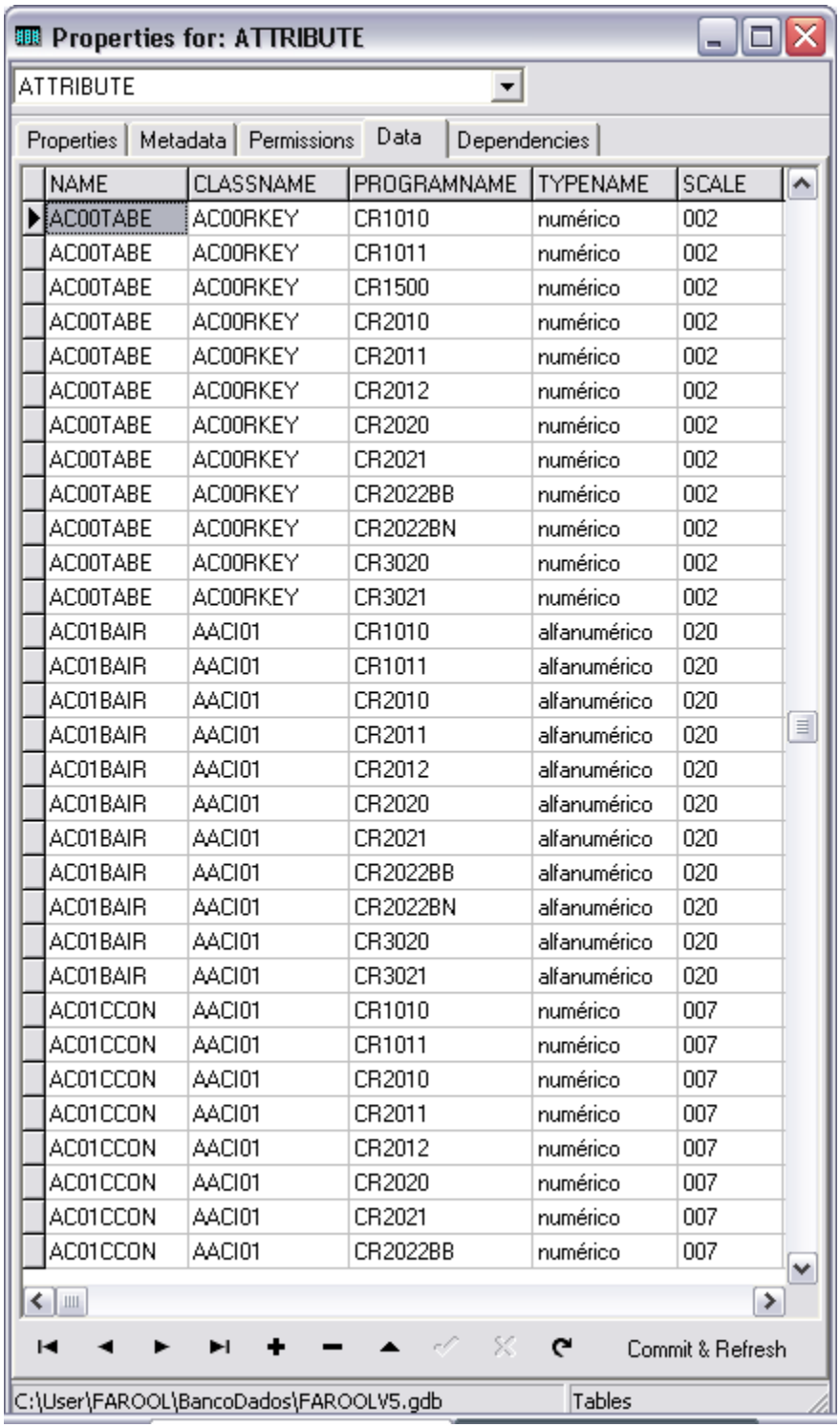


Figura 113 - Visualização dos resultados obtidos com a Determinação dos Atributos

A tabela 5 exibe alguns dos atributos candidatos em NAME com suas respectivas classes em CLASSNAME e programas em PROGRAMNAME, bem como seus tipos em TYPENAME e tamanhos em SCALE. Esses atributos estão armazenados no banco de dados FAROOL.

O próximo passo cuida da determinação dos métodos candidatos encontrados no sistema Controle de Recibos, do armazenamento no banco FAROOL dos seus nomes, das suas respectivas classes e dos seus programas.

Tabela 5 – Alguns Atributos Candidatos do Sistema Controle de Recibos



NAME	CLASSNAME	PROGRAMNAME	TYPENAME	SCALE
AC00TABE	AC00RKEY	CR1010	numérico	002
AC00TABE	AC00RKEY	CR1011	numérico	002
AC00TABE	AC00RKEY	CR1500	numérico	002
AC00TABE	AC00RKEY	CR2010	numérico	002
AC00TABE	AC00RKEY	CR2011	numérico	002
AC00TABE	AC00RKEY	CR2012	numérico	002
AC00TABE	AC00RKEY	CR2020	numérico	002
AC00TABE	AC00RKEY	CR2021	numérico	002
AC00TABE	AC00RKEY	CR2022BB	numérico	002
AC00TABE	AC00RKEY	CR2022BN	numérico	002
AC00TABE	AC00RKEY	CR3020	numérico	002
AC00TABE	AC00RKEY	CR3021	numérico	002
AC01BAIR	AACI01	CR1010	alfanumérico	020
AC01BAIR	AACI01	CR1011	alfanumérico	020
AC01BAIR	AACI01	CR2010	alfanumérico	020
AC01BAIR	AACI01	CR2011	alfanumérico	020
AC01BAIR	AACI01	CR2012	alfanumérico	020
AC01BAIR	AACI01	CR2020	alfanumérico	020
AC01BAIR	AACI01	CR2021	alfanumérico	020
AC01BAIR	AACI01	CR2022BB	alfanumérico	020
AC01BAIR	AACI01	CR2022BN	alfanumérico	020
AC01BAIR	AACI01	CR3020	alfanumérico	020
AC01BAIR	AACI01	CR3021	alfanumérico	020
AC01CCON	AACI01	CR1010	numérico	007
AC01CCON	AACI01	CR1011	numérico	007
AC01CCON	AACI01	CR2010	numérico	007
AC01CCON	AACI01	CR2011	numérico	007
AC01CCON	AACI01	CR2012	numérico	007
AC01CCON	AACI01	CR2020	numérico	007
AC01CCON	AACI01	CR2021	numérico	007
AC01CCON	AACI01	CR2022BB	numérico	007

3- Determinação dos Métodos Candidatos – Para realização dessa tarefa foram analisados os procedimentos existentes para cada programa do sistema legado Controle de Recibos. A Figura 114 mostra, parcialmente, algumas informações que são consideradas nessa tarefa.

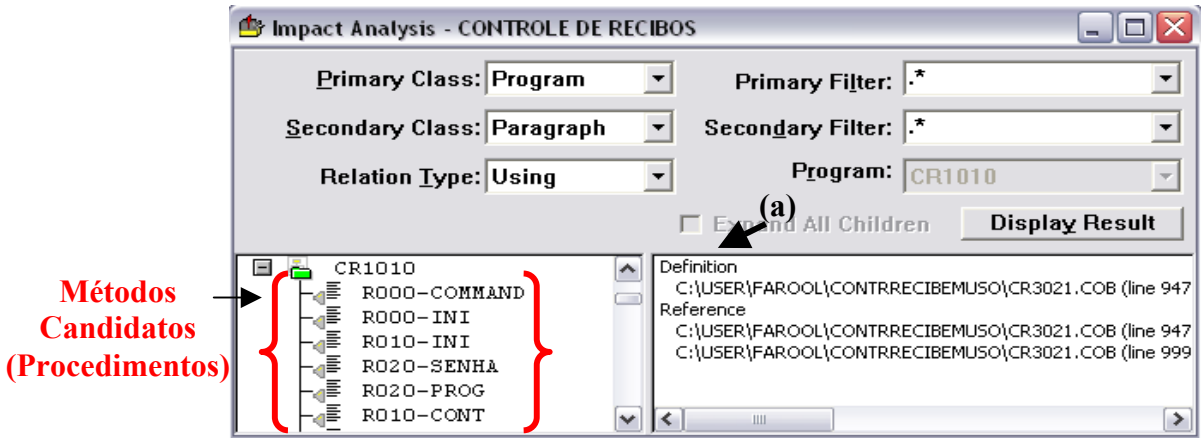


Figura 114 - Tela Impact Analysis para Determinação dos Métodos Candidatos

Para se obter o corpo do método é necessário um clique-duplo no caminho especificado em Definition, Figura 114(a). O trecho de código do programa escolhido anteriormente é exibido em uma tela com mostra a Figura 115.

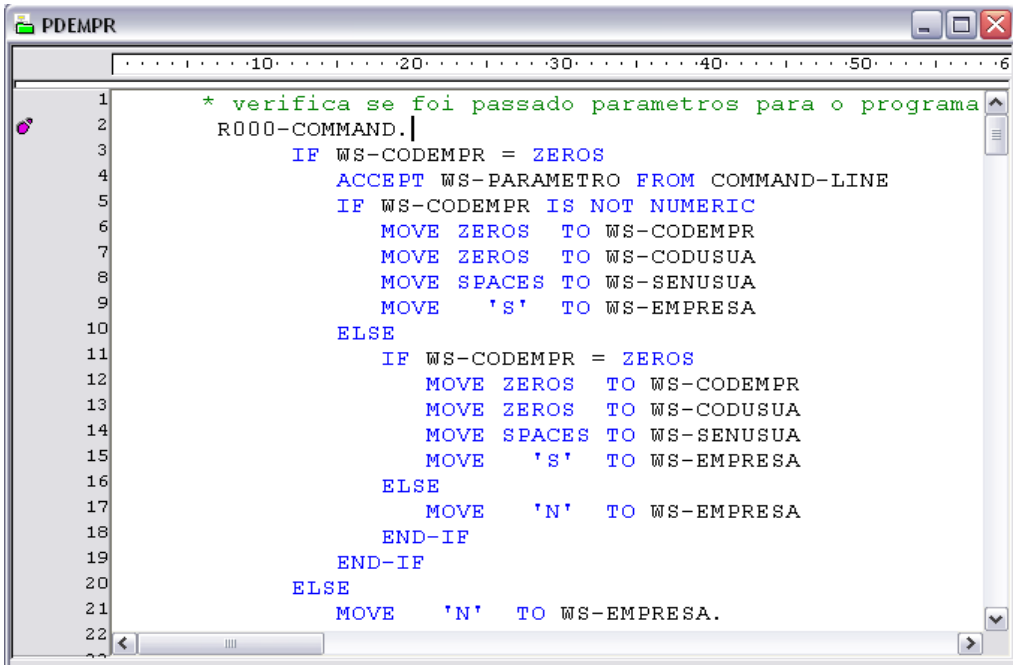


Figura 115 - Trecho de Código do procedimento ROOO-COMAND

A partir das informações exibidas na Figura 114, para cada programa, classe candidata (FD) e método encontrados no sistema legado, foram preenchidos os campos

Programa, Classe Candidata, Nome do Método e Corpo do Método, conforme apresentado na Figura 116.

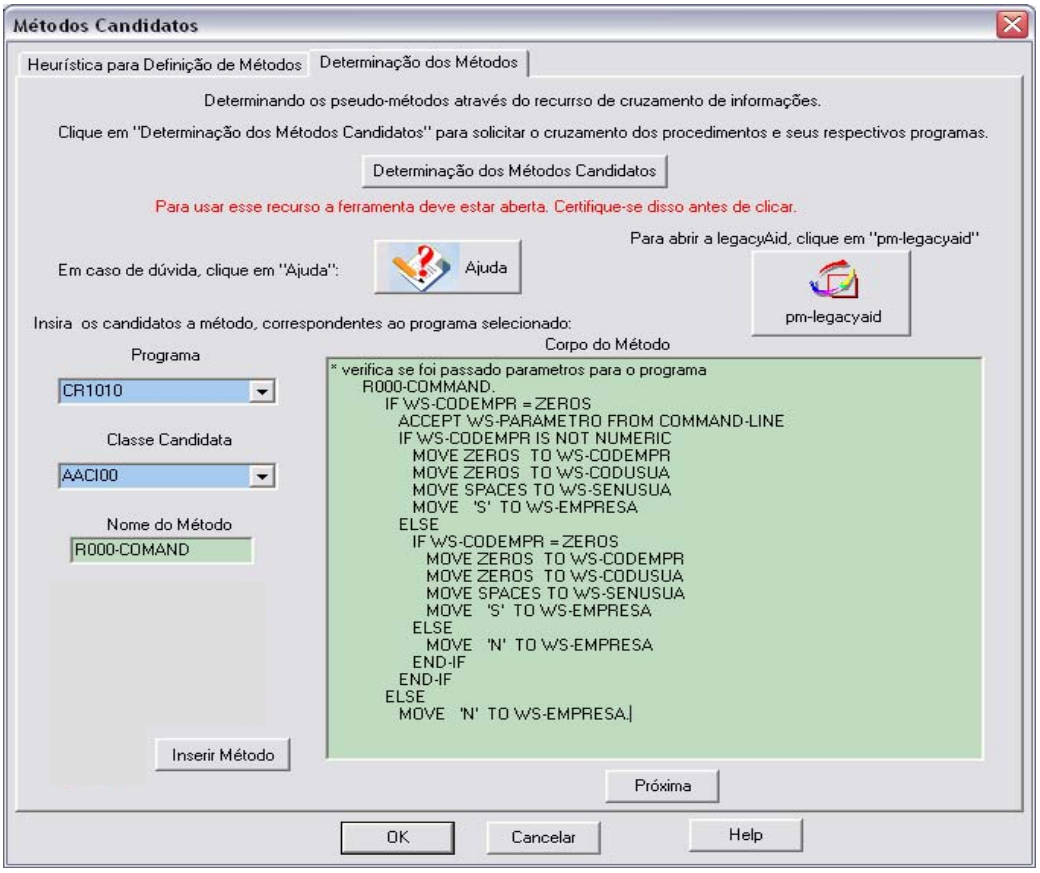


Figura 116 - Tela Métodos Candidatos aba Determinação dos Métodos do programa CR1010

Uma vez inseridos todos os métodos candidatos do sistema, os mesmos devem ser classificados em observadores, construtores ou de implementação utilizando as informações e selecionando os botões apropriados da tela da Figura 117. Alguns dos métodos, já classificados, são mostrados em NAME, Tabela 6, com seus respectivos programas em PROGRAMNAME e classes em CLASSNAME.

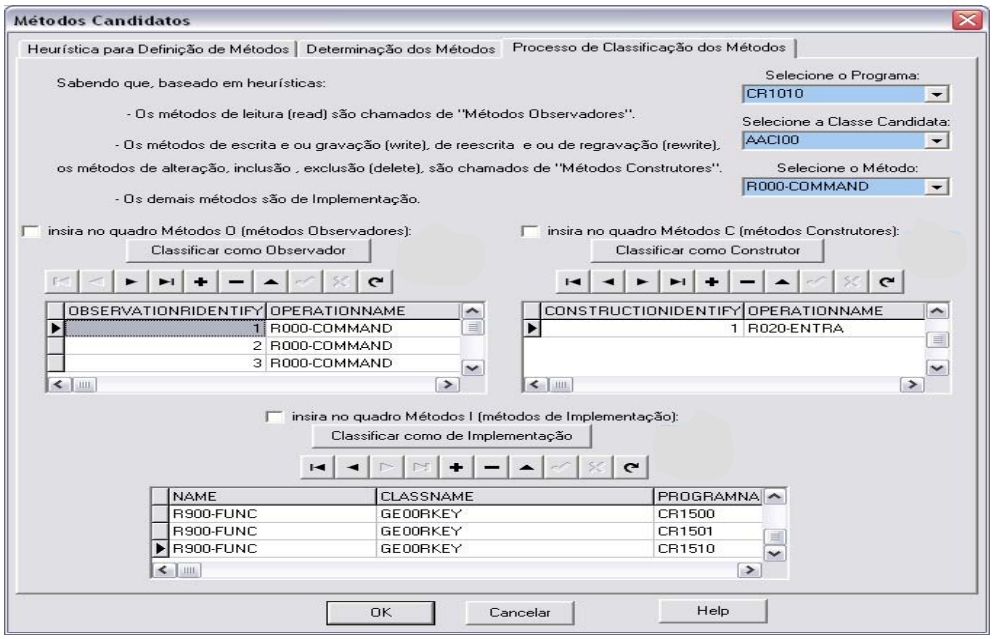
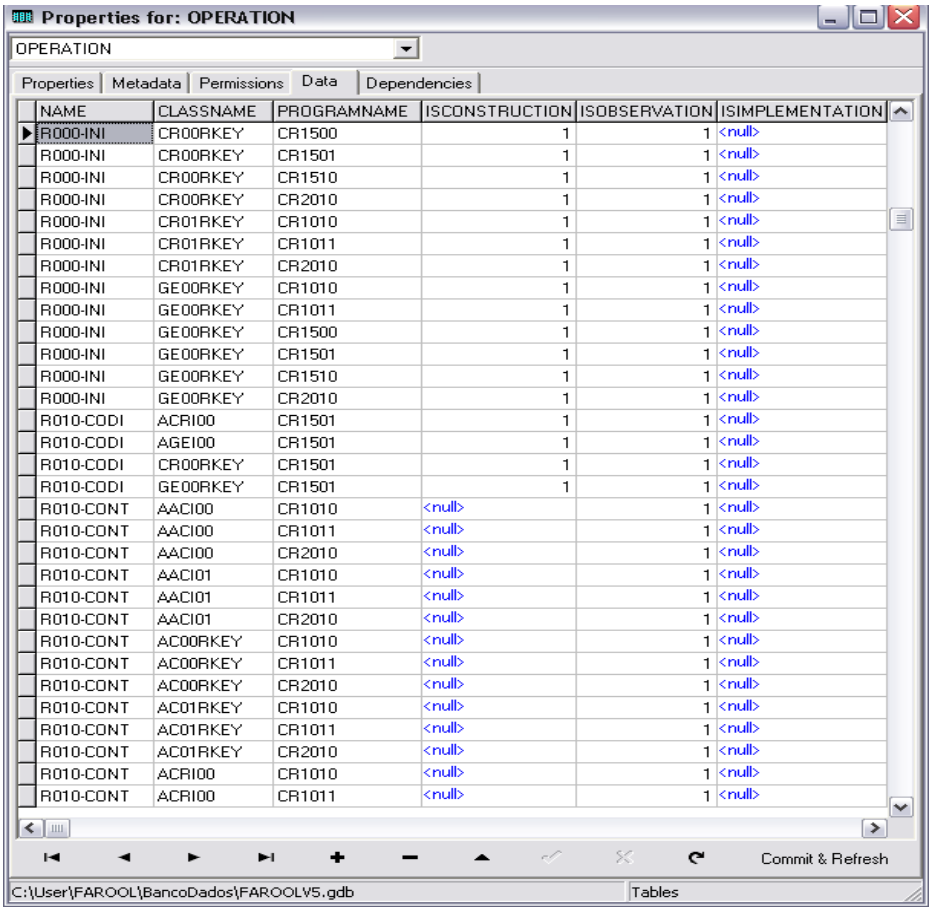


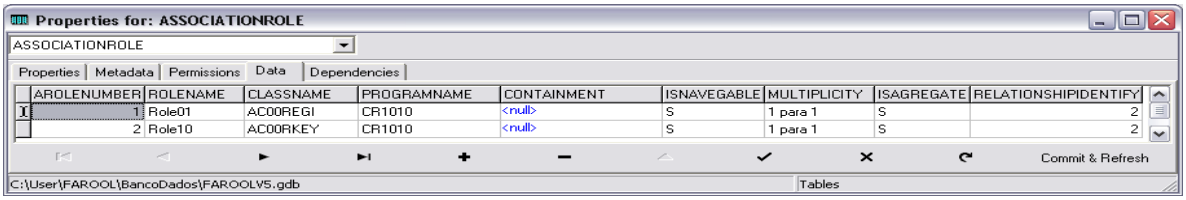
Figura 117 - Classificação dos Métodos em Observadores, Construtores e de Implementação

Tabela 6 - Alguns Métodos Candidatos do Sistema Controle de Recibos



NAME	CLASSNAME	PROGRAMNAME	ISCONSTRUCTION	ISOBSERVATION	ISIMPLEMENTATION
R000-INI	CR00RKEY	CR1500	1	1	<null>
R000-INI	CR00RKEY	CR1501	1	1	<null>
R000-INI	CR00RKEY	CR1510	1	1	<null>
R000-INI	CR00RKEY	CR2010	1	1	<null>
R000-INI	CR01RKEY	CR1010	1	1	<null>
R000-INI	CR01RKEY	CR1011	1	1	<null>
R000-INI	CR01RKEY	CR2010	1	1	<null>
R000-INI	GE00RKEY	CR1010	1	1	<null>
R000-INI	GE00RKEY	CR1011	1	1	<null>
R000-INI	GE00RKEY	CR1500	1	1	<null>
R000-INI	GE00RKEY	CR1501	1	1	<null>
R000-INI	GE00RKEY	CR1510	1	1	<null>
R000-INI	GE00RKEY	CR2010	1	1	<null>
R010-CODI	ACRI00	CR1501	1	1	<null>
R010-CODI	AGEI00	CR1501	1	1	<null>
R010-CODI	CR00RKEY	CR1501	1	1	<null>
R010-CODI	GE00RKEY	CR1501	1	1	<null>
R010-CONT	AACI00	CR1010	<null>	1	<null>
R010-CONT	AACI00	CR1011	<null>	1	<null>
R010-CONT	AACI00	CR2010	<null>	1	<null>
R010-CONT	AACI01	CR1010	<null>	1	<null>
R010-CONT	AACI01	CR1011	<null>	1	<null>
R010-CONT	AACI01	CR2010	<null>	1	<null>
R010-CONT	AC00RKEY	CR1010	<null>	1	<null>
R010-CONT	AC00RKEY	CR1011	<null>	1	<null>
R010-CONT	AC00RKEY	CR2010	<null>	1	<null>
R010-CONT	AC01RKEY	CR1010	<null>	1	<null>
R010-CONT	AC01RKEY	CR1011	<null>	1	<null>
R010-CONT	AC01RKEY	CR2010	<null>	1	<null>
R010-CONT	ACRI00	CR1010	<null>	1	<null>
R010-CONT	ACRI00	CR1011	<null>	1	<null>

4- Determinação dos Relacionamentos – Baseando-se em análises feitas no código fonte e nos conhecimentos obtidos com o levantamento das informações gerais do sistema realizado até o momento no processo de engenharia reversa, devem ser determinados novos relacionamentos, bem como seus papéis (*roles*), cardinalidade e navegabilidade. As telas oferecidas para essa determinação já foram expostas na seqüência em que elas aparecem nas Figuras 107 a 109. Os relacionamentos definidos nesta atividade são armazenados no banco FAROOL como mostra a Figura 118.



AROLENUMBER	ROLENAME	CLASSNAME	PROGRAMNAME	CONTAINMENT	ISNAVEGABLE	MULTIPLICITY	ISAGREGATE	RELATIONSHIPIDENTIFY
1	Role01	AC00REGI	CR1010	<null>	S	1 para 1	S	2
2	Role10	AC00RKEY	CR1010	<null>	S	1 para 1	S	2

Figura 118 - Relacionamentos definidos no sistema Controle de Recibos

Como exemplo de relacionamento entre classes são consideradas as classes AC00REGI e AC00KEY do sistema Controle de Recibos. Primeiramente, analisando o

código fonte do sistema legado em uso, AC00REGI e AC00KEY eram itens de grupo, isto é, AC00REGI era um dos atributos candidatos da classe AACI00 oriunda do FD AACI00, especificada conforme apresentado anteriormente quando da determinação das classes candidatas e AC00KEY era um item de grupo de AC00REGI. Os atributos AC00REGI e AC00KEY foram definidos como classes a partir da opção Criar Nova Classe, Figura 107, da determinação dos atributos candidatos. A partir dessa definição, foi determinado o relacionamento de associação entre essas novas classes, onde a cardinalidade é de 1 para 1 com base em heurísticas pré-determinadas e definida no campo Cardinalidade, Figura 111. Como a associação é navegável tanto partindo da classe todo AC00REGI como partindo da classe parte ACI00KEY, é definido S em IsNavegable? para as especificações da classe todo e parte, Figura 111.

A determinação dos relacionamentos foi realizada para todas as classes que mantinham algum tipo de relacionamento que pode ser de dependência, associação, interface e generalização, tanto os previstos com base em heurísticas e sugeridos pela FAROOL, quanto os definidos com os conhecimentos do próprio engenheiro de software. Alguns casos de associação são especiais e por isso tratados a parte pela FAROOL como se segue.

5- Encontrar Associações entre Classes – essas associações são oriundas de casos especiais de referências entre as chaves das classes, determinadas nas classes candidatas. Para o caso específico do sistema Controle de Recibos, primeiramente, é selecionado o programa, no caso, CR4010 que cuida da eliminação dos contratos inativos do sistema, a classe ABKI01 que descreve os contratos e sua chave BK01RkEY como pode ser visto na Figura 119. A partir dessa chave são verificadas as possíveis classes do sistema que a referenciam através da análise do código fonte dos demais programas do sistema, ou se há algum Item Elementar dessa chave em outras classes do sistema caso ela seja definida no código como um Item de Grupo. O código do programa CR4010 é visualizado no quadro (a) a partir da opção Visualizar Código do Programa e da seleção do mesmo na tela Abrir, Figura 119.

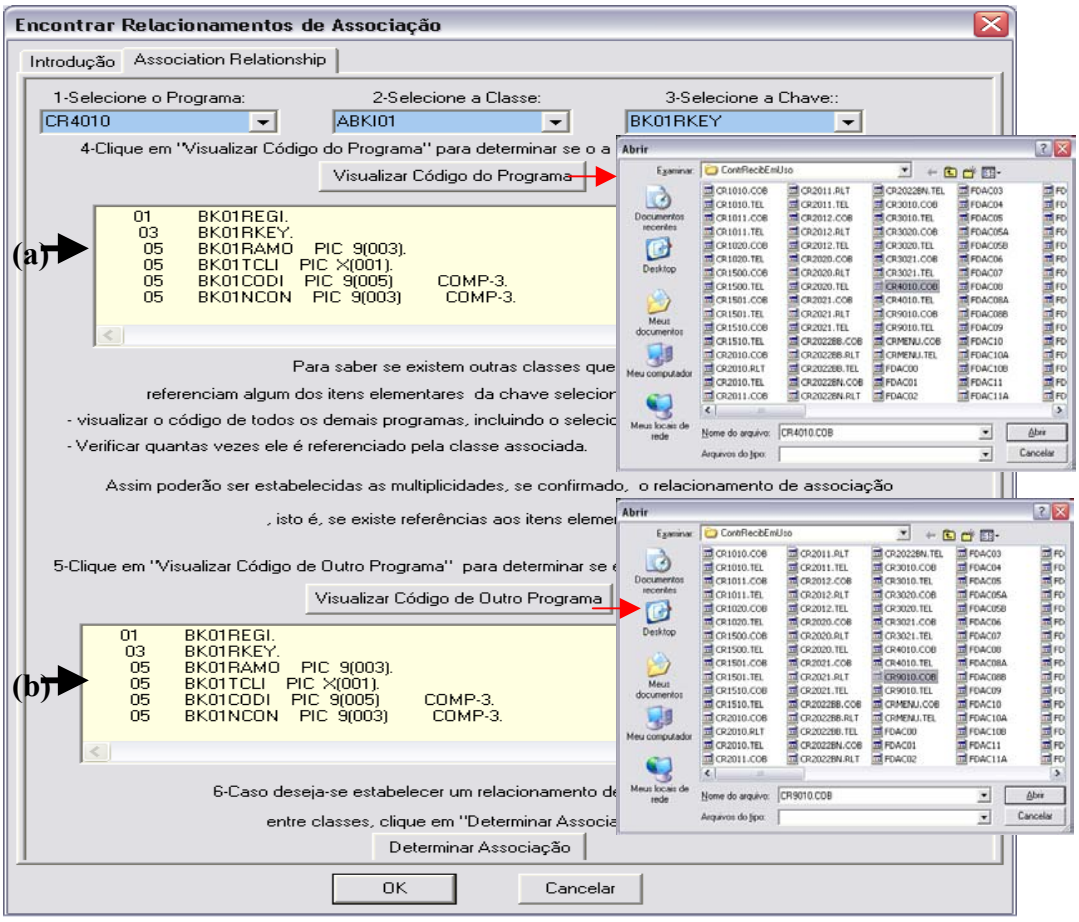


Figura 119 - Encontrando novas associações a partir das chaves das classes

Existem três itens elementares que definem a chave: BK01TCLI que define o tipo de cliente quando do momento da identificação do mesmo, BK01CCODI que é o código do cliente também usado para identificação do cliente e BK01NCON que contém o número do contrato feito com o cliente identificado, conforme analisado no código do programa exibido em (a) da Figura 119. Para saber se outros programas contêm classes que referenciam essa, deve-se analisar todo o código fonte dos demais programas do sistema. Esses programas podem ser escolhidos a partir das informações que são disponibilizadas pela seleção do botão Visualizar Código de Outro Programa e em seguida a escolha do programa desejado. O código é exibido no quadro da Figura 119(b). Como resultado da análise é verificado que somente o programa rotulado de CR9010, que converte os arquivos de contrato, continha uma classe, de mesmo nome da selecionada anteriormente, ABKI01, que também cuida da descrição dos contratos, que referenciava, não somente um, mas todos os itens elementares da chave. Os tipos e tamanhos dos itens elementares do programa

CR4010 foram comparados com os encontrados no programa CR9010. Reconhecida a igualdade desses, foi determinada a cardinalidade do relacionamento de associação, no caso 1 para 1 e, posteriormente, definida a associação utilizando a opção Definir Associação, conforme comentado anteriormente em 4-Determinação dos Relacionamentos.

Na Figura 120 é mostrado o modelo de classes do sistema em uso (MASA) obtido até o momento com a aplicação da FAROOL. O modelo de classes consta das classes, seus atributos e relacionamentos. Os métodos foram omitidos para simplificar a representação do modelo.

O sistema Controle de Recibos é entendido através dos programas, classes e breve descrição dos programas identificados no modelo oferecidos na Tabela 7, colunas: Programa, Classe e Descrição Geral, respectivamente. Muitos dos programas identificados possuem classes de mesmo nome e código, com a mesma funcionalidade, porém utilizados em contextos diferentes, ora no estabelecimento dos contratos, ora na emissão dos recibos, ora no cadastro dos clientes e fornecedores. Isso pode ser notado na Tabela 7 onde a classe AACRI02 existe para os programas CR2020, CR2020, CR2022BB e CR2022BN. As funcionalidades de cada classe são apresentadas na Tabela 8.

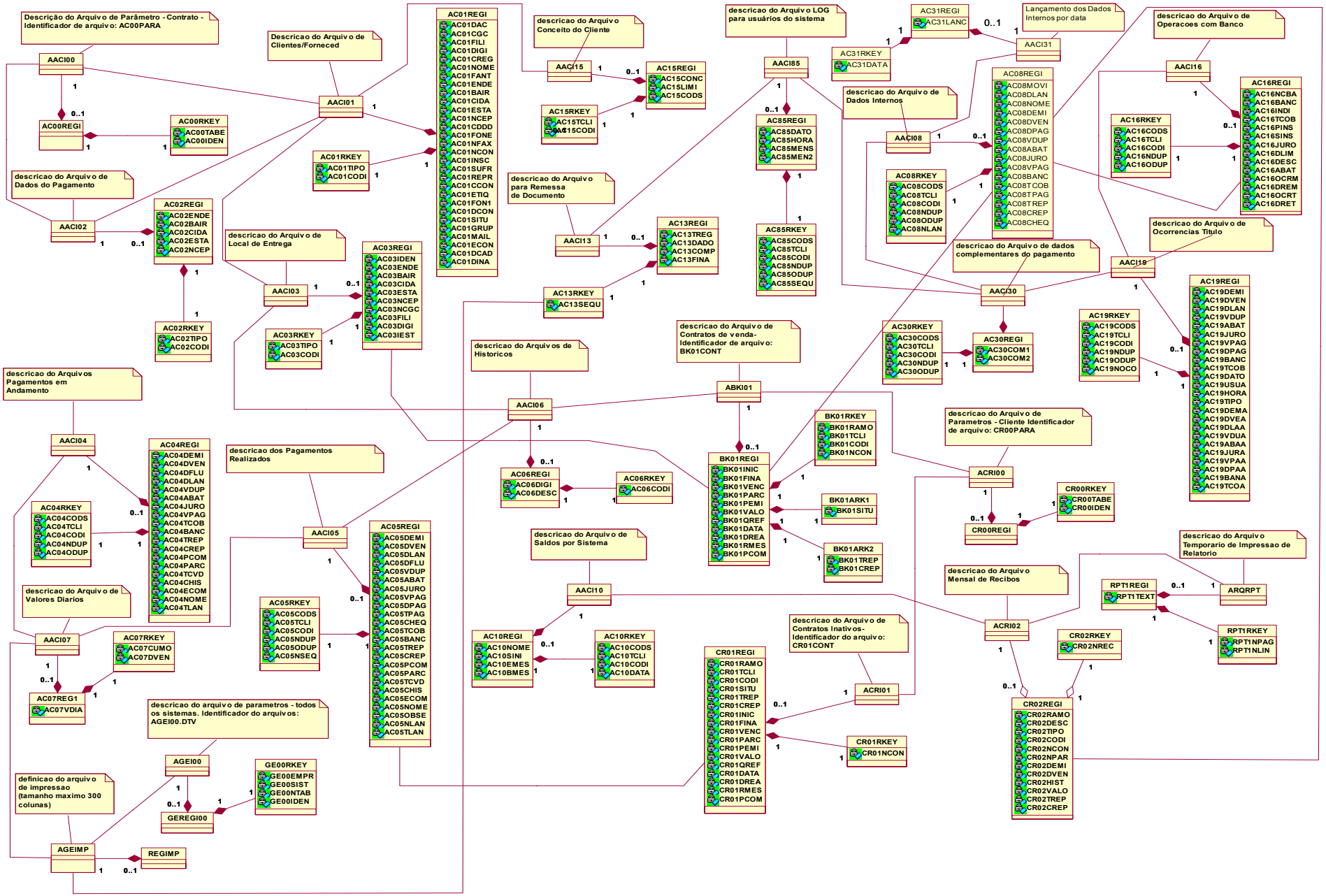

Descrição do Arquivo de Parâmetro - Contrato - Identificador de arquivo: AC00PARA
AACI00
AC00REGI
AC00RKEY
AC00TABE
AC00IDEN
Descricao do Arquivo de Clientes/Forneced
AACI01
AC01REGI
AC01DAC
AC01CGC
AC01FILI
AC01DIGI
AC01CREG
AC01NOME
AC01FANT
AC01ENDE
AC01BAIR
AC01CIDA
AC01ESTA
AC01NCEP
AC01CDDD
AC01FONE
AC01NFAX
AC01NCON
AC01INSC
AC01SUFR
AC01REPR
AC01CCON
AC01ETIQ
AC01FON1
AC01DCON
AC01SITU
AC01GRUP
AC01MAIL
AC01ECON
AC01DCAD
AC01DINA
AC01RKEY
AC01TIPO
AC01CODI
descricao do Arquivo Conceito do Cliente
AACI15
AC15REGI
AC15CONC
AC15LIMI
AC15CODS
AC15RKEY
AC15TCLI
AC15CODI
descricao do Arquivo de Dados do Pagamento
AACI02
AC02REGI
AC02ENDE
AC02BAIR
AC02CIDA
AC02ESTA
AC02NCEP
AC02RKEY
AC02TIPO
AC02CODI
descricao do Arquivo de Local de Entrega
AACI03
AC03REGI
AC03IDEN
AC03ENDE
AC03BAIR
AC03CIDA
AC03ESTA
AC03NCEP
AC03NCGC
AC03FILI
AC03DIGI
AC03IEST
AC03RKEY
AC03TIPO
AC03CODI
descricao do Arquivo para Remessa de Documento
AACI13
AC13REGI
AC13TREG
AC13DADO
AC13COMP
AC13FINA
AC13RKEY
AC13SEQU
descricao do Arquivo LOG para usuários do sistema
AACI85
AC85REGI
AC85DATO
AC85HORA
AC85MENS
AC85MEN2
AC85RKEY
AC85CODS
AC85TCLI
AC85CODI
AC85NDUP
AC85ODUP
AC85SEQU
AC31REGI
AC31LANC
AC31RKEY
AC31DATA
AACI31
Lançamento dos Dados Internos por data
descricao do Arquivo de Dados Internos
AACI08
AC08REGI
AC08MOVI
AC08DLAN
AC08NOME
AC08DEMI
AC08DVEN
AC08DPAG
AC08VDUP
AC08ABAT
AC08JURO
AC08VPAG
AC08BANC
AC08TCOB
AC08TPAG
AC08TREP
AC08CREP
AC08CHEQ
AC08RKEY
AC08CODS
AC08TCLI
AC08CODI
AC08NDUP
AC08ODUP
AC08NLAN
descricao do Arquivo de Operacoes com Banco
AACI16
AC16REGI
AC16NCBA
AC16BANC
AC16INDI
AC16TCOB
AC16PINS
AC16SINS
AC16JURO
AC16DLIM
AC16DESC
AC16ABAT
AC16OCRM
AC16DREM
AC16OCRT
AC16DRET
AC16RKEY
AC16CODS
AC16TCLI
AC16CODI
AC16NDUP
AC16ODUP
descricao do Arquivo de Ocorrencias Titulo
descricao do Arquivo de dados complementares do pagamento
AACI19
AC19REGI
AC19DEMI
AC19DVEN
AC19DLAN
AC19VDUP
AC19ABAT
AC19JURO
AC19VPAG
AC19DPAG
AC19BANC
AC19TCOB
AC19DATO
AC19USUA
AC19HORA
AC19TIPO
AC19DEMA
AC19DVEA
AC19DLAA
AC19VDUA
AC19ABAA
AC19JURA
AC19VPAA
AC19DPAA
AC19BANA
AC19TCOA
AC19RKEY
AC19CODS
AC19TCLI
AC19CODI
AC19NDUP
AC19ODUP
AC19NOCO
AACI30
AC30REGI
AC30COM1
AC30COM2
AC30RKEY
AC30CODS
AC30TCLI
AC30CODI
AC30NDUP
AC30ODUP
descricao do Arquivo de Contratos de venda- Identificador de arquivo: BK01CONT
ABKI01
BK01REGI
BK01INIC
BK01FINA
BK01VENC
BK01PARC
BK01PEMI
BK01VALO
BK01QREF
BK01DATA
BK01DREA
BK01RMES
BK01PCOM
BK01RKEY
BK01RAMO
BK01TCLI
BK01CODI
BK01NCON
BK01ARK1
BK01SITU
BK01ARK2
BK01TREP
BK01CREP
descricao do Arquivo de Historicos
AACI06
AC06REGI
AC06DIGI
AC06DESC
AC06RKEY
AC06CODI
descricao do Arquivo de Parametros - Cliente Identificador de arquivo: CR00PARA
ACRI00
CR00REGI
CR00RKEY
CR00TABE
CR00IDEN
descricao dos Pagamentos Realizados
AACI05
AC05REGI
AC05DEMI
AC05DVEN
AC05DLAN
AC05DFLU
AC05VDUP
AC05ABAT
AC05JURO
AC05VPAG
AC05DPAG
AC05TPAG
AC05CHEQ
AC05TCOB
AC05BANC
AC05TREP
AC05CREP
AC05PCOM
AC05PARC
AC05TCVD
AC05CHIS
AC05ECOM
AC05NOME
AC05OBSE
AC05NLAN
AC05TLAN
AC05RKEY
AC05CODS
AC05TCLI
AC05CODI
AC05NDUP
AC05ODUP
AC05NSEQ
descricao do Arquivo de Saldos por Sistema
AACI10
AC10REGI
AC10NOME
AC10SINI
AC10EMES
AC10BMES
AC10RKEY
AC10CODS
AC10TCLI
AC10CODI
AC10DATA
descricao do Arquivo de Contratos Inativos- Identificador do arquivo: CR01CONT
ACRI01
CR01REGI
CR01RAMO
CR01TCLI
CR01CODI
CR01SITU
CR01TREP
CR01CREP
CR01INIC
CR01FINA
CR01VENC
CR01PARC
CR01PEMI
CR01VALO
CR01QREF
CR01DATA
CR01DREA
CR01RMES
CR01PCOM
CR01RKEY
CR01NCON
descricao do Arquivo Mensal de Recibos
ACRI02
CR02REGI
CR02RAMO
CR02DESC
CR02TIPO
CR02CODI
CR02NCON
CR02NPAR
CR02DEMI
CR02DVEN
CR02HIST
CR02VALO
CR02TREP
CR02CREP
CR02RKEY
CR02NREC
descricao do Arquivo Temporario de Impressao de Relatorio
ARQRPT
RPT1REGI
RPT1TEXT
RPT1RKEY
RPT1NPAG
RPT1NLIN
descricao do Arquivos Pagamentos em Andamento
AACI04
AC04REGI
AC04DEMI
AC04DVEN
AC04DFLU
AC04DLAN
AC04VDUP
AC04ABAT
AC04JURO
AC04VPAG
AC04TCOB
AC04BANC
AC04TREP
AC04CREP
AC04PCOM
AC04PARC
AC04TCVD
AC04CHIS
AC04ECOM
AC04NOME
AC04TLAN
AC04RKEY
AC04CODS
AC04TCLI
AC04CODI
AC04NDUP
AC04ODUP
descricao do Arquivo de Valores Diarios
AACI07
AC07REGI
AC07VDIA
AC07RKEY
AC07CUMO
AC07DVEN
descricao do arquivo de parametros - todos os sistemas. Identificador do arquivos: AGEI00.DTV
AGEI00
GEREGI00
GE00RKEY
GE00EMPR
GE00SIST
GE00NTAB
GE00IDEN
definicao do arquivo de impressao (tamanho maximo 300 colunas)
AGEIMP
REGIMP

Figura 120 - Modelo de Análise do Sistema Controle de Recibos (MASA)

Tabela 7 - Classes do MASA para cada programa do sistema Controle de Recibos

Programa	Classe	Descrição Geral
CR1010	**AACI00; AACI01; ACRI00; ACRI01; AGEI00.**	**Manutenção do Cadastro de Contratos.**
CR1011	**AACI00; AACI01; ACRI00; ACRI01; AGEI00.**	**Manutenção na Situação do Cliente.**
CR1020	**AACI00; AACI01; ACRI00; ACRI01; ACRI02; AGEI00.**	**Manutenção no arquivo mensal de contratos.**
CR1500	**AACI00; ACRI00; AGEI00.**	**Manutenção nos parâmetros do sistema.**
CR1501	**ACRI00; AGEI00.**	**Alteração da Data do Sistema.**
CR1510	**ACRI00; AGEI00.**	**Manutenção da Tabela dos Ramos de Atividade.**
CR2010	**AACI00; AACI01; ACRI00; ACRI01; AGEI00.**	**Relatório do Contrato.**
CR2011	**AACI00; AACI01; ACRI00; ACRI01; AGEI00.**	**Relatório do Contrato por Representante/Cliente.**
CR2012	**AACI00; AACI01; ACRI00; ACRI01; AGEI00.**	**Relatório por Vencimento de Contrato.**
CR2020	**AACI00; AACI01; ACRI00; ACRI01; ACRI02; AGEI00.**	**Relatório da Movimentação Mensal.**
CR2021	**AACI00; AACI01; ACRI00; ACRI01; ACRI02; AGEI00.**	**Emite Recibos Mensais.**
CR2022BB	**AACI00; AACI01; AACI02; ACRI00; ACRI01; ACRI02; AGEI00.**	**Emite Aviso de Cobrança (Banco do Brasil).**
CR2022BN	**AACI00; AACI01; AACI02; ACRI00; ACRI01; ACRI02; AGEI00.**	**Emite Aviso de Cobrança (Banco Banespa).**
CR3010	**ACRI00; ACRI01; AGEI00.**	**Reajustes dos Valores do Contrato.**
CR3020	**AACI00; AACI01; ACRI00; ACRI01; ACRI02; AGEI00.**	**Emissão dos Recibos.**
CR3021	**AACI00; AACI01; AACI02; AACI03; AACI04; AACI05; AACI06; AACI07; AACI08; AACI10; AACI13; AACI15; AACI16; AACI19; AACI30; AACI31; AACI40; AACI85; ACRI00; ACRI01; ACRI02; AGEI00.**	**Integração com Contas a Receber.**
CR4010	**ABKI01; ACRI00; ACRI01; AGE00.**	**Eliminação dos Contratos Inativos.**
CR9010	**ABKI01; ACRI00; ACRI01; AGE00.**	**Conversão de Contratos.**
CRmenu	**AGEI00.**	Seletor de Programas.

Tabela 8 - Funcionalidades das Classes identificadas no MASA

Classe	Funcionalidade
AACI00	Classe responsável pela descrição dos arquivos de parâmetros – Cliente.
AACI01	Classe responsável pelo Cadastro dos Clientes e Fornecedores da empresa.
AACI02	Classe com os dados do pagamento.
AACI03	Classe com o local de entrega dos produtos encomendados nos contratos.
AACI04	Classe com a descrição dos pagamentos em andamento.
AACI05	Classe com os pagamentos realizados.
AACI06	Classe que armazena o histórico com os dados internos do sistema para os contratos realizados.
AACI07	Classe com os valores referentes aos contratos realizados no dia.
AACI08	Classe que armazena os dados internos do sistema Controle de Recibos, como movimentação da empresa, banco, cheques que recebidos, etc..
AACI10	Classe responsável pelo armazenamento e controle dos saldos da empresa.
AACI13	Classe que contém detalhes sobre os documentos a serem remetidos.
AACI15	Classe que contém o conceito dos clientes que servem para identificá-los no sistema.
AACI16	Classe que guarda as principais operações realizadas no banco para pagamento, consulta e recebimento na conta.
AACI19	Classe responsável por armazenar o número de ocorrências de juros, cheques, pagamentos, fechamento de contratos, abatimento, usuários, etc..
AACI30	Classe responsável por armazenar os dados complementares dos pagamentos caso necessário.
AACI31	Classe responsável pelo lançamento dos dados internos por data.
AACI85	Classe responsável pelo *login* do usuário do sistema.
ACRI00	Classe responsável pela descrição dos arquivos de parâmetros – Contrato.
ACRI01	Classe que armazena a descrição dos arquivos de contrato inativos.
ACRI02	Classe responsável pela emissão do recibo mensal.
ABKI01	Classe que armazena a descrição dos arquivos de contrato de venda de produtos.
AGEI00	Classe responsável pela impressão de arquivos.

7.3.3 Fase de Elaboração do MAS

O banco de dados da FAROOL armazena informações que devem ser analisadas para que um modelo completamente orientado a objetos seja obtido. Assim, as atividades que devem ser realizadas para que esse modelo seja obtido são apresentadas no decorrer desta seção.

1- Remoção das Classes que representam datas – Primeiramente deve-se considerar os possíveis casos candidatos à remoção mostrados no quadro (a) da Figura 121, relativas ao sistema legado em uso.

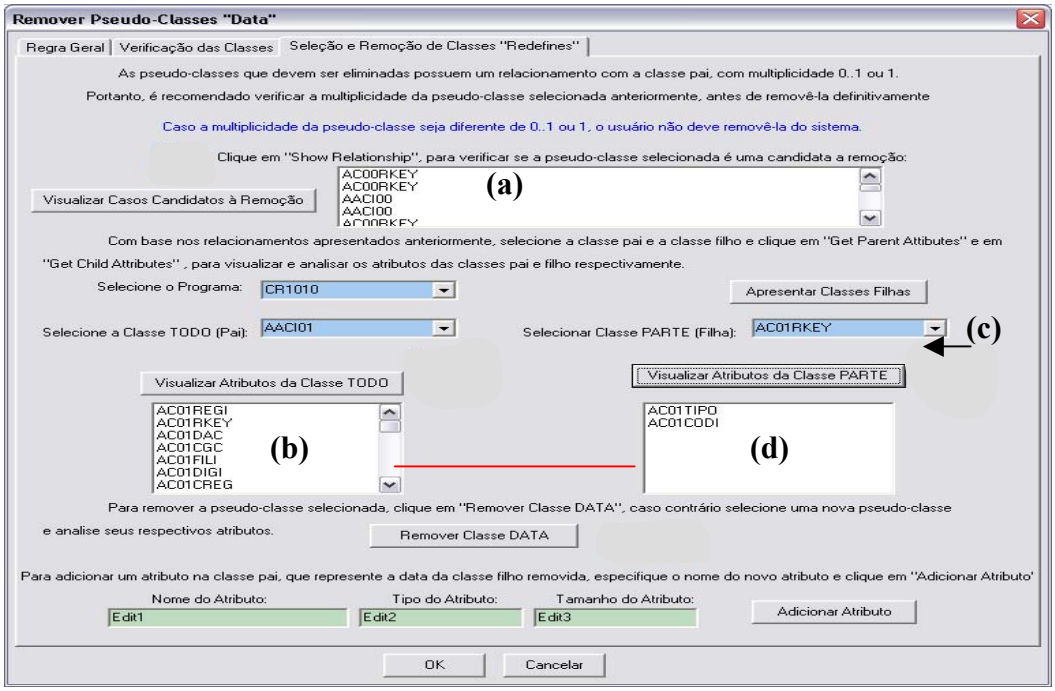


Figura 121 - Removendo classes DATA

Para cada caso candidato à remoção foram selecionados o programa e a classe todo do relacionamento correspondentes a ele, para a visualização de seus atributos (Visualizar Atributos da Classe TODO), quadro (b) da Figura 121. Uma vez selecionada a classe todo, FAROOL disponibiliza todas as classes parte(s) associadas a ela, no *combo* (c), através da opção Apresentar Classes Filhas. Seus atributos podem ser visualizados em (d) através da opção Visualizar Atributos da Classe PARTE. Esses atributos são então

comparados com os da classe todo selecionada anteriormente para verificar se esses representam data, no formato dia, mês e ano como itens elementares. Quando isso ocorre, deve-se optar por Remover Classe DATA, ou seja a classe data do MASA é retirada do banco e, conseqüentemente, do modelo de análise do sistema. O sistema Controle de Recibos em uso não apresenta esse tipo de dados, portanto, nenhuma classe do modelo foi removida nessa atividade.

2- Mudança dos Nomes das Classes – Para exemplificar o processo de mudança dos nomes das classes do sistema Controle de Recibos, considere a Figura 122, com o código do programa CR1010, especificamente o trecho de código referente à classe AACI00. Ao se analisar o sistema legado, pode-se observar que esse FD trata da descrição de parâmetros, assim o nome escolhido para a classe foi ParametroDoContrato como mostra a tela da Figura 123.

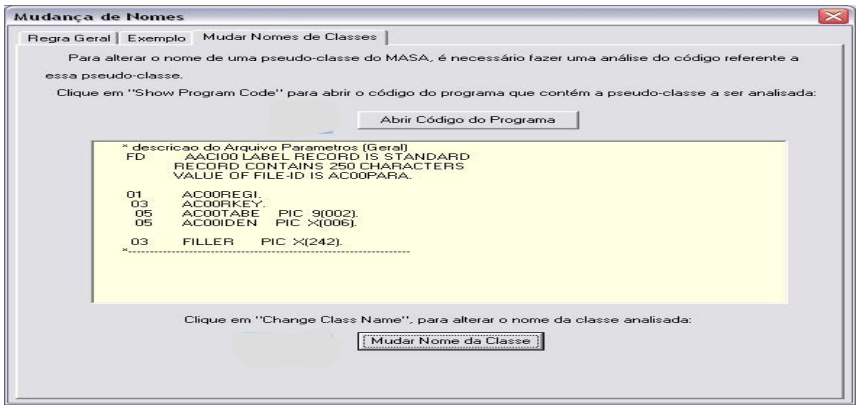


Figura 122 - Trecho de código do FD AACI00

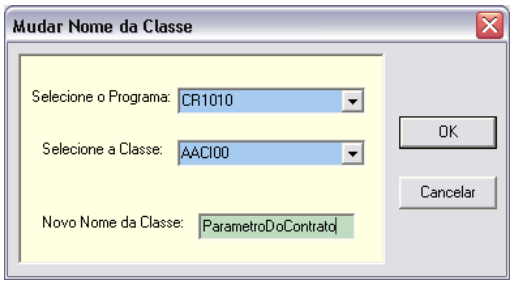


Figura 123 - Mudança do Nome da Classe AACI00

A tabela 9 apresenta alguns exemplos de mudança de nomes de classes que ocorreram. A coluna NomeMASA, refere-se ao nome existente no sistema legado e

NomeMAS, ao escolhido para o modelo orientado a objetos. A descrição da funcionalidade exercida pela classe é mostrada no campo Funcionalidade da Tabela 8 apresentada anteriormente.

Tabela 9 - Mudança de alguns dos nomes das classes do sistema Controle de Recibos

NomeMASA	NomeMAS
AACI00	ParametroDoContrato
AACI01	CadastroCliente_Fornecedor
AACI02	DadoDePagamento
AACI03	LocalDeEntrega
AACI04	PagamentoEmAndamento
AACI05	PagamentoRealizado
AACI06	Historico
AACI07	ValorDiario
AACI08	DadoInterno
AACI10	Saldo
AACI13	PagamentoEnviado
AACI15	ConceitoDoCliente
AACI16	OperacaoDoBanco
AACI19	OcorrenciaTitulo
AACI30	DadosComplementares
AACI31	DadoInternoEmLancamento
AACI85	LoginDoUsuario
ACRI00	ParametroDoCliente
ACRI01	ContratoInativo
ACRI02	ReciboMensal
ABKI01	ContratoDeVendaDeProduto
AGEI00	s
	DefinicaoDaImpressora

3- Mudança dos Nomes dos Atributos – Continuando com o exemplo anterior, após a mudança de nome da classe, observou-se, através de análises do código fonte do sistema, que os atributos estão rotulados com nomes não significativos e, portanto, devem ser renomeados utilizando os recursos da FAROOL como mostram as telas da Figura 124.

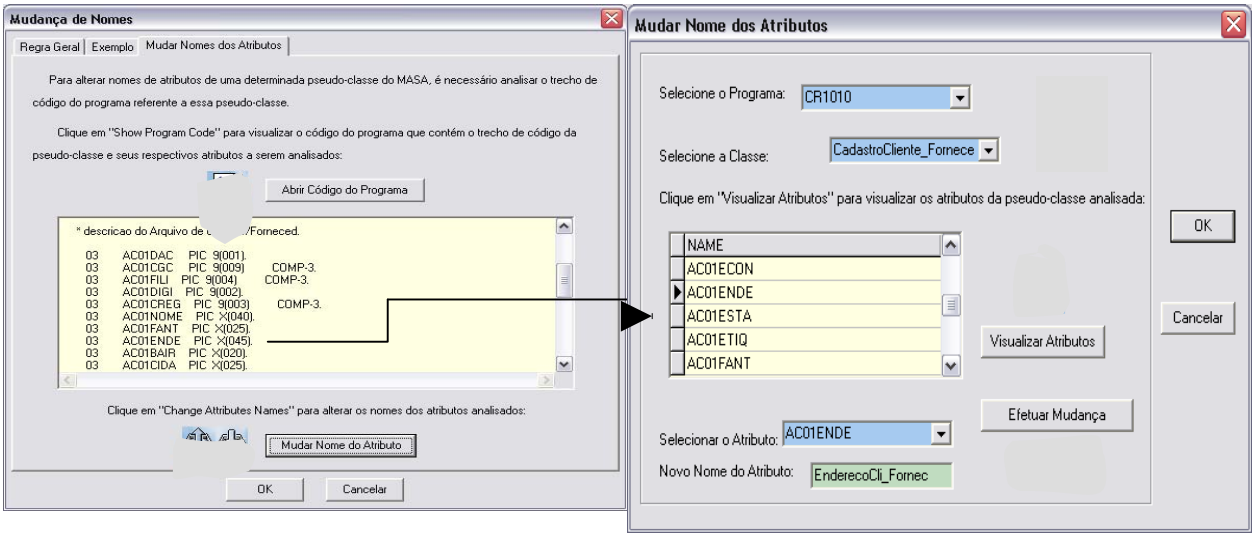


Figura 124 - Mudança do Nome do Atributo AC01ENDE da classe CadastroCliente_Fornecedor

O atributo AC01ENDE é renomeado para EnderecoCli_Fornec uma vez que o arquivo que o contém cuida dos arquivos de cliente e fornecedores cadastrados. Esse procedimento de alteração de nomes foi aplicado a todos os atributos encontrados no sistema. As Tabelas 10 e 11 mostram a mudança dos nomes dos atributos para as classes Saldo e CadastroCliente_Fornecedor, respectivamente. Na coluna NomeMASA, os nomes ainda não modificados e em NomeMAS, os nomes alterados para o modelo orientado a objeto.

Tabela 10 - Atributos renomeados do MASA para o MAS (Classe Saldo)

NomeMASA	NomeMAS
AC10NOME	NomeDaConta
AC10SINI	SaldoInicial
AC10EMES	MesDeEmissao
AC10BMES	BaseDoMes
AC10CODS	CodigoDoSaldo
AC10TCLI	TipoDeCliente
AC10CCODI	CodigoDoCliente
AC10DATA	DataDeEmissaoDoSaldo

OBS: Os atributos AC10CODS, AC10TCLI, AC10CCODI e AC10DATA são oriundos da classe AC10RKEY que, no MASA, mantém um relacionamento de 1 para 1 com a classe AACI10, agora denominada Saldo.

Tabela 11 - Atributos renomeados do MASA para o MAS (Classe CadastroCliente-Fornecedor)

NomeMASA	NomeMAS
AC01CGC	CGC
AC01FILI	Filiacao
AC01DIGI	DigitoIdentificador
AC01CREG	Identidade
AC01NOME	Nome
AC01FANT	FornecedorAnterior
AC01ENDE	EnderecoCli_Fornec
AC01BAIR	Bairro
AC01CIDA	Cidade
AC01ESTA	Estado
AC01NCEP	CEP
AC01CDDD	DDD
AC01FONE	Telefone
AC01NFAX	FAX
AC01NCON	NumerpContrato
AC01INSC	Inscrição
AC01CCON	CodigoDoContrato
AC01DCON	DataDoContrato
AC01SITU	SituacaoDoContrato
AC01GRUP	Grupo
AC01FON1	TelefoneDeContato
AC01MAIL	E_mail
AC01ECON	EmissaoDoContrato
AC01DCAD	NumeroDoCadastro
AC01DINA	DataInicial
AC01TIPO	TipoDoCliente_Fornecedor
AC01CODI	CodigoDoCliente_Fornecedor

OBS: Os atributos AC01TIPO e AC01CODI são oriundos da classe AC01RKEY que, no MASA, mantém um relacionamento de 1 para 1 com a classe AACI01, agora denominada CadastroCliente_Fornecedor.

4- Refinamento dos Métodos – Continuando o processo de Elaboração do MAS, os métodos classificados como construtores e observadores, conforme apresentado na tela da Figura 125, devem ser renomeados para mnemônicos mais significativos, como mostrado na tela da Figura 124 para o caso do sistema Controle de Recibos.

O programa CR1010, a classe ParametroDoContrato e o método R030-LER foram selecionados e conforme mostrado em (a) da Figura 125, trata-se de um método observador. Analisando o código exibido em (b), é observado que o método tem como única finalidade a leitura dos arquivos de contrato da empresa. Assim, o nome sugerido para ele é R030-LER foi LerContrato().

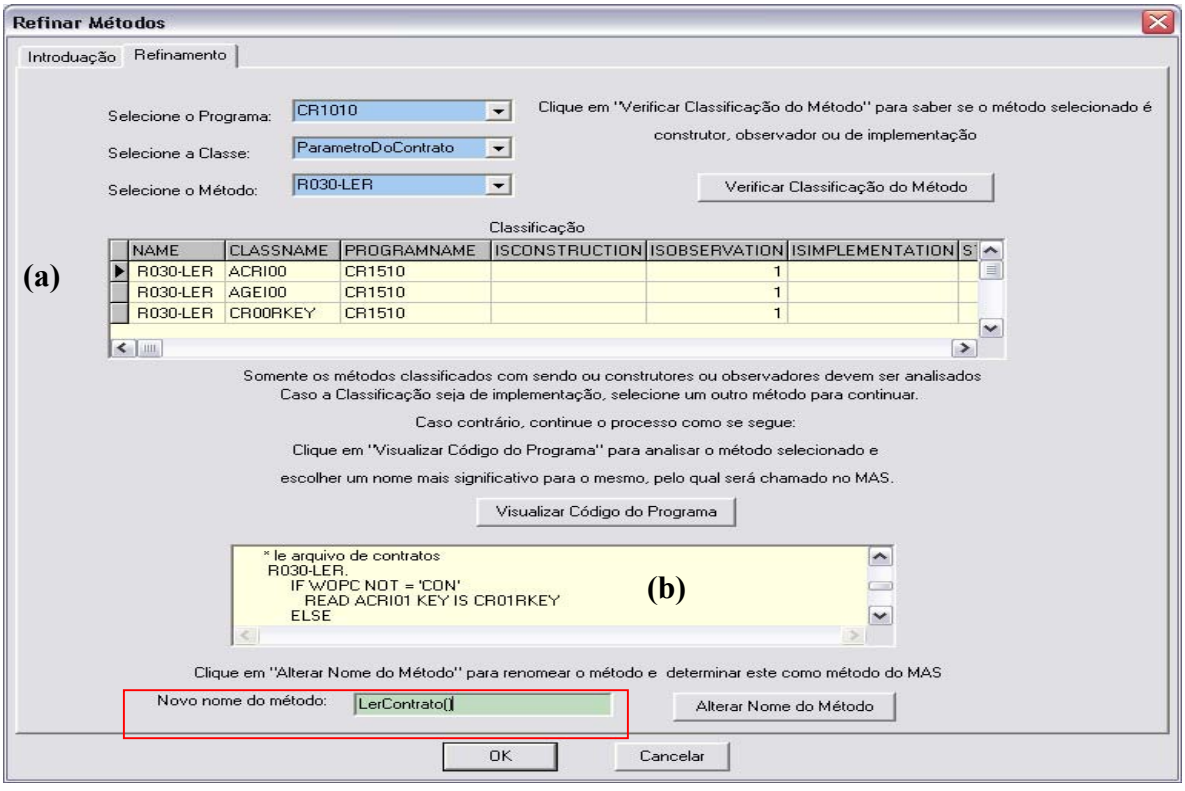


Figura 125 - Mudança do Nome do Atributo AC00REG da classe ACI00

A Tabela 12 mostra na primeira coluna os procedimentos (pseudo-métodos) do MASA, na segunda, os nomes a eles atribuídos no MAS e na terceira coluna, a descrição de sua funcionalidade para a classe ParametroDoContrato.

Tabela 12 - Métodos renomeados do MASA para o MAS na classe ParametroDoContrato

NomeMASA	NomeMAS	Funcionalidade

NomeMASA	NomeMAS	Funcionalidade
R000-COMMAND	VerificarParametroPrograma()	Verifica se os parâmetros do programa estão de acordo para uso do mesmo.
R000-INI	InicializarConfiguracao()	Inicializa corretamente o sistema com as cores, fontes, telas e demais configurações.
R010-INI	ChecarErroEmpresa()	verifica se a empresa existe através de cálculos de cifras e códigos das empresas e qual é a data do sistema operacional.
R020-SENHA	VerificarCodigoSenhaUsuario()	Consite a senha do usuário do sistema para permitir o uso do mesmo.
R020-PROG	VerificarDisponibilidadeDoPrograma()	Verifica se o programa desejado pode ser disponibilizado ou se está em manutenção.
R010-CONT	FecharArquivo()	Fecha o arquivo que controla a permissão do uso do sistema e é responsável por carregar as suas configurações.
R020-OPC	AceitarOpcao()	Verifica se o código da opção desejada pelo usuário é conhecido.
R020-ENTRA	ExecutarOpcao()	Caso a opção seja aceita e esteja disponível, executa a opção desejada.
R020-SENH2	VerificarDisponibilidadeDaOpcao()	Verifica a disponibilidade da opção.
R030-ABR00	AbrirArquivoCliente()	Abre o arquivo dos clientes para consulta e identificação dos mesmos.
R030-ABR01	AbrirArquivoProduto()	Abre o arquivo com os produtos que estão à venda na empresa.
R030-CHAVE	AceitarChaveDoArquivoContrato()	Verifica se o número digitado é conhecido comparando-o com a chave do contrato.
(*) R030-CODI	VerificarCodigoContrato()	Caso a chave exista, verifica se o código do contrato é válido,

NomeMASA	NomeMAS	Funcionalidade
		retornando o contrato.
R030-LER	LerContrato()	Permite a leitura do contrato já reconhecido.
R030-TELA	MostrarContratoNaTela()	Exibe o contrato na tela para o usuário do sistema.
R620-ENTRA	ConsistirContratoNaTela()	Consistir contrato exibido na tela e alterado pelo usuário.
R628-CONF	AceitarConfirmacao()	Aceita a confirmação de alterações no contrato exibido.
R629-SAI	SairDoArquivoDeContrato()	Sai do arquivo de contratos após alterações.
R700-AC01	MostrarNomeRepresentante()	Exibe os nomes dos representantes para seleção.
R740-GRAVA	GravarArquivo()	Grava os arquivos desejados.
R740-LER	LerParametroCadastrado()	Faz a leitura dos parâmetros de cadastro dos clientes e fornecedores.
R740-SAI	Sair()	Permite a saída do arquivos de clientes e fornecedores.
ERRO	EsperarTeclaENTER()	Aguarda o uso da tecla ENTER em caso de erros.
R700-E	MostrarErro()	Exibe o erro cometido após a seleção da tecla ENTER.
R700-STAT	RedefinirErro()	Redefini o erro caso o usuário assim o deseje.
R800-FUNC	ControlarFuncaoViaTecla()	Controla as opções via uso do teclado como comunicação.
R900-FECHA	FecharTodosOsArquivos()	Fecha todos os arquivos em uso.
R900-FIM	FinalizarOperacao()	Finaliza a operação realizada.

* Continuação da Tabela 12

5- Determinação dos Relacionamentos baseados em Casos de Redefinição – Para o sistema Controle de Recibos, foi analisado, primeiramente, o caso de redefinição que determinada um relacionamento de associação entre classes cuja cardinalidade é de 0..1 para classe parte. Para cada programa selecionado em (a) na Figura 126, no caso CR9010, são analisados se existem relacionamentos entre classes candidatas à associação com a cardinalidade 0..1 para a classe parte.

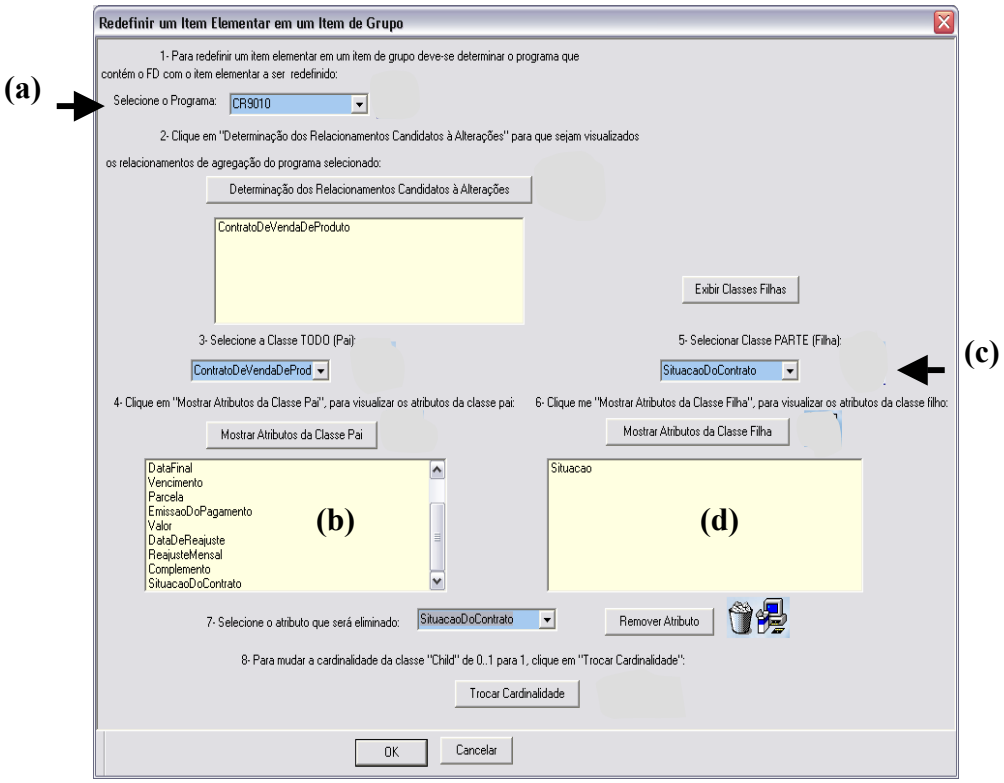


Figura 126 - Encontrando Relacionamentos - caso de redefinição 1

Existe apenas um caso candidato à alteração de cardinalidade para o programa selecionado. Assim, a classe todo ContratoDeVendaDeProdutos é selecionada para análise dos atributos exibidos em (b). Com base na classe todo, são mostradas todas as classes parte(s) que mantém esse caso particular de associação, no caso apenas a classe SituacaoDoContrato, no *combo* (c) e seus atributos são exibidos em (d). Os atributos da classe todo disponíveis em (b) são comparados com os da classe parte selecionada, exibidos em (c). Como existe redundância entre os atributos, isto é, a classe todo ContratoDeVendaDeProdutos possui um atributo de mesmo nome da classe parte SituacaoDoContrato, desempenhando a mesma funcionalidade; o atributo da classe todo,

SituacaoDoContrato, é eliminado e a cardinalidade do relacionamento em relação à parte é mudada de 0.1 para 1. Essa alteração é realizada selecionando-se o botão Trocar a Cardinalidade. Esse procedimento foi aplicado a todos os atributo redundantes nas classes todo(s) e cuja cardinalidade em relação as classes parte(s) é de 0..1.

O segundo caso de redefinição, que são encontrados os relacionamentos de generalização baseados em *redefines*, não foi identificado na análise do sistema Controle de Recibos. Quando selecionando o botão Selecione o Programa, Figura 127 (a), a mesma mensagem de não existência desse tido de relacionamento no sistema analisado, aparece na janela (b) dessa figura.

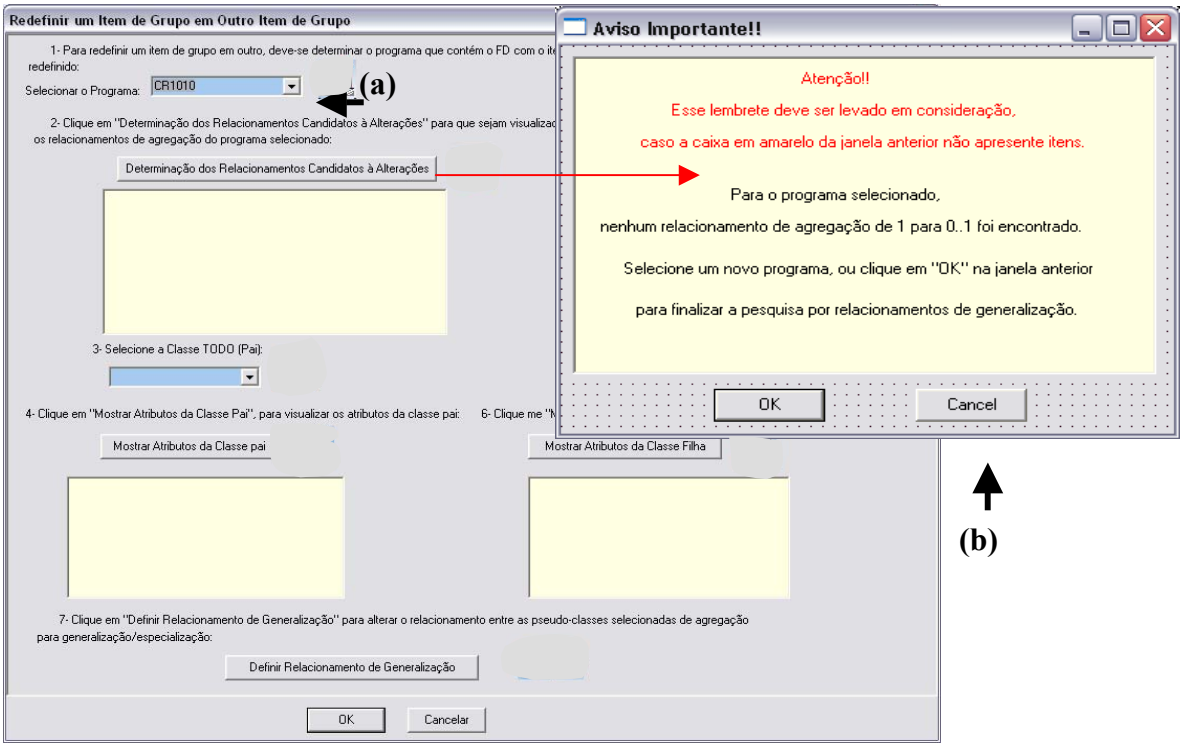


Figura 127 - Notificação de que não foram encontrados relacionamentos – caso de redefinição 2

Após esse processo, o engenheiro de software consegue elaborar o modelo de análise do sistema (MAS) conforme ilustrada na Figura 128, são omitidos os métodos para permitir legibilidade. Salienta-se que esse modelo é elaborado utilizando-se uma ferramenta, tipo *Rational Rose*® (Rose, 2001) ou *MVCase*® (Lucredio, 2001) manualmente, com as informações armazenadas no banco de dados.

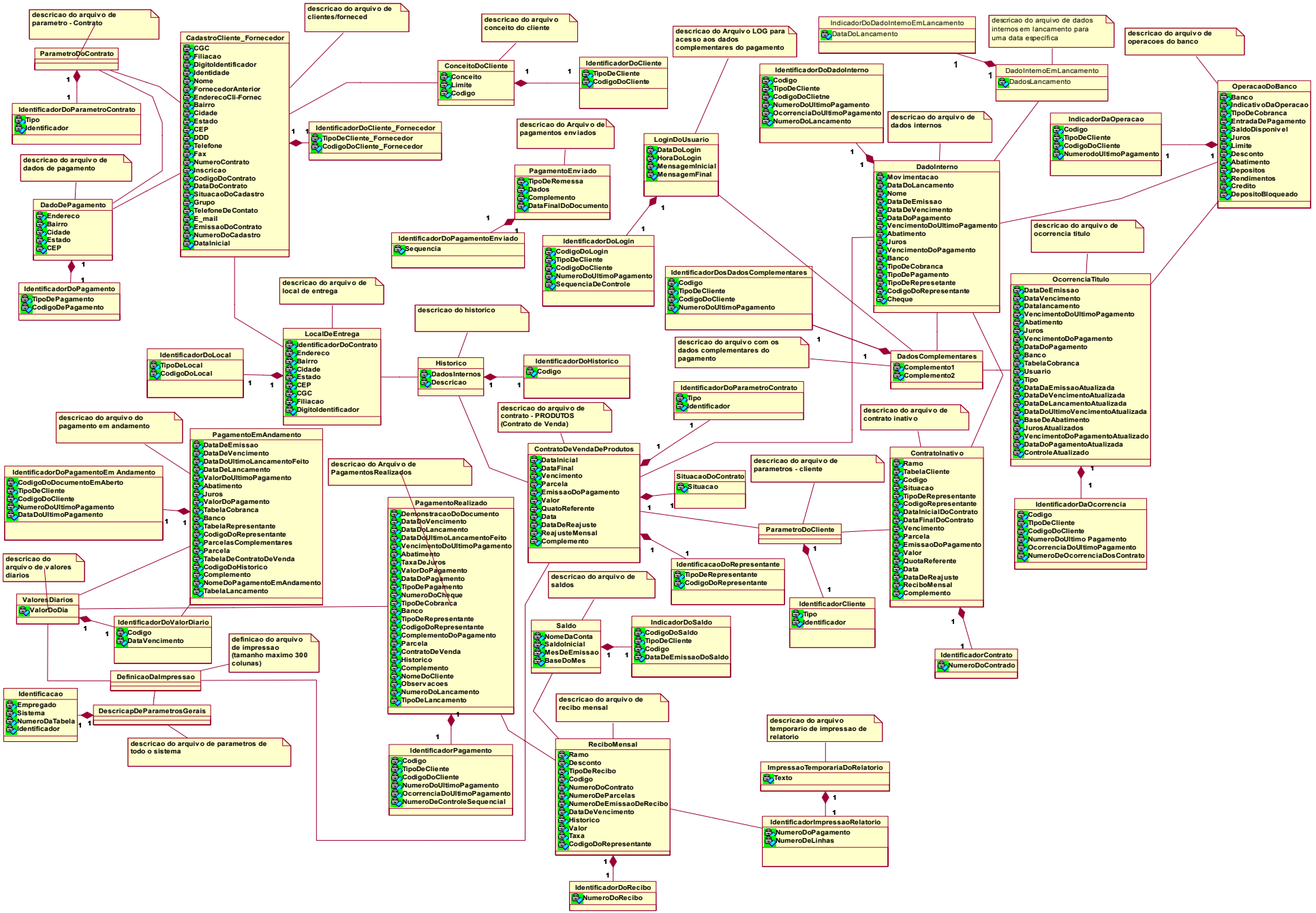

descricao do arquivo de parametro - Contrato
ParametroDoContrato
IdentificadorDoParametroContrato
Tipo
Identificador
descricao do arquivo de dados de pagamento
DadoDePagamento
Endereco
Bairro
Cidade
Estado
CEP
IdentificadorDoPagamento
TipoDePagamento
CodigoDePagamento
descricao do arquivo de clientes/forneced
CadastroCliente_Fornecedor
CGC
Filiacao
DigitoIdentificador
Identidade
Nome
FornecedorAnterior
EnderecoCli-Fornec
Bairro
Cidade
Estado
CEP
DDD
Telefone
Fax
NumeroContrato
Inscricao
CodigoDoContrato
DataDoContrato
SituacaoDoCadastro
Grupo
TelefoneDeContato
E_mail
EmissaoDoContrato
NumeroDoCadastro
DataInicial
IdentificadorDoCliente_Fornecedor
TipoDeCliente_Fornecedor
CodigoDoCliente_Fornecedor
descricao do arquivo de local de entrega
LocalDeEntrega
IdentificadorDoContrato
Endereco
Bairro
Cidade
Estado
CEP
CGC
Filiacao
DigitoIdentificador
IdentificadorDoLocal
TipoDeLocal
CodigoDoLocal
descricao do historico
Historico
DadosInternos
Descricao
IdentificadorDoHistorico
Codigo
descricao do arquivo conceito do cliente
ConceitoDoCliente
Conceito
Limite
Codigo
IdentificadorDoCliente
TipoDeCliente
CodigoDoCliente
descricao do Arquivo de pagamentos enviados
PagamentoEnviado
TipoDeRemessa
Dados
Complemento
DataFinalDoDocumento
IdentificadorDoPagamentoEnviado
Sequencia
descricao do Arquivo LOG para acesso aos dados complementares do pagamento
LoginDoUsuario
DataDoLogin
HoraDoLogin
MensagemInicial
MensagemFinal
IdentificadorDoLogin
CodigoDoLogin
TipoDeCliente
CodigoDoCliente
NumeroDoUltimoPagamento
SequenciaDeControle
IndicadorDoDadoInternoEmLancamento
DataDoLancamento
descricao do arquivo de dados internos em lancamento para uma data especifica
DadoInternoEmLancamento
DadosLancamento
descricao do arquivo de operacoes do banco
OperacaoDoBanco
Banco
IndicativoDaOperacao
TipoDeCobranca
EntradaDePagamento
SaldoDisponivel
Juros
Limite
Desconto
Abatimento
Depositos
Rendimentos
Credito
DepositoBloqueado
IdentificadorDoDadoInterno
Codigo
TipoDeCliente
CodigoDoCliente
NumeroDoUltimoPagamento
OcorrenciaDoUltimoPagamento
NumeroDoLancamento
descricao do arquivo de dados internos
DadoInterno
Movimentacao
DataDoLancamento
Nome
DataDeEmissao
DataDeVencimento
DataDoPagamento
VencimentoDoUltimoPagamento
Abatimento
Juros
VencimentoDoPagamento
Banco
TipoDeCobranca
TipoDePagamento
TipoDeRepresetante
CodigoDoRepresentante
Cheque
IndicadorDaOperacao
Codigo
TipoDeCliente
CodigoDoCliente
NumerodoUltimoPagamento
descricao do arquivo de ocorrencia titulo
OcorrenciaTitulo
DataDeEmissao
DataVencimento
DataLancamento
VencimentoDoUltimoPagamento
Abatimento
Juros
VencimentoDoPagamento
DataDoPagamento
Banco
TabelaCobranca
Usuario
Tipo
DataDaEmissaoAtualizada
DataDeVencimentoAtualizada
DataDeLancamentoAtualizada
DataDoUltimoVencimentoAtualizada
BaseDeAbatimento
JurosAtualizados
VencimentoDoPagamentoAtualizado
DataDoPagamentoAtualizada
ControleAtualizado
IdentificadorDaOcorrencia
Codigo
TipoDeCliente
CodigoDoCliente
NumeroDoUltimo Pagamento
OcorrenciaDoUltimoPagamento
NumeroDeOcorrenciaDosContrato
IdentificadorDosDadosComplementares
Codigo
TipoDeCliente
CodigoDoCliente
NumeroDoUltimoPagamento
descricao do arquivo com os dados complementares do pagamento
DadosComplementares
Complemento1
Complemento2
IdentificadorDoParametroContrato
Tipo
Identificador
descricao do arquivo de contrato - PRODUTOS (Contrato de Venda)
ContratoDeVendaDeProdutos
DataInicial
DataFinal
Vencimento
Parcela
EmissaoDoPagamento
Valor
QuatoReferente
Data
DataDeReajuste
ReajusteMensal
Complemento
SituacaoDoContrato
Situacao
descricao do arquivo de parametros - cliente
ParametroDoCliente
descricao do arquivo de contrato inativo
ContratoInativo
Ramo
TabelaCliente
Codigo
Situacao
TipoDeRepresentante
CodigoRepresentante
DataInicialDoContrato
DataFinalDoContrato
Vencimento
Parcela
EmissaoDoPagamento
Valor
QuotaReferente
Data
DataDeReajuste
ReciboMensal
Complemento
IdentificacaoDoRepresentante
TipoDeRepresentante
CodigoDoRepresentante
IdentificadorCliente
Tipo
Identificador
IdentificadorContrato
NumeroDoContrado
descricao do arquivo do pagamento em andamento
PagamentoEmAndamento
DataDeEmissao
DataDeVencimento
DataDoUltimoLancamentoFeito
DataDeLancamento
ValorDoUltimoPagamento
Abatimento
Juros
ValorDoPagamento
TabelaCobranca
Banco
TabelaRepresentante
CodigoDoRepresentante
ParcelasComplementares
Parcela
TabelaDeContratoDeVenda
CodigoDoHistorico
Complemento
NomeDoPagamentoEmAndamento
TabelaLancamento
IdentificadorDoPagamentoEm Andamento
CodigoDoDocumentoEmAberto
TipoDeCliente
CodigoDoCliente
NumeroDoUltimoPagamento
DataDoUltimoPagamento
descricao do Arquivo de PagamentosRealizados
PagamentoRealizado
DemonstracaoDoDocumento
DataDoVencimento
DataDoLancamento
DataDoUltimoLancamentoFeito
VencimentoDoUltimoPagamento
Abatimento
TaxaDeJuros
ValorDoPagamento
DataDoPagamento
TipoDePagamento
NumeroDoCheque
TipoDeCobranca
Banco
TipoDeRepresentante
CodigoDoRepresentante
ComplementoDoPagamento
Parcela
ContratoDeVenda
Historico
Complemento
NomeDoCliente
Observacoes
NumeroDoLancamento
TipoDeLancamento
descricao do arquivo de saldos
Saldo
NomeDaConta
SaldoInicial
MesDeEmissao
BaseDoMes
IndicadorDoSaldo
CodigoDoSaldo
TipoDeCliente
Codigo
DataDeEmissaoDoSaldo
descricao do arquivo de recibo mensal
ReciboMensal
Ramo
Desconto
TipoDeRecibo
Codigo
NumeroDoContrato
NumeroDeParcelas
NumeroDeEmissaoDeRecibo
DataDeVencimento
Historico
Valor
Taxa
CodigoDoRepresentante
IdentificadorDoRecibo
NumeroDoRecibo
descricao do arquivo temporario de impressao de relatorio
ImpressaoTemporariaDoRelatorio
Texto
IdentificadorImpressaoRelatorio
NumeroDoPagamento
NumeroDeLinhas
descricao do arquivo de valores diarios
ValoresDiarios
ValorDoDia
IdentificadorDoValorDiario
Codigo
DataVencimento
definicao do arquivo de impressao (tamanho maximo 300 colunas)
DefinicaoDaImpressao
Identificacao
Empregado
Sistema
NumeroDaTabela
Identificador
DescricapDeParametrosGerais
descricao do arquivo de parametros de todo o sistema
IdentificadorPagamento
Codigo
TipoDeCliente
CodigoDoCliente
NumeroDoUltimoPagamento
OcorrenciaDoUltimoPagamento
NumeroDeControleSequencial

Figura 128 - Modelo de Análise do Sistema Controle de Recibos Recuperado (MAS)

Refinamentos podem ser realizados no modelo de classes, MAS, pelo engenheiro de software com base nos conhecimentos no modelo de classes do sistema legado e da sua prática no desenvolvimento de sistemas orientados a objeto. Assim, a Figura 129 apresenta um diagrama de classes (omitindo os métodos) com refinamentos do tipo:

- Remoção de algumas classes com cardinalidade de 1 para 1 e inserção dos seus atributos nas classes todo, como pode ser observado no caso das classes ParametroDoContrato e IdentificadorDoParametroContrato, que mantinham uma associação de 1 para 1 no MAS e no MASRefinado a classe parte IdentificadroDoParametroContrato foi eliminada e seus atributos inseridos na classe ParametroDoContrato. O mesmo procedimento foi feito para as classes:

    - ConceitoDoCliente e IdentificadorDoCliente;
    - LocalDeEntrega e IdentificadorDoLocal;
    - PagamentoEmAndamento e IdentificadorDoPagamentoEmAndamento;
    - ValorDiario e IdentificadorDoValorDiario;
    - Saldo e IdentificadorDoSaldo;
    - ReciboMensal e IdentificadorDoRecibo;
    - DadoInterno e IdentificadorDoDadoInterno;
    - DefinicaoDaImpressao e Identificação;
    - DadoDePagamento e IdentificadorDoPagamento;
    - OcorrenciaTitulo e IdentificadorDaOcorrencia;
    - DadoInternoEmLancamento e IdentificadorDoDadoInternoEmLancamento;
    - ContratoInativo e IdentificadorDoContrato
    - ContratoDeVendaDeProdutos e IdentificacaoDoRepresentante e IdenticadorDoParametroContrato;
    - Historico e IdentificadorDoHistorico;
    - LoginDoUsuario e IdentificadorDoLogin;
    - DadosComplementares e IdentificadorDosDadosComplementares.

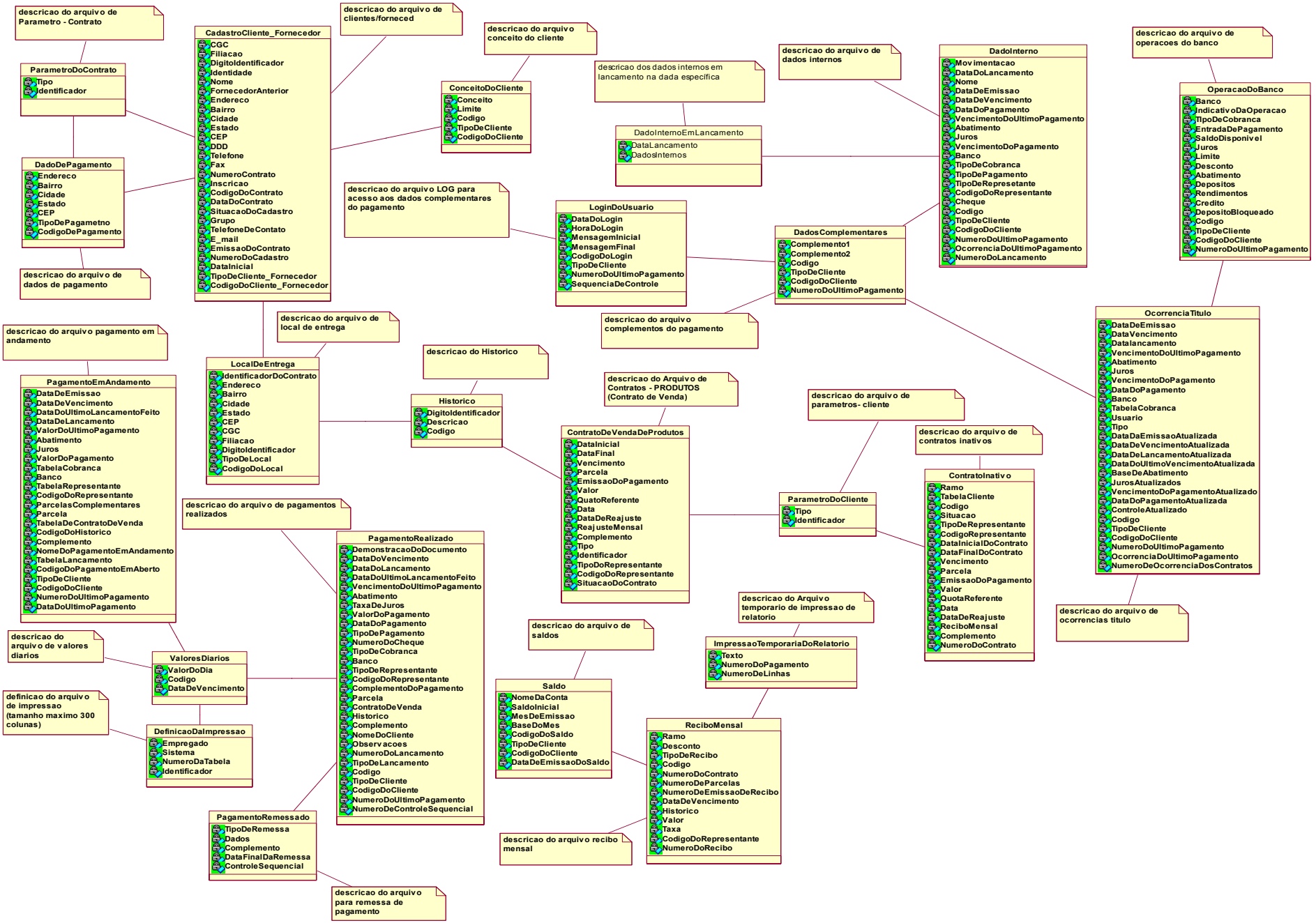

descricao do arquivo de Parametro - Contrato
ParametroDoContrato
Tipo
Identificador
DadoDePagamento
Endereco
Bairro
Cidade
Estado
CEP
TipoDePagametno
CodigoDePagamento
descricao do arquivo de dados de pagamento
CadastroCliente_Fornecedor
CGC
Filiacao
DigitoIdentificador
Identidade
Nome
FornecedorAnterior
Endereco
Bairro
Cidade
Estado
CEP
DDD
Telefone
Fax
NumeroContrato
Inscricao
CodigoDoContrato
DataDoContrato
SituacaoDoCadastro
Grupo
TelefoneDeContato
E_mail
EmissaoDoContrato
NumeroDoCadastro
DataInicial
TipoDeCliente_Fornecedor
CodigoDoCliente_Fornecedor
descricao do arquivo de clientes/forneced
descricao do arquivo conceito do cliente
ConceitoDoCliente
Conceito
Limite
Codigo
TipoDeCliente
CodigoDoCliente
descricao dos dados internos em lancamento na dada especifica
DadoInternoEmLancamento
DataLancamento
DadosInternos
descricao do arquivo de dados internos
DadoInterno
Movimentacao
DataDoLancamento
Nome
DataDeEmissao
DataDeVencimento
DataDoPagamento
VencimentoDoUltimoPagamento
Abatimento
Juros
VencimentoDoPagamento
Banco
TipoDeCobranca
TipoDePagamento
TipoDeRepresetante
CodigoDoRepresentante
Cheque
Codigo
TipoDeCliente
CodigoDoCliente
NumeroDoUltimoPagamento
OcorrenciaDoUltimoPagamento
NumeroDoLancamento
descricao do arquivo de operacoes do banco
OperacaoDoBanco
Banco
IndicativoDaOperacao
TipoDeCobranca
EntradaDePagamento
SaldoDisponivel
Juros
Limite
Desconto
Abatimento
Depositos
Rendimentos
Credito
DepositoBloqueado
Codigo
TipoDeCliente
CodigoDoCliente
NumeroDoUltimoPagamento
descricao do arquivo LOG para acesso aos dados complementares do pagamento
LoginDoUsuario
DataDoLogin
HoraDoLogin
MensagemInicial
MensagemFinal
CodigoDoLogin
TipoDeCliente
NumeroDoUltimoPagamento
SequenciaDeControle
DadosComplementares
Complemento1
Complemento2
Codigo
TipoDeCliente
CodigoDoCliente
NumeroDoUltimoPagamento
descricao do arquivo complementos do pagamento
descricao do arquivo pagamento em andamento
PagamentoEmAndamento
DataDeEmissao
DataDeVencimento
DataDoUltimoLancamentoFeito
DataDeLancamento
ValorDoUltimoPagamento
Abatimento
Juros
ValorDoPagamento
TabelaCobranca
Banco
TabelaRepresentante
CodigoDoRepresentante
ParcelasComplementares
Parcela
TabelaDeContratoDeVenda
CodigoDoHistorico
Complemento
NomeDoPagamentoEmAndamento
TabelaLancamento
CodigoDoPagamentoEmAberto
TipoDeCliente
CodigoDoCliente
NumeroDoUltimoPagamento
DataDoUltimoPagamento
descricao do arquivo de local de entrega
LocalDeEntrega
IdentificadorDoContrato
Endereco
Bairro
Cidade
Estado
CEP
CGC
Filiacao
DigitoIdentificador
TipoDeLocal
CodigoDoLocal
descricao do Historico
Historico
DigitoIdentificador
Descricao
Codigo
descricao do Arquivo de Contratos - PRODUTOS (Contrato de Venda)
ContratoDeVendaDeProdutos
DataInicial
DataFinal
Vencimento
Parcela
EmissaoDoPagamento
Valor
QuatoReferente
Data
DataDeReajuste
ReajusteMensal
Complemento
Tipo
Identificador
TipoDoRepresentante
CodigoDoRepresentante
SituacaoDoContrato
descricao do arquivo de parametros- cliente
ParametroDoCliente
Tipo
Identificador
descricao do arquivo de contratos inativos
ContratoInativo
Ramo
TabelaCliente
Codigo
Situacao
TipoDeRepresentante
CodigoRepresentante
DataInicialDoContrato
DataFinalDoContrato
Vencimento
Parcela
EmissaoDoPagamento
Valor
QuotaReferente
Data
DataDeReajuste
ReciboMensal
Complemento
NumeroDoContrato
OcorrenciaTitulo
DataDeEmissao
DataVencimento
Datalancamento
VencimentoDoUltimoPagamento
Abatimento
Juros
VencimentoDoPagamento
DataDoPagamento
Banco
TabelaCobranca
Usuario
Tipo
DataDaEmissaoAtualizada
DataDeVencimentoAtualizada
DataDeLancamentoAtualizada
DataDoUltimoVencimentoAtualizada
BaseDeAbatimento
JurosAtualizados
VencimentoDoPagamentoAtualizado
DataDoPagamentoAtualizada
ControleAtualizado
Codigo
TipoDeCliente
CodigoDoCliente
NumeroDoUltimoPagamento
OcorrenciaDoUltimoPagamento
NumeroDeOcorrenciaDosContratos
descricao do arquivo de ocorrencias titulo
descricao do arquivo de pagamentos realizados
PagamentoRealizado
DemonstracaoDoDocumento
DataDoVencimento
DataDoLancamento
DataDoUltimoLancamentoFeito
VencimentoDoUltimoPagamento
Abatimento
TaxaDeJuros
ValorDoPagamento
DataDoPagamento
TipoDePagamento
NumeroDoCheque
TipoDeCobranca
Banco
TipoDeRepresentante
CodigoDoRepresentante
ComplementoDoPagamento
Parcela
ContratoDeVenda
Historico
Complemento
NomeDoCliente
Observacoes
NumeroDoLancamento
TipoDeLancamento
Codigo
TipoDeCliente
CodigoDoCliente
NumeroDoUltimoPagamento
NumeroDeControleSequencial
descricao do Arquivo temporario de impressao de relatorio
ImpressaoTemporariaDoRelatorio
Texto
NumeroDoPagamento
NumeroDeLinhas
descricao do arquivo de saldos
Saldo
NomeDaConta
SaldoInicial
MesDeEmissao
BaseDoMes
CodigoDoSaldo
TipoDeCliente
CodigoDoCliente
DataDeEmissaoDoSaldo
descricao do arquivo de valores diarios
ValoresDiarios
ValorDoDia
Codigo
DataDeVencimento
definicao do arquivo de impressao (tamanho maximo 300 colunas)
DefinicaoDaImpressao
Empregado
Sistema
NumeroDaTabela
Identificador
ReciboMensal
Ramo
Desconto
TipoDeRecibo
Codigo
NumeroDoContrato
NumeroDeParcelas
NumeroDeEmissaoDeRecibo
DataDeVencimento
Historico
Valor
Taxa
CodigoDoRepresentante
NumeroDoRecibo
descricao do arquivo recibo mensal
PagamentoRemessado
TipoDeRemessa
Dados
Complemento
DataFinalDaRemessa
ControleSequencial
descricao do arquivo para remessa de pagamento

Figura 129 - Modelo de Análise do Sistema Controle de Recibos Recuperado Refinado (MASRefinado)

7.4 Considerações Finais

Esse capítulo apresentou, de forma resumida, as principais diretrizes, heurísticas, utilizadas no processo de engenharia reversa do sistema legado Controle de Recibos utilizando à **FAROOL**. A aplicação dessas heurísticas, específicas para programas implementados em *COBOL*, gera dois modelos de análises, MASA e MAS, que podem ser utilizados tanto para tarefas de manutenção como também para o processo de reengenharia orientada a objetos, ambos realizados após o processo de engenharia reversa.

Essas heurísticas são apresentadas na forma de guia, tomando como base o trabalho desenvolvido em Camargo (2001a), para auxiliar o engenheiro de software no processo de engenharia reversa orientada a objeto e permitir a automação de parte desse processo na ferramenta FAROOL. Portanto, a ferramenta de apoio FAROOL pode ser entendida como um guia para aplicação do método *Fusion/RE*, elaboração dos modelos de análise MASA e MAS, e conseqüentemente, para a modelagem de sistemas orientados a objetos a partir de legados procedimentais, implementados em *COBOL*, através do processo de engenharia reversa.

Com auxílio da ferramenta *CASE Legacy Aid*®, integrada à ferramenta FAROOL e de interações com o engenheiro de software, foram determinadas as pseudo-classes, os pseudo-atributos e os pseudo-métodos do sistema legado Controle de Recibos. Posteriormente, com os recursos implementados na ferramenta de apoio FAROOL, foram determinadas as classes, os atributos e os métodos, bem como os relacionamentos, cardinalidades, papéis, navegabilidade e demais detalhes resultantes do processo de engenharia reversa como um todo.

No caso do sistema Controle de Recibos, os resultados obtidos podem ser apresentados resumidamente como sendo um sistema com 95 classes, cada classe com em média 10 atributos e 10 métodos. Em termos de relacionamentos predominam relacionamentos de agregação com cardinalidade de 1 para 1, não ocorrendo, no entanto, relacionamentos de generalização ou de dependência. Algumas das classes identificadas,

criadas a partir de itens de grupo, mantém relacionamentos de agregação de 1 para 1. Mudanças de nomes de classes e de atributos forma feitas para exemplificar a facilidade de se fazer a manutenção em um sistema mais bem documentado. Nenhuma classe representando data foi encontrada, o que simplificou o processo de engenharia reversa. Todas as informações pertinentes e oriundas desse processo foram armazenadas em um banco de dados, feito especificamente para a ferramenta a partir de um metamodelo desenvolvido com as principais classes, atributos e relacionamentos entre classes.

Capítulo 8

Conclusões

8.1 Considerações Iniciais

Este projeto teve como objetivo o desenvolvimento de uma ferramenta de apoio ao processo de engenharia reversa orientada a objetos de sistemas legados procedimentais, implementados especificamente em *COBOL*, denominada FAROOL (**F**erramenta de **A**poio à Engenharia **R**eversa **O**rientada a **O**bjetos de **L**egados). Essa ferramenta contribui tanto para manutenção do sistema legado, uma vez que facilita a sua re-documentação, quanto para o processo de engenharia reversa orientada a objetos.

As heurísticas pré-determinadas em Camargo (2001a) e consolidadas manualmente em outros trabalhos foram utilizadas de forma a dar apoio computadorizado à parte do processo de engenharia reversa. FAROOL conta com uma interface interativa e um banco de dados relacional que permite o armazenamento das informações do sistema legado, facilitando o processo de engenharia reversa orientada a objetos.

O ambiente *Delphi*®, o SGBD *Interbase*® e a ferramenta *Legacy Aid*® foram utilizados para o desenvolvimento da FAROOL versão inicial. As contribuições e pontos fracos da ferramenta desenvolvidas nesta dissertação, FAROOL, são apresentados na seção 8.2 e as sugestões para ampliação da mesma, em trabalhos futuros, na seção 8.3

8.2 Resultados e contribuições da Ferramenta FAROOL

Com a primeira versão da FAROOL, o engenheiro de software tem um apoio computadorizado para realizar engenharia reversa orientada a objetos. Algumas das contribuições deste projeto são:

- Facilitar a realização do processo de engenharia reversa como um todo;
- Armazenar automaticamente as informações necessárias seja para manutenção do sistema ou realização do processo de engenharia reversa orientada a objetos;
- Oferecer maior eficiência e rapidez na recuperação das informações de sistemas legados, bem como no armazenamento dessas, obtidas durante o processo de engenharia reversa;
- Garantir que parte do processo de engenharia reversa conta com apoio computacional, facilitando a aplicação do mesmo por parte dos engenheiros de software;
- Apresentar uma possível forma de automação do processo de engenharia reversa com integração de ferramentas *CASE(s)* auxiliares e de interação com o engenheiro de software, e
- Enfatizar a importância do engenheiro de software no processo de engenharia reversa de sistemas legados, pois esse processo não pode ser totalmente automatizado.

Resumindo, pode-se destacar os seguintes pontos relevantes deste trabalho:

- **Facilidade de Uso:** o engenheiro de software conta com uma interface interativa que disponibiliza recursos para que o engenheiro de software recupere o sistema otimizando esforços;

- **Redução de Tempo:** o engenheiro de software reduz o tempo necessário na recuperação do sistema, uma vez que parte do trabalho, antes realizada manualmente por ele, conta agora com apoio computacional. Exemplos desse apoio automatizado são: a lista chama-chamado, grafos de fluxos de programa e arquivos para documentação que a ferramenta *CASE Legacy Aid*® realiza quando ativada a partir da FAROOL.

- **Condução para Análise de Pontos Relevantes:** o engenheiro de software pode tratar pontos relevantes de análise do sistema através do guia que lhe é

fornecido, conduzindo-o, passo a passo para a recuperação completa do sistema legado com mudança do paradigma de procedimental para orientado a objeto.

Os pontos fracos dessa primeira versão da ferramenta são:

- A não geração automática de um modelo de classes, a partir das informações geradas e armazenadas no banco de dados;
- O excesso de informações constantes das interfaces que exige muita atenção do engenheiro de software;
- A falta de um *parser* para ler o código *COBOL* diretamente e, de acordo com as heurísticas, gerar um modelo de classes, e

8.3 Sugestões para Trabalhos Futuros

As sugestões abaixo relacionam alguns trabalhos que podem ser realizados com o propósito de facilitar ainda mais a recuperação dos sistemas legados procedimentais implementados em *COBOL*, bem como para auxiliar a manutenção dos mesmos e a aplicação do processo de engenharia reversa orientada a objetos visando reengenharia.

- Implementar um *parser* para a ferramenta FAROOL, permitindo assim uma menor dependência da ferramenta *CASE Legacy Aid*® no processo de determinação de pseudo-classes, pseudo-atributos e pseudo-métodos.
- Possibilitar a geração de modelos orientados a objetos a partir das informações armazenadas no banco de dados, utilizando linguagens do tipo *XMI* ou *XML*. Dessa forma, pode haver a interação da FAROOL com *CASE* tipo *Rational Rose* (Rose, 2001), *Argo* (Argo, 2003) ou *MVCase* (Lucredio, 2001) na diagramação e geração inicial do código em linguagens orientadas a objeto.

- Estudar e implementar recursos para ampliar a aplicação da ferramenta FAROOL a fim de realizar o processo de reengenharia e permitir a mudança de linguagem do sistema legado, de procedimental (*COBOL*) para uma linguagem mais moderna. Podem ser incorporadas técnicas que auxiliem o engenheiro de software quanto à

decisão do tipo de arquitetura a ser usada quando o sistema for disponibilizado para *Web*: *Thin ou Fat Client*; utilização de *servlet* , *jsp* ou recursos mais atuais.

- Fazer um tratamento estatístico com as informações obtidas no processo de engenharia reversa a partir de um conjunto de sistemas legados implementados em *COBOL*, de forma a comparar aumento/diminuição do número de classes, atributos e métodos entre o legado e o que passou pelo processo de engenharia reversa orientada a objetos.
- Tratar a elaboração de relatórios bem estruturados para facilitar a manutenção de sistemas.
- Realizar estudos referentes à usabilidade da FAROOL, melhorando-a quanto à interface.

Referências Bibliográficas:

Arakaki, R.; Arakaki, J.; Angerami, P. M.; Aoki, O. L.; Salles, D. de S. – (**1990**). Fundamentos de Programação C. 2ª. Edição, Rio de Janeiro: LTC - Livros Técnicos e Científicos, Editora S.A..

Bastos, A. C. – (**1983**). Programação COBOL. 4ª. Edição, Rio de Janeiro: LTC - Livros Técnicos e Científicos, Editora S.A..

Bellay, B.; Gall, H. C. – (**1998**). An Evaluation of Reverse Engineering Tool Capabilities. Journal of Software Maintenance. V 10, páginas 305-331.

Bennet, K. H.; Ward, M. P. – (**1995**). Formal Methods for Legacy Systems. Journal of Software Maintenance: Research and Practice. Volume 7, número 3, páginas 203-219.

Bray, T., Paoli, J., and McQueen, M. S. - (**1998**). Extensible Markup Languange (XML) 1.0. http://www.w3.org/TR/REC-xml, 1998.

Sobrinho, A. - (**1997**). Análise de Sistemas II - O Mito do RAD. Revista Byte Brasil. Originalmente publicado na Revista Developer's Magazine em Abril de 1997.

Camargo, V. V. de. – (**2001a**). Reengenharia Orientada a Objetos de Sistemas COBOL com a Utilização de Padrões de Projeto e Servlets - Dissertação de Mestrado. Universidade Federal de São Carlos.

Camargo, V. V. de. – (**2001b**). Diretrizes para a Realização da Engenharia Reversa de Sistemas COBOL utilizando Fusion/RE. In Proceedings do CLEI 2001 – México.

Chicofsky, J. E. and Cross, J.H. – (**1990**). Reverse Engineering and Design Recovery: A Taxonomy. IEEE Software. Volume 7, número 1, páginas 13-17.

Coleman, Dl. – (**1994**) Object-Oriented Development – The Fusion Method. Prentice Hall,

Curtis, B.; Hefley, W.; Miller, S. - (**1995**) The Overview of the People Capability Maturity Model (P-CMM) http://www.sei.cmu.edu/publications/documents/95.reports/.

Demeyer, S.; Ducasse, S.; Nierstrasz, O. - (**2000**) A Pattern Language for Reverse Engineering Proceedings, of the 5th Europen Conference on Pattern Languages of Programming and Computing, (EuroPLOP'2000), Andreas Ruping (Ed.).

Furlan, J.D. – (**1998**) Modelagem de Objetos através da UML. Makron Books, 1a.ed.,.

Jacobson, I; Lindström, F. – (**1991**). Re-engineering of old systems to na object oriented architecture. SIGPLAN Notices. Volume 26, número 11, páginas 340-350.

Lemos, G. S. – (**2002**) Garantia de Qualidade nos Processos de Engenharia Reversa e Reengenharia – Dissertação de Mestrado. Universidade Federal de São Carlos.

Lucrédio, D., Prado, A. F. – (**2001**) Ferramenta MVCASE - Estágio Atual: Especificação, Projeto e Construção de Componentes. Sessão de Ferramenta do XV Simpósio Brasileiro de Engenharia de Software - SBES'2001. Pág. 368-373. CDU: 681.31:061.68. Rio de Janeiro-RJ, Brasil. 03-05 de Outubro.

Mecenas, I.– (**2002**). Interbase 6 Guia do Desenvolvedor. Ed. Book Express.

Melo, A. C. – (**2002**). Desenvolvendo Aplicações com UML. Ed. Brasport.

Oliveira, C. H. P. de. – (**2002**). SQL- Curso Prático. Ed. Novatec.

Penteado, R.A.D; Masiero, P. C.; Braga, R. T. V.; Prado, A. F. – (**1998**). Reengineering of Legacy Systems Using the Object Oriented Paradigm. V working Conference on Reverse Engineering –IEEE. Honolulu, Hawaii.

Penteado, R. A. D.; Masiero, P. C.; Cagnin, M. I. – (**1999**). An Experiment of Legacy Code Segmentation to Improve Maintainability. III European Conference on Software Maintenance and Reengineering – IEEE. Amsterdã, Holanda. Páginas 111-119.

Penteado, R.A.D; Germano F.; Masiero, P. C. – (**1996b**). An Overall Process Based on Fusion to Reverse Engineer Legacy Code. III working conference on Reverse Engineering – IEEE. Monterey, California.páginas 179-188.

Pressman, R.S. – (**2001**). Software Engineering: A Practitioner's Approach. Fifth Edition. McGraw-Hill Higher education.

Prado, Milene – (**2002**). Ferramenta de Apoio à Engenharia Reversa Orientada a Objetos de Legados – Qualificação de Mestrado, Departamento de Computação – UFSCar, São Carlos.

Prado, Milene – (**2003**). Manual da Ferramenta de Apoio à Engenharia Reversa Orientada a Objetos de Legados – Qualificação de Mestrado, Departamento de Computação – UFSCar, São Carlos.

Recchia, E.L. – (**2002a**) Engenharia Reversa e Reengenharia Baseadas em Padrões, São Carlos – SP, Dissertação de Mestrado apresentado ao PPGCC – UFSCar.

Recchia, E. L;Penteado, R – (**2002b**) Família de padrões para Conduzir Processos de Engenharia Reversa Orientada a Objetos de Sistemas Legados Orientados a Procedimentos. Artigo Submetido ao The Second latin American Conference on Pattern languages of Programming.

Sommerville. I. – (**1995**). Software Engineering. International Computer Science Series. 5a. edição. Addison Wesley Publishing Co.

Sonninno, B. – (**2002**). Desenvolvendo Aplicações em Delphi 6.0. Ed. Makron Books.

Vilanova, L.O. – (**1997**). Reengenharia para Reutilização de Software: Uma abordagem para recuperação de projetos orientados a objetos. Tese de mestrado, COPPE/UFRJ.

Sites Consultados

(Accacia, 2002) http://www.research.att.com/sw/tools/Acacia/ consultado em dez 2002.

(AnalSoftSpec, 2002) http://www.uni-koblenz.de/~clange/IWPCPaper.pdf consultado em dez 2002.

(Argo, 2003) http://argouml.tigris.org consultado em janeiro de 2003

(Bauhaus, 2002) http://www.informatik.uni-stuttgart.de/ifi/ps/bauhaus/index-english.html consultado em dez de 2002.

(Borland, 2003) http://www.borland.com.br/ consultado em maio de 2003.

(Ciao, 2002) http://www.research.att.com/~ciao/ consultado em dez 2002.

(Cosmos, 2002) http://www.emendo.com consultado em dez 2002.

(DBMain, 2002) http://www.fundp.ca.be/ consultado em dez 2002.

(Dpute, 2002) http://www.cs.umn.edu/Research/softeng/sm/dpute/dpute.tar.gz consultado em dez 2002.

(ERStudio, 2003) http://www.embarcadero.com/products/erstudio/index.asp consultado em maio 2003

(ErWin, 2002) http://www.cai.com/evaluate/download/erwin_standard.htm consultado em dez 2002.
http://www.linsoft.se/info/erwin/ até dez 2002.

(Grasp, 2002) http://www.eng.auburn.edu/department/cse/research/grasp/ consultado em dez 2002.

(Gupro, 2002) http://www.uni-koblenz.de/~ist/gupro.en.html consultado em dez 2002.

(Legacy Aid, 2002) http://www.casemaker.com consultado em dez 2002.

(MacroMagic, 2002) http://www.iolo.com/support/ consultado em maio de 2003

(OMG, 2003) http://www.omg.org consultado em maio 2003.

(Refine - C, 2002) http://www.reasoning.com consultado em dez 2002.

(Rigi, 2002) http://www.rigi.csc.uvic.ca/ consultado em dez 2002.

(RoadMap, 2002) http://www.raincode.com/cobolroad/ consultado em dez 2002.

(Rose, 2001) http://www.rational.com/tryit/rose/tutorials/rose-tutorials.jsp
Tutorial Rational Rose – (2001) , consultado em dez 2001.

(Tksee, 2002) http://www.site.uottawa.ca/~tcl/kbre/ consultado em dez 2002.

(Windows, 2002) http://www.microsoft.com/brasil/default.asp consultado em maio 2003.

(WithClass, 2002) http://www.microgold.com consultado em dez 2002.

ApêndiceI

Mapeamento entre o MASA e o MAS:

O engenheiro de software, quando do término do MASA, tem a possibilidade de: Gerar um banco de dados com o MASA (Arquivo de extensão .gbk no *Interbas®e*) e Recuperar o banco de dados com o MASA (Arquivo de extensão .gdb no *Interbase®*).

O primeiro permite que o engenheiro de software gere um cópia das informações armazenadas no banco de dados da FAROOL ao término da Elaboração do MASA. O segundo permite ao engenheiro de software recuperar esse modelo, carregando-o no *IBConsole* do SGBD *Interbase®*. Isso possibilita o mapeamento entre o pseudo modelo orientado a objetos (MASA) e o modelo orientado a objetos (MAS) após a elaboração desse último. As janelas disponíveis ao engenheiro de software para realização desses passos encontram-se nas Figuras 130 e 131 respectivamente.

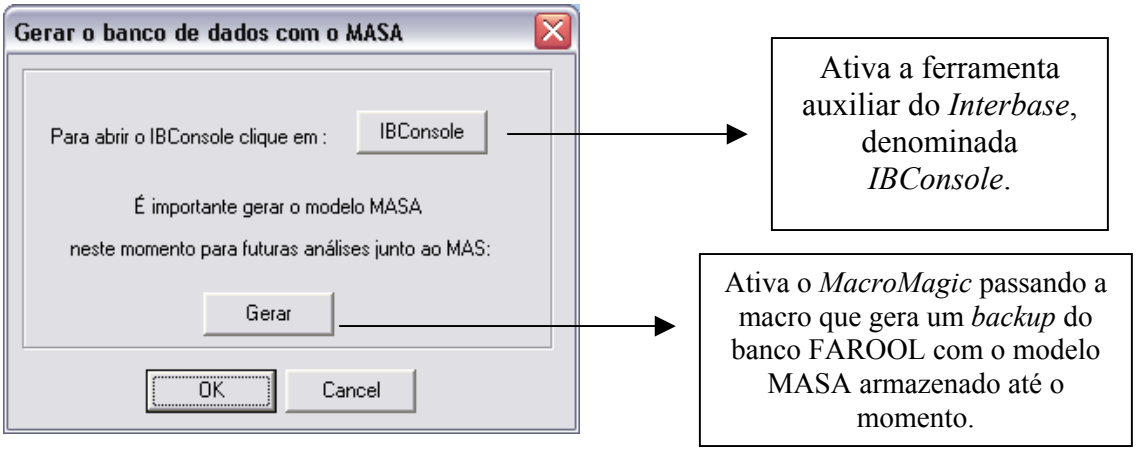


Figura 130 - Geração de uma cópia do MASA no Banco de Dados

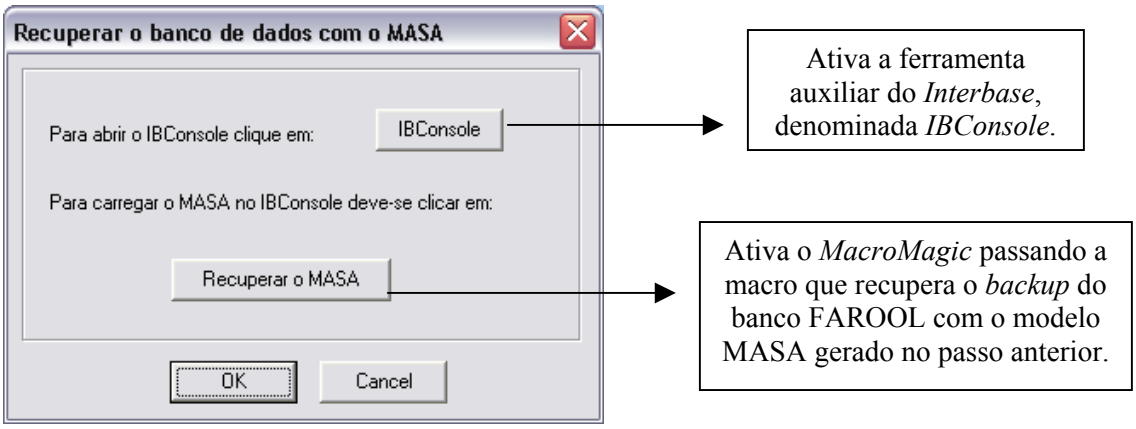


Figura 131 - Recuperação do banco com o MASA

A macro que responsável por gerar uma banco com uma cópia do MASA está programada para preencher as informações da janela apresentada na Figura 132 de forma transparente ao engenheiro de software a partir do acionamento do botão Gerar exibido na tela da Figura 131. A macro que dispara a recuperação do banco de dados com o MASA está programada para preencher as informações da tela da Figura 133 a partir de uma clique no botão Recuperar o MASA.

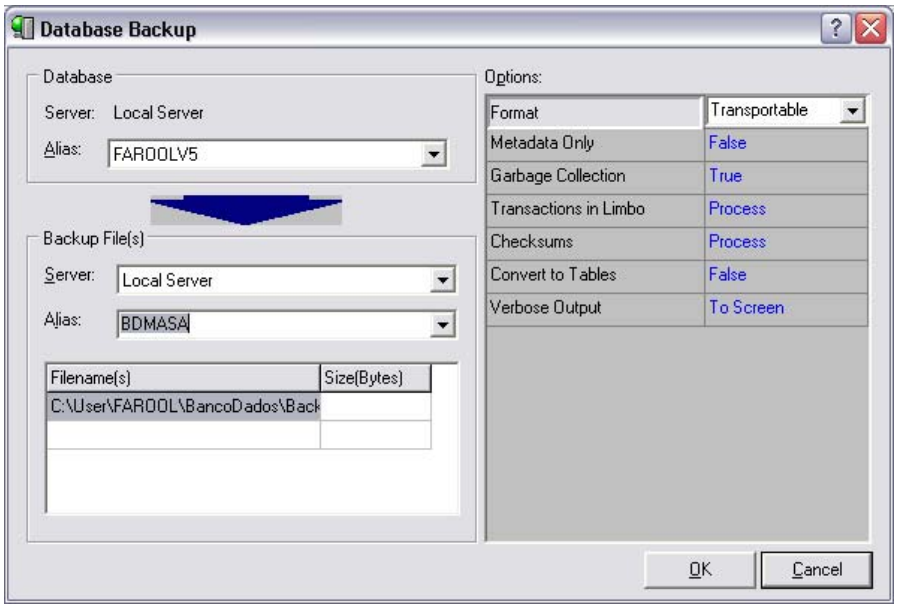


Figura 132 - Tela para geração de uma cópia do MASA no banco FAROOL

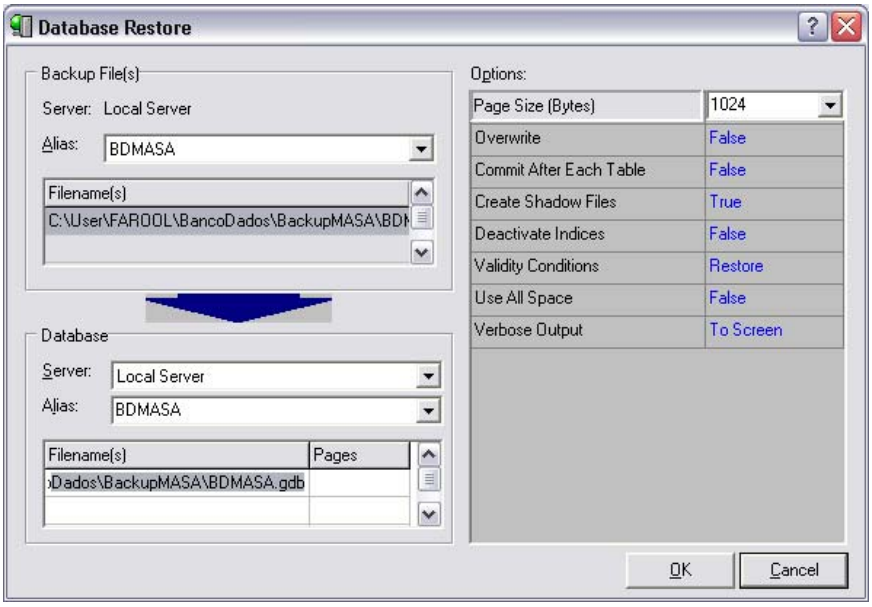


Figura 133 - Tela para recuperação do MASA gerado

A Figura 134 mostra o banco com o modelo MAS em FAROOLV5 e o modelo MAS em MASABD.

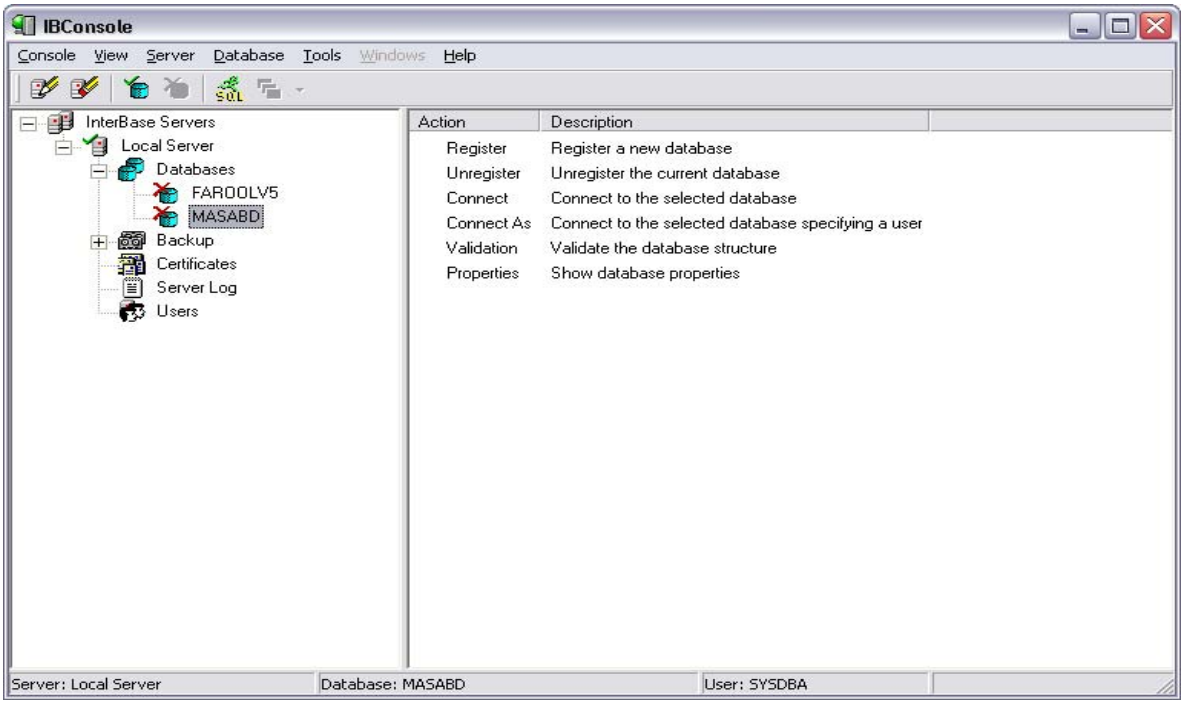


Figura 134 - Os bancos com o MASA e o MAS

Ainda foi inserido um novo campo no banco de dados da FAROOL para armazenamento do nome antigo da classe, permitindo ao engenheiro de software fazer a ligação entre o nome anterior e o novo nome dado quando da mudança de nomes das classes por mnemônicos mais significativos. Foi também oferecido um novo recurso na tela para mudança de nome das classes que permite guardar os nomes dos programas, os nomes antigos das classes e os atuais definidos nessa tela. O engenheiro de software clica em Abrir Arquivo Texto para Mapeamento quando é aberto um documento no *Word Pad* e posteriormente em Mapeamento quando é ativada uma macro que escreve o nome do programa, o antigo nome da classe e o novo nome sugerido pelo engenheiro de software, separados por barras (/) no arquivo do *Word Pad.* A Figura 135 ilustra o processo.

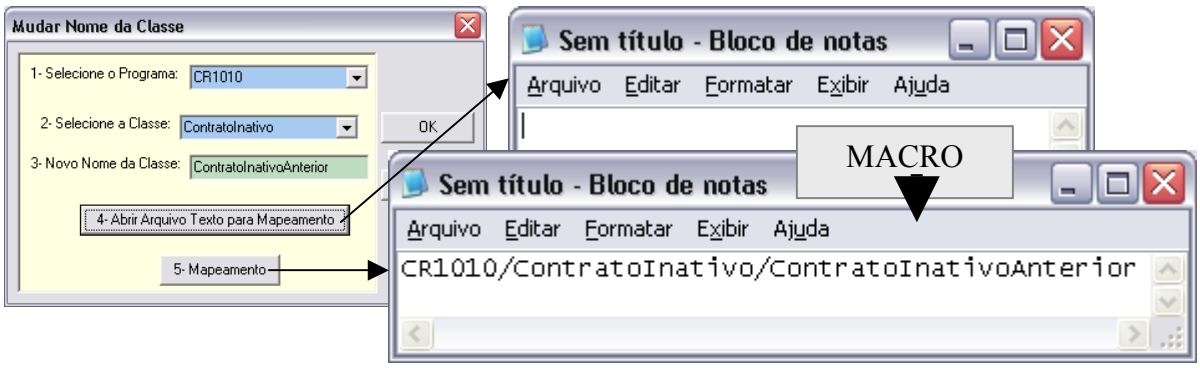


Figura 135 - Mapeamento entre o nome antigo (MASA) e o novo nome da classe (MAS)