KÁTIA REZENDE FERREIRA

REENGENHARIA DE SOFTWARE BASEADA NOS PADRÕES DA FAPRE/OO

Trabalho de conclusão de curso apresentado ao Curso de Ciência da Computação.

UNIVERSIDADE PRESIDENTE ANTÔNIO CARLOS

Orientador: Prof. Eduardo Macedo Bhering
Co-orientador: MSc. Frederico de Miranda Coelho

BARBACENA
2004

KÁTIA REZENDE FERREIRA

REENGENHARIA DE SOFTWARE BASEADA NOS PADRÕES DA FAPRE/OO

Este trabalho de conclusão de curso foi julgado adequado à obtenção do grau de Bacharel em Ciência da Computação e aprovado em sua forma final pelo Curso de Ciência da Computação da Universidade Presidente Antônio Carlos.

Barbacena – MG, 23 de Junho de 2004.

_____

Prof. Eduardo Macedo Bhering - Orientador do Trabalho

_____

MSc. Frederico de Miranda Coelho - Membro da Banca Examinadora

_____

MSc. Wender Magno Cota - Membro da Banca Examinadora

AGRADECIMENTOS

Agradeço a Deus que sempre me protege e me dá coragem pra seguir em frente; aos meus irmãos e aos meus pais agradeço pelo esforço para que eu pudesse concluir o curso. Ao meu orientador Eduardo Bhering pela dedicação, e pelo acompanhamento dado a este trabalho e a todos os meus mestres, o meu muito obrigada. Aos meus colegas que através de momentos alegres e difíceis estamos juntos no final de mais uma etapa de nossa vida, ao meu namorado e a todos que direta ou indiretamente me deram força. Obrigada.

RESUMO

O propósito deste trabalho de conclusão de curso é apresentar as etapas da reengenharia de um sistema legado. Estes sistemas são, na maioria das vezes, desenvolvidos através de técnicas de desenvolvimento informal e, aliado à necessidade de manutenção ao longo do tempo, possuem documentação incompleta, não refletindo a situação do software existente. Esta situação torna problemática a sua manutenção.

Nesse contexto, a reengenharia assume importância essencial, possibilitando recuperar as informações de projeto perdidas durante a fase de desenvolvimento, e de documentar o real estado do sistema.

Para documentar o sistema será adotada a metodologia de desenvolvimento baseado nos padrões da Família de Padrões de Reengenharia Orientada a Objetos, proposto por Rosângela Penteado e Edson Luiz Recchia (Universidade Federal de São Carlos/SP). A FaPRE/OO engloba todas as fases do processo de engenharia reversa e engenharia avante alcançando um projeto do sistema em um alto nível de abstração, assegurando a sua manutenção e tornando-o de fácil compreensão.

.

Palavras-chave: Reengenharia, Engenharia Reversa, Engenharia Avante.

SUMÁRIO

LISTAS.....6

LISTAS.....7

1 INTRODUÇÃO.....8

2 REVISÃO BIBLIOGRÁFICA.....11

3 REENGENHARIA BASEADA NOS PADRÕES DA FAPRE/OO.....23

4 ENGENHARIA AVANTE BASEADA NOS PADRÕES DA FAPRE/OO.....58

5 CONCLUSÕES.....62

REFERÊNCIAS BIBLIOGRÁFICAS.....64

ANEXO A – RELÁTORIO DXF.....65

ANEXO B – CHAVE PRIMÁRIA E CHAVE ESTRANGEIRA OBTIDA DO DOCUMENTADOR DO ACCESS 67

ANEXO C – SCRIPTS (SQL).....78

LISTAS

FIGURA 1 - PROCESSO DE REENGENHARIA:(ELISA, 2004).....12

FIGURA 2 - ENGENHARIA REVERSA E ENGENHARIA AVANTE:(IAN, 2003).....13

FIGURA 3 - CATEGORIAS DA ENGENHARIA REVERSA E SUAS VISÕES:(ELISA, 2004).....17

FIGURA 4 - FAMÍLIA DE PADRÕES DE REENGENHARIA ORIENTADOS A OBJETOS: (RECCHIA, 2004).....21

FIGURA 5 - MODELO ENTIDADE RELACIONAMENTO INICIAL:.....26

FIGURA 6 - MER DO SISTEMA LEGADO:.....29

FIGURA 7 - MASA DO SISTEMA LEGADO:.....31

FIGURA 8 - MENU INICIAL DO SISTEMA DE CADASTRO DE SONDAGENS:.....32

FIGURA 9 - MENU AJUDA:.....34

FIGURA 10 - DIAGRAMA DE CASO DE USO PARA O CENÁRIO SISTEMA:.....35

FIGURA 11 - DIAGRAMA DE CASO DE USO PARA O CENÁRIO CADASTRO:.....35

FIGURA 12 - DIAGRAMA DE CLASSES (CONSTRUÇÃO DA CLASSE):.....45

FIGURA 13 - DIAGRAMA DE CLASSES (DEFINIÇÃO DOS ATRIBUTOS DAS CLASSES):.....47

FIGURA 14 - DIAGRAMA DE CLASSES (DEFINIÇÃO DOS MÉTODOS DAS CLASSES):.....50

FIGURA 15 - DIAGRAMA DE SEQÜÊNCIA DO CASO DE USO PARA O CENÁRIO SISTEMA:.....53

FIGURA 16 - DIAGRAMA DE SEQÜÊNCIA DO CASO DE USO PARA O CENÁRIO CADASTRO:..54

FIGURA 17 - DIAGRAMA DE SEQÜÊNCIA DO CASO DE USO PARA O CENÁRIO CADASTRO:..55

FIGURA 18 - DIAGRAMA DE SEQÜÊNCIA DO CASO DE USO PARA O CENÁRIO CADASTRO:..56

FIGURA 19 - DIAGRAMA DE IMPLANTAÇÃO:.....57

LISTAS

1 INTRODUÇÃO

1.1 CONSIDERAÇÕES INICIAIS

O número de problemas que envolvem manutenção de sistema aumenta constantemente fazendo com que alguns sistemas tornem-se ineficientes, apresentando características desnecessárias para a aplicação específica; necessitando de sucessivas atualizações e adaptações forçadas o que dificulta a localização da origem dos problemas, e conseqüentemente o processo de manutenção (ALVARO, 2004).

A dificuldade em atualizar sistemas para utilização de novas tecnologias tem motivado pesquisadores a analisar novas soluções buscando a diminuição de custos de manutenção e desenvolvimento, prolongando assim o tempo de vida útil do sistema; pois a falta de uma documentação consistente que permita um melhor entendimento do domínio da aplicação é constante na maioria dos sistemas (ALVARO, 2004).

Existe um amplo número de empresas que continuam trabalhando com sistemas nomeados legados (definido de maneira informal como aquele que foi desenvolvido

utilizando-se de técnicas atualmente consideradas absoletas, mas que fornecem tarefas e serviços benéficos para a organização), cuja manutenção se torna cada vez mais complexa e arriscada. Esses sistemas, ainda são de muita utilidade aos seus usuários e muitas vezes, sua reconstrução usando técnicas modernas de desenvolvimento de software, pode ser a solução para sua reutilização sem a necessidade de construção de um novo sistema. Normalmente, o código legado possui lógica de programação e requisitos de usuários que podem ser recuperados, facilitando a construção de sua documentação. Ocorre também a existência de sistemas legados que não possuem documentação ou quando possuem, a mesma está desatualizada, devido ao grande número de alterações e adaptações neles implementadas. Fatores como estes tornam a manutenção difícil e de alto custo (JAMES, 1999).

Diante disso surgiu a motivação de analisar os conceitos de análise e projeto de software facilitando a manutenção dos sistemas. Neste contexto, o processo de Reengenharia torna-se uma forma de produzir um sistema novo de fácil manutenção, consistindo da engenharia reversa seguida da engenharia avante.

1.2 OBJETIVO

O objetivo deste trabalho é realizar a reconstrução de algo do mundo real através do processo de reengenharia que tem como propósito à busca por melhorias.

Dado um sistema legado será feito o processo de reengenharia (análise e modelagem), sendo sua implementação uma sugestão para trabalhos futuros.

O sistema no qual será realizada a reengenharia de software foi desenvolvido na linguagem Delphi em 1998, para resolver o problema específico de uma empresa de sondagens e fundações.

Através da recuperação do sistema, será realizada a sua modelagem visando-se a obtenção de uma representação clara.

Após o processo de análise, a finalidade é oferecer um produto que atenda não apenas às necessidades de uma determinada empresa, mas que torne um sistema genérico de alta qualidade e eficiente.

1.3 ORGANIZAÇÃO DO TRABALHO

De maneira que conduza o leitor ao entendimento do conjunto de idéias e conceitos alcançados durante o desenvolvimento deste trabalho, é apresentado inicialmente o material pesquisado e estudado que fundamenta o presente trabalho na seguinte ordem:

O capítulo 1 descreve a importância de um sistema documentado e seu objetivo.

O capítulo 2 aborda a revisão bibliográfica, os principais conceitos de Reengenharia e a Família de Padrões para a Reengenharia Orientada a Objetos de Sistemas Legados Procedimentais, FaPRE/OO.

No capítulo 3, o uso dos padrões da FaPRE/OO é exemplificado para um Sistema de Cadastro de Sondagens implementado na linguagem DELPHI.

O capítulo 4 apresenta a engenharia avante do sistema, baseada nos modelos obtidos no capítulo 3.

Por fim, será tratada no capítulo 5 a conclusão final deste trabalho e propostas para realizações de trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Para satisfazer as metas propostas para este projeto, foi feita uma revisão da literatura, abordando um breve resumo dos principais conceitos que foram aplicados.

2.1 REENGENHARIA DE SOFTWARE

Para Peters (JAMES, 1999), realizar reengenharia significa aperfeiçoar a documentação, compreender e descrever aquilo que o sistema faz, suas entradas, saídas e ações, projetando e reestruturando um sistema existente de forma a obter uma forma mais aceitável do mesmo. Com isso, a reengenharia pode ser utilizada em um processo de software onde existe uma descrição de requisitos que permita a identificação das estruturas de software, codificação e implementação física do projeto de sistema.

Segundo Pressman (PRESSMAN, 2002), a reengenharia de Software é uma atividade de reconstrução que identifica e avalia os processos existentes, criando processos revisados que satisfazem melhor os objetivos atuais, e versões dos programas existentes com mais qualidade. Uma das grandes motivações para a utilização da reengenharia é a diminuição

nos altos custos de manutenção de sistema. A contínua manutenção desses sistemas faz com que a implementação fique incoerente com a documentação do projeto original, e o código cada vez mais difícil de se compreender e sujeito a erros. Ou seja, reengenharia é o processo de criar uma descrição abstrata do sistema, de forma reutilizável que permite obter o entendimento do domínio da aplicação, recuperar as informações das etapas de análise e projeto, organizando-as de forma coerente e reutilizável e então implementá-las no sistema.

A principal característica da reengenharia de software e reimplementar sistemas legados, tendo como principal objetivo um software de fácil manutenção. A reengenharia tem a capacidade de redocumentar, organizar e reestruturar o sistema, traduzir o sistema para uma linguagem de programação mais moderna, modificar e atualizar a estrutura e os valores dos dados do sistema (IAN, 2003).

Segundo Jacobson e Lindstrom (ELISA, 2004) o processo de reengenharia de software é constituído de duas fases distintas, como é ilustrada na Figura 1. Na primeira fase o software objeto de reconstrução é “desmontado”, visando seu entendimento. Já na segunda fase, o software é reconstruído na forma desejada, a partir do produto da primeira fase, sendo incluídos os ajustes que se fizerem necessários.

FIGURA 1 - Processo de Reengenharia:(ELISA, 2004)

Onde △ pode ser de dois tipos:

Alterações de funcionalidade: Devido à necessidade do usuário são adicionados ao sistema novos requisitos. Essas alterações são conseguidas com maior facilidade devido à fase anterior, visto já haver informações em nível de abstração mais alto (ELISA, 2004);

Alterações de implementação: Ocorrem alterações no ambiente de operação do software e ou alteração da linguagem de implementação (protocolos, sistema operacional, linguagens,etc.) (ELISA, 2004).

Segundo Ian Sommerville (IAN, 2003), a principal distinção entre ambas é o ponto de partida para o desenvolvimento. Em vez de iniciar com uma especificação escrita, o sistema legado atua como uma especificação para um novo sistema. A diferença está ilustrada na Figura 2. A engenharia reversa percorre o caminho inverso da engenharia avante: começa com um sistema já existente, que envolve o processo de compreender e analisar o sistema, possibilitando seu entendimento e a obtenção da documentação. Já a engenharia avante irá partir de um nível de abstração alto, iniciando com uma especificação do sistema que envolve o projeto e a implementação, chegando ao nível físico de um novo sistema.

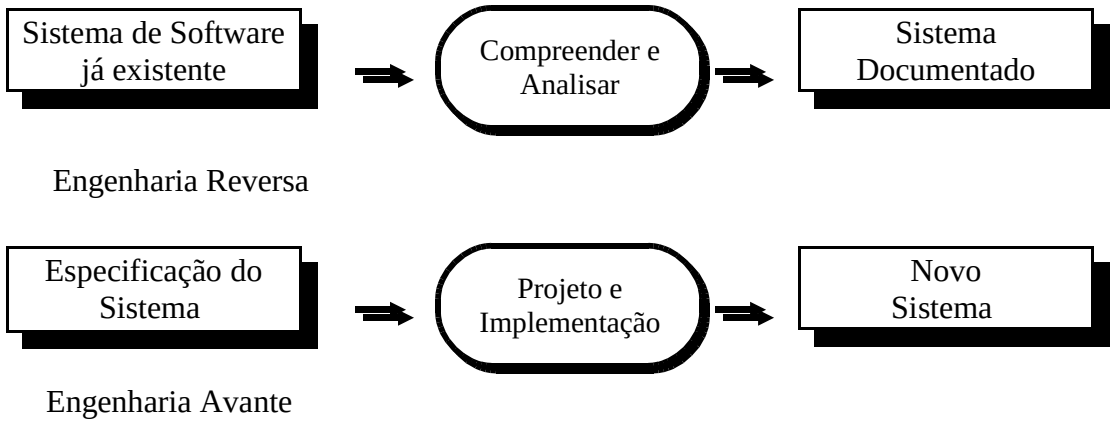


FIGURA 2 - Engenharia Reversa e Engenharia Avante:(IAN, 2003)

2.2 ENGENHARIA REVERSA

A atividade de manutenção de sistema é reconhecidamente uma fase problemática. Tais problemas são causadores de custos substanciais quando comparados com os custos das outras fases do ciclo de vida do software. A facilidade de manutenção, caracterizada principalmente pelo entendimento do sistema, está fortemente relacionada à disponibilidade de documentação do software, que na maioria das vezes é incompleta, desatualizada ou até mesmo inexistente, causando assim o aparecimento de um dilema na decisão sobre o futuro de um sistema, a "reconstrução". A reconstrução é necessária quando o software tem vida limitada e alterações na sua estrutura podem adulterá-la.

- Software deve evoluir continuamente ou tornar-se-á menos útil no mundo real;
- Software legado que executa tarefas úteis mas que foi desenvolvido com técnicas obsoletas que dificultam a sua manutenção e evolução, como desestruturação e dificuldade de entendimento do código, documentação desatualizada, programadores que não desenvolveram o produto e efeitos colaterais causados por alterações.

Nesse sentido a Engenharia Reversa é primordial.

Para Pressman (PRESSMAN, 2002), a engenharia reversa de software é um processo de recuperação de projeto, consistindo em analisar um programa, na tentativa de criar uma representação do mesmo, em um nível de abstração mais alto que o código-fonte.

Segundo Peters (JAMES, 1999), o ponto principal da engenharia reversa é a análise de um sistema existente, de forma a identificar seus dados e as inter relações dos mesmos. Um dos principais benefícios da engenharia reversa aplicada a sistemas de software, está relacionado à obtenção de uma melhor compreensão e recuperação de informação e estruturas úteis em nível de projeto para auxiliar a manutenção. Exemplos de itens que podem ser recuperados são os modelos de dados reutilizáveis, as estruturas de controle, as descrições de interface, projeto, propriedades comportamentais, requisitos funcionais e de desempenho. Durante o processo de engenharia reversa, são extraídos elementos de projeto, coletando-se da análise do sistema descrições menos dependentes de implementação relativas ao sistema em questão. As informações a respeito do controle de tarefa (por exemplo, arquivos criados, arquivo de controle de fonte), os comentários feitos pelos desenvolvedores na explicitação do código, as estruturas de dados e os esquemas de banco de dados são exemplos de elementos úteis na engenharia reversa.

Para Garcia (GARCIA, 2004), a engenharia reversa tem como objetivo identificar as informações do projeto de um sistema, as especificações funcionais e, eventualmente, requisitos utilizados na sua implementação, criando representações abstratas, para assim auxiliar na manutenção e adaptação desse sistema a novas tecnologias e plataformas. Um tipo de Engenharia Reversa comum é a redocumentação de um sistema para obter o seu projeto, representado através de modelos gráficos e textual, sendo que as formas resultantes desta

representação são consideradas visões alternativas, utilizadas para uma melhor compreensão humana do sistema analisado, e em um alto nível de abstração.

Sendo assim a Engenharia Reversa deve produzir documentos que possibilitem aumentar o entendimento suficiente em nível de projeto para auxiliar a manutenção, reutilização, teste e controle de qualidade do software.

2.3 CATEGORIAS DA ENGENHARIA REVERSA

De acordo com o nível de entendimento obtido do sistema são definidas duas categorias de engenharia reversa (ELISA, 2004):

A. Visualização de código

Denominada também como redocumentação, a visualização de código é a criação num mesmo nível de abstração. O processo de visualização de código cria representações a partir de informações obtidas apenas da análise do código fonte, que são consideradas visões alternativas, cuja finalidade é aprimorar a concepção do sistema global (ELISA, 2004).

A forma mais simples e mais antiga de engenharia reversa é a visualização de código. A intenção é recuperar a documentação que já existiu, ou que deveria ter existido, sobre o sistema. A ênfase é a criação de visões gráficas, que não foram criadas durante o processo original (ELISA, 2004).

Visões mais ambiciosas como, função do sistema, finalidade ou a essência do sistema exige um nível de entendimento maior e são definidas como entendimento de programa (ELISA, 2004).

B. Entendimento de programa

Também denominada recuperação de projeto, entender um programa significa poder explicar o sistema, seu comportamento e sua estrutura, além de conhecer o domínio das informações externas. As deduções são adicionadas às observações feitas sobre o sistema

através do exame do mesmo de modo a obter informações com nível mais alto de abstração (ELISA, 2004).

O entendimento de programa possibilita abstrair do projeto todas as informações necessárias (combinação de código, documentação existente, domínio de aplicação), para se entender completamente como, por que e o que o sistema faz (ELISA, 2004).

O entendimento de sistemas é essencial para o processo de manutenção. Qualquer que seja a manutenção a ser feita no software haverá a necessidade de primeiramente entender o programa, para depois efetuar as alterações (ELISA, 2004).

Entendimento de programa distingue-se de visualização de código porque objetiva entender e explicar o processo pelo qual o sistema foi desenvolvido, seus efeitos e seus relacionamentos com o domínio da aplicação, ao invés de simplesmente fornecer visões alternativas para auxiliar o usuário a entender o sistema (ELISA, 2004).

É apresentada na Figura 3 a relação entre as duas categorias de engenharia reversa, relacionada com o escopo das informações utilizadas (código fonte ou base de conhecimento) e o nível de visualização que pode ser classificado em (ELISA, 2004):

Visão em nível implementacional: abstrai características da linguagem de programação e característica especifica da implementação;

Visão em nível estrutural: abstraem detalhes da linguagem de programação para revelar sua estrutura a partir de diferentes perspectivas. O resultado é uma representação especificada das dependências entre os componentes do sistema;

Visão em nível funcional: relaciona partes do programa às suas funções, procurando revelar as relações lógicas entre elas.

Visão em nível de domínio: abstrai o contexto em que o sistema está operando, ou seja, o porquê do sistema a ser desenvolvido.

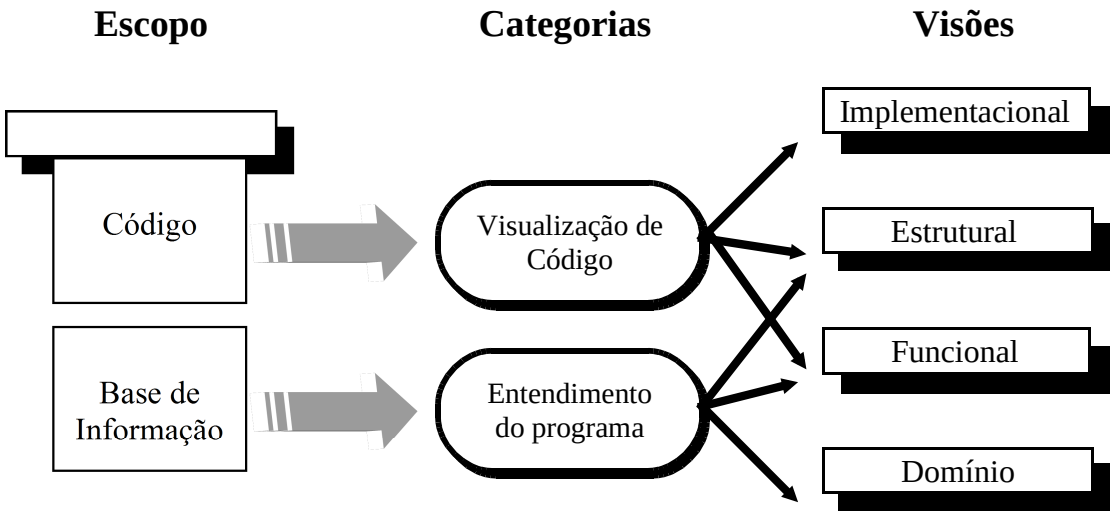


FIGURA 3 - Categorias da Engenharia Reversa e suas Visões:(ELISA, 2004)

2.4 ENGENHARIA AVANTE

A engenharia avante é a segunda parte da reengenharia de software, que através da documentação obtida do processo de engenharia reversa é possível acrescentar os requisitos necessários reformulando o sistema legado.

Segundo Pressman (PRESSMAN, 2002) a engenharia avante é o processo que irá partir de um nível de abstração alto, representado pelos requisitos do sistema, e chega ao nível físico, de implementação do sistema. Assim, a engenharia avante reconstrói as especificações de um sistema usando métodos modernos de engenharia de software e as informações extraídas durante o processo de engenharia reversa. O processo de engenharia avante tanto recupera as informações do projeto de um sistema existente, mas também usa informações na sua alteração ou reconstrução, procurando aperfeiçoá-lo para garantir melhoria de qualidade na sua funcionalidade, combinando novos requisitos, adequando as novas tecnologias e assim possibilitando diminuição dos custos com a sua manutenção.

2.5 CONTRIBUIÇÕES RELEVANTES PARA O PROCESSO DE REENGENHARIA

Com o constante aumento da complexidade dos sistemas, bem como do alto grau de competição do mercado, que exige cada vez mais a diminuição do tempo e buscando uma automatização na transformação do código e qualidade na recuperação das informações do código legado; é necessário portanto para auxiliar o processo de reengenharia o uso de ferramentas que auxiliam no desenvolvimento de projetos de sistemas em geral. A maioria das ferramentas oferece ao desenvolvedor várias metodologias criadas para dar suporte à modelagem de sistemas, ajudando sobremaneira na interação homem-máquina com relação a usuários leigos (ELISA, 2004) .

Para conduzir o processo de engenharia reversa e engenharia avante do sistema, foi adotada uma metodologia de desenvolvimento que engloba todas as fases deste processo, sendo que a abordagem proposta para resolver esse problema é uma Família de Padrões de Reengenharia Orientada a Objeto - FaPRE/OO, tendo como alvo sistemas Orientado a Objetos. A notação escolhida foi à utilização da UML (*Unified Modeling Language*) que é uma linguagem que permite a construção de modelos gráficos para visualização, especificação, construção e documentação de informações, tendo seu foco voltado para a representação conceitual e física de um sistema (BOOCH, 2000).

Para auxiliar na modelagem será utilizada a ferramenta CASE *Rational Rose* baseada na notação da UML, através da qual obtém-se uma melhor representação dos diagramas gráficos do processo de desenvolvimento da engenharia reversa. O resultado desse processo é encaminhado para a engenharia avante, finalizando, assim, o processo de reengenharia.

2.6 FAMÍLIA DE PADRÕES DE REENGENHARIA ORIENTADA A OBJETOS – FAPRE/OO

A definição de padrões de reengenharia tem sido abordada pela necessidade de caminhos mais consistentes para orientar na realização desse processo. Aliada a isso, a preocupação com a qualidade do produto gerado, contribui para a descrição da reengenharia sob a forma de padrões com o objetivo de torná-la mais clara para a engenharia de software.

Para condução desse processo foi abordada a técnica, denominada FaPRE/OO.

A FaPRE/OO é uma Família de Padrões de Reengenharia para gerar processos de engenharia reversa e de engenharia avante orientados a objetos, a partir de sistemas legados procedimentais (PENTEADO, 2004).

A FaPRE/OO é composta de quatro *clusters* sendo que cada um agrupa os padrões relacionados à reengenharia. A etapa da engenharia reversa é dividida em três *clusters*, e a engenharia avante possui um cluster. A Figura 4 ilustra graficamente os *clusters* e os padrões existentes em cada um deles (PENTEADO, 2004).

O primeiro *Cluster,* "Modelar os Dados do Legado" agrupa padrões que extraem informações a partir dos dados e do código fonte do sistema legado gerando o MER - Modelo Entidade Relacionamento (visão procedimental dos dados) e o MASA - Modelo de Análise do Sistema Atual – Diagrama de Pseudo-Classes (visão orientada a objetos dos dados). Esses padrões orientam o engenheiro de software no primeiro contato com o sistema. Fazem parte desse *cluster* os seguintes padrões:

- Iniciar Análise dos Dados;
- Definir Chaves;
- Identificar Relacionamentos;
- Criar Visão OO dos Dados.

O segundo *Cluster,* "Modelar a Funcionalidade do Sistema", agrupa padrões destinados à obtenção da funcionalidade do sistema, criando modelos que recuperem as Regras de Negócio da Empresa, contidas no sistema legado. Esses padrões habilitam o engenheiro de software a obter um entendimento detalhado dos componentes do sistema, aprofundando sua compreensão. Fazem parte desse *cluster* os seguintes padrões:

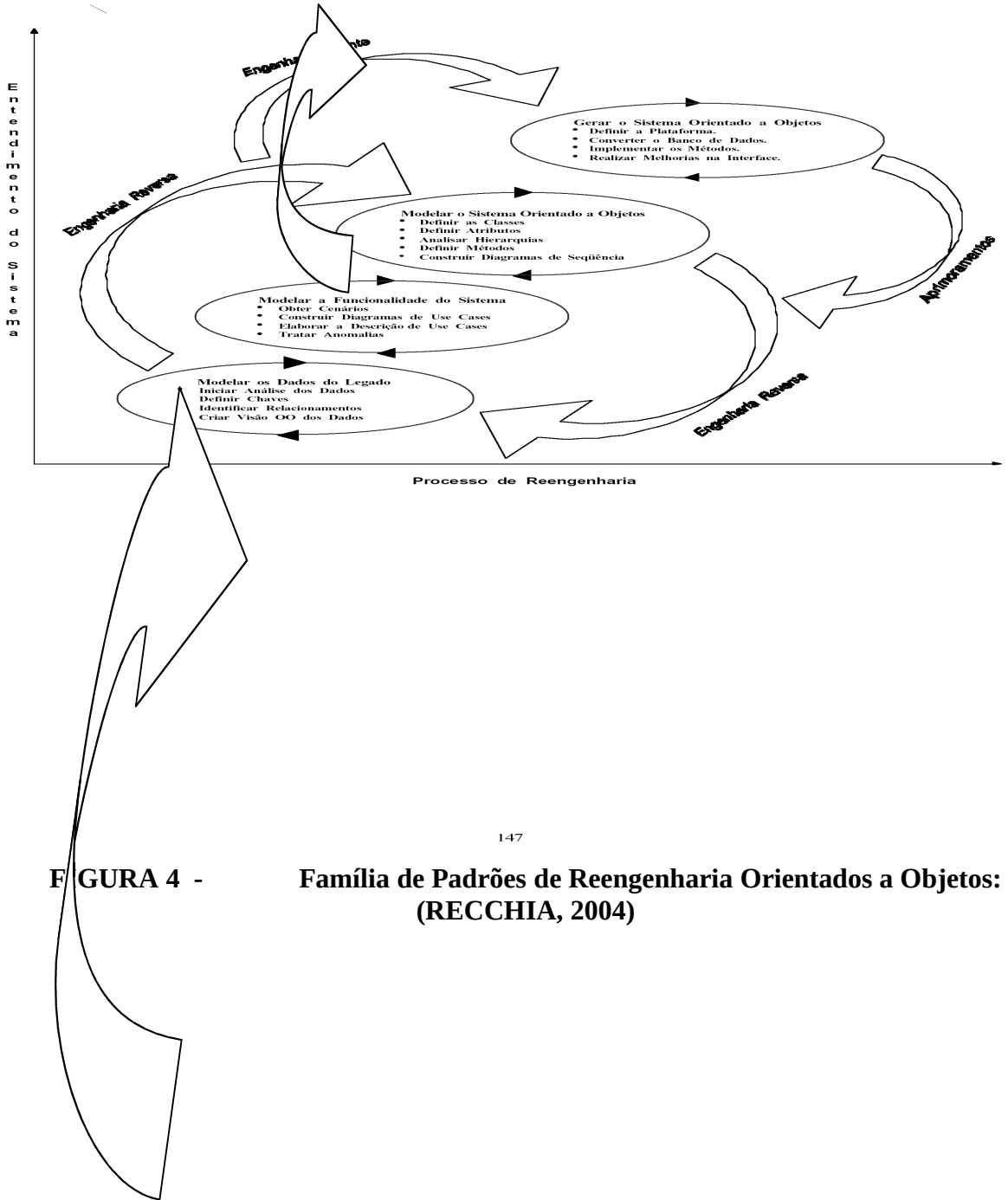
- Obter Cenários;
- Construir Diagramas de Caso de Uso;
- Elaborar a Descrição de Caso de Uso;
- Tratar Anomalias.

O terceiro *Cluster,* "Modelar o Sistema Orientado a Objetos", agrupa padrões para se obter o Diagrama de Classes e os Diagramas de Seqüência do sistema, através da interação dos produtos obtidos pelos padrões dos *clusters* anteriores. Esses padrões habilitam o engenheiro de software a obter o MAS-Modelo de Análise do Sistema, sendo o modelo orientado a objetos a servir de suporte ao processo de reengenharia. Fazem parte desse *cluster* os seguintes padrões:

- Definir as Classes;
- Definir Atributos;
- Analisar Hierarquias;
- Definir Métodos;
- Construir Diagramas de Seqüência.

O quarto *Cluster,* "Gerar o Sistema Orientado a Objetos", agrupa padrões que completam o processo de engenharia avante do sistema, transformando o sistema legado, do paradigma procedimental para o paradigma orientado a objetos. Fazem parte desse *cluster* os seguintes padrões:

- Definir a Plataforma:
    - escolher o ambiente operacional;
    - escolher o banco de dados relacional;
    - escolher o pacote da linguagem de implementação;
- Converter o Banco de Dados;
- Implementar os Métodos;
- Realizar Melhorias na Interface.



147

FIGURA 4 - Família de Padrões de Reengenharia Orientados a Objetos: (RECCHIA, 2004)

Na Figura 4, pode-se observar que muitos padrões terão como entrada e saída de algum padrão aplicado anteriormente, dando a idéia de ser uma aplicação seqüencial. O modelo é evolutivo, podendo-se, de qualquer padrão, avançar ou retroceder após a sua aplicação. Pode-se observar também que a aplicação seqüencial ocorre apenas nos padrões do *cluster* 1.

Os padrões do processo de engenharia reversa obedecem o seguinte formato: Nome, Intuito, Problema, Contexto (Influências), Solução, Avaliação, Justificativa, Usos Conhecidos, Padrões Relacionados e Produtos Obtidos. O item **Solução** a ser adotada é apresentado na forma de passos, sempre que necessário e os itens de **Avaliação** e **Justificativa** devem conter a opinião dos engenheiros de software quando da utilização desses padrões em seus processos de engenharia reversa. O item **Usos Conhecidos** é apresentado como sendo o uso dos conceitos de cada padrão, sem estarem na forma de padrão, que foram reconhecidos nos processos de engenharia reversa e os procedimentos utilizados para solucionar problemas análogos.

Já os padrões de engenharia avante apresentam o seguinte formato: Nome, Intuito, Contexto, Problema, Solução, Conseqüência, Usos Conhecidos e Padrões Relacionados. O item **Solução** a ser adotada é apresentado na forma de passos, sempre que necessário e o item **Conseqüência** não é apresentado, mas deve conter a opinião dos engenheiros de software quando da utilização desses padrões em seus processos de reengenharia.

De acordo com (PENTEADO, 2004) e (RECCHIA, 2004), o formato para os padrões foi adotado com base nos padrões já existentes na literatura (Demeyer et al).

2.7 CONCLUSÕES

No capítulo 2 foi apresentada a revisão bibliográfica sobre Reengenharia de Software. As definições abordadas neste capítulo serão aplicadas no próximo capítulo tendo como finalidade dar uma visão inicial sobre o desenvolvimento do projeto.

3 REENGENHARIA BASEADA NOS PADRÕES DA FAPRE/OO

3.1 CONSIDERAÇÕES INICIAIS

As etapas do processo de engenharia reversa e o uso dos padrões da Família de Padrões de Reengenharia Orientada a Objetos – FaPRE/OO será aplicada neste capítulo que tem como intuito a obtenção da documentação do Sistema de Cadastro de Sondagens, no qual estará apta para a etapa do processo de engenharia avante.

3.2 VISÃO DO SISTEMA

Designado Sistema de Cadastro de Sondagens - CadSond, este tem por intuito facilitar o trabalho de engenheiros e desenhistas gerando relatórios sobre sondagens sobre tipo de solo realizadas em um determinado local, como é mostrado no Anexo A. O sistema registra as informações gerais sobre a sondagem, os tipos de camadas do terreno e os tipos de

solo encontrados em cada uma delas. São registradas também as informações colhidas nas amostras obtidas durante a perfuração, incluindo a resistência do material encontrado. A partir do registro descritivo de uma sondagem, o sistema gera o croqui esquemático da perfuração, de acordo com o modelo da empresa. O croqui é exportado no formato AutoCAD DXF, para posterior edição e impressão através deste software. Essa escolha foi feita considerando a precisão do AutoCAD, bem como os recursos oferecidos para a edição do desenho gerado.

3.3 ENGENHARIA REVERSA DO SISTEMA DE CADASTRO DE SONDAGENS APLICANDO OS PADRÕES DA FAPRE/OO

De acordo com as definições bibliográficas apresentadas anteriormente será apresentado daqui a diante as etapas do processo de engenharia reversa aplica no Sistema de Cadastro de Sondagens baseado no modelo de padrões FaPre/OO.

Como dito anteriormente essa etapa é agrupada em 3 clusters que envolvem 13 padrões, onde cada padrão da Engenharia reversa é composto de itens que serão analisados e completados de acordo com a análise do sistema.

CLUSTER 1 - MODELAR OS DADOS LEGADOS:

1. Nome: Iniciar Análise dos Dados

Intuito: Dar início à construção do MER, Modelo Entidade Relacionamento, utilizando uma tabela que relaciona todos os programas/módulos que fazem parte do sistema, com seus respectivos arquivos de dados.

Influências:

O sistema é composto por um único programa (CadSond.EXE), que agrupa as funções (módulos);

Ausência de documentação adicional sobre a base de dados. A documentação existente refere-se aos metadados contidos no próprio arquivo de dados, em formato MDB (Microsoft® Access DataBase) ;

Nesses metadados estão descritos quais são as tabelas que compõe a base de dados do sistema, bem como os atributos que assumem os papéis de chaves primária e estrangeira;

Na lista de arquivos manipulados pelo sistema, encontra-se um arquivo em formato texto, contendo a descrição do cabeçalho de um arquivo de dados em formato DXF (formato de transferência de dados do AutoCAD®), o qual é utilizado para compor o arquivo de dados de saída.

Solução:

I. Construir a tabela Programas x Arquivos, com duas colunas. A primeira coluna contém os nomes dos programas e, a segunda, o arquivo de dados tratado em cada programa. O nome original desses arquivos deve ser mantido, como é ilustrado na tabela 1.

II. Iniciar a construção do MER, a partir da tabela construída no passo 1. Considerando cada tabela do sistema legado como uma entidade do MER, Figura 5.

Avaliação: De acordo com a construção da Figura 5, pode-se visualizar um primeiro esquema do sistema para a construção do MER.

Justificativa: Com a utilização desse padrão é apresentado o primeiro contato com o sistema.

Padrões Relacionados: Esse padrão é a entrada para o padrão Identificador Relacionamentos.

Produtos Obtidos:

MER – Modelo Entidade Relacionamento Inicial

Tabela de Programas X Arquivos.

TABELA 1 – Tabela de Programas X Arquivos:

Programas	Arquivos
Cadastro de Sondagens.EXE	BD_Sondagens. MDB ▪ Sondagens ▪ Amostras ▪ Camadas ▪ Classificação de Camadas ▪ Entidade de Preenchimento ▪ Observações ▪ Padrões de Preenchimento ▪ Parâmetros Base.DXF ▪ Cabeçalho para o arquivo de saída

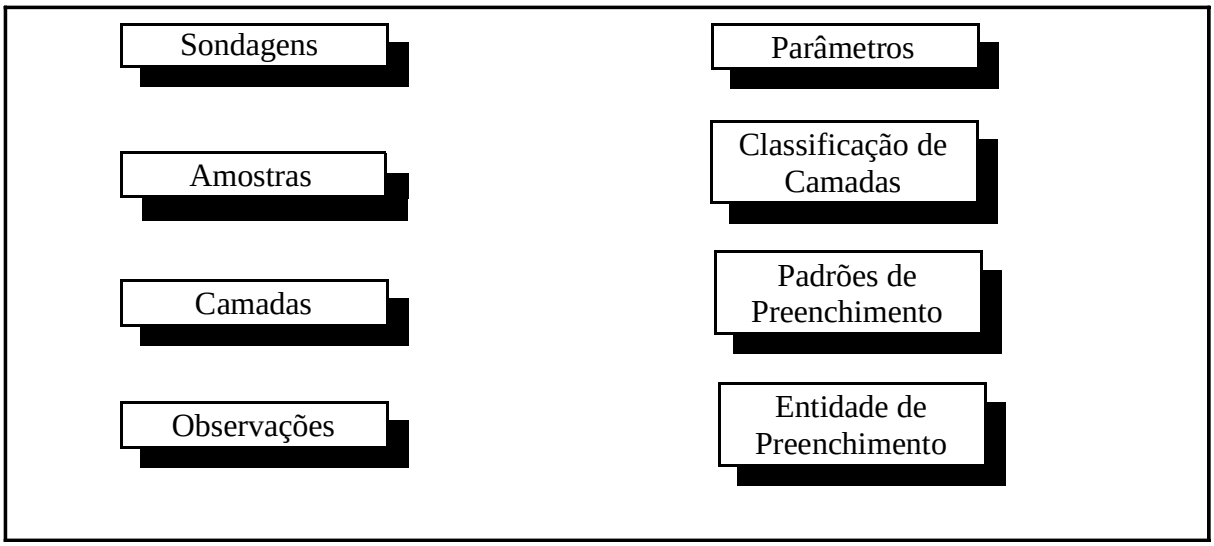


FIGURA 5 - Modelo Entidade Relacionamento Inicial:

2. Nome: Definir Chaves

Intuito: Obter as chaves primárias e estrangeiras para cada entidade identificada pelo padrão Iniciar Análise dos Dados, a partir de arquivos de dados do sistema legado, a fim de identificar os relacionamentos entre as entidades que compõem o MER, iniciado no passo anterior.

Influências:

Para obter as chaves primárias e estrangeiras, bem como as informações sobre relacionamentos entre as entidades, pode-se utilizar a ferramenta de documentação de uma ferramenta CASE ou do próprio gerenciador de banco de dados (no caso o Microsoft® Access®).

Solução:

I. Construir a Tabela Entidades x Chaves. A primeira coluna contém os nomes das entidades e a segunda as chaves primárias identificadas para essas entidades.

II. Utilizando a ferramenta apropriada, identificar as chaves estrangeiras das entidades, adicionando-as na Tabela Entidades x Chaves em uma nova coluna (chaves estrangeiras).

Através do documentador do próprio Access, pode-se identificar a chave Primária e a chave Estrangeira para cada entidade analisando o referente arquivo.mdb. O resultado do processo de documentação automática do Access é apresentado no Anexo B.

Avaliação: De acordo com a análise inicial dos dados, tem-se a capacidade de extrair um MER inicial a partir das entidades identificadas do Sistema. Com a identificação destas entidades pode-se localizar as chaves primárias e estrangeiras que o sistema possui.

Justificativa: De acordo com esse padrão é obtido as chaves e os atributos das entidades em questão.

Produto Obtido:

Tabela Entidade X chaves;

A Tabela 2 mostra a Tabela ENTIDADES X CHAVES, contendo três colunas.

TABELA 2 – Tabela de Entidades X Chaves:

Entidades	Chaves Primárias	Chaves Estrangeiras
Amostras	NumAmostra, Lk_Sondagem	Lk_Sondagem
Camadas	NumCamada, Lk_Sondagem	Lk_Sondagem
Classificação de Camadas	Classif	_____
Entidades de Preenchimento	AutoID	Lnk_Padrao
Observações	AutoCod	Lk_Sondagem
Padrões de Preenchimento	Nome	_____
Parâmetros	Tipo, Descrição	_____
Sondagens	Código	_____

3. **Nome:** Identificar Relacionamentos

Intuito: Identificar os relacionamentos entre as entidades do MER.

Influências:

Como descrito anteriormente, as informações sobre relacionamentos entre as entidades, pode ser feita através da ferramenta de documentação de uma ferramenta CASE ou do próprio gerenciador de banco de dados (no caso o Microsoft® Access®).

Solução:

I. Utilizando-se a documentação descrita no Anexo B, obtém-se a descrição dos relacionamentos entre as entidades e as informações de cardinalidade associadas aos relacionamentos.

De acordo com a tabela 2, deve-se observar que a existência de uma chave estrangeira de uma entidade, indica a existência de um relacionamento entre entidades do MER.

Avaliação: De acordo com os itens alcançados, tem-se a capacidade de identificar os relacionamentos entre as entidades do sistema, contribuindo para a construção do MER.

Justificativa: De acordo com os resultados obtidos nos padrões anteriores, será utilizado como entrada o MER completo no próximo padrão.

Produto Obtido:

MER – Modelo Entidade Relacionamento do sistema legado.

A Figura 6 ilustra os relacionamentos das entidades, estabelecendo o MER do sistema legado.

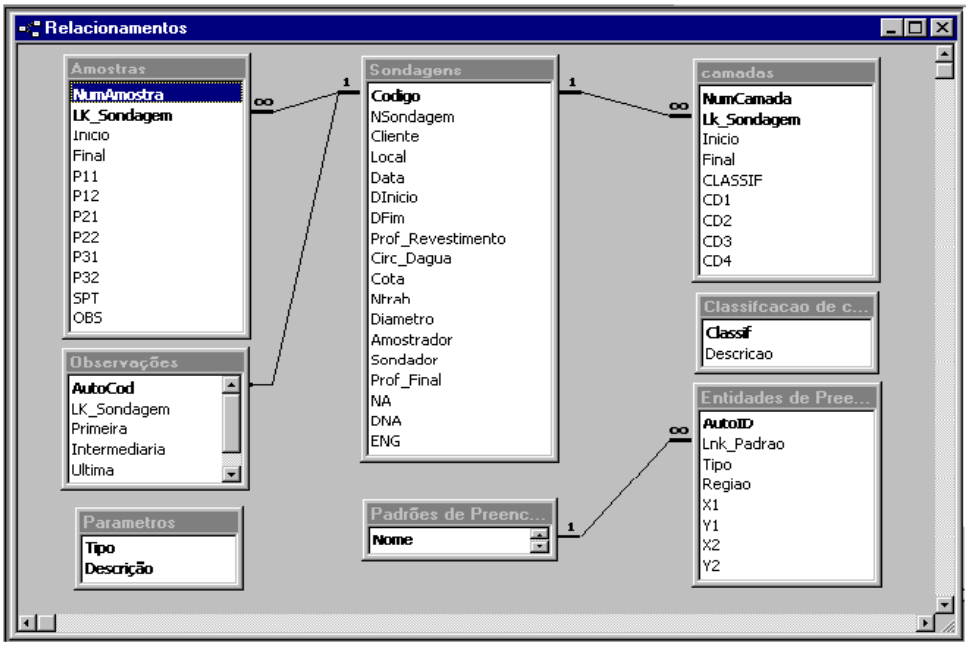


FIGURA 6 - MER do Sistema Legado:

4. Nome: Criar Visão OO dos Dados

Intuito: Criar uma visão orientada a objetos dos dados.

Problema: Transformar um modelo de dados desenvolvido de forma procedimental em um modelo de dados orientado a objetos.

Influências:

O resultado obtido no passo anterior, demonstrado através da Figura 6 fornece a base para analisar o relacionamento básico entre as entidades. Deve ser dada atenção especial para os casos de agregação existentes no modelo: Sondagens-Amostras e Sondagens-Camadas;

Solução:

I. A partir do MER, construir o Diagrama de Pseudo-Classes (candidatas a classes) do sistema legado, gerando, assim, o MASA, Modelo de Análise do Sistema Atual. O nome original das entidades do MER deve ser mantido.

II. Considerar cada entidade do MER como uma pseudo-classe.

III. Buscar pares de relacionamentos n-para-n, no MER, que podem ser representados como *link* de atributo no modelo orientado a objetos, de acordo com a funcionalidade do sistema legado e com os conceitos da orientação a objetos.

IV. Buscar nos relacionamentos um-para-n, no MER, aqueles que podem representar o Princípio Todo-Parte (Agregação por: Referência ou Valor), considerando a funcionalidade do sistema legado e os conceitos da orientação a objetos.

Avaliação: O Cluster "Modelar os Dados do Legado" termina com o padrão que nos fornece a visão OO do Modelo de Análise do Sistema Atual (MASA), que através dele será continuado a etapa de engenharia reversa do Sistema.

Justificativa: Os resultados encontrados até o presente momento, serão utilizados mais adiante para dar continuidade à análise do sistema legado.

Produto Obtido:

MASA - Modelo de Analise do Sistema Atual: Diagrama de Pseudo-Classes.

A Figura 7 ilustra o MASA do sistema legado.

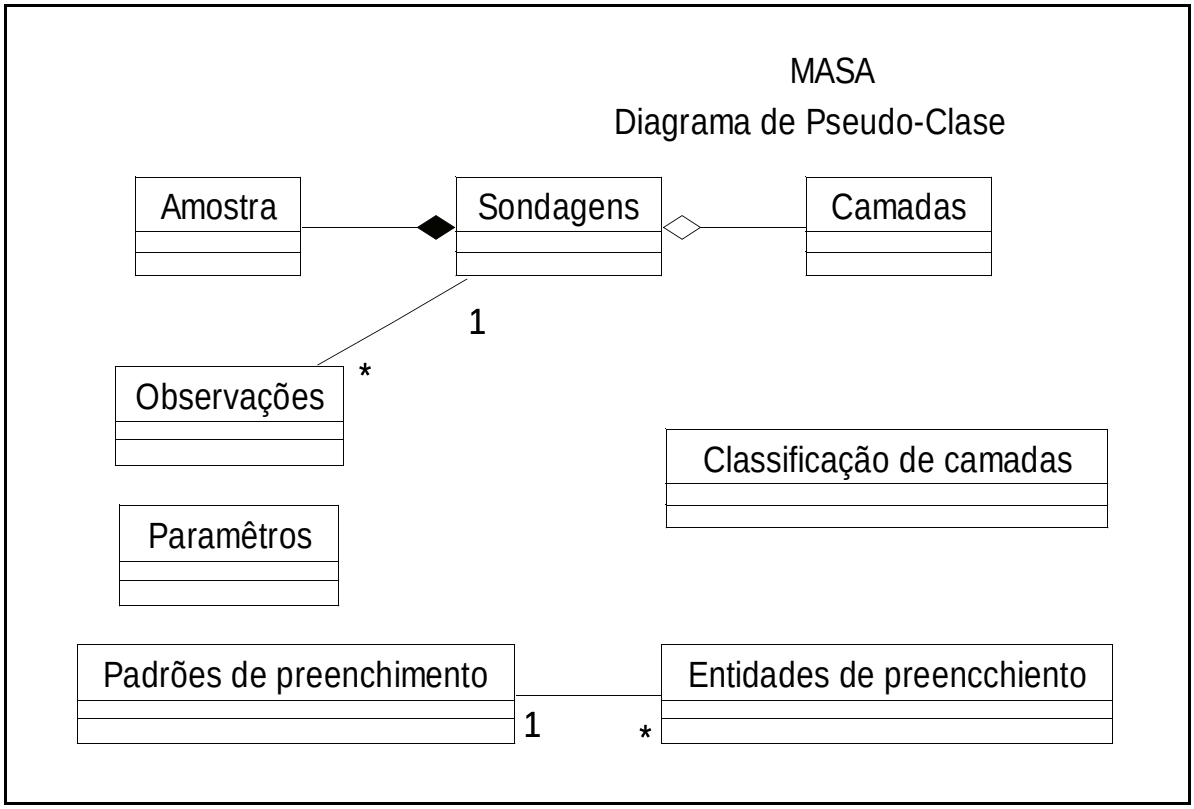


FIGURA 7 - MASA do Sistema Legado:

CLUSTER 2: MODELAR A FUNCIONALIDADE DO SISTEMA

5. Nome: Obter Cenários

Intuito: Obter os Cenários do Sistema através da análise das interfaces do sistema em operação.

Problema: Há grande variedade de interfaces nos sistemas legados. No entanto, é necessário identificar os Cenários do Sistema para obter um modelo que represente a sua funcionalidade.

Influências:

Sistemas implementados em linguagens como Delphi, etc., utilizam telas de menus para o acesso à funcionalidade.

Cada Cenário, em sistemas legados, é representado por um Menu e cada opção do menu corresponde a um ou mais programas fonte.

Solução:

Construir a Tabela Cenários do Sistema, com duas colunas. A primeira coluna contém as opções do menu principal e, a segunda, os submenus de cada opção do menu principal. Para essa construção, obter os cenários do sistema através da observação do sistema legado em operação.

A Figura 8 mostra a interface inicial do Sistema CadSond; e a Tabela 3 mostra os cenários obtidos dessa interface.

FIGURA 8 - Menu inicial do Sistema de Cadastro de Sondagens:

Avaliação: Através da interface do sistema, tem-se como objetivo obter todos os menus, que são os cenários e submenus do mesmo, caso exista.

Justificativa: Esse padrão é utilizado para a construção dos diagramas de Caso de Usos.

Padrões Relacionados: Esse padrão é a entrada para o padrão Construir Diagramas de Caso de Usos.

Produto Obtido:

Tabela Cenários do Sistema.

TABELA 3 – Tabela Cenário do Sistema:

Cenários **Opções do Menu Principal**	**Conteúdo** **Opções do sub-menus de cada opção principal**
Sistema	Sair
Cadastro	Sondagens Parâmetros Padrões
Ajuda	Sobre

6. Nome: Construir Diagramas de Caso de Usos

Intuito: Construir todos os Diagramas de Caso de Usos do sistema a partir da Tabela Cenários do Sistema elaborada pelo padrão Obter Cenários.

Problema: Documentar as informações da funcionalidade de um sistema legado procedimental, durante a engenharia reversa orientada a objetos.

Influências:

Obter os Caso de Usos a partir do código procedimental é difícil.

Os cenários obtidos a partir das opções de menu, viabilizam a construção dos respectivos diagramas de Caso de Usos.

Solução:

I. Considerar como Caso de Uso cada item da segunda coluna da Tabela Cenários do Sistema. Observe que cada item da coluna cenário “possui um conteúdo com várias opções como mostrado na tabela 3. Cada item da segunda coluna (Conteúdo), passa a corresponder a um Caso de Uso no Diagrama em questão.

Não há Diagrama de Caso de Uso para o Cenário ajuda, pois o menu Ajuda que contém o sub-menu Sobre não realiza nenhum procedimento, como é ilustrado na Figura 9.

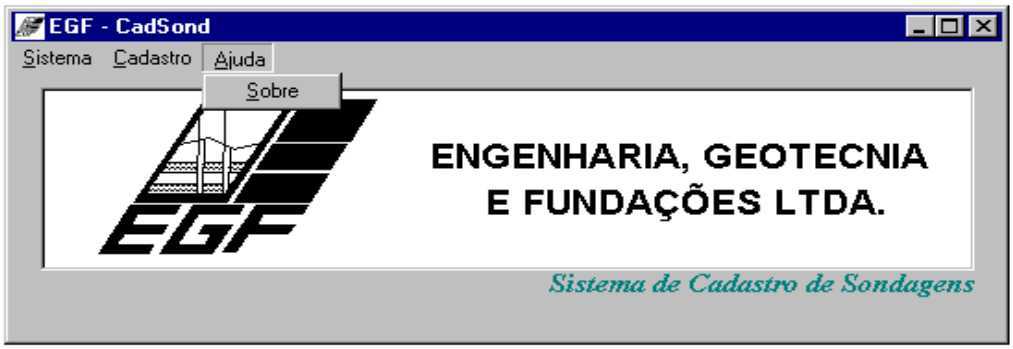


FIGURA 9 - Menu Ajuda:

Avaliação: A partir desse padrão pode-se elaborar uma documentação da funcionalidade do sistema.

Justificativa: Com os Diagramas obtidos pode-se utilizá-los para descrever os Caso de Uso.

Produto Obtido:

Diagramas de Caso de Uso do sistema.

As Figuras 10 e 11 exibem os Diagramas de Caso de Uso do Sistema de Cadastro de Sondagens que foram representados num nível de abstração alto. O nome para cada Caso de Uso deve ser definido pelo próprio engenheiro de software, bem como os eventos associados a cada Caso de Uso, pela observação do sistema legado em operação, quando da ativação de cada opção do Menu correspondente.

FIGURA 10 - Diagrama de Caso de Uso para o Cenário Sistema:

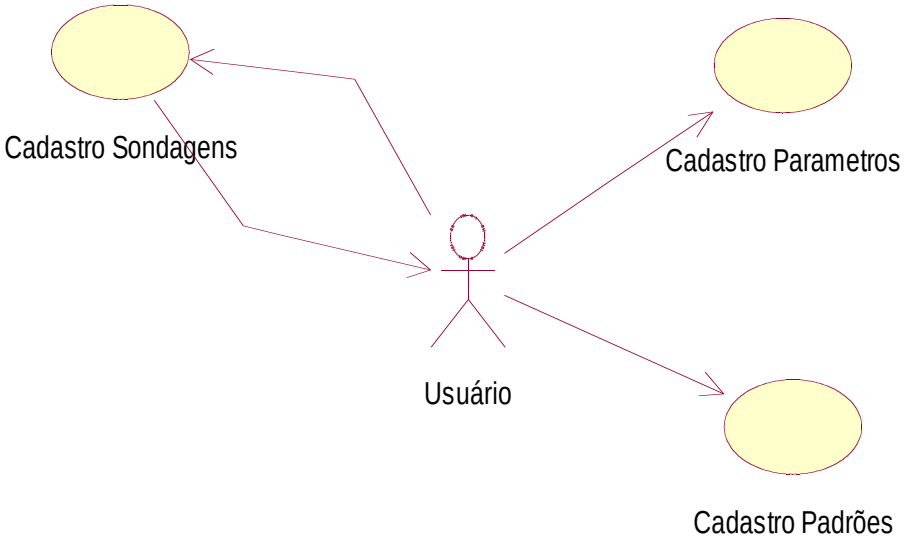


FIGURA 11 - Diagrama de Caso de Uso para o Cenário Cadastro:

7. Nome: Elaborar a Descrição de Caso de Usos

Intuito: Elaborar a Descrição correspondente a cada Caso de Uso obtido pelo padrão Construir Diagramas de Caso de Uso.

Problema: É necessário registrar a lógica da funcionalidade do sistema para facilitar sua futura reengenharia.

Influências:

Há falta de documentação do sistema que registre a lógica da funcionalidade.

Tem-se o código fonte disponível para obter a lógica da funcionalidade.

O engenheiro de software tem experiência de uso da linguagem de programação.

Solução:

I. Para cada Caso de Uso obtido pelo padrão Construir Diagramas de Caso de Usos, elaborar a sua Descrição a partir de trechos do código fonte correspondente à opção do Menu ativada, quando da construção do respectivo Caso de Uso.

Avaliação: Com o uso desse padrão pode-se entender de forma mais detalhada a funcionalidade do sistema.

Justificativa: Sem o uso desse padrão seria mais difícil dar continuidade ao processo de engenharia reversa.

Padrões Relacionados: Esse padrão tem como entrada o padrão Construir Diagramas de Caso de Usos e é utilizado como entrada para o padrão Tratar Anomalias.

Produto Obtido:

Descrição de cada Caso de Uso do sistema.

A seguir são apresentadas as descrições dos casos de uso do sistema:

- **Caso de Uso**: **Sair do Sistema**.

Ator: Usuário.

Pré-condição: Não Há.

Invariante: Não Há.

Pós-condição: A execução do programa chega ao fim, e os dados que estão na tela são salvos.

1. Usuário solicita o menu saída;
2. Sistema salva e fecha a conexão dos dados;
3. Sistema finaliza execução.

- **Caso de Uso: Cadastro de Parâmetros.**

Ator: Usuário.

Pré-condição: Não Há.

Invariante: Não Há.

Pós-condição: É possível a realização de cadastro de sondagem.

1. Usuário solicita cadastro de Parâmetros;
2. Sistema exibe na tela os Parâmetros;
3. Usuário entra com os valores dos Parâmetros (Engenheiro, Desenhista, Sondadores e Diretório) e com o Diretório default para a criação dos arquivos de saída;
4. Sistema grava automaticamente os valores para os parâmetros informados;
5. Fecha a tela de edição dos parâmetros, Sistema retorna ao menu Cadastro.

- **Caso de Uso**: Cadastro Sondagens.

Ator: Usuário.

Pré-condição: Os parâmetros e padrões devem estar previamente cadastrados.

Invariante: Os valores dos parâmetros não podem mudar.

Pós-condição: Após uma sondagem estar cadastrada é inserido um registro no banco de dados..

1. Usuário solicita cadastro de Sondagens;
2. Sistema exibe a listagem de sondagens cadastradas, e dá ao usuário a opção de localizar uma sondagem específica, através da navegação pelos registros cadastrados, ou através de um valor para busca e um critério de ordenação;
3. O sistema disponibiliza ao usuário a opção de selecionar um dos comandos:
    - Alterar os dados de uma sondagem já cadastrada, através da visualização dos detalhes da sondagem;
    - Criar uma nova sondagem. Caso o usuário escolha essa opção, o sistema exibirá automaticamente os detalhes da nova sondagem criada, para que o usuário proceda ao preenchimento dos dados da nova sondagem.;
    - Excluir os dados da sondagem selecionada. Para concluir essa operação é solicitada a confirmação do usuário;
    - Para uma sondagem cadastrada, o usuário pode solicitar a geração do arquivo DXF correspondente;
    - Fechar o cadastro de sondagens, retornando à tela principal do programa.

- **Caso de Uso**: Cadastro Padrões.

Ator: Usuário.

Pré-condição: Não Há.

Invariante: Não Há.

Pós-condição: É possível a realização de cadastro de sondagem.

1. Usuário solicita cadastro de padrões;
2. Sistema abre uma tela que permite que o usuário realize manutenção do cadastro de padrões, incluindo, alterando ou excluindo um padrão existente;
3. Para cada padrão em edição o usuário deve informar os objetos gráficos (pontos e linhas) que compõe a representação gráfica do padrão;
4. O sistema grava automaticamente os valores informados;
5. Fecha a tela de edição de padrões e retorna ao menu cadastro.

8. Nome: Tratar Anomalias

Intuito: Analisar as Descrições de Caso de Usos para tratar as anomalias, definindo, assim, os possíveis métodos das pseudo-classes (candidatas à classe) do sistema, obtidos pelo padrão Criar Visão OO dos Dados.

Anomalia: Quando a Descrição do Caso de Uso faz acesso/atualização a arquivos de dados que não pertencem à pseudo-classe a que ele se relaciona, então se infere que esse procedimento (Descrição do Caso de Uso) é anômalo. A anomalia pode ser do tipo:

o+ Quando o procedimento é observador de duas ou mais classes;

c+ Quando o procedimento é construtor de duas ou mais classes;

oc Quando o procedimento é observador de uma classe e construtor de outra;

o+c Quando o procedimento é observador de duas ou mais classes e construtor de outra;

oc+ Quando o procedimento é observador de uma classe e construtor de duas ou mais classes;

o+c+ Quando o procedimento é observador de duas ou mais classe e construtor de duas ou mais classe;

i Quando o procedimento é dependente da implementação e que não se refere à classe alguma.

Problema: Em sistemas construídos de forma procedimental pode existir um número elevado de anomalias em cada procedimento (código fonte). Deve-se eliminar essas anomalias uma vez que, em sistemas orientados a objetos, os métodos estão associados a uma única classe. Então, é necessário transformar o código fonte correspondente ao procedimento anômalo (Descrição do Caso de Uso) em métodos.

Influências:

Quando o sistema não tem implementação orientada a objetos, a análise dos procedimentos pode ser complexa.

Requer do engenheiro de software experiência para extrair as anomalias a partir da Descrição do Caso de Uso.

Têm-se as Descrições de Caso de Usos geradas a partir do código fonte.

Solução:

Para cada Descrição de Caso de Uso, obtida pelo padrão Elaborar a Descrição de Caso de Usos, construa a Tabela Detalhes de Implementação, com seis colunas:

A primeira coluna contém o nome do Caso de Uso;

A segunda coluna contém o nome de todos os programas/módulos (códigos fontes) utilizados na construção do Caso de Uso;

A terceira coluna contém o nome das pseudo-classes do MASA correspondentes aos arquivos de dados a que se tem acesso em cada programa/módulo;

A quarta coluna contém a classificação quanto ao tipo de acesso ao arquivo de dados, realizado pelo procedimento;

A quinta coluna contém os nomes dos possíveis métodos quando as anomalias forem eliminadas;

A sexta coluna contém os nomes das pseudo-classes revisadas, a que os respectivos métodos estão associados.

I. Iniciar a construção da Tabela Detalhes de Implementação, verificando para cada Descrição de Caso de Uso o tipo de acesso aos arquivos. Transformar o trecho fonte que consulta/altera diversos arquivos em possíveis métodos, eliminando assim o múltiplo acesso a arquivos. A parte do código que somente consulta um arquivo é retirada desse, e uma chamada a esse método é feita para que a funcionalidade do sistema seja mantida.

II. Continuar a construção da Tabela Detalhes de Implementação, renomeando as pseudoclasses que não representam completamente a informação dentro do contexto do sistema.

III. Acrescentar na Tabela Detalhes de Implementação, os nomes de entidades do MER (arquivo de dados) existentes no código fonte que foram usados apenas para atender a necessidades de implementação. Esses arquivos são temporários e não fazem parte da funcionalidade do sistema. Portanto, toda pseudo-classe que é criada por algum procedimento classificado com a anomalia **i** deve ser desconsiderada.

IV. Acrescentar na coluna pseudo-classes revisadas, da Tabela Detalhes de Implementação, as pseudo-classes correspondentes a trechos do código fonte usados para implementar informações adicionais em memória. Por

exemplo, tabelas de descontos em função da quantidade comprada, que foram implementadas em trechos de código ao invés de estarem persistidas em arquivos de dados. Em sistemas procedimentais esses artifícios eram comuns para melhorar o desempenho do sistema. Como essas pseudo-classes são geradas diretamente do código fonte, elas não aparecem como entidades do MER e, conseqüentemente, como pseudo-classes do MASA, pelo fato de não existir um arquivo de dados correspondente.

Avaliação: Através desse padrão pode-se visualizar quais arquivos fontes se relacionam com quais arquivos de dados e classificar o tipo das anomalias existentes no código.

Justificativa: Esse padrão é importante pois, com o resultado final obtido dele, podemos continuar a análise do sistema, nos padrões do próximo Cluster (Modelar o Sistema Orientado a Objetos).

Padrões Relacionados: Esse padrão tem como entrada o padrão Elaborar a Descrição de Caso de Usos e é utilizado como entrada para os padrões do *Cluster* Modelar o Sistema Orientado a Objetos.

Produto Obtido:

Tabela Detalhes de Implementação.

TABELA 4 – Tabela Detalhes de Implementação:

Caso de Uso	Códigos Fontes Apropriados	Pseudo Classes (MASA)	Tipo de Anomalia	Possíveis Métodos	Pseudo Classes Revisadas

Cadastro Sondagens	CadSond.EXE	Sondagens	O+C+	Editar()	Sondagens
				Incluir()	Sondagens
				Excluir()	Sondagens
				Imprimir()	Sondagens
		Camadas	O+C+	Editar()	Camadas
				Incluir()	Camadas
				Excluir()	Camadas
		Amostras	O+C+	Editar()	Amostras
				Incluir()	Amostras
				Excluir()	Amostras
		Observação	O+C+	Editar()	Observação
				Incluir()	Observação
				Excluir()	Observação
		Padrões de Preenchimento	C+	Consultar()	Padrões de Preenchimento
		Classificação de camadas	C+	Consultar()	Classificação de camadas
Cadastro Parâmetros	CadSond.EXE	Parâmetros	C+	Editar()	Parâmetros
				Incluir()	Parâmetros
				Excluir()	Parâmetros
Cadastro Padrões	CadSond.EXE	Padrões de Preenchimento	O+C+	Editar()	Padrões de Preenchimento
				Incluir()	Padrões de Preenchimento
				Excluir()	Padrões de Preenchimento
		Entidade de Preenchimento	O+C+	Editar()	Entidade de Preenchimento
				Incluir()	Entidade de Preenchimento
				Excluir()	Entidade de Preenchimento

CLUSTER 3: MODELAR O SISTEMA ORIENTADO A OBJETOS

9. Nome: Definir as Classes

Intuito: Iniciar a construção do Diagrama de Classes do sistema.

Problema: Existem várias informações levantadas pelo processo de engenharia reversa: MER, MASA, Diagramas de Caso de Usos, Descrições de Caso de Usos e Tabela Detalhes de Implementação. No entanto, elas precisam ainda ser consolidadas e apresentadas na visão de orientação a objetos.

Influências:

As classes são obtidas por meio da coluna Pseudo-Classes Revisadas, da Tabela Detalhes de Implementação.

Solução:

I. Transformar cada pseudo-classe revisada, da Tabela Detalhes de Implementação, em uma classe no Diagrama de Classes, conforme mostra a Figura 14 .

II. Gerar os relacionamentos no diagrama de classes a partir dos relacionamentos existentes do MASA. Além disso, para as pseudo-classes que foram geradas diretamente do código fonte, criar relacionamentos que representam o Princípio Todo-Parte (Agregação por: Referência ou Valor), bem como *link* de atributo, conforme os conceitos da orientação a objetos, discutidos pelo padrão Criar Visão OO dos Dados. A Figura 16 ilustra os respectivos relacionamentos.

Avaliação: O objetivo desse padrão é obter todas as classes e seus relacionamentos, visando já uma transformação para a abordagem orientada a objetos.

Justificativa: Esse padrão é capaz de definir os atributos e continuar a construção do diagrama de classe final.

Produto Obtido:

Diagrama de Classes (Inicial) do sistema: classes definidas.

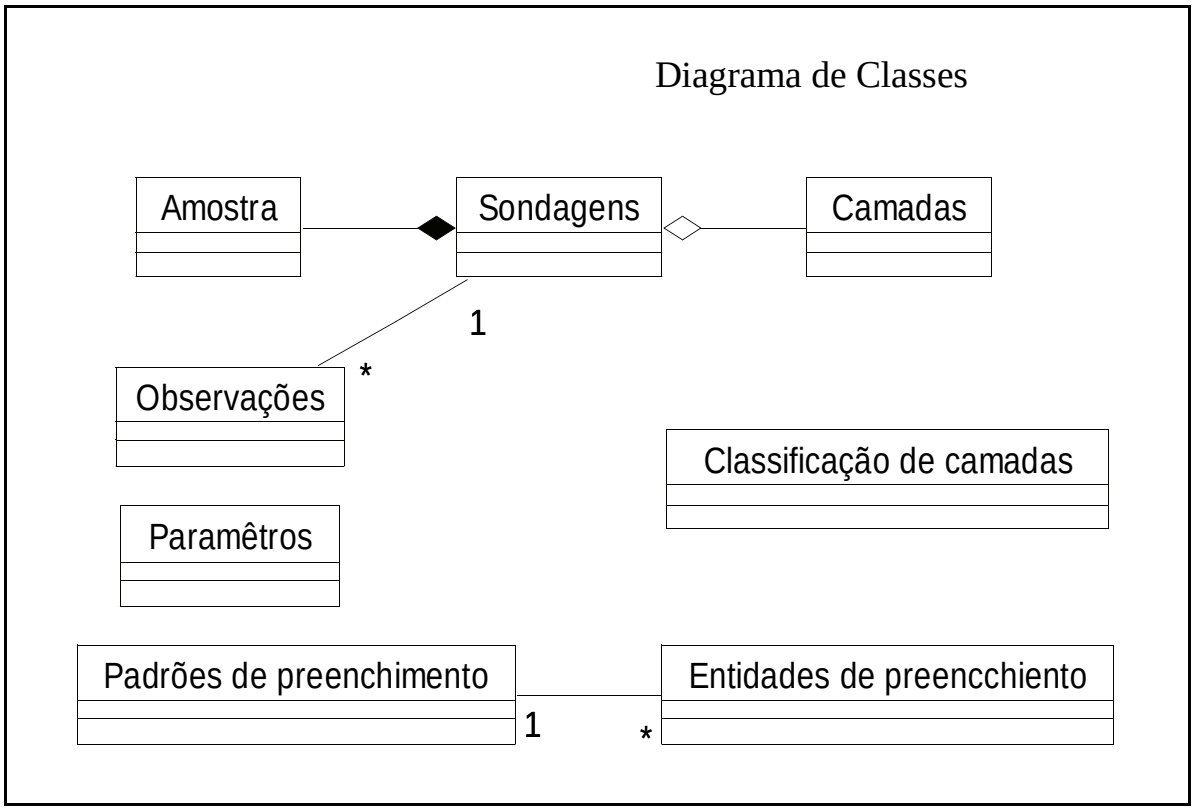


FIGURA 12 - Diagrama de Classes (Construção da Classe):

10. Nome: Definir Atributos

Intuito: Definir os atributos das classes pertencentes ao Diagrama de Classes em construção.

Influências:

Para obter os atributos, bem como as informações sobre relacionamentos entre as entidades, pode-se utilizar a ferramenta do próprio Access.

Existem ferramentas que auxiliam a análise das estruturas dos dados do legado.

Solução:

Os campos dos arquivos de dados, representados como pseudo-classes no MASA são, na sua maioria, transformados em atributos das classes correspondentes. Também pode-se visualizar na Figura 6 os atributos de cada classe apresentados através da ferramenta Access. A Figura 13 ilustra os Diagrama de Classes com os atributos.

Os casos em que não há correspondência direta deverão ser tratados em paralelo avaliando-se as questões de hierarquias abordadas pelo padrão Analisar Hierarquias.

Avaliação: Neste padrão o Diagrama de Classes é melhor detalhado.

Padrões Relacionados: Esse padrão tem como entrada o padrão Definir as Classes e serve como entrada para o padrão Definir Métodos. Pode ser aplicado, se necessário, em paralelo ao padrão Analisar Hierarquias.

Produto Obtido:

Diagrama de Classes (continuação) do Sistema: classes e atributos definidos.

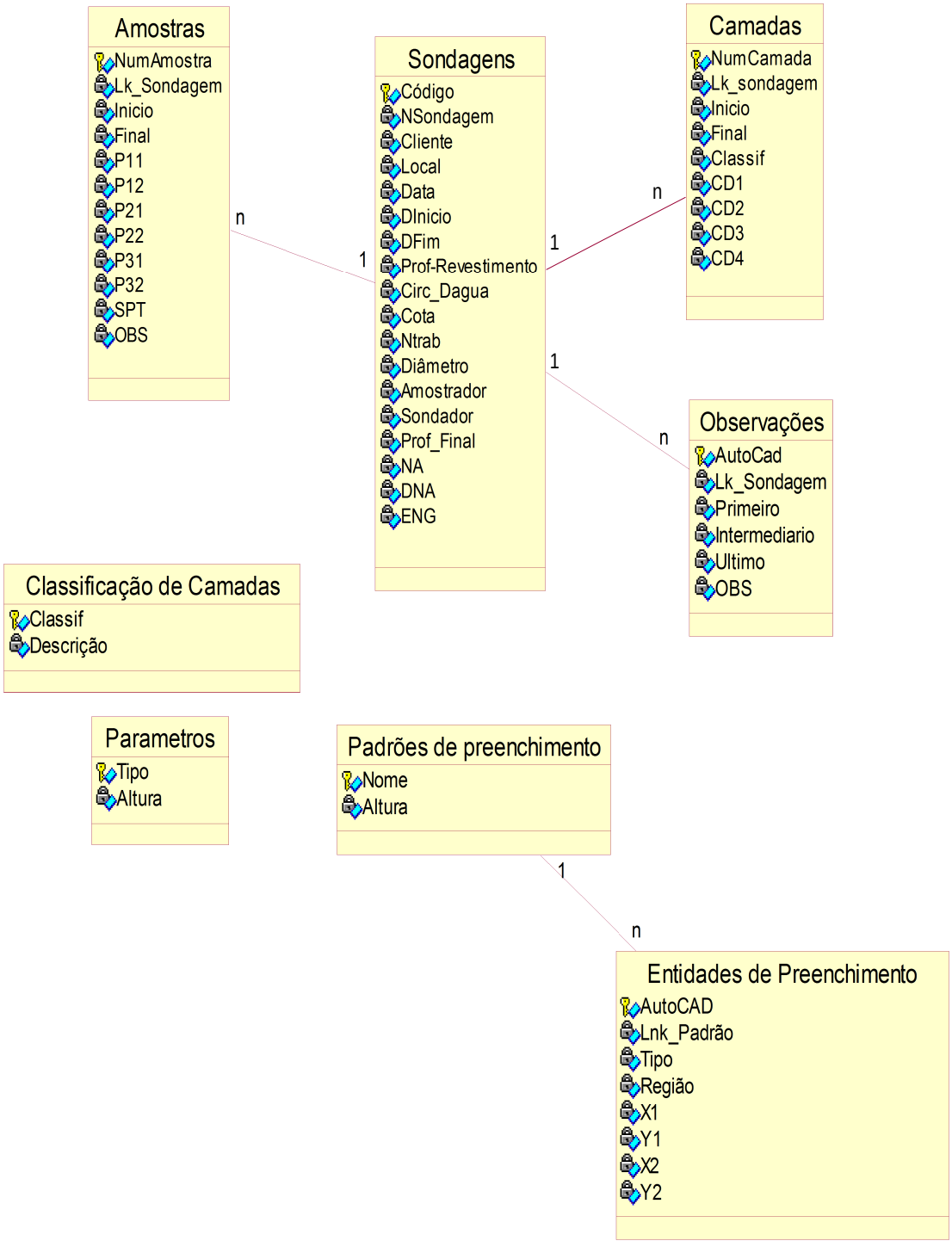


FIGURA 13 - Diagrama de Classes (Definição dos Atributos das Classes):

11. Nome: Analisar Hierarquias

Intuito: Fornecer um mecanismo para descobrir possíveis papéis (mais de uma função) e hierarquias de herança que possam estar contidas nos arquivos de dados do sistema legado.

Justificativa: Esta etapa não foi realizada, pois o sistema em questão não apresenta Hierarquia, ou seja, não possui campos opcionais (campos que assumem valores dependendo de um contexto. Por exemplo, um arquivo de dados de nome Cliente possui os campos CPF e CGC, sendo que para cada cliente, somente um desses campos assumirá valor, dependendo do tipo do cliente – pessoa física ou jurídica); e também não contém relacionamentos do tipo um-para um.

12. Nome: Definir Métodos

Intuito: Definir os métodos das classes pertencentes ao Diagrama de Classes em construção.

Problema: Identificar métodos de cada classe a partir da tabela detalhes de implementação.

Influências:

Definir cada um dos métodos a partir dos procedimentos anômalos existentes no código fonte legado é difícil.

Existe indício dos prováveis métodos na Tabela Detalhes de Implementação.

Solução:

Representar os métodos que estão relacionados na coluna Possíveis Métodos, da Tabela Detalhes de Implementação, nas classes respectivas às pseudo-classes, especificadas na coluna Pseudo-Classes Revisadas, da mesma tabela. Assim, obtêm-se os métodos das classes no Diagrama de Classes em construção por meio da Descrição do Caso de Uso correspondente.

A Figura 14 ilustra o Diagrama de Classes já com os métodos definidos.

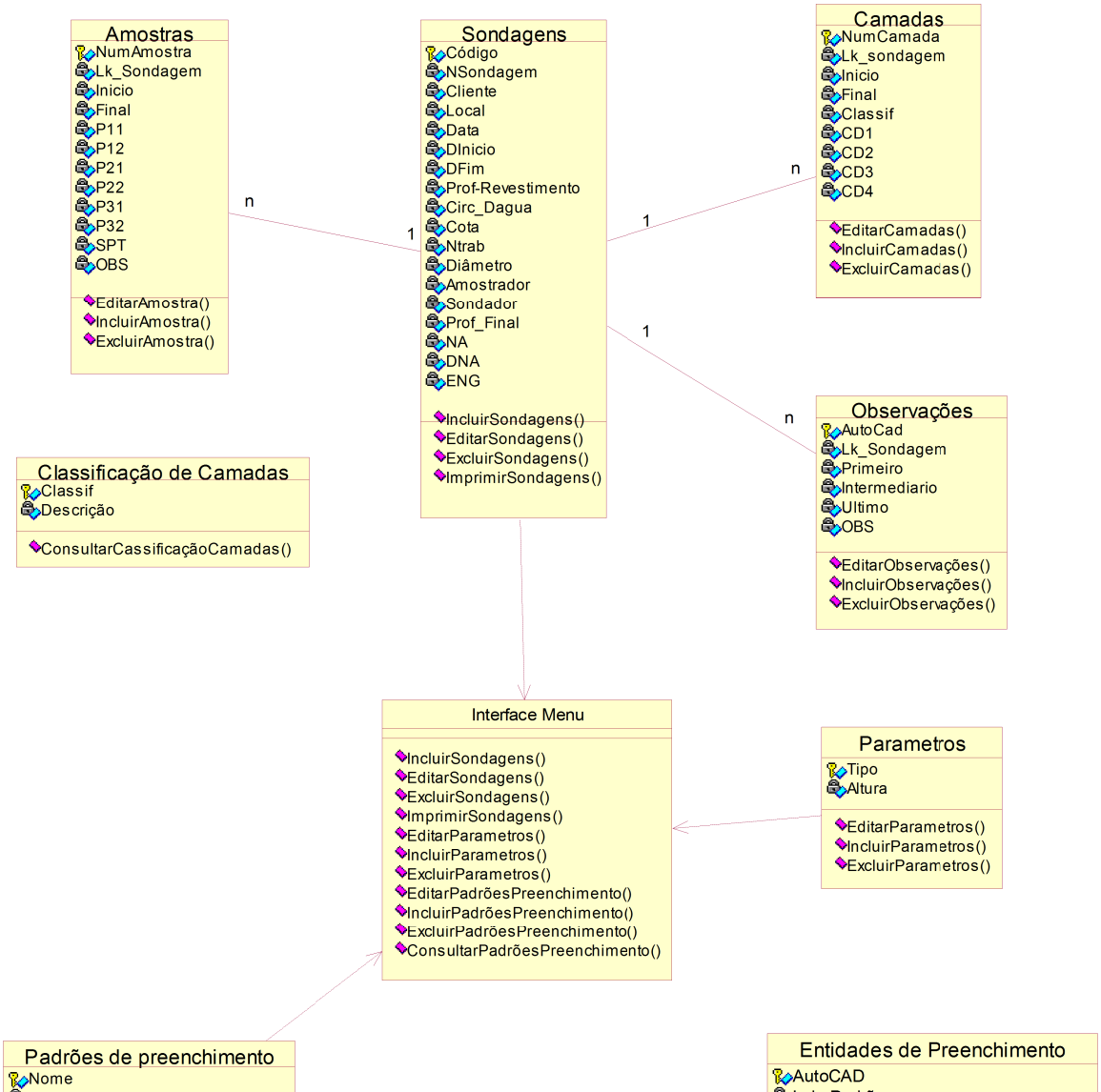


FIGURA 14 - Diagrama de Classes (Definição dos Métodos das Classes):

Informações Adicionais: O engenheiro de software deve observar trechos do código fonte que possam ser caracterizados como Polimorfismo, gerando tantos métodos quantos forem necessários, consolidando, assim, o diagrama de classes do sistema orientado a objetos.

Avaliação: Este padrão é capaz de determinar a construção do Diagrama de Classes, com seus respectivos métodos e atributos.

Produto Obtido:

Diagrama de Classes do Sistema.

13. Nome: Construir Diagramas de Seqüência

Intuito: Documentar as Regras de Negócio do sistema legado para facilitar a sua futura reengenharia.

Influências:

Métodos já definidos a partir das Descrições de Caso de Usos, viabilizam o problema.

Solução:

I. Construir todos os Diagramas de Seqüência, a partir de cada Descrição de Caso de Uso, obtida pelo padrão Elaborar Descrição de Caso de Usos. Considere, para a sua construção, o processo de eliminação de anomalias efetuado pelo padrão Tratar Anomalias, quando foram gerados tantos métodos quantos foram necessários, para manter a mesma funcionalidade do sistema legado. Agora, no Diagrama de Seqüência, é apresentada graficamente a interação desses métodos.

Avaliação: Com a aplicação deste padrão pode-se detalhar mais as funções que começaram a ser especificadas no padrão Elaborar Descrição de Caso de Uso.

Justificativa: Este padrão conclui a etapa de engenharia Reversa de acordo com o padrão adotado.

Padrões Relacionados:

Esse padrão tem como entrada os padrões Elaborar Descrição de Caso de Usos, Definir Métodos e Tratar Anomalias.

Produtos Obtidos:

Diagramas de Seqüência do Sistema.

A Figura 15 ilustra o Diagrama de Seqüência para o Caso de Uso Sair do Sistema.

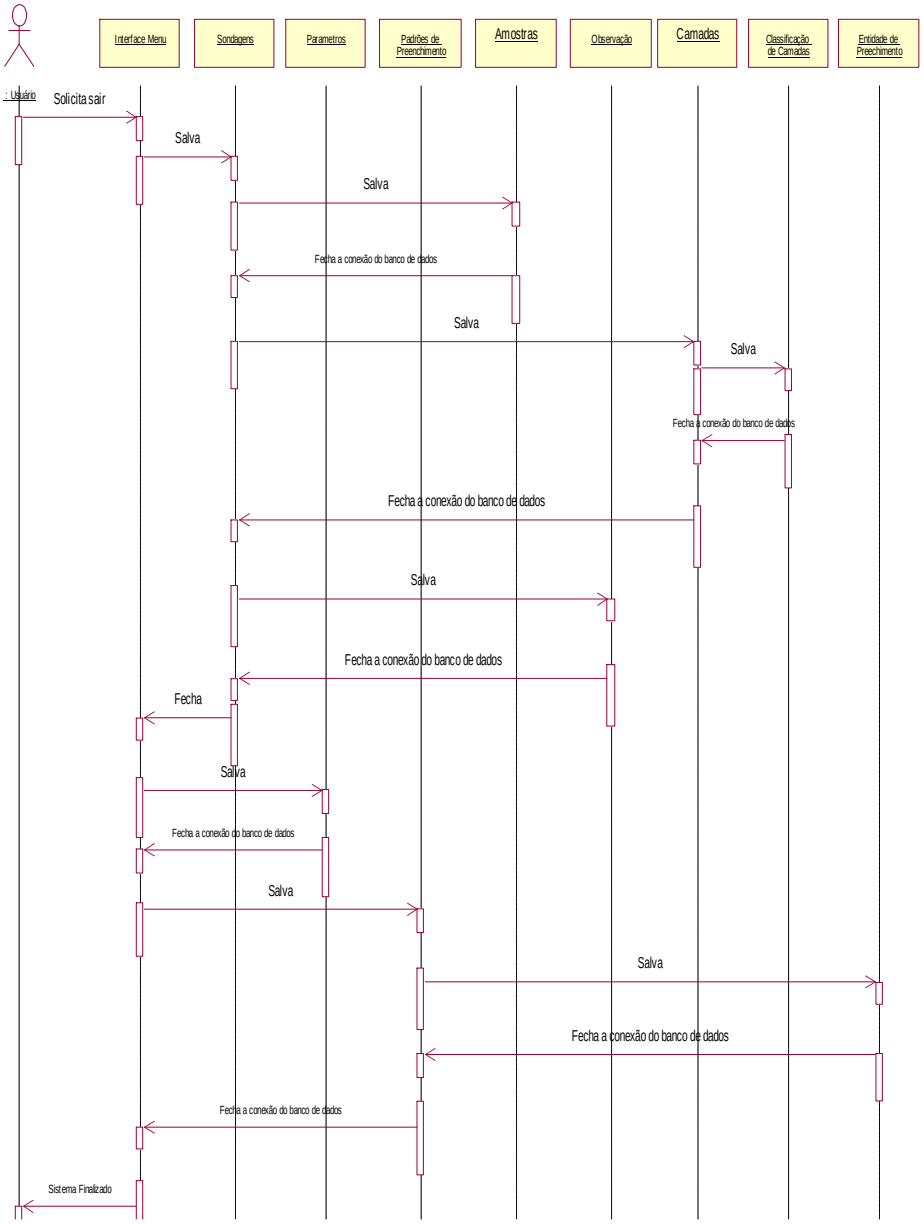


FIGURA 15 - Diagrama de Seqüência do Caso de Uso para o Cenário Sistema:

A Figura 15 ilustra o Diagrama de Seqüência para o Caso de Uso Sair do Sistema.

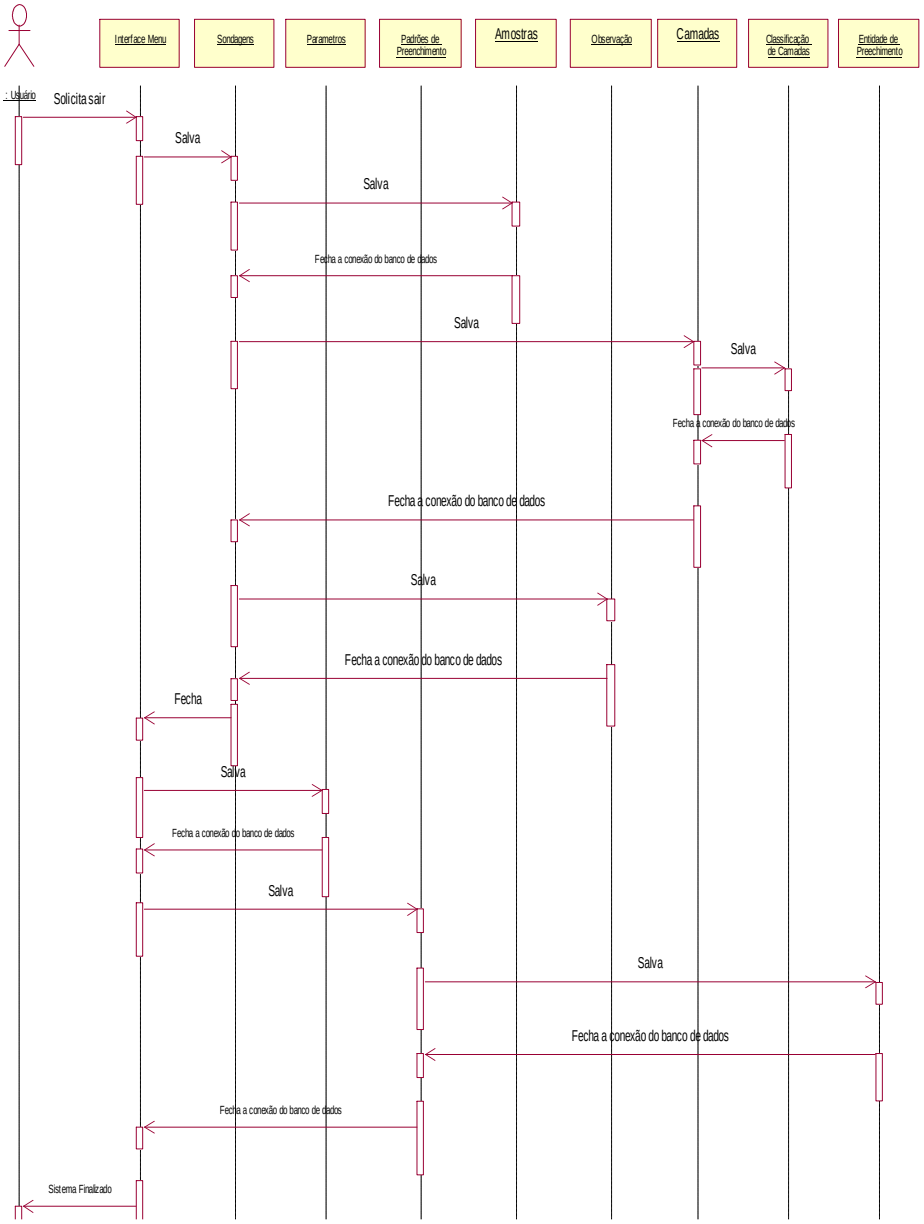


FIGURA 15 - Diagrama de Seqüência do Caso de Uso para o Cenário Sistema:

A Figura 16 ilustra o Diagrama de Seqüência para o Caso de Uso Cadastro de Sondagens.

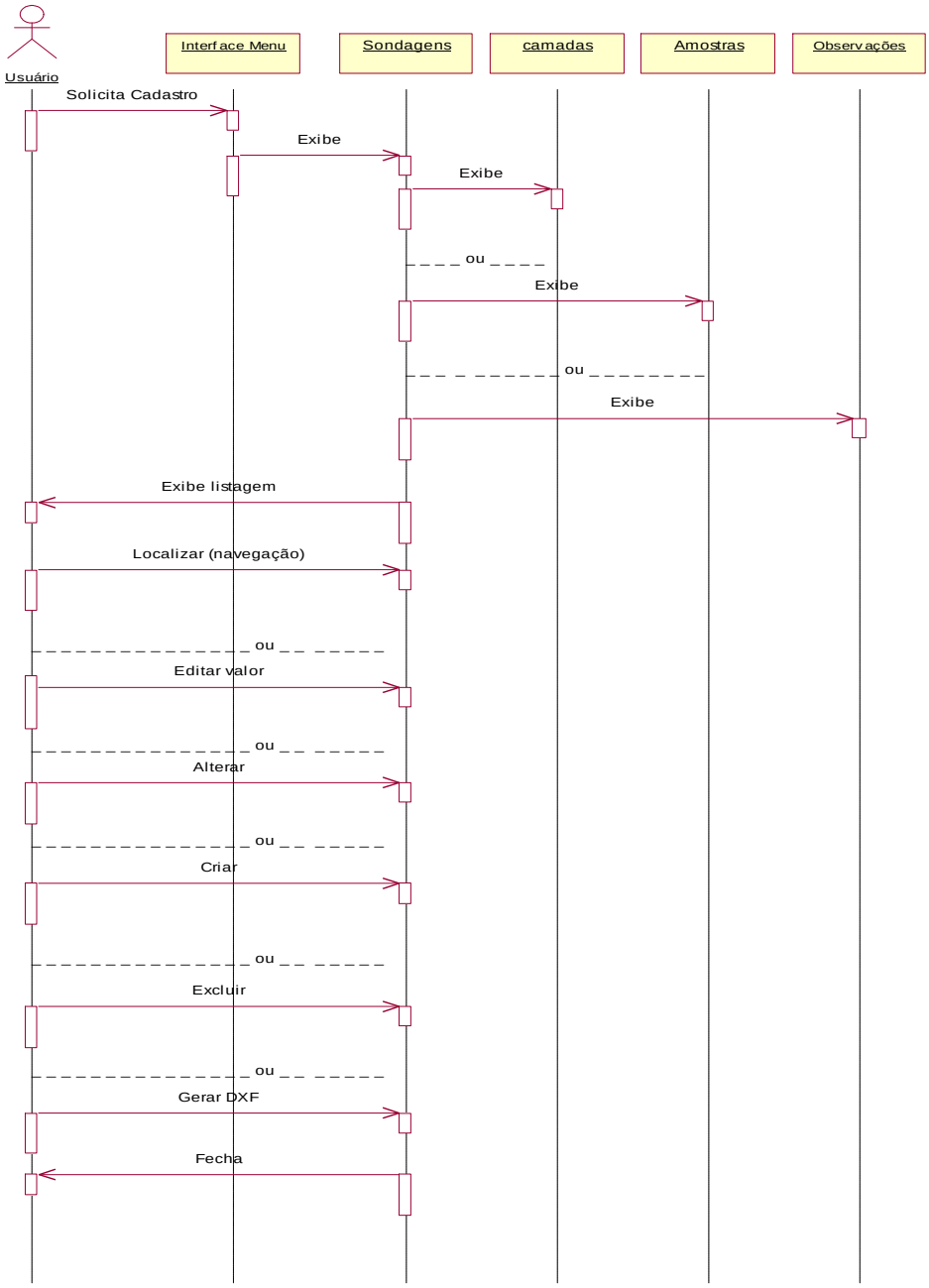


FIGURA 16 - Diagrama de Seqüência do Caso de Uso para o Cenário Cadastro:

A Figura 17 ilustra o Diagrama de Seqüência para o Caso de Uso Cadastro de Parâmetros.

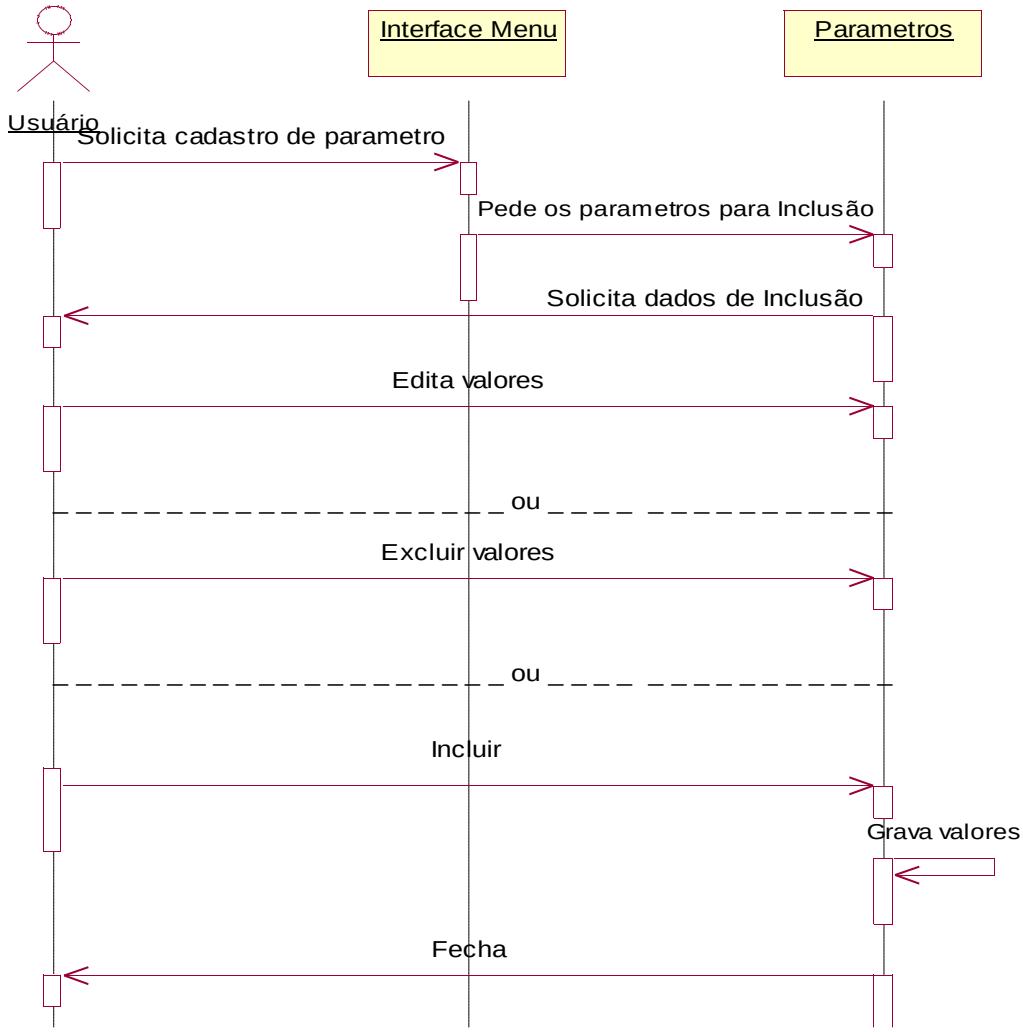


FIGURA 17 - Diagrama de Seqüência do Caso de Uso para o Cenário Cadastro:

A Figura 18 ilustra o Diagrama de Seqüência para o Caso de Uso Cadastro de Padrões.

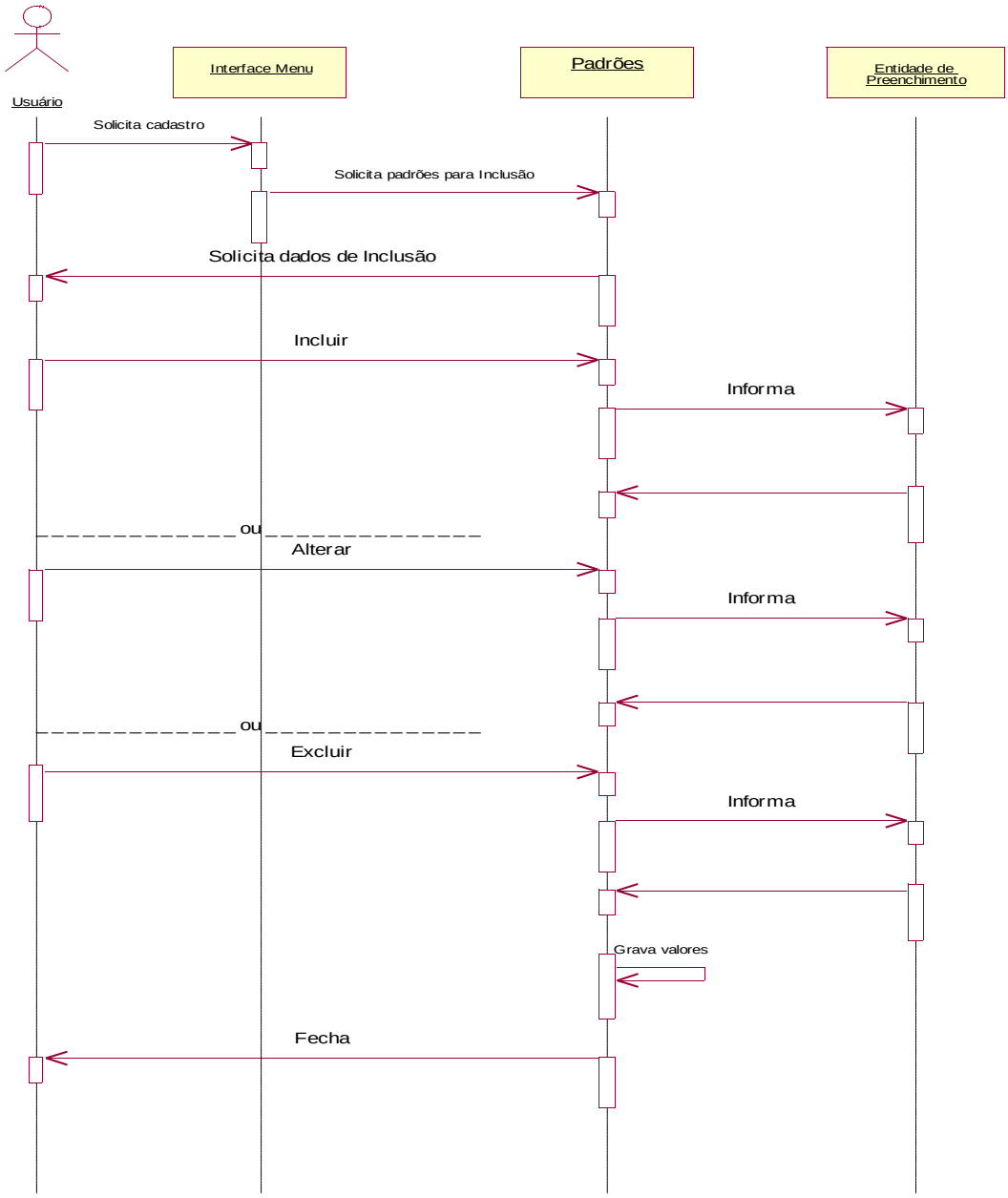


FIGURA 18 - Diagrama de Seqüência do Caso de Uso para o Cenário Cadastro:

Diagrama de Implantação

Os diagramas de implantação são empregados para a modelagem dos aspectos físicos de um sistema orientado a objetos. O diagrama de implantação mostra a configuração dos nós de processamento em tempo de execução e os componentes que nele existem, sendo importantes para o gerenciamento de sistemas executáveis por meio de engenharia reversa (BOOCH, 2000).

Os diagramas de implantação são empregados para modelagem da visão estática da implantação de um sistema. Na maior parte, isso envolve a modelagem da topologia do hadware em que o sistema é executado. Os diagramas de implantação são essencialmente diagramas de classes que focalizam os nós do sistema (BOOCH, 2000).

A Figura 19 ilustra o diagrama de implantação do Sistema de Cadastro de Sondagens, e sua interação com o software AutoCAD.

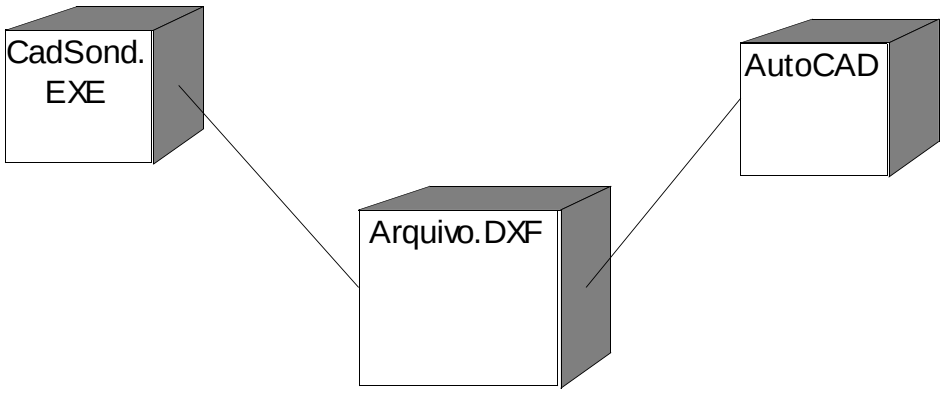


FIGURA 19 - Diagrama de Implantação:

3.4 CONCLUSÕES

Neste capítulo foi apresentado passo a passo as etapas do processo de engenharia reversa de acordo com a metodologia de Família de Padrões de Reengenharia aplicado no Sistema de Cadastro de Sondagens. No próximo capitulo será aplicado à etapa de engenharia avante baseada na mesma família de padrões.

4 ENGENHARIA AVANTE BASEADA NOS PADRÕES DA FAPRE/OO

A etapa do processo de engenharia avante baseada nos Padrões será aplicado neste capítulo no Sistema de Cadastro de Sondagens, visto não ser a implementação o objetivo principal deste trabalho.

CLUSTER 4: GERAR O SISTEMA ORIENTADO A OBJETOS:

14 Nome: Definir a Plataforma:

Intuito: Definir a plataforma de desenvolvimento do novo sistema conforme as exigências do negócio e dos objetivos estratégicos da organização empresarial.

Influências:

O sistema atual foi desenvolvido para aplicação em uma empresa de Sondagens e fundações sediada na cidade de Belo Horizonte – MG. Não havia, na época do desenvolvimento, a expectativa de utilizá-lo em outras empresas da mesma área. Por esse

motivo, o sistema atual suporta exclusivamente a plataforma composta por computadores pessoais operando com o sistema operacional Windows.

Um dos objetivos, é que o “novo” sistema possa ser utilizado em qualquer empresa da área de maneira genérica. Dessa forma, as exigências em termo de Hardware e software devem ser mantidas em níveis baixos, ou seja, o software deve ser o menos exigente possível em termos desses recursos.

Solução:

I. Escolher o Ambiente Operacional:

II. Escolher o Sistema Gerenciador de Banco de dados (SGBD):

III. Escolher a Linguagem de Implementação e o Ambiente de Desenvolvimento:

Produtos Obtidos:

Plataforma de implementação definida:

- A plataforma de implementação será a linguagem Borland Delphi 7.0. A justificativa por essa plataforma se dá, principalmente, pelo fato do sistema original ter sido desenvolvido em uma plataforma semelhante (Delphi 3.0). Futuramente pode-se optar pela plataforma Linux, utilizando-se o software Kylix (2.0).

Ambiente operacional;

- O software será implementado para o ambiente Windows 32 bits, considerando que esta ainda é o ambiente mais difundido nas empresas.

Banco de Dados Relacional;

- O software será desenvolvido para acessar a base de dados através de qualquer SGBD relaciona, através da interface OBDC.

Pacote de Linguagem de Implementação.

- De acordo com a escolha anterior, citada na Plataforma de implementação, será utilizada a linguagem Object Pascal.

15 Nome: Converter o Banco de Dados:

Intuito: Criar as estruturas de Banco de Dados do sistema, fisicamente, de acordo com o Diagrama de Classes.

Influências:

Deve ser criado o Conjunto de comandos em linguagem SQL (Padrão para SGBD's relacionais) para a definição da estrutura do BD. Dessa forma, o esquema do BD pode ser definido de maneira independente do SGBD utilizado.

Produtos obtidos:

Banco de dados do Sistema;

Os scripts (SQL) gerados a partir do diagrama de classes estão incluídos no Anexo C.

16 Nome: Implementar os Métodos:

Intuito: Escrever os métodos obtidos na engenharia reversa na linguagem de programação escolhida para o processo de engenharia avante.

Produtos Obtidos:

Os métodos identificados no Diagrama de Classes desenvolvidos na etapa de Engenharia Reversos do Sistema não foram efetivamente implementados, visto não ser a implementação o objetivo principal deste trabalho.

17 Nome: Realizar melhorias na Interface**:**

Intuito: Conhecer sistemas que atendam, cada vez melhor, as necessidades dos usuários em relação não apenas a critérios de funcionalidade (conjunto de tarefas desempenhadas pelo sistema), mas também a usabilidade.

Influências:

O sistema deve manter uma interface amigável com o usuário, e ao mesmo tempo produtiva. Com esse objetivo, é interessante também observar o estilo de interface utilizada por softwares de domínio dos usuários previstos, ou seja engenheiros da área de construção civil.

Produtos Obtidos:

As interfaces não foram implementadas, uma vez que essa etapa não fez parte do escopo do trabalho.

5 CONCLUSÕES

5.1 CONSIDERAÇÕES FINAIS

Foi proposto neste trabalho uma abordagem para reengenharia de sistema, no qual foi aplicada a metodologia de FaPRE/OO.

Um sistema desenvolvido na linguagem Delphi foi submetido ao processo de engenharia reversa seguindo, passo a passo o reconhecimento de todos os padrões; obtendo-se uma documentação do sistema atual. Pode-se observar que através da reengenharia aplicando o reconhecimento de padrões proposto o sistema torna-se eficiente e de fácil entendimento.

A contribuição mais significativa é o fornecimento de uma estratégia para o reconhecimento de padrões FaPRE/OO tanto na engenharia reversa que pode ser reconhecido após a recuperação do sistema legado quanto na engenharia avante que pode ser reconhecido após o projeto do sistema e utilizado na implementação; garantido também ao engenheiro de software executar de forma correta a reengenharia do sistema legado. O uso desse padrão

implicou no aumento de produtividade, maior uniformidade na estrutura do software e redução da complexidade.

O resultado obtido a partir do reconhecimento desse padrão é um sistema com melhor qualidade e documentação completa e de fácil manutenção que pode surgir de acordo com as necessidades futuras.

5.2 SUGESTÃO PARA FUTUROS TRABALHOS

O sistema aqui estudado passou pelo processo de reengenharia, mas não foi concluída a implementação do mesmo. Aos alunos que vierem a ter acesso a este trabalho fica como sugestão implementar o sistema de acordo com o que foi proposto.

Após o reconhecimento desse padrão, fica como sugestão sua utilização em desenvolvimento de trabalhos futuros possibilitando aos desenvolvedores um ponto de partida que os estimule ao uso de padrões de software.

REFERÊNCIAS BIBLIOGRÁFICAS

(ALVARO, 2004) ALVARO, Alexandre**. TB-REPP - Padrões de Processo para a Engenharia Reversa baseado em Transformações** disponível em: http://www.dc.ufscar.br/~aalvaro/papers/TB-REPP.pdf. Acesso em 01 março 2004.

(BOOCH, 2000) BOOCH Grady, James Rumbaugh, Ivar Jacobson. **UML Guia do Usuário**. Tradução de Fábio Freitas da Silva. – Rio de Janeiro : Campus, 2000.

(ELISA, 2004) ELISA, Ana**. Reengenharia de Software: o quê, por quê e como,** disponível em: http://www.unicentro.br/pesquisa/editora/revistas/exatas/v1n2/Reengenharia.pdf. Acesso em 27 março 2004.

(GARCIA, 2004) GARCIA**,** Vinícius Cardoso. **Uma Abordagem Orientada a Aspectos para Reengenharia de Software usando Transformações,** disponível em: http://www.dc.ufscar.br/~vinicius/papers/Phoenix_FAPESB.pdf. Acesso em 01 março 2004.

(IAN, 2003) IAN Sommerville. **Engenharia de Software**. Tradução André Mauricio de Andrade Ribeiro. – São Paulo: Addison Wesley, 2003.

(JAMES, 1999) JAMES F. Peters, witold Pedrycz. **Engenharia de Software**. Tradução de Ana Patrícia Garcia. – Rio de Janeiro: Campus, 1999.

(PENTEADO, 2004) PENTEADO, Rosângela, Edson Recchia. **FaPRE/OO: Uma Família de Padrões para Reengenharia Orientada a Objeto de Sistemas Legados procedimentais Procedimentais**, disponível em: http://www.cos.ufrj.br/~sugarloafplop/2002/papers/Paper1C.pdf. Acesso em 03 maio 2004.

(PRESSMAN, 2002) PRESSMAN Roger S. **Engenharia de Software**. – *5.ed*. – Rio de Janeiro : McGraw-Hill, 2002.

(RECCHIA, 2004) RECCHIA Edson, Rosângela Penteado. **Padrões para Processo de Engenharia Avante na Reengenharia Orientada a Objetos de Sistemas Legados Procedimentais,** disponível em: http://www.cin.ufpe.br/~sugarloafplop/articles/ww/PadroesEngenhariaAvante.pdf. Acesso em 03 maio 2004.

ANEXO A – RELÁTORIO DXF

Laboratório Eng.Civil

FURO Nº. 5

CLIENTE - Carlos Antonio Balbino.
LOCAL - R. Santana Fundo c/ Rio S. Bartolomeu.

COTA	NÍVEL D'AGUA	PROF.	CLASSIFICAÇÃO	PERFIL	PENETRAÇÕES GOLPES/15cm 1ª	2ª	3ª	SPT	GRÁFICO SPT x PROFUNDIDADE
96,10									0 10 20 30 40 50
95			Silte argilosa micáceo marrom, mole.	1	0	0	3	3	
				2	0	0	1	1	
		2,70	Areia siltosa, micácea a variegada, muito fofa.	3	0	0	2	2	
	4,00	3,70	Silte argilo arenoso micáceo cinza, mole.	4	0	0	1	1	
		4,90	Areia siltosa micáce cinza, média.	5	0	0	16	16	
90		5,60	Areia siltosa, micácea a variegada, muito fofa.	6	0	0	4	4	
		6,80	Areia siltosa micácea coinza, mádia	7	0	0	11	11	
		7,60	Areia grossa com pedregulhos, muito compacta (PEDREGULHOS)	8	0	0	50	50	
		9,45		9	0	0	60	60	

DADOS DA EXECUÇÃO

INÍCIO 12/12/1999
TÉRMINO
PROF. REVESTIMENTO 0,00 m
CIRCULAÇÃO D'ÁGUA A PARTIR DE 0,00 m

OBSERVAÇÕES

1 - Limite do Furo : 0,00m
2 - Profundidade do NA : 4,00m
3 - Escala do Perfil : 1:100
4 - Fonte: ABC Sondágens e Fundacoes.
Tipo de fundacao: Estaca Struss.

SOND.
DES.
ENGº.
DATA 11/05/2001
DES. Nº. 1
REL. Nº. 1

ANEXO B – CHAVE PRIMÁRIA E CHAVE ESTRANGEIRA OBTIDA DO DOCUMENTADOR DO ACCESS

D:\egf.mdb sexta-feira, 18 de junho de 2004
Tabela: Amostras Página: 1

Propriedades

DateCreated:	27/5/1999 17:06:40	LastUpdated:	29/6/1999 20:46:51
OrderByOn:	Falso	RecordCount:	1839
Updatable:	Verdadeiro		

Colunas

Nome	Tipo	Tamanho
NumAmostra	Inteiro longo	4
LK_Sondagem	Inteiro longo	4
Inicio	Duplo	8
Final	Duplo	8
P11	Inteiro	2
P12	Inteiro	2
P21	Inteiro	2
P22	Inteiro	2
P31	Inteiro	2
P32	Inteiro	2
SPT	Inteiro	2
OBS	Texto	40

Relacionamentos

SondagensAmostras

Sondagens			**Amostras**
Código	1	∞	LK_Sondagem

Attributes:	Imposto; Propagar atualizações; Propagar exclusões
RelationshipType:	Um-para-muitos

Índices da tabela

Nome	Número de campos
CODSONDAGE	1
Campos:	
LK_Sondagem	Crescente
PrimaryKey	2
Campos:	
NumAmostra	Crescente
LK_Sondagem	Crescente
sondagemAmostras	1
Campos:	
LK_Sondagem	Crescente

D:\egf.mdb sexta-feira, 18 de junho de 2004
Tabela: camadas Página: 2

Propriedades

DateCreated:	27/5/1999 17:06:43	LastUpdated:	29/6/1999 20:46:51
OrderByOn:	Falso	RecordCount:	482
Updatable:	Verdadeiro		

Colunas

Nome	Tipo	Tamanho
NumCamada	Inteiro longo	4
Lk_Sondagem	Inteiro longo	4
Inicio	Duplo	8
Final	Duplo	8
CLASSIF	Memorando	-
CD1	Texto	20
CD2	Texto	20
CD3	Texto	20
CD4	Texto	20

Relacionamentos

Sondagenscamadas

Sondagens			**camadas**
Codigo	1	∞	Lk_Sondagem

Attributes:	Imposto; Propagar atualizações; Propagar exclusões
RelationshipType:	Um-para-muitos

Índices da tabela

Nome	Número de campos
Lk_Sondagem	1
Campos:	
Lk_Sondagem	Crescente
PrimaryKey	2
Campos:	
NumCamada	Crescente
Lk_Sondagem	Crescente
sondagemcamadas	1
Campos:	
Lk_Sondagem	Crescente

D:\egf.mdb sexta-feira, 18 de junho de 2004
Tabela: Classifcacao de camadas Página: 3

Propriedades

DateCreated:	27/5/1999 17:06:45	LastUpdated:	29/6/1999 20:46:51
OrderByOn:	Falso	RecordCount:	1
Updatable:	Verdadeiro		

Colunas

Nome	Tipo	Tamanho
Classif	Texto	20
Descricao	Memorando	-

Índices da tabela

Nome	Número de campos
PrimaryKey	1
Campos:	
Classif	Crescente

Propriedades

DateCreated:	2/6/1999 10:59:05	LastUpdated:	29/6/1999 20:46:51
OrderByOn:	Falso	RecordCount:	74
Updatable:	Verdadeiro		

Colunas

Nome	Tipo	Tamanho
AutoID	Inteiro longo	4
Lnk_Padrao	Texto	16
Tipo	Texto	1
Regiao	Inteiro	2
X1	Duplo	8
Y1	Duplo	8
X2	Duplo	8
Y2	Duplo	8

Relacionamentos

Padrões de PreenchimentoEntidades de Preenchimento

Padrões de			**Entidades de**
Nome	1	∞	Lnk_Padrao

Attributes:	Imposto; Propagar atualizações; Propagar exclusões
RelationshipType:	Um-para-muitos

Índices da tabela

Nome	Número de campos
Lnk_Padrao	1
Campos:	
Lnk_Padrao	Crescente
Padrões de PreenchimentoEntidades de	1
Campos:	
Lnk_Padrao	Crescente
PrimaryKey	1
Campos:	
AutoID	Crescente

D:\egf.mdb sexta-feira, 18 de junho de 2004
Tabela: Observações Página: 5

Propriedades

DateCreated:	2/7/1999 15:53:41	LastUpdated:	2/7/1999 16:11:13
OrderByOn:	Falso	RecordCount:	198
Updatable:	Verdadeiro		

Colunas

Nome	Tipo	Tamanho
AutoCod	Inteiro longo	4
LK_Sondagem	Inteiro longo	4
Primeira	Sim/Não	1
Intermediaria	Sim/Não	1
Ultima	Sim/Não	1
OBS	Memorando	-

Índices da tabela

Nome	Número de campos
LK_Sondagem	1
Campos:	
LK_Sondagem	Crescente
PrimaryKey	1
Campos:	
AutoCod	Crescente

Propriedades

DateCreated:	2/6/1999 10:52:08	LastUpdated:	29/6/1999 20:46:51
OrderByOn:	Falso	RecordCount:	9
Updatable:	Verdadeiro		

Colunas

Nome	Tipo	Tamanho
Nome	Texto	16
Altura	Duplo	8

Relacionamentos

Padrões de PreenchimentoEntidades de Preenchimento

Padrões de			**Entidades de**
Nome	1	∞	Lnk_Padrao

Attributes:	Imposto; Propagar atualizações; Propagar exclusões
RelationshipType:	Um-para-muitos

Índices da tabela

Nome	Número de campos
PrimaryKey	1

Campos:

Nome	Crescente

D:\egf.mdb sexta-feira, 18 de junho de 2004
Tabela: parâmetros Página: 7

Propriedades

DateCreated:	31/5/1999 08:58:47	LastUpdated:	14/11/2000 09:33:34
OrderByOn:	Falso	RecordCount:	4
Updatable:	Verdadeiro		

Colunas

Nome	Tipo	Tamanho
Tipo	Texto	10
Descrição	Texto	50

Índices da tabela

Nome	Número de campos
PrimaryKey	2

Campos:

Tipo	Crescente
Descrição	Crescente

D:\egf.mdb sexta-feira, 18 de junho de 2004
Tabela: Sondagens Página: 8

Propriedades

DateCreated:	27/5/1999 17:06:47	LastUpdated:	29/6/1999 20:46:51
OrderByOn:	Falso	RecordCount:	99
Updatable:	Verdadeiro		

Colunas

Nome	Tipo	Tamanho
Codigo	Inteiro longo	4
NSondagem	Texto	10
Cliente	Texto	60
Local	Texto	60
Data	Data/Hora	8
DInicio	Data/Hora	8
DFim	Data/Hora	8
Prof_Revestimento	Duplo	8
Circ_Dagua	Duplo	8
Cota	Duplo	8
Ntrab	Texto	10
Diametro	Texto	6
Amostrador	Texto	6
Sondador	Texto	40
Prof_Final	Duplo	8
NA	Duplo	8
DNA	Data/Hora	8
ENG	Texto	20

Relacionamentos

SondagensAmostras

Sondagens			**Amostras**
Codigo	1	∞	LK_Sondagem

Attributes: Imposto; Propagar atualizações; Propagar exclusões
RelationshipType: Um-para-muitos

Sondagenscamadas

Sondagens			**camadas**
Codigo	1	∞	Lk_Sondagem

Attributes: Imposto; Propagar atualizações; Propagar exclusões

RelationshipType: Um-para-muitos

SondagensSondagens

Sondagens	**Sondagens**
Codigo	Codigo

Attributes:	Não imposto
RelationshipType:	Um-para-um

Índices da tabela

Nome	Número de campos
Cliente	1
Campos:	
Cliente	Crescente
Local	1
Campos:	
Local	Crescente
NSondagem	1
Campos:	
NSondagem	Crescente
Ntrab	1
Campos:	
Ntrab	Crescente
PrimaryKey	1
Campos:	
Codigo	Crescente

ANEXO C – SCRIPTS (SQL)

```
create database Cadastro de Sondagens;

create table Amostras (
    NumAmosta char(15) not null,
    Lk-Sondagem varchar(15) not null,
    Inicio char(15) not null,
    Final char(15) not null,
    P11 char(15) not null,
    P12 char(15) not null,
    P21 char(15) not null,
    P22 char(15) not null,
    P31 char(15) not null,
    P32 char(15) not null,
    SPT varchar(30) not null,
    OBS varchar(50) not null,
    Código char(15) not null,
    primary key (NumAmosta, Lk-Sondagem));

create table Camadas (
    NumCamada char(15) not null,
    Lk_Sondagem char(15) not null,
    Inicio char(15) not null,
    Final char(15) not null,
    CLASSIF varchar(50) not null,
    CD1 char(15) not null,
    CD2 char(15) not null,
    CD3 char(15) not null,
    CD4 char(15) not null,
    Código char(15) not null,
    primary key (NumCamada, Lk_Sondagem));

create table Classificação de Camadas (
    Classif varchar(50) not null,
    Descrição varchar(50) not null,
    primary key (Classif));

create table Entidade de Preenchimento (
    AutoCAD char(15) not null,
    Lnk_Padrão char(15) not null,
    Tipo char(20) not null,
    Região char(15) not null,
    X1 char(15) not null,
    Y1 char(15) not null,
    X2 char(15) not null,
    Y2 char(15) not null,
    Nome varchar(20) not null,
    primary key (AutoCAD));

create table Padrões de Preenchimento (
    Nome varchar(20) not null,
    Altura char(15) not null,
    primary key (Nome));

create table Parametros (
    Tipo varchar(20) not null,
```

```
    Descrição varchar(50) not null,
    primary key (Tipo, Descrição));


create table Observações (
    AutoCad char(15) not null,
    Lk_sondagem char(15) not null,
    Primeiro char(15) not null,
    Intermédiario char(15) not null,
    Ultimo char(15) not null,
    OBS char(15) not null,
    Código char(15) not null,
    primary key (AutoCad, Lk_sondagem));

create table Sondagens (
    Código char(15) not null,
    NSondagem char(15) not null,
    Cliente varchar(50) not null,
    Local varchar(50) not null,
    Data date not null,
    DInicio char(15) not null,
    DFim char(15) not null,
    Prof-Revestimento char(15) not null,
    Circ_Dagua char(15) not null,
    Cota char(15) not null,
    Ntrab char(15) not null,
    Diâmetro char(15) not null,
    Amostrador varchar(50) not null,
    Sondador varchar(50) not null,
    Prof_Final char(15) not null,
    NA char(15) not null,
    DNA char(15) not null,
    ENG varchar(50) not null,
    primary key (Código));


alter table Amostras add constraint FKPossui
    foreign key (Código)
    references Sondagens;

alter table Camadas add constraint FKContém
    foreign key (Código)
    references Sondagens;

alter table Entidade de Preenchimento add constraint FKTem
    foreign key (Nome)
    references Padrões de Preenchimento;

alter table Observações add constraint FKFaz
    foreign key (Código)
    references Sondagens;

create unique index IDAmostras
    on Amostras (NumAmosta, Lk-Sondagem);
```

```
create index FKPossui
    on Amostras (Código);

create unique index IDCamadas
    on Camadas (NumCamada, Lk_Sondagem);

create index FKContém
    on Camadas (Código);

create unique index IDClassificação de Camadas
    on Classificação de Camadas (Classif);


create unique index IDEntidade de Preenchimento
    on Entidade de Preenchimento (AutoCAD);

create index FKTem
    on Entidade de Preenchimento (Nome);

create unique index IDPadrões de Preenchimento
    on Padrões de Preenchimento (Nome);

create unique index IDParametros
    on Parametros (Tipo, Descrição);

create unique index IDObservações
    on Observações (AutoCad, Lk_sondagem);

create index FKFaz
    on Observações (Código);

create unique index IDSondagens
    on Sondagens (Código);
```