Universidade Federal de Santa Catarina

Cleber Gomes Caldana

INFUSION: Uma Experiência de Engenharia Reversa Orientada a Objetos para Sistemas Legados

Florianópolis
2003

Cleber Gomes Caldana

INFUSION: Uma Experiência de Engenharia Reversa Orientada a Objetos para Sistemas Legados

Dissertação submetida ao Curso de Pós Graduação, em Ciência da Computação, da Universidade Federal de Santa Catarina como parte dos requisitos à obtenção do título de Mestre.
Orientador: Prof. Rosvelter João C. da Costa

Florianópolis
2003

Cleber Gomes Caldana

INFUSION: Uma Experiência de Engenharia Reversa Orientada a Objetos para Sistemas Legados

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação Área de Concentração Sistemas de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Prof. Dr. Fernando Alvaro Ostuni Gauthier
Coordenador do Curso

Banca Examinadora

Prof. Orientador Dr. Rosvelter João C. da Costa
UFSC – Universidade Federal de Santa Catarina

Prof. Dr.Ricardo Pereira e Silva
UFSC – Universidade Federal de Santa Catarina

Prof. Dr. Vitório Bruno Mazzola
UFSC – Universidade Federal de Santa Catarina

Florianópolis, ___ de_____ de_____.

Dedico este trabalho a Joseane Pivetta, minha companheira, amiga, confidente e esposa que esteve presente e me apoiou em todos os momentos da minha vida.

Agradecimentos

Ao meu orientador Prof. Rosvelter pelo apoio, compreensão e dedicação aplicada ao desenvolvimento deste trabalho.

A Empresa Jabur Informática por possibilitar o desenvolvimento de uma significativa parte deste trabalho cedendo o Sistema SIGLO® e seus respectivos códigos fontes para serem utilizados como estudo de caso.

Aos professores e funcionários do Departamento de Informática e Estatística da Universidade Federal de Santa Catarina que sempre demonstraram compreensão e boa vontade para me auxiliar nas dificuldades encontradas durante o desenvolvimento deste trabalho.

CALDANA, Cleber Gomes. **INFUSION: Uma Experiência de Engenharia Reversa Orientada a Objetos para Sistemas Legados.** 2003. 97f. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Santa Catarina, Florianópolis, 2003.

RESUMO

Este trabalho apresenta um estudo sobre métodos de engenharia reversa para Sistemas Legados. Sistemas Legados caracterizam-se por apresentarem elevado custo de manutenção devido a não utilização dos conceitos de engenharia de software e são mantidos por desenvolvedores que não participaram da sua concepção. A Engenharia Reversa surge como uma abordagem para melhorar a manutenção de Sistemas Legados através de estudos sobre o código fonte e a documentação existente, caso exista, revitalizando a documentação inerente as etapas de Análise e Projeto, recuperando assim o entendimento do sistema. Também é apresentado um estudo de caso baseado no método de Engenharia Reversa FUSION/RE. O desenvolvimento do estudo de caso é descrito através de documentações geradas e dos procedimentos aplicados durante a aplicação do método FUSION/RE na Engenharia Reversa. Como contribuições deste trabalho são apresentadas alterações nos procedimentos do método FUSION/RE e uma proposta de validação da documentação gerada durante o processo de Engenharia Reversa.

Palavras Chaves: Manutenção de Software, Sistemas Legados, Engenharia Reversa.

CALDANA, Cleber Gomes. **INFUSION: An Experience of the Reverse Engineering Object Oriented for Legacy Systems..** 2003. 97f. Dissertation (Master´s of Science Degree Dissertation) – Federal University of Santa Catarina, Florianópolis, 2003.

ABSTRACT

This work represents a study about Reverse Engineering for Legacy Systems. Legacy Systems are caracterized by high cost of maintenance due to the non-utilization of software engineering concepts, and are kept updated by developers that do not participated of the conception of the System. The Reverse Engineering emerge as an approach to improve the maintenance of the Legacy Systems through studies over the source code and the documentation, case it exists, revitalizing the documentation to the phases of Analysis and Project, recovering this way the understanding of the system. It is also presented a case study based on the FUSION/RE Reverse Engineering method. The development of the case study is described throuhg documentation generated and the procedures applied during the application of the FUSION/RE method on the Reverse Engineering. The contributions of this work focus on the alterations to the procedures on the FUSION/RE method and to a proposal of validation of the documentation generated during the Reverse Engineering proccess.

Key Words: Software Maintenance, Legacy Systems, Reverse Engineering.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

1.1. Motivação e Objetivos

No ambiente comercial é grande a existência de sistemas que foram desenvolvidos há anos e sem aplicação correta dos conceitos da engenheira de software. Esses sistemas atendem aos requisitos do usuário mas necessitam de muito esforço na tarefa de manutenção. Sistemas com as características citadas, denominados Sistemas Legados, necessitam de melhorias quer quanto à interface como na facilidade na tarefa de manutenção. Assim este trabalho enfoca esses sistemas e métodos que propiciam aos engenheiros de software realizarem a manutenção de forma mais organizada reduzindo assim o esforço despendido nessa tarefa.

A engenharia reversa, também denominada manutenção preventiva (PRESSMAN, 1995), surge como um método para melhorar a manutenção e fornece base para possível reengenharia de sistemas legados. Entre os métodos de engenharia reversa destaca-se o Fusion/RE (PENTEADO, 1996), um método de engenharia reversa orientado a objetos.

O Fusion/RE é um método de engenharia reversa baseado no método Orientado a Objetos Fusion (COLEMAN, 1994). O Fusion/RE é aplicável em sistemas legados procedimentais e sua aplicação gera uma documentação com nível de abstração mais alto que o código fonte e pode ser usada tanto para melhorar a manutenibilidade do sistema como para realizar a sua reengenharia preservando a funcionalidade.

Após a engenharia reversa pode-se realizar a reengenharia de sistemas visando diversos objetivos: a mudança do paradigma procedimental para o orientado a objetos; uma atualização dos sistemas em relação à linguagem, plataforma, ambiente, suporte de ferramentas case, etc.

Este trabalho tem como principal objetivo aplicar o método de engenharia reversa orientado a objetos Fusion/RE em um sistema comercial implementado em linguagem procedimental.

1.2. Organização do Trabalho

Este trabalho está organizado da seguinte forma. No Capítulo 2 é realizada uma revisão bibliográfica analisando a literatura especializada referente aos temas aqui discutidos. O Capítulo 3 aborda o tema orientação a objetos e apresenta o método Fusion, temas estes que estruturam o método de engenharia reversa FUSION/RE apresentado em seguida. O capítulo 4 especifica uma aplicação do método de engenharia reversa Fusion/RE em um estudo de caso e sugere diretrizes de qualidade e procedimentos quanto à utilização desse método com relação ao modelo de processo original.

2. REVISÃO BIBLIOGRÁFICA

2.1. Considerações Iniciais

A manutenção é uma constante preocupação dos engenheiros de software pois consome grande esforço. Esse esforço está relacionado a problemas na engenharia do sistema, entre eles: a falta de documentação, a não utilização dos conceitos da engenharia de software, a equipe atual de manutenção não participou do desenvolvimento do sistema, entre outros.

A engenharia reversa pode ser utilizada para melhoria da manutenibilidade de sistemas legados e para possibilitar a sua reengenharia. Vários métodos de engenharia reversa foram propostos com o objetivo de recuperar a especificação do sistema e o entendimento de suas funcionalidades. Entre esses métodos há os que oferecem diretrizes para a realização da engenharia reversa em sistemas procedimentais gerando uma especificação orientada a objetos.

Após a engenharia reversa pode-se realizar a reengenharia de sistemas. A reengenharia visa a reimplementação do sistema para acrescentar funcionalidades, utilizar novas tecnologias da engenharia de software ou alterar a arquitetura para orientada a objetos.

Este capítulo está organizado da seguinte forma: a Seção 2.2 trata da terminologia utilizada pela engenharia de software para referenciar os conceitos abordados neste trabalho. A etapa de manutenção de software e suas características são apresentadas na Seção 2.3. Na Seção 2.4 são apresentados os processos de Engenharia Reversa e Reengenharia de Software e na Seção 2.5 as considerações finais.

2.2. Terminologia

Para racionalizar os termos utilizados para referenciar as tecnologias de análise e entendimento de sistemas existentes, Chikofsky (1990) apresenta uma terminologia empregada na engenharia de software. Os termos especificados são: engenharia avante; engenharia reversa que compreende redocumentação e recuperação de projeto; reestruturação e reengenharia.

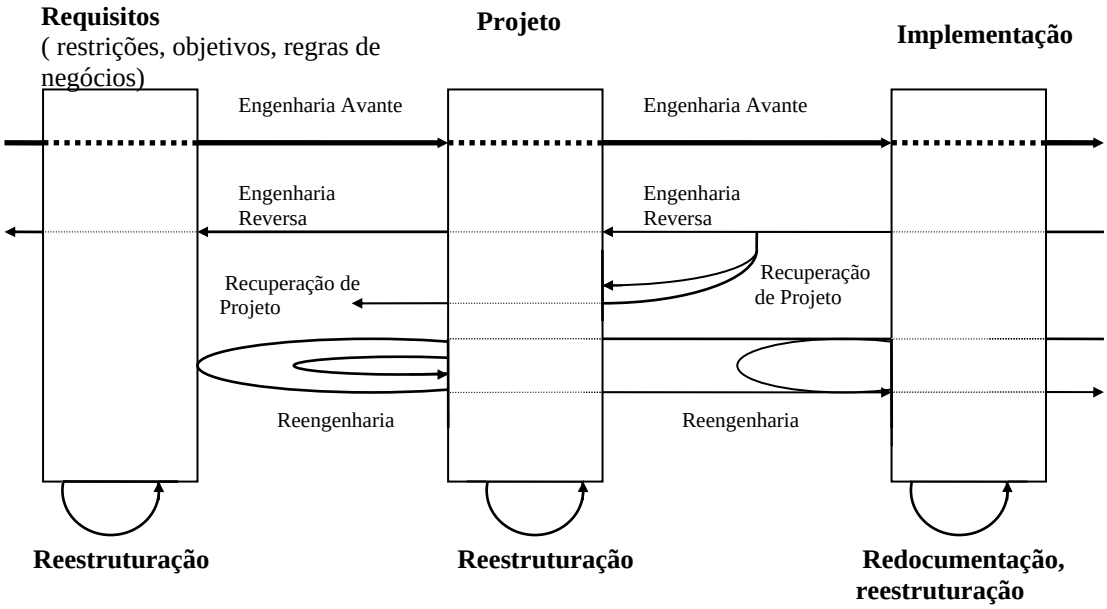


Ilustração 1 - Terminologia da Engenharia de Software (CHIKOFSKY,1990)

Engenharia avante é o processo tradicional de partir de um nível de abstração alto, de requisitos do sistema, e chegar ao nível físico, de implementação do sistema. Reestruturação é a transformação de uma forma de representação para outra no mesmo nível relativo de abstração, preservando o comportamento externo do sistema (funcionalidade e semântica). Como exemplos de reestruturação podem-se citar estruturação de programas e normalização de dados.

Os termos engenharia reversa e reengenharia são definidos na seção 2.4. O relacionamento entre os termos (Ilustração 1) é feito considerando-se que o ciclo de vida do software possui três grandes etapas (requisitos, projeto e implementação) com claras diferenças no nível de abstração. Requisitos tratam da especificação do problema, incluindo objetivos, restrições e regras de negócio. Projeto trata da especificação da solução. Implementação trata da codificação, teste e entrega do sistema em operação.

A Ilustração 1 mostra a direção seguida pela engenharia avante, do nível mais alto para o nível mais baixo de abstração. Mostra, também, que a engenharia reversa percorre exatamente o caminho inverso, podendo utilizar-se da recuperação de projeto para aumentar o nível de abstração. A reengenharia geralmente inclui alguma forma de engenharia reversa, seguida de alguma forma de engenharia avante ou reestruturação.

2.3. Manutenção de Software

Manutenção de software é definida como o processo de modificação de um software, após a sua instalação, de forma a corrigi-lo, melhorá-lo ou adaptá-lo (CORDEIRO, 2003). Segundo o Comitê de Engenharia Reversa – IEEE (2003) a manutenção *é "a modificação de um produto, após a entrega, para corrigir defeitos, para melhorar desempenho ou outros atributos, ou para adaptar o produto a um ambiente alterado".*

Uma pesquisa realizada pela SISTM (Society for Information Systems Technology and Management) apurou que 94,6% dos engenheiros de software despendem a maior parte do tempo na manutenção de sistemas (GLASS,?). Esse esforço está relacionado com quatro tipos de manutenção especificadas a seguir:

- *Manutenção corretiva:* Atividade que envolve o diagnóstico e a correção de erros não encontrados pela etapa de testes e detectados pelo usuário durante a utilização do sistema;
- *Manutenção adaptativa*: Manutenção responsável pelas alterações no sistema com a finalidade de adaptação a novos elementos do ambiente (hardware, sistema operacional, etc.) ou acréscimos e alterações de funções solicitadas pelo usuário;
- *Manutenção perfectiva:* Manutenção voltada para softwares bem-sucedidos, ou seja, softwares que apresentam elevada aceitação do usuário. Nesses softwares são realizadas alterações para atender diversas solicitações de melhorias ou expansões nas funções do sistema;

- *Manutenção preventiva:* Atividade que utiliza as técnicas de engenharia reversa e reengenharia para melhorar características de qualidade de um software.

A manutenção de software é reconhecidamente uma etapa problemática e diversas características da mesma foram discutidas por Schneidewind (1987):

- Existe dificuldade em averiguar quais e como foram implantadas as manutenções realizadas em um software por não ter havido gerenciamento de configuração;
- Devido à estabilidade do pessoal da área de software ser baixa, muitas vezes o engenheiro de software responsável pela manutenção não é o que desenvolveu o sistema tornando difícil seu entendimento;
- A maioria dos sistemas foram construídos sem a utilização dos conceitos oferecidos pela engenharia de software, entre eles, uma documentação inexistente ou incompreensível e inconsistente.

Todos os problemas acima citados afetam a manutenibilidade do sistema. Manutenibilidade é definida pelo Comitê de Engenharia Reversa – IEEE (2003) como *"a facilidade que um sistema ou componente de software possui de ser modificado para corrigir as falhas, melhorar seu desempenho e outras características ou adaptação para a mudança de ambiente".*

Muitas organizações possuem sistemas antigos, denominados Sistemas Legados, com problemas de manutenção que não podem ser desativados por desempenharem atividades essenciais (SILVA).

Sistemas legados de software são sistemas que evoluíram durante muitos anos sendo considerados insubstituíveis por serem confiáveis para os usuários e apresentarem alta complexidade na reimplementação de suas funções (BABIKER, 1997).

Sistemas legados também foram mencionados por Lee (1997) como sendo sistemas implementados em tecnologias antigas e que automatizam

operações importantes para as organizações. Esses sistemas possuem uma especificação pobre e são mantidos por diversas equipes de engenheiros de software.

O descarte dos sistemas legados realizando a engenharia de um novo software utilizando métodos corretos e tecnologias mais recentes se torna inviável se analisada a confiabilidade apresentada pelo usuário e a importância nas atividades desempenhadas (BENEDUSI, 1996).

Sendo assim, soluções foram estudas e propostas para tais sistemas, entre elas a recuperação da documentação do sistema através da engenharia reversa amenizando os esforços e custos empregados na fase de manutenção.

2.4. Engenharia Reversa e Reengenharia de Sistemas Legados

A manutenção de sistemas legados é uma constante preocupação para aqueles que trabalham em computação. Várias pesquisas estão sendo realizadas buscando suporte para a manutenção desses sistemas (PENTEADO, 1999).

Os métodos de engenharia reversa foram desenvolvidos para melhorar a manutenibilidade de sistemas legados e apoiar o processo de reengenharia (CAGNIN, 2000).

O termo “engenharia reversa” teve sua origem no mundo do hardware. Uma empresa desmontava um hardware comercializado, num esforço para entender os “segredos” do projeto e manufatura do concorrente para reconstruí-lo (PRESSMAN, 1995). O conceito de engenharia reversa de software é similar ao de hardware. Porém, tradicionalmente o objetivo da engenharia reversa de hardware é replicar o produto, enquanto o objetivo da engenharia reversa de software freqüentemente é obter um entendimento do sistema (QUINAIA, 1999).

Os objetivos da engenharia reversa são o reuso e a manutenção de software. O reuso de software pode ser apoiado pela engenharia reversa tanto na identificação e composição de componentes a partir de partes reutilizáveis de sistemas existentes quanto na elaboração da documentação dos novos sistemas compostos. A manutenção de software pode contar com a ajuda de

uma documentação completa de todo o sistema. Nesse ponto, a engenharia reversa pode fornecer as visões em diversos níveis de abstração, permitindo a localização dos componentes a serem mantidos, além de melhorar a entendimento do software, pela documentação produzida (BRAGA, 1998).

Há muitas tarefas diferentes na engenharia reversa (TILLEY, 2003). Destacam-se entre as mais importantes:

- *Entendimento do código fonte:* Uma das tarefas mais comuns na engenharia reversa de sistemas. O código fonte é examinado para que seu entendimento seja representado em um nível mais alto de abstração;
- *Reconhecimento do padrão:* Engenheiros de software normalmente seguem um padrão para a elaboração do código fonte. Essa tarefa visa o reconhecimento desse padrão para auxiliar numa representação mais abstrata do código e identificar fragmentos de códigos duplicados que foram reutilizados via o procedimento “copiar / colar”. Esse procedimento reduza o tempo de desenvolvimento mas conduz para problemas de manutenção devido ao número elevado de código e propagação de erros quando realizadas alterações nesses códigos;
- *Abstração de Conceitos:* Tarefa de examinar trechos do código fonte abstraindo suas funcionalidades. Essa abstração permite ao engenheiro compreender o programa e a aplicação da qual se destina;
- *Redocumentação:* A falta de uma documentação precisa e atualizada do sistema faz com que o código fonte seja a única fonte fiel de informação. A redocumentação é uma tarefa da engenharia reversa que gera uma documentação completa do software existente;
- *Recuperação da arquitetura:* Para sistemas legados grandes e complexos no qual foram utilizadas diversas equipes de desenvolvimento, o entendimento de sua arquitetura é de extrema importância. A documentação existente para esses sistemas normalmente descreve partes isoladas, não contendo a estrutura

completa . Essa tarefa recupera a documentação que especifica a arquitetura global do sistema.

Para Wu (1997), a engenharia reversa auxilia no entendimento de sistemas legados e tem como objetivo “a identificação dos componentes dos sistemas e seus relacionamentos criando representações do sistema em um nível mais alto de abstração”. A tarefa de entendimento do sistema é agrupada em três técnicas (MÜLLER, 2000):

- *Rastrear o código sem apoio de método/ferramenta:* o engenheiro de software rastreia manualmente o código de fonte que se apresenta de forma impressa ou meio eletrônico. Essa técnica apresenta limitações baseadas na quantidade de informações que o engenheiro de software pode ser capaz de memorizar;
- *Recuperar requisitos e experiências implícitas no sistema:* Técnica que realiza entrevistas com desenvolvedores e usuários visando a recuperação das informações inerentes ao sistema. Essas entrevistas são muito importantes pois as pessoas envolvidas com o sistema podem conhecer requisitos importantes como: decisões de projeto, manutenções realizadas, subsistemas problemáticos, entre outros;
- *Apoio de Métodos de Engenharia Reversa:* A aplicação da técnica citada anteriormente nem sempre é possível pois: a equipe de desenvolvimento pode não ser a mesma que participou da concepção do sistema ou o sistema pode ter sido adquirido de outra companhia. Nesse caso faz-se necessária à utilização de métodos de engenharia reversa. Esses métodos visam administrar as complexidades de compreensão do sistema e auxiliar na elaboração de uma representação de alto nível a partir de especificações de baixo nível, como código de fonte.

Lee (2000) especifica um método de engenharia reversa orientado a objetos baseado na elaboração de formulários para recuperar o entendimento

da funcionalidade dos sistemas legados. Esse método proporciona, através de reuniões com os principais usuários, uma análise das telas e fonte de entrada de dados do sistema. A estrutura do método denominada FORE (Form Driven Oriented-Object Reverse Engineering) é demonstrada na Ilustração 2.

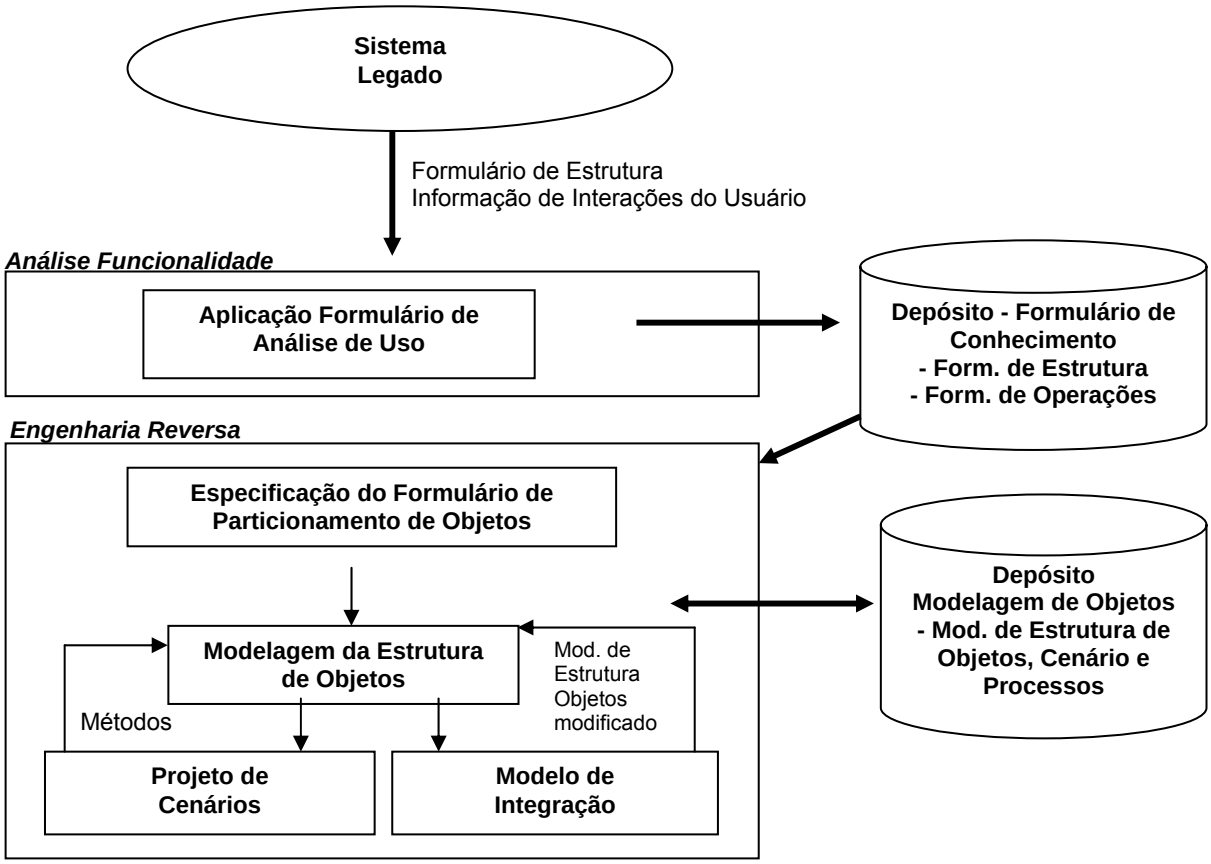


Ilustração 2 - Método de Engenharia Reversa FORE (LEE, 2000)

O método FORE é composto por cinco fases:

1. *Aplicação do Formulário de Análise de Uso:* Nesta são elaborados os formulários de estrutura de dados e adquiridas as informações sobre as interações do usuário com o sistema. Essas informações são capturadas durante a execução do sistema legado pelo usuário e especificadas no Formulário de Análise de Uso para serem utilizadas pelas fases seguintes;

2. *Especificação do Formulário de Particionamento de Objetos:* No Formulário de Particionamento de Objetos as especificações armazenadas na fase anterior são particionadas em unidades semânticas de acordo com os tipos de entradas (grupo de campos, operações, etc.).Essa fase em conjunto com a Modelagem da Estrutura de Objetos executa a engenharia reversa com a finalidade de realizar a modelagem orientada a objetos;

3. *Modelagem da Estrutura de Objetos:* A Modelagem da Estrutura de Objetos tem como objetivo principal abstrair os objetos da fase anterior e acrescentar os relacionamentos existentes entre os mesmos. Essa fase oferece como saída um modelo com os nomes dos objetos, seus atributos e relacionamentos;

4. *Projeto de Cenário:* No Projeto de Cenário as informações sobre operações levantadas na fase anterior são utilizadas para a modelagem de diagramas de seqüência para cada operação analisada;

5. *Modelo de Integração:* O Modelo de Integração é gerado através de refinamentos dos modelos das fases anteriores. Nessa fase princípios de generalização/especialização e agregação são inseridos gerando um modelo com nível de abstração mais alto;

Penteado (1996) apresenta o método de engenharia reversa orientado a objetos denominado FUSION/RE. O próximo capítulo apresenta um estudo detalhado deste método que foi aplicado ao estudo de caso. O método FUSION/RE apresenta as seguintes fases:

- *Revitalizar a Arquitetura do Sistema:* Tem a finalidade recuperar as informações relacionadas à arquitetura do sistema para que ele possa ser entendido/estendido pelo engenheiro de software, facilitando o desenvolvimento dos passos posteriores. Em sistemas

que não apresentam documentação o engenheiro deverá realizar análise no código fonte. Caso o sistema possua alguma documentação, é necessário identificá-la e organizá-la de forma a permitir que a funcionalidade do sistema seja completamente entendida;

- *Recuperar o Modelo de Análise da Solução Atual:* Deve-se desenvolver nesse passo uma modelagem considerando-se somente os aspectos da implementação atual. O resultado dessa fase é o Modelo de Análise do Sistema Atual (MASA) que consiste da especificação do sistema utilizando os conceitos de orientação a objetos. A partir da especificação existente (código legado), um modelo de análise orientado o objetos é elaborado. Neste modelo os objetos especificados são considerados pseudo-objetos;

- *Abstrair o Modelo de Análise do Sistema:* O Modelo de Análise do Sistema (MAS) é elaborado a partir do MASA e abstraí-se os conceitos orientados a objetos e os objetos que realmente devem fazer parte do sistema. Um Dicionário de Dados, manual ou automatizado, deve ser usado para armazenar informações sobre o MAS. Neste passo são utilizados os termos “atributos” e “métodos” para identificar os dados e os algoritmos associados a uma classe;

- *Mapear o Modelo de Análise do Sistema para o Modelo de Análise do Sistema Atual:* Este mapeamento é fundamental para futuros desenvolvimentos ou manutenções de partes do sistema atual, facilitando o reuso. O mapeamento entre os documentos do MAS e correspondentes do MASA devem ser cuidadosamente documentado e checado para não conter inconsistências entre os esses modelos. Esse mapeamento facilita tanto o entendimento como uma nova implementação do sistema, pois mostra facilmente como os elementos do MAS estão atualmente implementados facilitando

futuras reutilizações. Pode-se avaliar nesse ponto quanto da implementação pode ser reutilizada caso se decida continuar com a mesma abordagem atualmente utilizada. Em termos de notação a correspondência pode ser feita por tabelas, mas a informação pode também estar armazenada no Dicionário de Dados.

Posterior ao passo de engenharia reversa pode-se realizar a reengenharia de software, que segundo Chikofsky (1990) a reengenharia de software consiste na reimplementação de um sistema em uma forma mais fácil de ser mantido. A reengenharia de sistemas apresentou um rápido crescimento nos últimos anos devido à necessidade das empresas em conter custos na manutenção de seus sistemas.

Segundo o Comitê de Engenharia Reversa – IEEE (2003), reengenharia é submeter um sistema a um exame para reconstituí-lo em uma nova forma e conseqüentemente implementá-lo nessa nova forma. Sneed (1995) relata que a reengenharia de software possui quatro objetivos principais:

- *Melhorar a Manutenibilidade:* Melhoria que pode ser obtida com a reestruturação da arquitetura do sistema realizando uma melhor divisão nos módulos. Essa melhoria não é obtida somente com a reengenharia pois a manutenibilidade do sistema também necessita de uma estrutura de manutenção com métodos, procedimentos, equipes, etc;

- *Migração:* Reengenharia que realiza a migração do sistema para uma outra plataforma. O novo sistema é validado verificando sua funcionalidade com o sistema antigo;

- *Aumentar Confiabilidade:* A aplicação de testes exaustivos após a reengenharia para provar a equivalência funcional poderá revelar antigos erros inerentes ao sistema que devem ser corrigidos. Esse

objetivo pode ser facilmente validado através da análise de erros verificados no sistema;

- *Preparação para Melhoria Funcional:* Realizar a reengenharia para decompor programas em pequenos módulos isolados facilitando a mudança ou acréscimo de funcionalidades num determinado módulo sem afetar outros, alterar as estruturas de dados e relacionamentos para facilitar o acréscimo de novos atributos ou entidades. Através desse procedimento pode-se transformar um sistema implementado no paradigma estruturado para orientado a objetos.

A reengenharia de software é descrita como a reorganização e modificação parcial ou completa de sistemas de software existentes, para torná-los manuteníveis. A reengenharia de sistemas é classificada conforme o grau de profundidade da execução (SOMMERVILLE, 1995):

- *Conversão de programas fontes:* Reengenharia de sistemas simples em que os códigos são reescritos utilizando uma versão mais nova da linguagem de programação utilizada ou mudando de linguagem de programação. Essa reengenharia é utilizada quando ocorre troca de plataforma de hardware ou alterações de padrões de desenvolvimento utilizados pela empresa;

- *Reestruturação dos programas fontes:* Utilizada quando a estrutura do sistema está falha ou complexa, dificultando o entendimento. Programas que utilizam essa reengenharia apresentam estruturas condicionais complexas, inúmeros, desvios, rotinas não chamadas e/ou duplicadas, etc;

- *Reengenharia de dados:* Envolve análise, reorganização das estruturas de dados e conversão dos valores contidos nessas estruturas. A reengenharia de dados é efetuada quando há uma

migração de determinada base de dados para outra. Nessa reengenharia é necessária a criação de programas de suporte para que a conversão dos valores seja feita;

A reengenharia de software pode apresentar mudança de paradigmas de engenharia convencional para o paradigma orientado a objetos. Para esse cenário foram realizadas diversas pesquisas, algumas são especificadas a seguir.

A reengenharia pode ser definida como sendo a Σ = Engenharia reversa + ∆ (delta) + engenharia avante. Sendo que a engenharia reversa é o primeiro passo a ser realizado e consiste em uma representação mais abstrata e mais compreensível do sistema. O segundo passo, identificado pelo símbolo ∆ (delta), representa as possíveis modificações funcionais e de implementação do sistema.

A engenharia avante, último passo, refere-se a engenharia de software convencional, isto é, a implementação do sistema em alguma linguagem de programação. Nesse caso, uma linguagem que seja orientada a objetos. A reengenharia de sistemas com mudança de paradigma possui a seguinte classificação (JACOBSON,1991):

- *Com troca total da linguagem de implementação sem troca da funcionalidade:* esse processo passa por três passos: 1) Preparar um modelo de análise do sistema, 2) Mapear cada objeto de análise para a implementação do sistema antigo e 3) Reprojetar o sistema;

- *Com troca parcial da linguagem de implementação sem troca da funcionalidade:* o objetivo desse cenário é fazer com que a parte do sistema, que foi implementada utilizando o paradigma orientado a objetos, consiga se comunicar com a parte do sistema que não utiliza esse paradigma;

- *Com troca parcial da funcionalidade:* pode-se modificar o modelo de análise que foi obtido a partir de qualquer um dos dois métodos citados acima, incluindo novas funcionalidades e, em um processo de engenharia avante, reimplementar o sistema

Rosenberg (2003) declara que a reengenharia se inicia com entendimento do código fonte de um sistema legado e tem sua conclusão com a implementação e os testes do código fonte do sistema refeito. A Ilustração 3 apresenta o modelo geral para a reengenharia de software indicando os processos para todos os níveis de abstração existentes na engenharia de software.

Nesse modelo a reengenharia se inicia com a engenharia reversa analisando o código fonte e aumentando a abstração possibilitando reavaliar a arquitetura e os requisitos do sistema existente. Após a engenharia reversa os engenheiros de software realizam a reengenharia desenvolvendo o sistema destino seguindo o ciclo de vida clássico (cascata). A autora ainda relata que a reengenharia de sistemas pode ser justificada pela melhoria da qualidade do sistema destino com aumento da confiabilidade e redução do esforço na manutenção mantendo a funcionalidade do sistema existente.

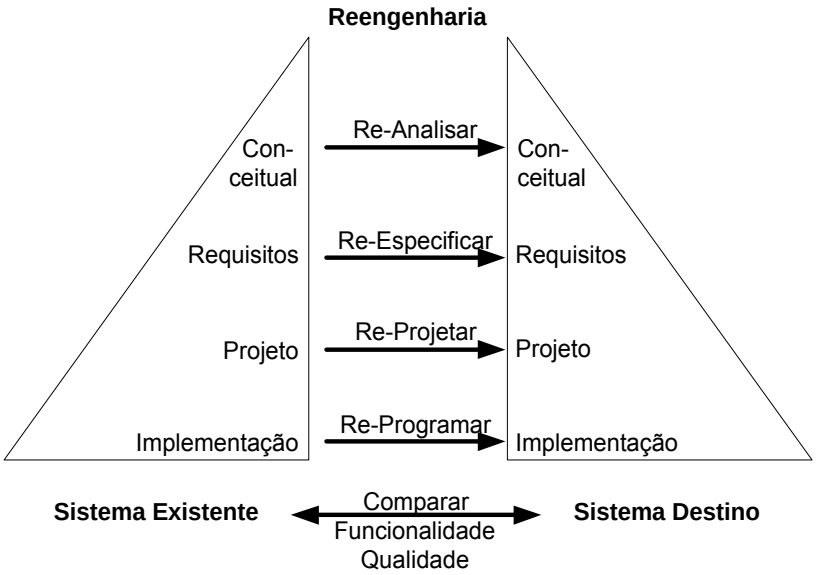


Ilustração 3 - Modelo Geral para Reengenharia de Software (ROSENBERG,2003)

A reengenharia de sistemas procedimentais para orientados a objetos é um processo importante pois tem como objetivos possibilitar a reusabilidade dos programas reduzindo os custos de desenvolvimento de software e melhorar a qualidade do produto final. Esses objetivos serão alcançados desde que sejam aplicados corretamente os procedimentos de engenharia de software que envolvem a reengenharia (GALL,1994).

A Ilustração 4 apresenta uma método de reengenharia conhecido como COREM (Capsule Oriented Reverse Engineering Method) (GALL,1993), este método utiliza o conhecimento do domínio da aplicação. Esse método consiste de quatro passos principais: Recuperação do Projeto, Modelagem da Aplicação, Mapeamento dos Objetos e Transformação do Sistema.

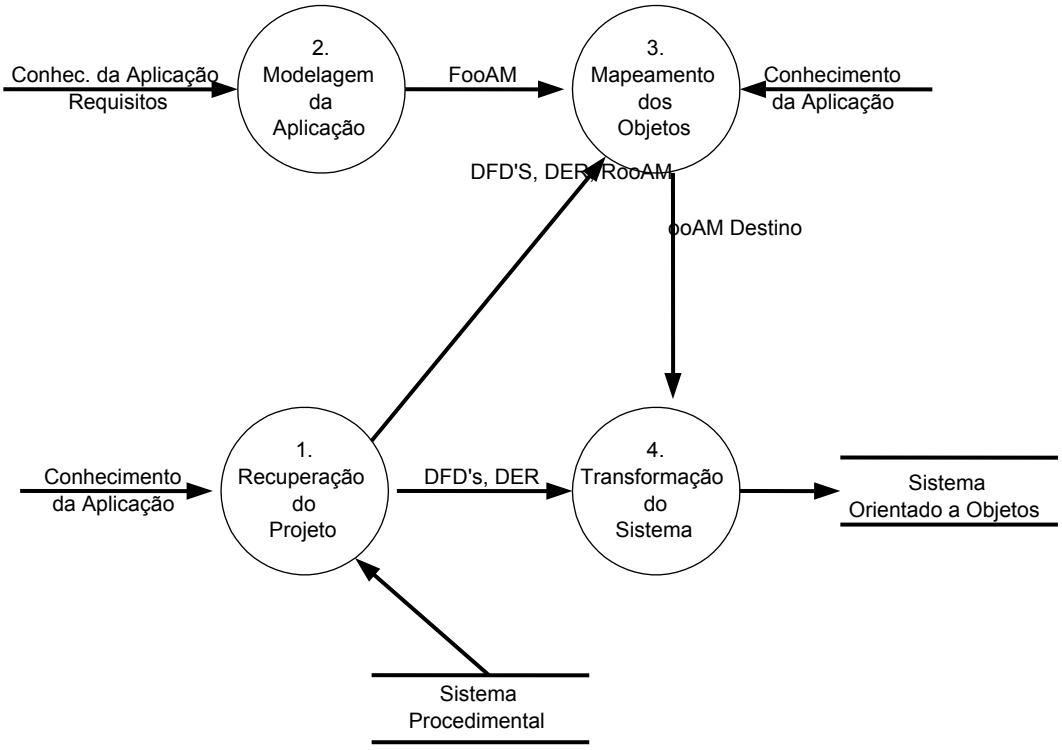


Ilustração 4 - Método de Reengenharia COREM (GALL, 1993)

- *Recuperação do Projeto:* geração de diversos documentos do projeto a partir do código fonte procedimental entre eles: Diagrama de Fluxo de Dados, Diagrama de Entidade e Relacionamentos e um modelo de aplicação orientado a objetos denominado Reverse Generated Object-Oriented Application Model (RooAM);

- *Modelagem da Aplicação:* a partir da análise dos requisitos do programa procedimental e domínio da aplicação é gerado o modelo de aplicação orientado a objetos Forward Generated Object-Oriented Model Application (FooAM);

- *Mapeamento dos Objetos:* os elementos do RooAM são mapeados para os elementos do FooAM, resultando em um modelo de aplicação orientado a objetos destino (ooAM – Objetct-Oriented Application Model). Nessa fase algumas partes podem permanecer em um dos modelos de aplicação, sem correspondência, devido à diferença dos paradigmas;

- *Transformação do Sistema:* a partir do código fonte e dos resultados obtidos nos passos anteriores é realizado o processo de transformação do programa resultando em três tipos diferentes de objetos: objetos da aplicação (definidos durante o processo de mapeamento entre RooAM e FooAM), objetos de dados (resultantes do encapsulamento dos dados globais) e objetos de implementação (resultantes das partes restantes do sistema procedimental que não possuem correspondência com a implementação orientada a objetos). A automação completa desse processo não é possível pois é imprescindível a aquisição de conhecimento adicional do engenheiro de software, mas muitos passos do processo podem ser auxiliados por ferramentas.

Ainda Gall e Klösch (GALL,1995) declaram que devido às divergências existentes entre os paradigmas, procedural e orientado a objetos, não é possível mapear diretamente todos os elementos de uma aplicação procedimental para uma orientada a objetos. Portanto, o sistema orientado a objetos resultante consiste de duas partes: a parte orientada a objetos e a parte restante procedimental. Assim é necessário realizar adaptações na implementação através da identificação de objetos, a partir da análise de funcionalidades do sistema.

Os autores ressaltam que para transformar o código procedimental em código orientado a objetos, é necessário preocupar-se com as seguintes considerações: adaptações no programa com a transformação de elementos de dados e procedimentos em atributos e métodos, respectivamente; adaptações nas interfaces dos serviços, uma vez que o estado interno de um objeto é acessível para cada serviço; isolamento e encapsulamento dos itens de dados globais em objetos separados (nomeados como objetos de dados, pois contêm somente atributos) para assegurar uma completa transformação orientada a objetos.

Babiker (1997) apresenta um método de reengenharia de sistema para a arquitetura orientada a objeto composto por três fases (Ilustração 5). Na primeira fase, a engenharia reversa é realizada para extração dos requisitos, entendimento da funcionalidade do sistema e definição das entidades do software. Na segunda fase, novos requisitos e funções podem ser inseridas à especificação da fase anterior e uma nova modelagem é gerada (Conhecimentos e Requisitos Filtrados). Na fase final é realizado o desenvolvimento do sistema utilizando o método orientado a objetos.

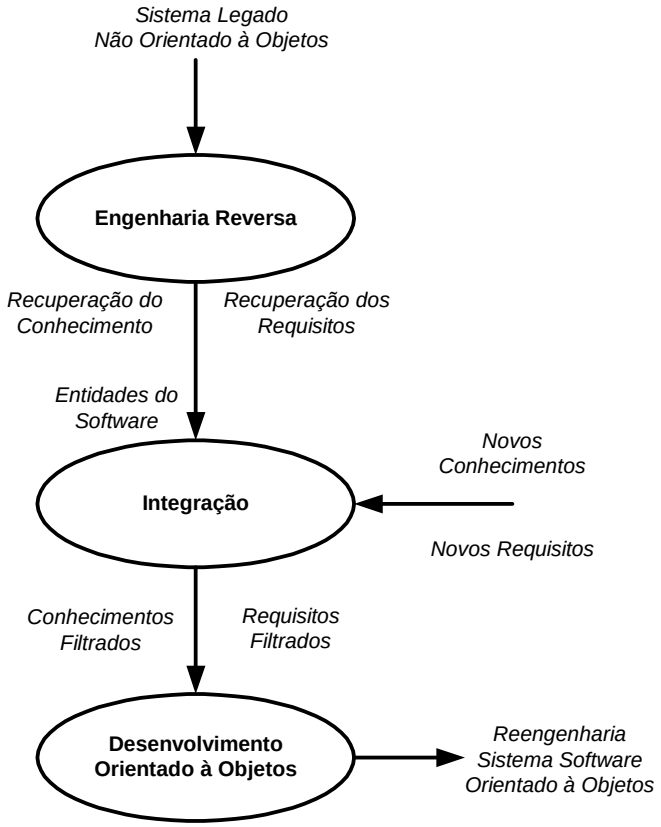


Ilustração 5 - Modelo de Reengenharia (BABIKER, 1997)

Esse método foi aplicado na reengenharia de um sistema implementado na linguagem “C” para a linguagem Smalltalk. Após essa aplicação o autor concluiu que o sistema reimplementado oferece diversas vantagens sobre o antigo:

- *Facilidade de Extensão:* O sistema que passou pelo processo de reengenharia é facilmente estendido, pois arquitetura orientada a objetos facilita as extensões de funcionalidades;

- *Aquisição do Entendimento do Software:* A primeira fase desse modelo oferece suporte para recuperação das informações do

sistema existente. Essa recuperação facilita futuras manutenções pois todos os requisitos estão contidos na documentação produzida;

- *Portabilidade:* o sistema pode sofrer mudanças para plataformas diferentes. Essa característica é apresentada pela existência da documentação produzida.

2.5. Considerações Finais

Nesta seção foram abordados temas que contribuirão para o desenvolvimento e pesquisa desta dissertação. Inicialmente foi apresentada a terminologia utilizada para os conceitos abordados neste trabalho. Após foi realizado um estudo sobre os problemas de manutenção em sistemas legados ressaltando assim a necessidade de métodos para minimizar estes problemas. Posteriormente, métodos de engenharia reversa que facilitam a manutenção e possibilitam a reengenharia de sistemas foram discutidos.

O método de engenharia reversa Fusion/RE e os conceitos correlatos serão abordados e detalhados no próximo capítulo para embasar a aplicação do estudo de caso onde será realizada uma engenharia reversa de um sistema legado procedimental.

3. MÉTODO DE ENGENHARIA REVERSA ORIENTADA A OBJETOS – FUSION/RE

3.1. Considerações Iniciais

O paradigma orientado a objetos utiliza um conjunto de conceitos específicos para o desenvolvimento de software. Estes conceitos ao serem utilizados na modelagem e implementação de um sistema oferecem ao engenheiro de software fatores positivos como facilidade de compreensão e alteração, reusabilidade e reaproveitamento de código.

Por apresentar os fatores positivos descritos anteriormente diversas pesquisas e métodos foram desenvolvidos abordando este paradigma. Entre eles destaca-se o método Fusion (COLEMAN, 1994), método orientado a objetos que tem como base os procedimentos e modelos dos métodos OMT (RUMBAUGH,1991), CRC (WIRFS - BROCK,1990), Booch (1997) e formais.

Tendo como base o método FUSION uma proposta de engenharia reversa orientada a objetos é elaborada, esta proposta apresenta o método FUSION/RE. Este método visa o entendimento, a recuperação da arquitetura e uma documentação orientada a objetos de sistemas legados.

A estrutura deste capítulo está organizada da seguinte forma: na Seção 3.2 são apresentados os conceitos orientados a objetos e os fatores positivos que estes conceitos trazem no desenvolvimento de sistemas. Uma pesquisa sobre o método FUSION abordando suas fases, modelos e procedimentos é apresentada na seção 3.3. Na Seção 3.4 descreve os passos e os documentos utilizados para a engenharia reversa de sistemas utilizando o método FUSION/RE.

3.2. Paradigma Orientado a Objetos

O paradigma de orientação a objetos tem como objetivo melhorar a manutenibilidade, reusabilidade e extensibilidade do sistema implementado (PENTEADO, 1996). Programas desenvolvidos segundo esse enfoque são mais fáceis de serem lidos e entendidos do que os desenvolvidos de acordo com a abordagem convencional. A complexidade do programa pode ser

controlada através da hierarquia funcional sobre os detalhes do programa e da ocultação de detalhes que não são relevantes ao programador em determinado momento.

O termo objeto é definido como a representação de uma entidade, unidade ou item identificável, individual, real ou abstrato, com um papel bem definido no domínio do problema (BOOCH,1997). Como exemplos de objetos, pode-se citar um aluno e sua matrícula como sendo objetos relevantes em um sistema acadêmico; um livro é considerado um objeto em um sistema de biblioteca, ou seja, objetos são especificados de acordo com o sistema que se deseja modelar e são caracterizados pelas suas informações e comportamentos associados.

As informações sobre um objeto são fornecidas pelos seus atributos e a utilização desses é viabilizada pelo comportamento ou pelos métodos que o objeto possui e disponibiliza a outros objetos no ambiente em que atuam. Portanto, os métodos podem ser considerados como meios de recuperação e criação de informações dos objetos. Quando um objeto necessita de uma informação de outro objeto, um pedido, através de uma mensagem, é realizado de um objeto para o outro. O objeto que sofre o pedido “atende” à mensagem executando o método correspondente, disponibilizando assim a informação solicitada.

Rumbaugh (1991) descreve o objeto como uma entidade independente, assíncrona e concorrente, que armazena dados, encapsula serviços, troca mensagens com outros objetos, e é modelado para executar as funções finais do sistema, como apresentado na Ilustração 6.

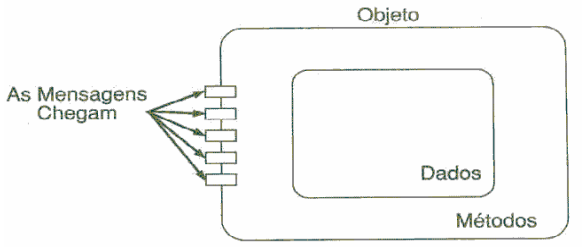


Ilustração 6 - Arquitetura do Objeto (RUMBAUGH, 1991)

No paradigma orientado a objetos há diversos conceitos relacionados a objetos, são eles: Classes, Mensagens, Herança e Polimorfismo, . Estes conceitos estão descritos a seguir nesta seção.

As classes são os blocos de construção mais importante de qualquer sistema orientado a objetos. Uma classe é uma descrição de um conjunto de objetos que compartilham os mesmo atributos, operações, relacionamentos e semântica (BOOCH, 1997). Em outras palavras, as classes contêm os moldes para a criação de objetos. A representação de uma classe é demonstrada na Ilustração 7 (WINBLAD, 1993).

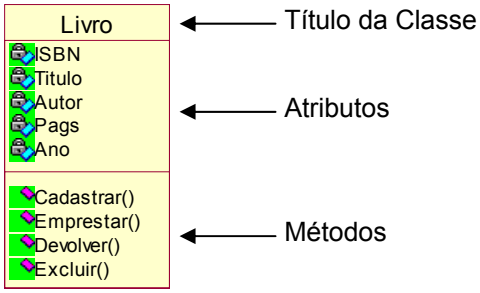


Ilustração 7 - Representação de Classe (WINBLAD, 1993)

Para a definição de novas classes na orientação a objetos é permitido utilizar uma classe já existente, este procedimento é denominado herança (VOSS, 1991). A classe criada (subclasse ou classe derivada) automaticamente herda todos os atributos e métodos da classe já existente (superclasse). O mecanismo de herança permite ainda que a subclasse inclua ou sobreponha novas variáveis e métodos da superclasse. Existem dois tipos de mecanismos de implementação de herança: simples e múltipla. Na herança simples, a subclasse pode herdar variáveis e métodos apenas de uma classe, enquanto na herança múltipla, a subclasse pode herdar variáveis e métodos de mais de uma classe. O mecanismo de herança é recursivo, permitindo criar-se uma hierarquia de classes.

O conceito de polimorfismo exige a utilização do conceito de herança e aplica-se apenas aos métodos da classe, o protocolo de comunicação é estabelecido na classe mais alta da hierarquia, que será herdada por todas as subclasses definidas posteriormente. Este mecanismo cria um protocolo

padrão de comunicação com um conjunto de objetos, permitindo uma grande flexibilidade na agregação de objetos semelhantes, mas não idênticos (BARROS, 2003).

Polimorfismo é o conceito usado em orientação a objetos para denotar a característica de que a linguagem suporta a utilização do mesmo identificador (o mesmo nome) para métodos de classes diferentes (BOOCH, 1997).

A utilização dos conceitos orientados a objetos apresentados nesta seção para o desenvolvimento de software apresenta fatores positivos, entre eles (JACOBSON,1994):

- *Compreensão:* pois se torna possível quebrar hierarquicamente as classes, obtendo-se um entendimento completo do sistema que está sendo modelado;

- *Entendimento:* o sistema é descrito em termos de objetos os quais, freqüentemente, têm uma ligação direta com o mundo real;

- *Alterações:* mudanças são feitas localmente para uma dada classe. Não há necessidade de afetar outras classes do modelo;

- *Adaptabilidade:* é possível especializar, com a ajuda dos mecanismos de herança existentes nas classes. Um modelo pode servir a diferentes situações através de adaptação para classes mais abstratas;

- *Reusabilidade:* classes podem ser construídas e manuseadas como componentes. Quando novas classes são criadas, propriedades de classes já definidas podem ser reusadas.

Devido aos fatores positivos descritos na utilização do paradigma orientado a objetos, muitas pesquisas foram realizadas e metodologias desenvolvidas para apoiar a especificação de sistemas sob essa abordagem.

Entre essas, encontra-se o método Fusion detalhado a seguir pois representa a base para o método de engenharia reversa utilizada neste trabalho.

3.3. Método Fusion

O método Fusion, criado por Coleman (1994) e outros, reúne características, procedimentos e modelos utilizados pelos métodos OMT (RUMBAUGH,1991), CRC (WIRFS - BROCK,1990), Booch (1997) e os formais. A Ilustração 8 ilustra a contribuição de cada um dos métodos citados para a elaboração do método Fusion.

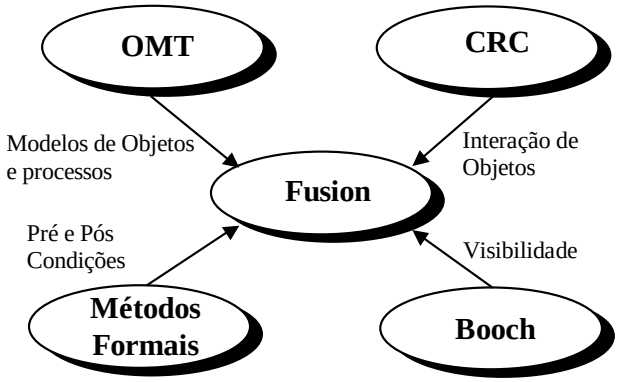


Ilustração 8 - Contribuição dos Métodos para o Fusion (COLEMAN, 1994)

A Ilustração 9 apresenta uma visão geral do método que possui três fases: análise, projeto e implementação. A interação existente entre essas três fases e os documentos produzidos em cada uma delas são também exibidos. O levantamento dos requisitos necessários produz um Documento de Requisitos, que é informal e utilizado como fonte de informações para a fase de análise.

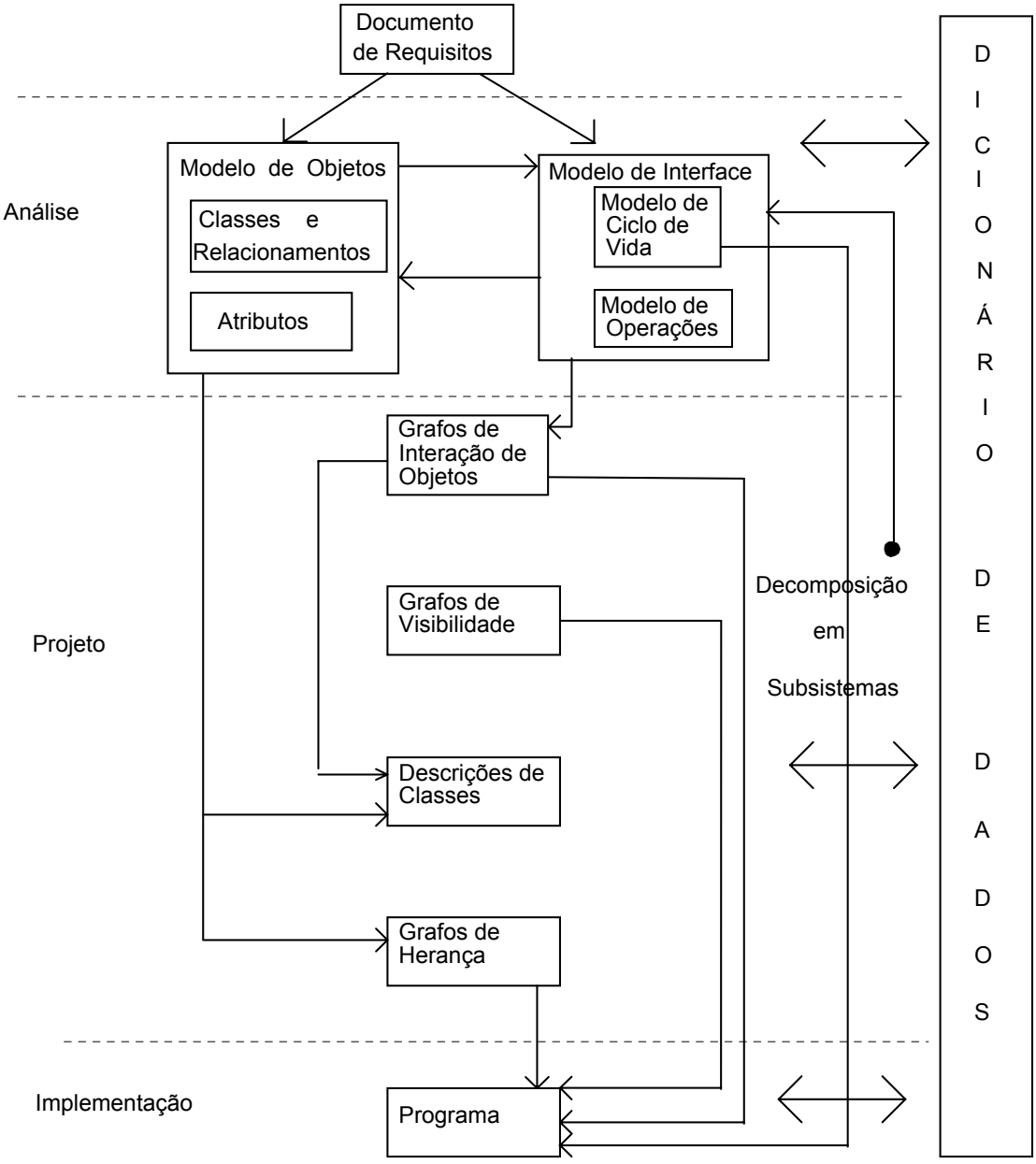


Ilustração 9 - Fases do Método Fusion (COLEMAN, 1994)

Para cada uma das fases existe um conjunto de ferramentas que facilitam a sua modelagem. A fase de análise, que tem o método OMT como base, oferece três modelos: o de Objetos, de Operação e de Ciclo de Vida. A fase de projeto, foi baseada em CRC e Booch e é composta dos seguintes modelos: Interação de Objetos, de Visibilidade, de Descrição de Classes e de Herança. A fase de implementação é realizada com apoio dos diagramas

elaborados na fase de projeto. Os modelos utilizados pelas fases são descritos a seguir.

- *Modelo de Objetos:* Tem como objetivo capturar e modelar as características relevantes do domínio do problema através de classes, relacionamentos entre elas, incluindo agregação e generalização/especialização, e atributos;

- *Modelo de Interface:* Formado pelo Modelo de Ciclo de Vida e Modelo de Operações, como pode ser visto na Ilustração 9;

- *Modelo de Ciclo de Vida:* Especifica através de expressões regulares as seqüências permitidas de interações com o ambiente em que o sistema pode participar, descrevendo o comportamento completo de como o sistema se comunica com o ambiente, desde a sua elaboração até o seu término;

- *Modelo de Operações:* Descreve o comportamento de uma atividade individual do sistema, chamada de operação, e é decorrente do Modelo de Ciclo de Vida. A cada evento de entrada corresponde uma atividade, que é especificada por um gabarito textual;

- *Grafo de Interação de Objetos:* Mostra os objetos que participam da execução de uma operação e a seqüência de trocas de mensagens entre eles.

- *Grafo de Visibilidade:* É construído para cada uma das classes existentes. Ele permite que se estude a necessidade de referências entre classes, a partir da troca de mensagens entre os objetos explicitada nos Grafos de Interação.

- *Descrição de Classes:* Através de uma notação textual abstrata, especifica os atributos de cada objeto e a sua interface externa, isto é seus métodos. As informações do Modelo de Objetos do Sistema, dos Grafos de Interação de Objetos e dos Grafos de Visibilidade são utilizadas para a derivação das descrições de classe.

- *Grafo de Herança:* Permite que novas abstrações possam ser acrescentadas, diferentes daquelas usadas no Modelo de Objetos do sistema, refletindo decisões de projeto.

Pode-se observar pela Ilustração 9 que as informações constantes do Modelo de Objetos são fonte de alimentação para os Grafos de Herança e para as Descrições de Classes, enquanto que as informações do Modelo de Interface alimentam o Grafo de Interação de Objetos.

O método Fusion recomenda o uso de um Dicionário de Dados durante as três fases, registrando os diversos componentes de cada modelo desenvolvido. Há também uma grande preocupação com os aspectos de qualidade e são recomendadas revisões ao final de cada fase, que visam a, principalmente, checar a complexidade da especificação e a consistência interna entre os modelos.

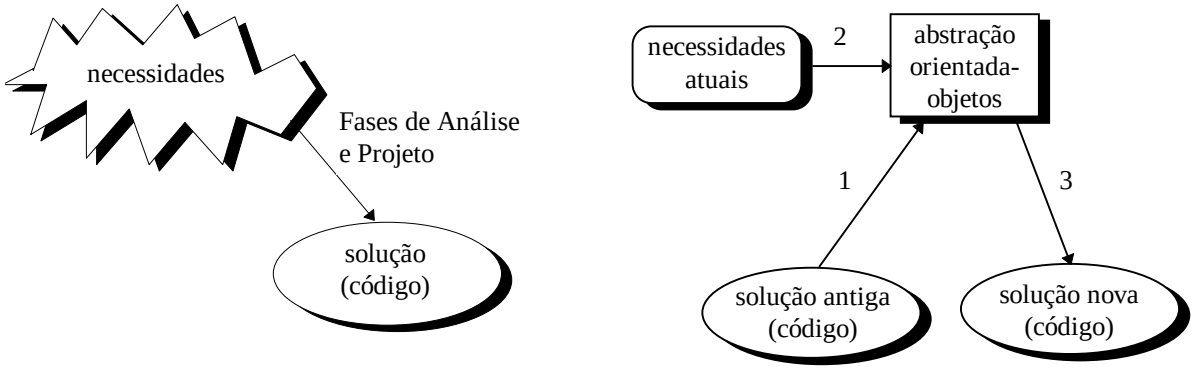
3.4. Método Fusion / RE

O Fusion/RE é um método de engenharia reversa orientado a objetos baseada no método Fusion (COLEMAN, 1994) e tem como objetivo a recuperação do projeto de sistemas legados. Para a realização do processo de engenharia reversa a partir dessa abordagem não há necessidade de se ter a documentação do sistema. A recuperação do projeto é possível apenas através do código fonte, análise do sistema através de sua execução e de entrevistas com os usuários, quando possíveis.

A Ilustração 10 mostra a diferença entre o desenvolvimento de software convencional e o processo de engenharia reversa Fusion/RE. Para realizar um desenvolvimento de software convencional, os requisitos do

sistema são obtidos e logo em seguida são realizadas as fases de análise, projeto e implementação. Esse processo tem como resultado o código fonte consistente com os requisitos solicitados, que visam atender completamente às necessidades dos usuários. Em um processo de engenharia reversa a recuperação do projeto inicia-se a partir do código fonte.

No método Fusion/RE o modelo recuperado possui características de orientação a objetos. Essa documentação completa do sistema pode ter várias utilidades, como: auxiliar em futuras manutenções do sistema, dar suporte a um processo de reengenharia, podendo alterar a funcionalidade do sistema, realizar a troca total ou parcial da implementação. Os números indicados nas setas representam a ordem em que as operações devem ser realizadas. Note que a operação (3) é realizada quando se opta pela alteração do código fonte.



a) Esquema de um desenvolvimento de software convencional

(b) Esquema da abordagem Fusion/RE

Ilustração 10 - Diferença entre Desenvolvimento Convencional e Fusion/RE

Os passos do método de engenharia reversa orientada a objetos FUSION/RE estão resumidos na Tabela 1. As atividades, modelos e documentações geradas por cada fase estão descritos em detalhes neste capítulo.

Tabela 1 - Passos do Método Fusion/RE

Passo	Objetivos	Artefatos
Revitalizar a Arquitetura do Sistema com base na documentação existente	Obter informações relacionadas à arquitetura do sistema para o seu entendimento.	Lista de todos os procedimentos, sua descrição e a relação chamam/chamado por
Recuperar o Modelo de Análise da Solução Atual	Obter um modelo de análise considerando somente os aspectos físicos	Documentos descritos nos passos 2.1., 2.2., 2.3., 2.4.
2.1. Definir Temas	Modelar em temas as informações armazenadas relativas às entradas, saídas, armazenamento permanente e temporário	Lista de Temas
2.2 Desenvolver o Modelo de Objetos	Elaborar um modelo com as classes e seus relacionamentos, extraído dos tipos abstratos de dados que compõem a base de dados do sistema	Modelo de Objetos, lista de atributos, procedimentos associados às classes, lista das anomalias existentes
2.3. Definir o Ciclo de Vida	Mostrar o comportamento global do sistema	Modelo de Ciclo de Vida
2.4. Abstrair e Desenvolver as operações	Obter as operações realizadas pelo sistema	Modelo de Operações

Passo	Objetivos	Artefatos
3. Abstrair o Modelo de Análise do Sistema	Obter um modelo de análise do sistema considerando os aspectos do domínio da aplicação	Documentos descritos nos passos 3.1., 3.2., 3.3.
3.1. Desenvolver o Modelo de Objetos	Elaborar um Modelo de Objetos considerando as classes e seus relacionamentos que devem ser tratados pelo sistema	Modelo de Objetos. Para cada classe: lista dos atributos e métodos
3.2. Elaborar o Modelo de Ciclo de Vida	Fornecer uma visão global do comportamento do sistema a partir da abstração realizada	Modelo de Ciclo de Vida
3.3. Especificar o Modelo de Operações	Descrever como as operações devem ser realizadas	Modelo de Operações
4. Mapear o Modelo de Análise do Sistema para o Modelo de Análise do Sistema Atual	Descrever a Relação entre os Modelos de Análise do Sistema atual e novo.	Mapeamento das Classes e dos Métodos do MAS para o MASA

3.4.1. Revitalizar a Arquitetura do Sistema com Base na Documentação Existente

O objetivo é recuperar as informações relacionadas à arquitetura do sistema para que ele possa ser entendido/estendido pelo engenheiro de software, facilitando o desenvolvimento dos passos posteriores. Caso o sistema possua documentação adequada, basta apenas identificá-la e organizá-la de forma a permitir que a funcionalidade do sistema seja completamente entendida.

A documentação deve descrever a função de cada um dos procedimentos utilizados para implementação do sistema, fazendo-se referência cruzada entre eles: os procedimentos utilizados (CHAMA) e os procedimentos utilizadores (CHAMADO POR). Essas informações são obtidas a partir do código fonte existente. Os parâmetros das chamadas dos procedimentos não são especificados aqui, embora eles possam ser obtidos a partir do código existente. A Ilustração 11 mostra um trecho da documentação produzida nesta fase do método.

De qualquer modo que se recuperem as informações para revitalizar a arquitetura do sistema são importante que o engenheiro de software tenha conhecimento dos métodos e técnicas usadas para projeto e implementação do sistema atual.

Módulo	Descrição
======	========
Módulo1.h	Este módulo refere-se às operações realizadas sobre módulo 1.h
Procedimentos	
proc1	Procedimento referente às operações realizadas por proc1. CHAMA: ======= proc2 proc3 (Módulo 2.h) proc4 CHAMADO POR: ============ proc3 (Módulo 3.h) proc5

Ilustração 11 - Modelo de Documento Chama x Chamado por

3.4.2. Recuperar o Modelo de Análise da Solução Atual

Deve-se desenvolver neste passo um modelo de análise do sistema atual considerando-se somente os seus aspectos físicos, ou seja, deve-se considerar somente a implementação atual. Os resultados deste passo são o Modelo de Objeto, de Operações e de Ciclo de Vida, esses modelos compõem o Modelo de Análise do Sistema Atual (MASA).

3.4.2.1. Definir Temas

O objetivo é modelar os grandes temas relativos às informações que o sistema manipula. Geralmente eles estão relacionados às entradas (interface), saídas e às informações armazenadas em caráter temporário e permanente. As informações de caráter temporário são aquelas em que o acesso é permitido somente durante o processamento, ou seja, relativas às operações do sistema executadas na memória RAM.

Já o termo permanente é utilizado para caracterizar aquelas informações que ficam armazenadas permanentemente em algum dispositivo de armazenamento não volátil como por exemplo, discos. O resultado deste sub-passo é uma lista de temas abrangendo todas as informações relacionadas ao sistema.

3.4.2.2. Desenvolver o Modelo de Objetos do Sistema Existente

Para cada um dos assuntos considerados no item anterior deve-se listar inicialmente todas as classes existentes, identificadas a partir das estruturas de dados implementadas. Os Modelos de Objetos criados neste sub-passo são abstrações realizadas visando a facilitar o entendimento do sistema existente, uma vez que a implementação atual não segue a abordagem de orientação a objetos. O termo Modelo de Objetos deve ser entendido como um modelo intermediário que representa uma abstração de um código não implementado usando os conceitos e técnicas orientadas a objetos. É construído e usado para a criação do Modelo de Análise do Sistema (fase 3).

A seguir são sugeridos os passos para a recuperação do Modelo de Objetos para três cenários possíveis de implementação do sistema atual resumidos na Ilustração 12.

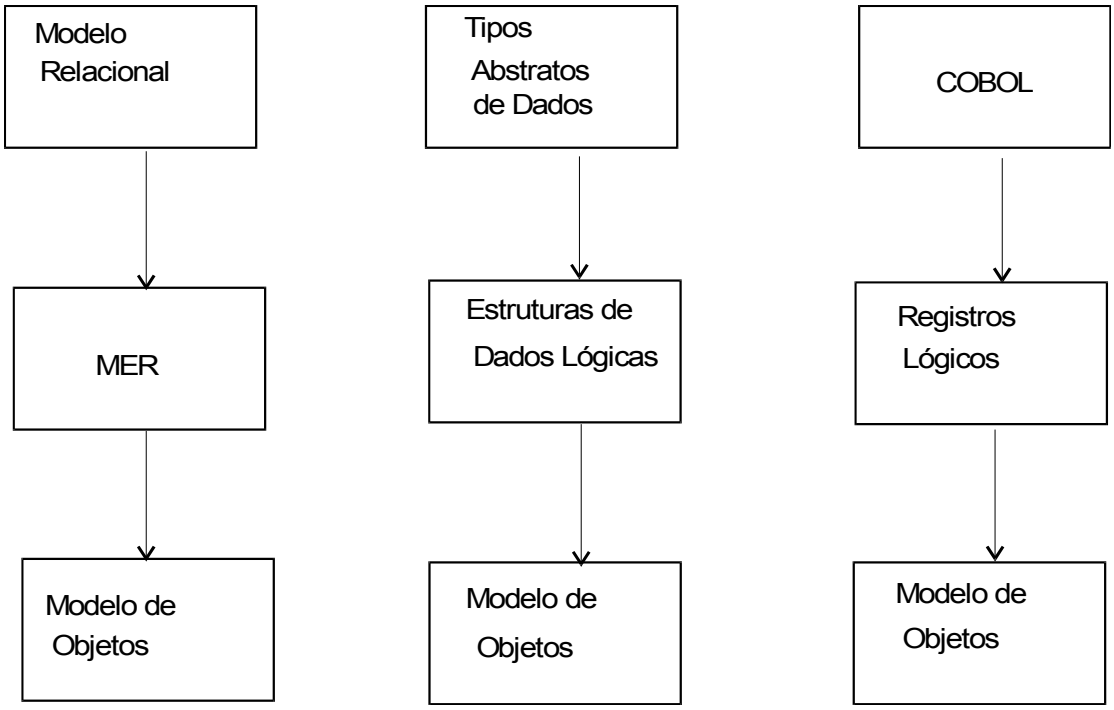


Ilustração 12 - Cenários Possíveis da Implementação Existente

Se a implementação foi realizada utilizando base de dados relacional o engenheiro de software deve consultá-la a fim de obter todas as entidades, relacionamentos e chaves das relações para projetar o modelo de entidade-relacionamento (MER). Como no modelo relacional os dados são armazenados através de tabelas e a duplicação de dado em tabelas indica que ele é chave de um relacionamento, obtêm-se as informações necessárias para recuperar as entidades e seus relacionamentos. A partir do MER pode-se facilmente adaptá-lo para o paradigma Orientado a Objetos e outras características adicionais podem ser acrescentadas ao modelo, tais como cardinalidade e papéis.

Se a técnica utilizada na implementação for Tipos Abstratos de Dados (TAD) deve-se consultar o código para descobrir quais são as estruturas de dados relevantes, de uso global ou local. Nessa técnica as estruturas de dados geralmente contêm ponteiros que são os responsáveis pelas ligações existentes entre elas, mas pode haver duplicação de chaves. Como passo intermediário pode-se construir um diagrama simples com as estruturas de dados lógicas recuperadas e os relacionamentos entre elas.

Esse diagrama pode então ser convertido para a notação do Fusion e as demais características semânticas (exemplo: cardinalidade, papéis, etc.) podem ser acrescentadas. A partir daí tem-se condições de se recuperar quais são os procedimentos que atuam sobre essas estruturas de dados lógicas e construir uma documentação de apoio a essa fase do método.

A partir dessas informações preliminares são definidos os registros lógicos. Em seguida verificam-se os relacionamentos entre eles, que podem ter sido implementados por índices, chaves duplicadas, contigüidade física, etc. Esse diagrama intermediário pode então ser convertido para o paradigma Orientado a Objetos e, assim como nos casos anteriores, outras características semânticas são acrescentadas.

É importante notar que para a construção do MASA não é necessário dar ênfase às estruturas de agregação e especialização/generalização, uma vez que provavelmente elas não fazem parte da implementação atual. Estruturas simples de agregação, como as que relacionam um arquivo e seus registros podem ser usadas sempre que relacionamentos nos dois níveis existirem.

A classe é descrita através de seus atributos e métodos. O relacionamento entre essas classes é obtido através da própria relação expressa entre os tipos de dados, isso é, os ponteiros existentes entre as diversas estruturas ou o esquema da base de dados. A partir das classes de objetos deve-se listar todos os atributos da classe. Os atributos são os mesmos que os existentes em cada uma das estruturas de dados selecionadas.

Os procedimentos são obtidos com base no conhecimento do engenheiro de software do sistema e dos documentos obtidos na fase anterior.

Deve-se analisar cada um dos procedimentos e verificar qual estrutura de dados está associada a ele. Além disso deve-se analisar a forma de associação entre a classe e o procedimento. O procedimento será associado à classe que ele consulta ou constrói. Adotou-se a convenção (c) para construtor, quando o procedimento altera a estrutura de dados, e (o) para observador, quando o procedimento somente consulta a estrutura.

Como já foi dito anteriormente, quando o sistema não tem uma implementação orientada a objetos, a análise dos procedimentos é complexa. Elaborou-se uma classificação de anomalias, sendo que as mais comuns se referem a procedimentos contidos na tabela 2:

Tabela 2 - Classificação de Procedimentos

observador de uma classe e construtor de outra	(oc)
observador de uma classe e construtor de mais de uma classe	(oc+)
observador de mais de uma classe e construtor de outra	(o+c)
observador de mais de uma classe e construtor de mais de uma classe	(o+c+)
construtor de duas classes (ou mais)	(c+)
observador de duas classes (ou mais)	(o+)

Além da classificação descrita seguem algumas considerações para a análise dos procedimentos:

- Existem procedimentos que não se referem a classe alguma, sendo dependentes da implementação, relacionados à interface ou ao módulo escalonador, etc., Esses procedimentos são classificados como (i);

- Quando um procedimento é construtor de uma classe e também observador dessa mesma classe, este deve ser classificado somente como construtor;

- Se mais de uma classe está associada a um procedimento então este será associado a uma única classe, seguindo a seguinte ordem de prioridade:
  1. À classe construída se for do tipo oc;
  2. À primeira classe construída analisada se for do tipo oc+;
  3. À primeira classe consultada analisada se for do tipo o+c.
  4. Quando não há anomalia o procedimento é classificado simplesmente como (o) ou (c).

Como produto deste passo tem-se: a) Modelo de Objetos descrevendo as classes e relacionamentos utilizando a notação do método Fusion; b) as classes, atributos e procedimentos identificados e; c) a classificação dos procedimentos com ou sem anomalias, como mostra a Ilustração 13.

Classe: X1			
Procedimentos			
a1	[o/m1]		o
a2	[c/m2]	[o/c1/t1]	oc
a3	[c/m1]		c
. . .			
a4	[c/m3]	[c/c2/t2]	c+

Ilustração 13 - Classificação dos Procedimentos por Classe

3.4.2.3. Definir o Ciclo de Vida

O objetivo deste sub-passo é definir o ciclo de vida do sistema. A observação em uso da interface do sistema atual é geralmente suficiente para definir todos os eventos de entrada e de saída que o sistema aceita. As seqüências de operações permitidas podem ser registradas em cenários que, reunidos e generalizados, são a base para a definição do ciclo de vida. Manuais de uso do sistema, quando existentes, também são úteis para auxiliar a definir as operações do sistema e a assegurar a completitude da análise.

Para auxiliar a construção do Modelo de Ciclo de Vida pode-se construir uma tabela, do tipo da Ilustração 14, contendo todas as informações existentes relacionadas com as chamadas de procedimentos quando o sistema atual é ativado. Devem ser listadas as opções da interface, as operações que estão associadas a essas opções e a hierarquia de chamadas, isto é , os nomes dos procedimentos que implementam as operações.

Deve-se listar os procedimentos até o segundo nível de chamada sempre que o primeiro for um procedimento tipo (i), já que esses são inerentes à forma utilizada e portanto não realizam operação específica.

OPÇÕES DE INTERFACE	OPERAÇÕES	HIERARQUIA DE CHAMADAS
---------	--------	--------
---------	--------	--------

Ilustração 14 - Opções Existentes na Interface do Ambiente

3.4.2.4. Abstrair e Desenvolver as Operações

Uma operação é normalmente implementada pelo procedimento chamado pelo módulo ou programa controlador da interface a partir da ocorrência de um evento de entrada. Muitas vezes há uma cadeia de chamadas até se chegar ao procedimento que realmente executa a operação, geralmente auxiliado por outros procedimentos subordinados. Esse deve ser o procedimento base para a especificação da operação, pois os demais cuidam de aspectos da interface e são dependentes da implementação. Consistências efetuadas podem dar origem a pré-condições.

As abstrações relevantes para a definição das operações são encontradas geralmente até no penúltimo nível da hierarquia de chamada de procedimentos a partir da interface. Essas operações são parte daquelas classificadas como (i), na seção 3.4.2.2. O nível mais baixo contém os procedimentos que constroem ou observam objetos, dando origem aos demais tipos da classificação apresentada na seção 3.4.2.2. A Ilustração 15 fornece

um esquema de onde podem ser obtidas as informações para a elaboração do Modelo de Operações.

Operação	Modelo de Ciclo de Vida
Descrição	Documento de Revitalização da Arquitetura do Sistema (descrição do procedimento com o mesmo nome que o da operação que está sendo descrita)
Lê	Código fonte
Modifica	Código fonte
Envia	Código fonte
Assume	Código fonte juntamente com os conhecimentos do engenheiro de Software e do Documento de Revitalização da Arquitetura do Sistema
Resultado	Código fonte

Ilustração 15 - - Fontes de informações para o Modelo de Operação

3.4.3. Abstrair para o Modelo de Análise do Sistema

Até o momento trabalhou-se com informações disponíveis na implementação existente, construindo-se o Modelo de Análise do Sistema Atual (MASA). A partir do passo 3 realiza-se a abstração da visão física criando-se a visão lógica do sistema, denominada de Modelo de Análise do Sistema (MAS). A visão lógica abstrai da visão física, sempre que possível, aspectos que deveriam ter sido especificados anteriormente e que não foram. O produto gerado é semelhante aos documentos gerados no passo 2, com exceção da lista de temas, dando-se enfoque no domínio da aplicação e não na implementação (domínio da solução). Um Dicionário de Dados, manual ou automatizado, deve ser usado para armazenar informações sobre o MAS. Neste passo são utilizados os termos “atributos” e “métodos” para identificar os dados e os algoritmos associados a certa classe.

3.4.3.1. Desenvolver o Modelo de Objetos

As classes do MAS são abstraídas das classes do MASA. É comum que um conjunto de classes e relações do MASA dê origem a uma classe do MAS, pois os objetos de implementação escolhidos (listas, vetores de ponteiros, etc.) devem ser abstraídos para um conceito do sistema. Pode ocorrer também que uma classe (tipo) do MASA com mais de uma instanciação concreta (instâncias) dê origem a mais de uma classe no MAS, correspondentes aos conceitos do sistema. Por exemplo, uma pilha pode armazenar informações relacionadas a mais de um tipo de objeto. Quando ela for especificada no MAS dará origem a tantas classes de objetos quantas as diferentes informações que ela instancia.

Os atributos são analisados e normalmente têm seu nome modificado para um mnemônico com significado mais próximo do conceito que ele representa no mundo real e com maior legibilidade (sem abreviações, por exemplo). Alguns atributos são desnecessários porque são dependentes da implementação e os que apontam para outras classes (referências) são substituídos por relacionamentos. Os atributos podem também se deslocar para a superclasse ou subclasse, conforme se identifiquem relações de generalização/especialização, respectivamente.

Alguns métodos canônicos, como aqueles que criam a classe e permitem acesso a atributos da classe (para atualizar ou consultar) podem ser incorporados ao MAS neste passo, mas o método Fusion não recomenda que isso seja feito, podendo ser deixado para as fases de projeto ou de implementação.

É importante notar que os métodos classificados como (i) não dão origem a métodos. Nesse passo também não são apresentados os parâmetros que acompanham os procedimentos, embora esses sejam facilmente encontrados no código fonte existente.

O Modelo de Objetos do sistema produzido neste sub-passo contém classes e seus respectivos relacionamentos, como o proposto no método Fusion. De acordo com o método Fusion depois da construção do Modelo de Objetos deve-se verificar a interface do sistema, ou seja, adicionar limites ao

modelo. Não há preocupação com esse passo no método de engenharia reversa orientada a objetos uma vez que o próprio MAS define o limite da implementação em função do estudo realizado pelo MASA anteriormente, devendo-se cuidar do Modelo de Ciclo de Vida do Sistema. Entretanto os agentes podem ser acrescentados ao modelo como classes, da maneira utilizada no Fusion.

3.4.3.2. Elaborar o Modelo de Ciclo de Vida

O Modelo de Ciclo de Vida para o MAS é praticamente o mesmo que o produzido no passo 3.4.2.3, mas tomando-se o cuidado de alterar os nomes dos eventos de entrada e de saída para nomes mais mnemônicos. Cada alteração realizada implica também em mudança do nome da operação correspondente.

Interfaces de comandos exigem, além da experimentação com o sistema, alguma documentação escrita com a especificação dos comandos. Interfaces guiadas por menus ou por manipulação direta de ícones na tela são mais simples de serem analisadas. Com base na tabela criada no sub-passo 3.4.2.3, como mostra a Ilustração 14, na documentação produzida quando da revitalização do sistema e do conhecimento do engenheiro de software pode-se elaborar o Modelo de Ciclo de Vida do MAS, com o mesmo “layout” apresentado quando da elaboração do MASA.

Se o objetivo principal do projeto de reengenharia dentro do qual está inserida a engenharia reversa for não alterar a funcionalidade do sistema o Modelo de Ciclo de Vida para o MAS é praticamente o mesmo que o produzido no passo 3.4.2.3, a menos de alterações dos nomes dos eventos de entrada e de saída.

A ampliação da funcionalidade implicaria em outras mudanças que fogem ao escopo deste trabalho mas poderiam ser integradas a esta abordagem. Por exemplo: seria necessária a verificação de outros relacionamentos entre as classes, a adição de novas operações e a elaboração do Modelo de Ciclo de Vida seguindo essas modificações realizadas.

3.4.3.3. Especificar as Operações

Para cada uma das operações do Modelo de Operações desenvolvido no sub-passo 3.4.2.4, possivelmente com seu nome alterado no sub-passo 3.4.3.2, deve-se reespecificar a operação para o nível de abstração do MAS. Deve-se respeitar a funcionalidade especificada mas atualizar a notação para o estilo do Fusion e considerar agora as classes, relacionamentos e atributos do Modelo de Objetos do MAS. O Modelo de Operações segue o exemplo mostrado na Ilustração 15.

3.4.4. Mapear o Modelo de Análise do Sistema para o Modelo de Análise do Sistema Atual

Este mapeamento é fundamental para o desenvolvimento futuro de todo ou de substituição de partes do sistema atual, facilitando o reuso. O próprio desenvolvimento do sub-passo 3.4.3.1. conduz para o mapeamento entre os documentos do MAS e correspondentes do MASA, que deve ser cuidadosamente documentado e checado em termos de completitude neste passo.

Esse mapeamento facilita tanto o entendimento como a implementação futura do sistema, pois mostra facilmente como os elementos do MAS estão atualmente implementados facilitando a reutilização. Pode-se avaliar neste ponto quanto da implementação pode ser reutilizada caso se decida continuar com a mesma abordagem atualmente utilizada e não passar a adotar uma nova abordagem como a de orientação a objetos. Em termos de notação a correspondência pode ser feita por tabelas conforme Ilustração 16, mas a informação pode também estar armazenada no Dicionário de Dados.

MAS	MASA
Modelo de Objetos Classe: nome dado no MAS	 nome dado no MASA
Atributos	Elementos de Dados
---------- ---------- ----------	---------- ---------- ----------
Métodos	Procedimentos
---------- ---------- ----------	---------- ---------- ----------

Ilustração 16 - Mapeamento MASA x MAS

Para cada uma das classes descritas no MAS elabora-se uma tabela conforme mostrada na Ilustração 16, relacionando-as com o seu correspondente no MASA. Essa correspondência deixa de existir para os objetos necessários quando o sistema sofre mudança de funcionalidade. Para cada classe são listados os atributos e seus elementos de dados correspondentes bem como os métodos associados e os correspondentes procedimentos. Pode ocorrer que um procedimento dê origem a mais de método e, nesse caso, o procedimento aparecerá repetidamente na coluna correspondente ao MASA.

3.5. Interligação dos Documentos Produzidos

A Ilustração 17 exibe o relacionamento existente entre os documentos atuais existentes do sistema e os documentos produzidos pelo método de engenharia reversa orientada a objetos. Pode-se notar que os procedimentos, estruturas de dados e o conhecimento dos usuários e profissionais são o subsídio para a recuperação da arquitetura do sistema. Se o sistema possuir alguma documentação, ela, juntamente com os documentos anteriormente citados é que dá apoio à geração dessa nova documentação. A partir dessa

documentação, recuperando-se a arquitetura do sistema, elaboram-se os documentos do MASA e posteriormente os do MAS.

Pode-se observar pela Ilustração 4.17 que os documentos produzidos pela fase de revitalização da arquitetura são fonte de alimentação tanto para os documentos do MASA como do MAS. Os Temas resultam das consultas às Estruturas de Dados e a partir deles e da consulta aos Procedimentos o Modelo de Objetos é criado. Com as informações armazenadas no Modelo de Objetos gera-se o Modelo de Ciclo de Vida.

Por sua vez, o Modelo de Operação tem origem a partir das informações encontradas no Modelo Ciclo de Vida, juntamente com o Modelo de Objetos. A partir dos documentos criados no MASA passa-se à elaboração do MAS. Para a criação dos documentos do MAS utilizam-se todos os modelos criados anteriormente e os documentos correspondentes à Revitalização da Arquitetura do Sistema. O último passo do método consta do Mapeamento entre os documentos produzidos no MASA e no MAS.

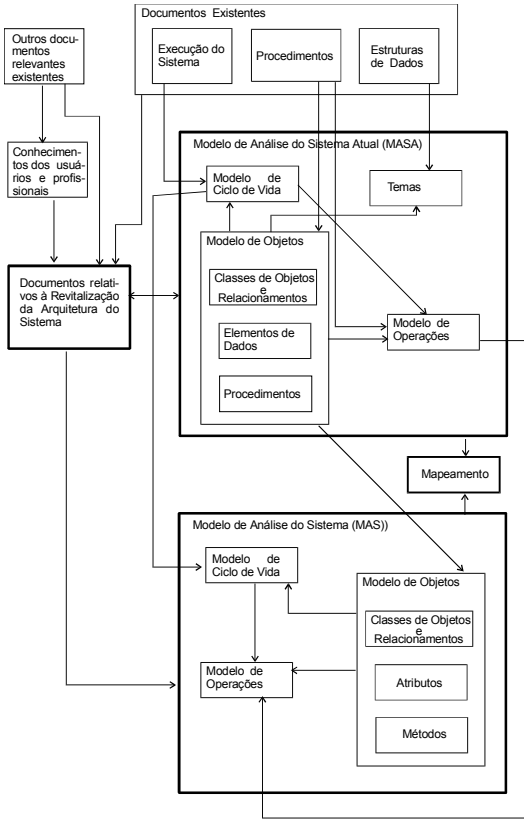


Ilustração 17 - Interligação dos documentos produzidos

3.6. Considerações Finais

Neste capítulo foram pesquisados conceitos e métodos que apoiarão a aplicação da engenharia reversa no estudo de caso deste trabalho. A primeira seção apresentou os conceitos utilizados para a modelagem e o desenvolvimento de sistemas no paradigma orientados a objetos. Em seguida as fases e modelos do método orientado a objetos FUSION foram descritas. Finalizando o capítulo, um detalhamento do método de engenharia reversa Fusion/RE foi apresentado descrevendo seus passos e documentos. Este detalhamento forneceu base para a aplicação da engenharia reversa no estudo de caso deste trabalho. Esta aplicação e uma avaliação dos procedimentos da engenharia reversa serão apresentados no próximo capítulo.

4. APLICANDO O MÉTODO FUSION/RE AO SISTEMA SIGLO REVENDAS®

4.1. Considerações Iniciais

Este capítulo apresenta a aplicação do método de engenharia reversa orientado a objetos Fusion/RE em um sistema comercial desenvolvido com linguagem procedimental e banco de dados estruturado. Serão apresentadas as aplicações dos passos do método Fusion/RE e suas respectivas documentações geradas após cada passo.

A aplicação do método de engenharia reversa Fusion/RE no estudo de caso proposto apresentou características variantes do modelo original devido ao tipo de sistema escolhido. Estas variações bem como uma proposta de diretrizes para validar a documentação gerada com o sistema analisado também são apresentadas neste capítulo.

4.2. Características Técnicas do Estudo de Caso

O estudo de caso escolhido é um sistema de informação denominado SIGLO REVENDAS® (JABUR INFORMÁTICA, 2003), utilizado e cedido para esta referida pesquisa pela empresa Jabur Informática. O SIGLO REVENDAS® é composto por 19 módulos e foi desenvolvido no ambiente de Programação Progress. O sistema SIGLO REVENDAS® apresenta todas as características de um sistema legado e para tornar este trabalho mais objetivo foi feita a engenharia reversa de um módulo deste sistema, o módulo de visitas.

O módulo de visitas tem por característica, cadastrar todos os visitantes (clientes preferenciais) para que os vendedores estabelecem contatos periódicos com esses clientes, seja pessoalmente ou por telefone, em um período de tempo pré-determinado para oferecer-lhes produtos seja para compra, venda ou somente orçamento. O módulo também gera relatórios demonstrando as visitas feitas pelos vendedores e seus respectivos resultados.

A implementação deste módulo apresenta as métricas especificadas na tabela 3:

Tabela 3 - Métricas do Estudo de Caso

Métrica	Media
Telas	23
Relatórios	07
Estruturas de Dados	55 (22 inerentes ao módulo de visitas)
Linhas de Código	20,88 Kloc (lines of code)

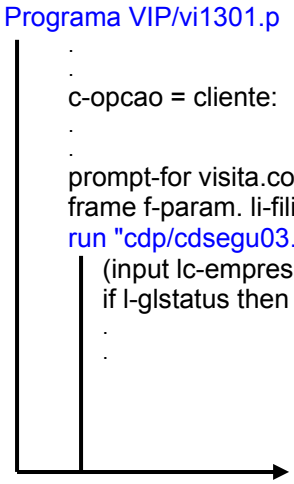
A engenharia reversa do sistema SIGLO REVENDAS® foi executada seguindo as diretrizes e passos inerentes ao método FUSION/RE. Devido ao tipo de sistema escolhido para o estudo de caso deste trabalho, a aplicação do método FUSION/RE apresentou variações se comparada ao modelo original. Estas variações referem-se a uma maior ou menor ênfase realizada em determinadas atividades contidas nos passos fornecidos pelo método.

A aplicação do estudo de caso gerou uma documentação extensa e detalhada, por isto as próximas seções apresentam trechos desta documentação gerada e seus respectivos comentários visando uma melhor clareza e objetividade deste trabalho.

4.3. Revitalizar a Arquitetura do Sistema

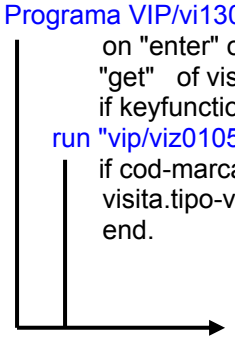
Conforme descrito no método Fusion/RE, o objetivo deste passo é recuperar as informações relacionadas à arquitetura do sistema para que ele possa ser entendido/estendido pelo engenheiro de software. Neste passo é analisado o código fonte para levantamento e especificação dos procedimentos e seus respectivos relacionamentos. A especificação gerada nesta fase é uma tabela denominada pelo método Fusion/RE como Chama x Chamado Por.

As Ilustrações 18 e 19 demonstram a atividade de leitura do código fonte e a análise para a especificação dos procedimentos e seus respectivos relacionamentos.



CHAMA	*CHAMADO*
Vip/vi1301.p	c-opcao = “Cliente”
c-opcao = “Cliente”	cdp/cdsegu03.p (verificacao de permissao de acesso)
:	:

Ilustração 18 - Análise de Procedimentos (Exemplo 1)



CHAMA	*CHAMADO*
Vip/vi1301.p	on "enter" visita.tipo-vis ou "get" visita.tipo-vis f-geral
on "enter" visita.tipo-vis ou "get" visita.tipo-vis f-geral	vip/viz0105.p (Zoom de marca de veiculos)
:	:

Ilustração 19 - Análise de Procedimentos (Exemplo 2)

A análise do código fonte e especificação dos procedimentos fornecem base para a geração de uma documentação que representa a arquitetura do sistema, este documento é denominado pelo método Fusion/RE com Tabela Chama x Chamado Por. A tabela x apresenta parcialmente o documento gerado neste passo.

Tabela 4 - Tabela de Procedimentos Chama x Chamado Por

Chama	Chamado Por	Descrição
vip/vi1301.p		Cadastrar Cliente
	cdp/cdsegu01.i	criação de variáveis para empresa principal
	cdp/cdsegu02.i	cria variáveis locais com valores das publicas
	cdp/cd0120.i	chama programa de verificação de segurança do sistema
	utp/ut0201b.p	verificação de segurança do sistema
	cd/cd9000.i	arquivo que contem variáveis comuns aos programas
	cdp/cd9900.i1	definição de variáveis do digito verificador de c.g.c
	cdp/cd9000.f	Formularia da tela padrão
Cdp/cdsegu04.p		validação inicial de fontes em geral (somente a senha de ADM pode acessar os programas pelo editor)
	on "enter" visita.tipo-vis	checa o tipo(marca) de carro da visita
. . .	. . .	. . .

4.4. Recuperar o Modelo de Análise do Sistema Atual

Neste passo é desenvolvido um modelo orientado a objetos que represente a análise do sistema atual. Este modelo é especificado a partir dos aspectos físicos, ou seja, deve-se considerar somente a implementação atual. Os documentos gerados neste passo são o Modelo de Objeto, de Operações e de Ciclo de Vida, esses modelos compõem o Modelo de Análise do Sistema Atual (MASA).

4.4.1. Definir Temas

Esta atividade tem como objetivo detalhar os grandes temas relativos às informações que o sistema manipula. Devido a este trabalho apresentar a aplicação do método Fusion/RE em somente um módulo do sistema SIGLO REVENDA® não se fez necessário elaborar a documentação desta atividade pois o tema ficou restrito a uma única funcionalidade: Controlar Visitas.

4.4.2. Desenvolver o Modelo de Objetos do Sistema Existente

Analisando as estruturas de dados implementadas são abstraídos os objetos que irão compor o Modelo de Objetos do Sistema Existente. Este Modelo de Objetos visa facilitar o entendimento do sistema existente, uma vez que a implementação atual não segue a abordagem de orientação a objetos e deve ser entendido como um modelo intermediário que representa uma abstração de um código não implementado usando os conceitos e técnicas orientadas a objetos.

A análise das estruturas de dados seguiu um dos cenários propostos pelo método Fusion/RE para a recuperação do Modelo de Objetos. O cenário aplicado (Ilustração 20) foi o que determina diretrizes para sistemas implementados em modelos relacionais.

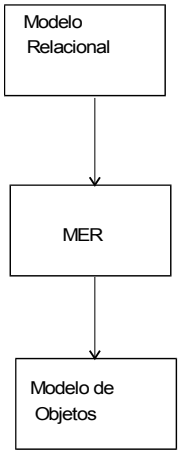


Ilustração 20 - Cenário utilizado para recuperação do Modelo de Objetos

A aplicação deste cenário deu-se através de consultas obtendo todas as entidades (tabela 5) , após levantamento das entidades foi realizada uma análise para verificação das funcionalidades de cada entidade especificada (tabela 6). Esta análise tem como objetivo fornecer entendimento do sistema ao engenheiro de software e nortear a nomenclatura dos objetos conforme suas funcionalidades.

Tabela 5 - Entidades do Sistema Existente

Entidade
Estabelec
Histórico
Vendedor
Visita
Emitente
Cad-cbpo
Repres
Hid-emit
Natur-oper
. . .

Tabela 6 - Tabela de Entidades x Funcionalidades

Tabelas	Funcionalidades
Vendedor	Tabela com as informações dos vendedores
Visita	Tabela com as visitas agendadas dos vendedores
Histórico	Tabela com os históricos as visitas realizadas pelos vendedores
Cad-cbpo	Tabela de Profissões
Grupo	Tabela de Classificação de Emitentes por Grupos
Classe	Tabela de Classificação de Emitentes por Classe
Estatist	Tabela de Estatísticas por Emitente
Veículo	Tabela de Veículos
. . .	. . .

Depois de obtidas e analisadas as entidades foram verificadas as chaves primárias e estrangeiras existentes para declaração dos relacionamentos. Através desta análise foi elaborado o projeto do modelo de entidade-relacionamento (MER), Ilustração 21.

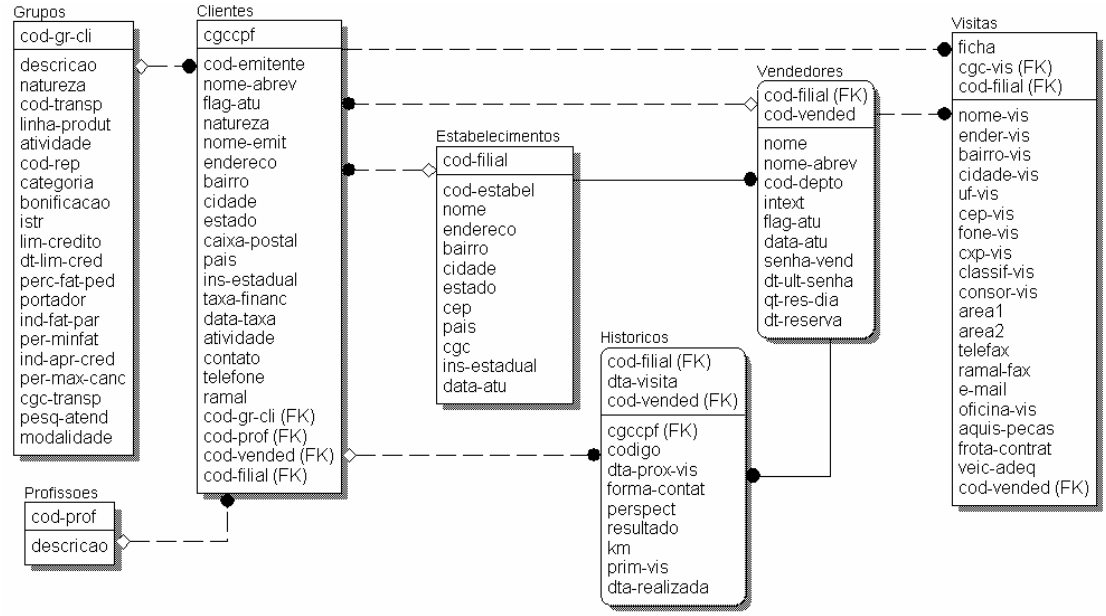


Ilustração 21 - Modelo de Entidade-Relacionamento

A partir do MER realizou-se a abstração para o Modelo de Objetos (Ilustração 22) e outras características adicionais podem ser acrescentadas ao modelo, tais como cardinalidade e papéis.

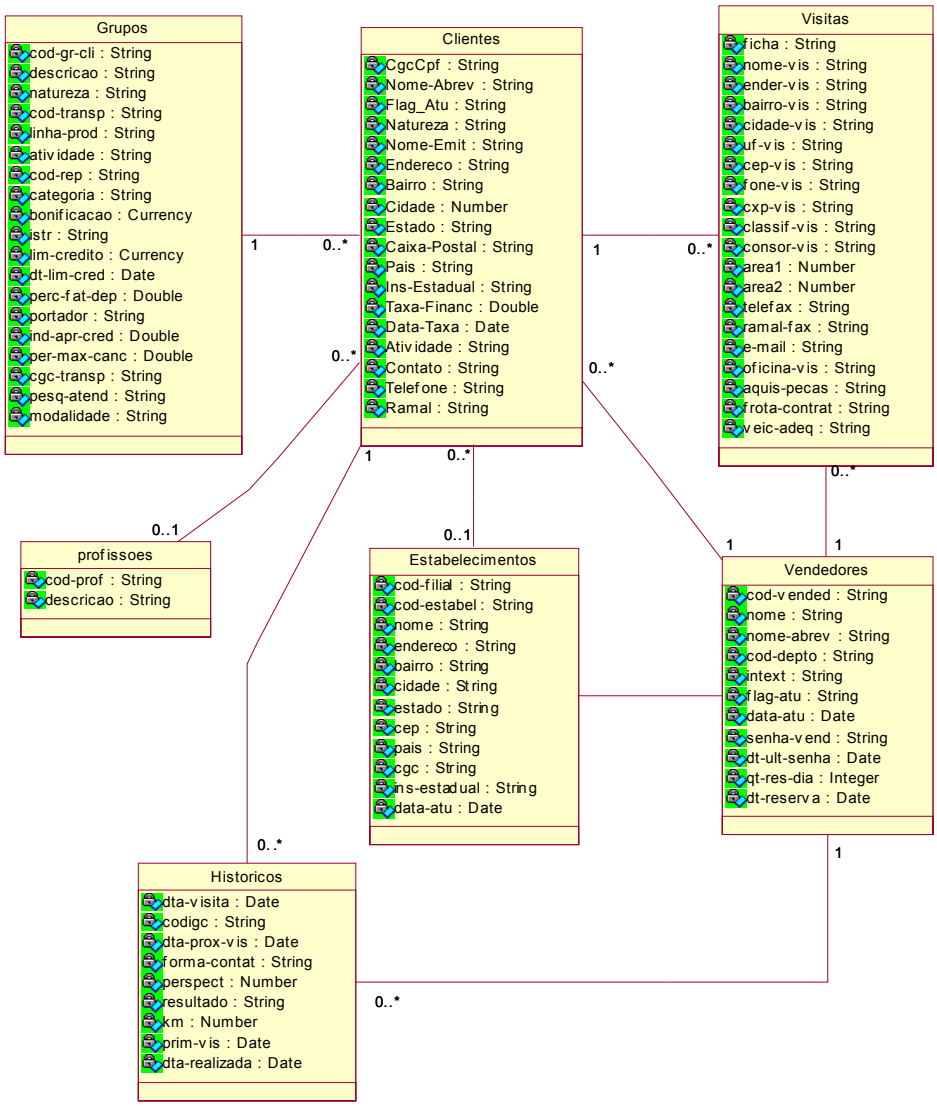


Ilustração 22 - Modelo de Objetos do Sistema Existente

A análise dos procedimentos foi obtida com base no conhecimento do engenheiro de software do sistema e nos documentos obtidos no passo anterior (Revitalizar a Arquitetura do Sistema). Esta análise verificou cada um dos

procedimentos e definiu a forma de associação entre as classes já especificadas e os procedimentos analisados. Esta associação seguiu as diretrizes e tabela de classificação (Tabela 2) descritas na seção 3.4.2.2. deste trabalho.

Após análise é gerada uma documentação (Ilustração 23) conforme modelo definido pelo Método FUSION/RE. Esta documentação representa a classe, os método e suas respectivas classificações.

Destaca-se na Ilustração 23 um exemplo de anomalia nos métodos Gravar_visita e Desativar Cliente. O procedimento Desativar_Cliente tinha em sua implementação rotinas de pesquisa nas classes Visitas e Históricos (classificação o+) e rotinas de alteração e gravação na classe Clientes (c). O procedimento gravar_visita continha rotinas de pesquisa nas classes clientes e vendedores (o+) e uma rotina de gravação e alteração de dados na classe Visitas (c). Nos casos exemplificados foi analisada a seguinte diretriz do método Fusion/RE para determinar o relacionamento do procedimento com a classe:

Se mais de uma classe está associada a um procedimento então este será associado a uma única classe, seguindo a seguinte ordem de prioridade:

1. À classe construída se for do tipo oc;
2. À primeira classe construída analisada se for do tipo oc+;
3. À primeira classe consultada analisada se for do tipo o+c.
4. Quando não há anomalia o procedimento é classificado simplesmente como (o) ou (c).

Conforme determina a 1a. prioridade desta diretriz os métodos Desativar_Cliente e Gravar_Visita por serem construtores das classes Clientes e Visitas ficam relacionados com as mesmas respectivamente.

Classe: Clientes	
Procedimentos	
Pesq_CnpjCpf	o
Criar	c
PesqCli_Fones	o
Desativar_Cliente	***o+c***
Atualizar_Cliente	c
...	

Classe: Visitas	
Procedimentos	
Gravar_Visita	***o+c***
Alterar_Data	c
....	

Ilustração 23 - Associação e Classificação de Métodos x Classes

4.4.3. Abstrair e Desenvolver as Operações

Analisando os procedimentos mapeados nos passos anteriores são efetuadas as especificações das operações. As abstrações relevantes para a definição das operações não devem considerar os procedimentos classificados como de interface pois estes cuidam de aspectos da interface e são dependentes da implementação. As Ilustrações 24 e 25 apresentam exemplos de operações especificadas nesta atividade.

Operação	PesqCli_Fones()
Descrição	Recupera os fones do cliente desejado
Lê	Clientes
Modifica	------
Envia	Agenda
Assume	Clientes já cadastrados
Resultado	Fone Residencial, Fone Comercial e Fone Celular

Ilustração 24 - Modelo da Operação PesqCli_Fones()

Operação	Gerar_Agenda()
Descrição	Criar agendas de visitas para vendedores
Lê	Cliente, Vendedor, Histórico, Agenda
Modifica	Agenda, Histórico
Envia	
Assume	Não pode ocorrer agenda duplicada de visitas (mesmo dia e horário) e são agendadas visitas para vendedores e clientes previamente cadastrados.
Resultado	Geração de uma agenda de visita.

Ilustração 25 - Modelo da Operação Gerar_Agenda()- Modelo da Operação Gerar_Agenda()

Após a elaboração de todos os modelos apresentados até este ponto, as informações relevantes do sistema atual foram obtidas e finalizou-se a recuperação do MASA. A seguir passa-se à elaboração do modelo de análise

do sistema (MAS) que é abstraído do modelo de análise do sistema atual (MASA).

4.5. Abstrair para o Modelo de Análise do Sistema (MAS)

Finalizada a análise e modelagem da implementação existente através da construção do Modelo de Análise do Sistema Atual (MASA), o próximo passo é realizar a abstração da visão física criando-se a visão lógica do sistema, denominada de Modelo de Análise do Sistema (MAS). As atividades e documentações geradas neste passo são semelhantes ao passo anterior, diferenciando-se pelo nível de abstração e detalhamento voltados para o domínio da aplicação e não na implementação. Neste passo são utilizados os termos “atributos” e “métodos” para identificar os dados e os procedimentos associados a certa classe.

4.5.1. Desenvolver o Modelo de Objetos

O modelo de objetos documentado no MAS é analisado e um novo modelo foi gerado reduzindo os detalhes de implementação e focalizando detalhes do sistema. Neste modelo foram inseridos os conceitos orientados a objetos necessários para uma correta apresentação da aplicação.

Os atributos e métodos foram analisados e, quando necessário, tiveram seus nomes alterados para melhor legibilidade (sem abreviações, por exemplo) ou melhor representar suas funcionalidades.

Abstraindo para um modelo que utilizou conceitos orientados a objetos diversas alterações foram realizadas no modelo do MAS, um exemplo é o deslocamento de atributos de uma subclasse para uma superclasse ou subclasse, conforme se relação de herança detalhada na Ilustração 26.

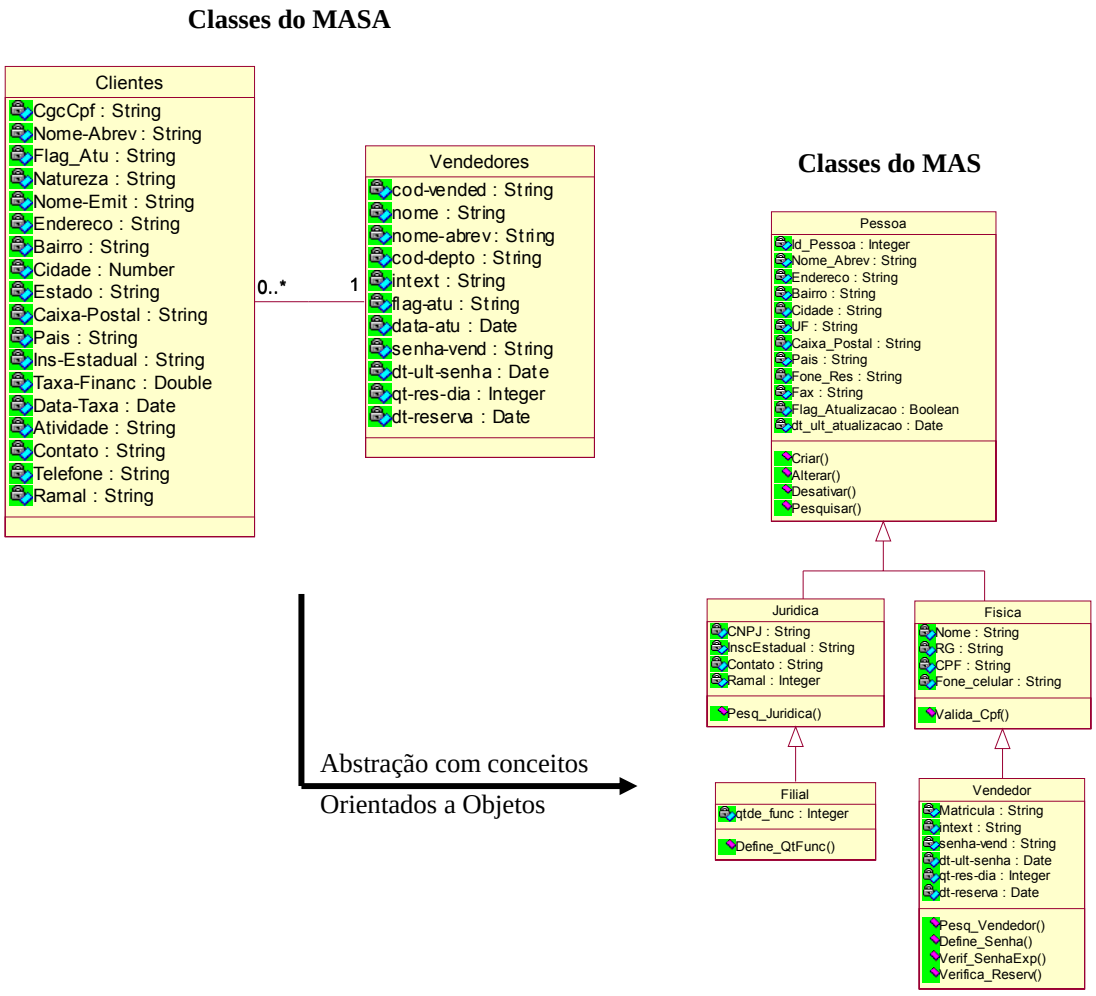


Ilustração 26 - Abstrações Realizadas do MASA para o MAS

A Ilustração 26 exemplifica uma abstração realizada no estudo de caso. Neste exemplo as classes Clientes, Vendedores e Estabelecimentos especificadas no Modelo de Objetos do MAS foram mapeadas através de uma herança (generalização/especialização), os atributos em comuns das classes analisadas foram deslocados para as superclasses. Outra abstração realizada foi a geração da classe filial a partir da análise dos atributos e funcionalidades da classe estabelecimentos.

Outras abstrações foram realizadas neste sub-passo inserindo conceitos orientados a objetos. A Ilustração 27 demonstra uma parte do modelo de objetos inerente ao MAS gerado após análise de todas as classes e seus respectivos métodos e atributos.

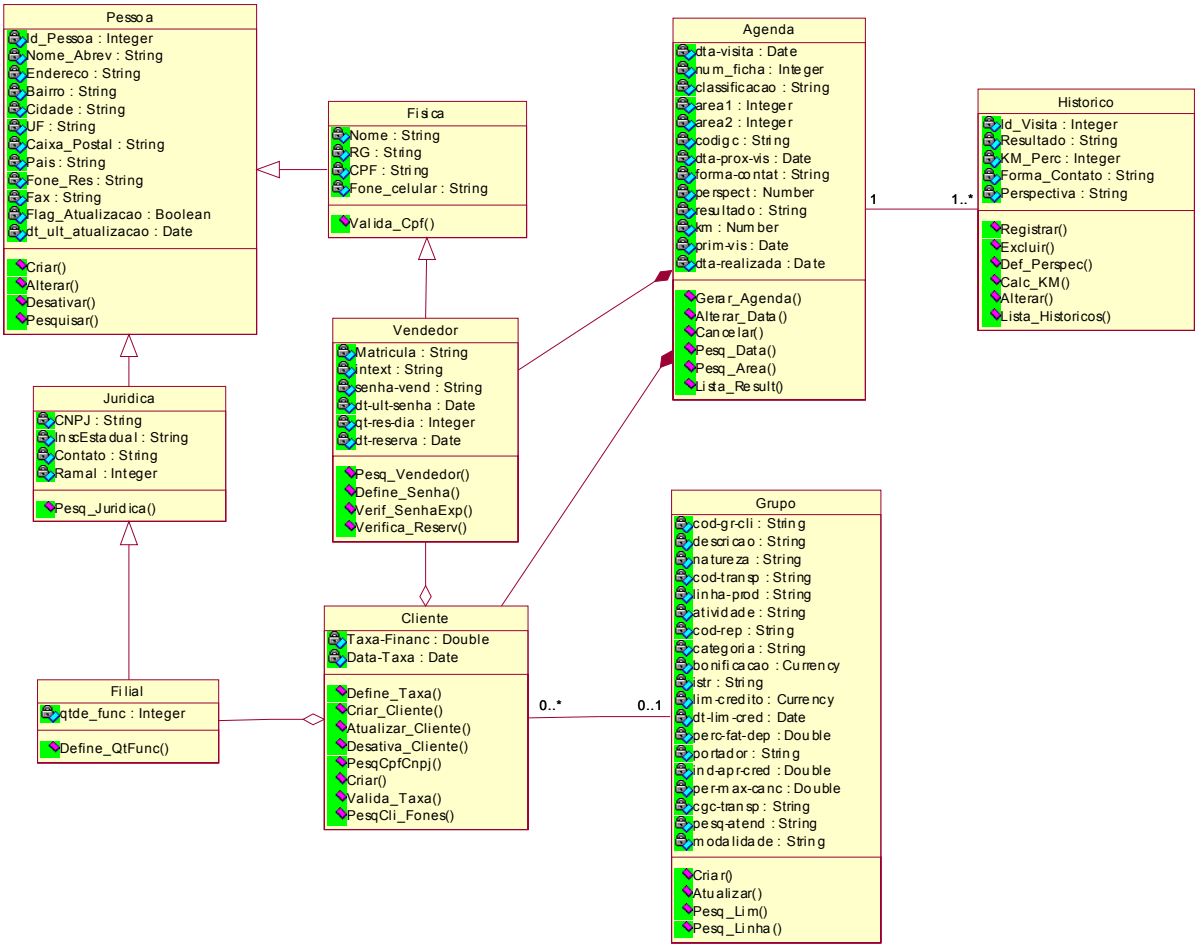


Ilustração 27 - Classes do Modelo de Objetos do MAS

4.6. Mapear o Modelo de Análise do Sistema para o Modelo de Análise do Sistema Atual

Neste passo foi realizado o mapeamento entre os modelos MASA e MAS, gerados durante a engenharia reversa do estudo de caso. Este mapeamento foi realizado através de tabelas conforme mostra a Ilustração 28. Para cada uma das classes descritas no MAS elabora-se uma tabela, relacionando-as com o seu correspondente no MASA. Para cada classe foram

listados os atributos e seus elementos de dados correspondentes bem como os métodos associados e os correspondentes procedimentos.

MAS	MASA
Agenda	Visitas
Atributos	Elementos de Dados
Num_ficha Classificação Resultado Área1 Área2 . . .	Ficha classif-vis result-vis área1 area2 . . .
Métodos	Procedimentos
Gerar_agenda() Alterar_Agenda() Alterar_Data() . . .	On enter op = ”Salvar” On enter op = “Alterar” On enter op = “Re-Agendar” . . .

Tabela de Mapeamento MASA x MAS

Pode ocorrer de classes, métodos ou atributos não estarem mapeados entre os modelos MASA e MAS, um exemplo deste não relacionamento entre os modelos pode ser representada pela classe Filial que foi abstraída no modelo MAS através da análise das classes clientes, vendedores e estabelecimento. Esta abstração está representada na Ilustração 26.

4.7. Validação do Modelo de Análise do Sistema

Ao término da engenharia reversa utilizando o método FUSION/RE foi obtida uma documentação orientada a objetos completa e detalhada sobre o sistema analisado. Porém um critério não compreendido pelo método FUSION/RE é a validação entre a documentação gerada e o sistema atual, ou seja, não há no método FUSION/RE um passo que verifique se a documentação gerada suporta as funcionalidades do sistema atual. Esta validação é de extrema importância pois tem com finalidade verificar se todas as funcionalidades do sistema foram analisadas e recuperadas no processo de engenharia reversa.

Este trabalho apresenta a proposta de um novo passo no método FUSION/RE para realizar a validação da documentação gerada. Esta validação teve como procedimento básico a execução de todas as funcionalidades do sistema atual e uma análise e modelagem das interfaces que suportam as funcionalidades analisadas.

A documentação gerada neste passo é o diagrama de seqüência por se tratar da ferramenta que permite mapear as classes e seus componentes (métodos e atributos) com suas respectivas interfaces e funcionalidades. Os diagramas de seqüência gerados a partir da execução e análise das funcionalidades são validados com a documentação da engenharia reversa para verificar se as classes especificadas nesta documentação suportam as funcionalidades modeladas.

A seguir são apresentados os procedimentos e documentos utilizados para a execução deste passo. Os exemplos demonstram situações que validaram a documentação da engenharia reversa gerada verificando sua consistência com as funcionalidades implementadas no sistema.

A execução deste passo é descrita a seguir utilizando a funcionalidade Cadastrar Clientes (Ilustração 28) como exemplo. Esta funcionalidade apresenta vários procedimentos implementados: Inclui, Modifica, Elimina, Histórico, Lista entre outros.

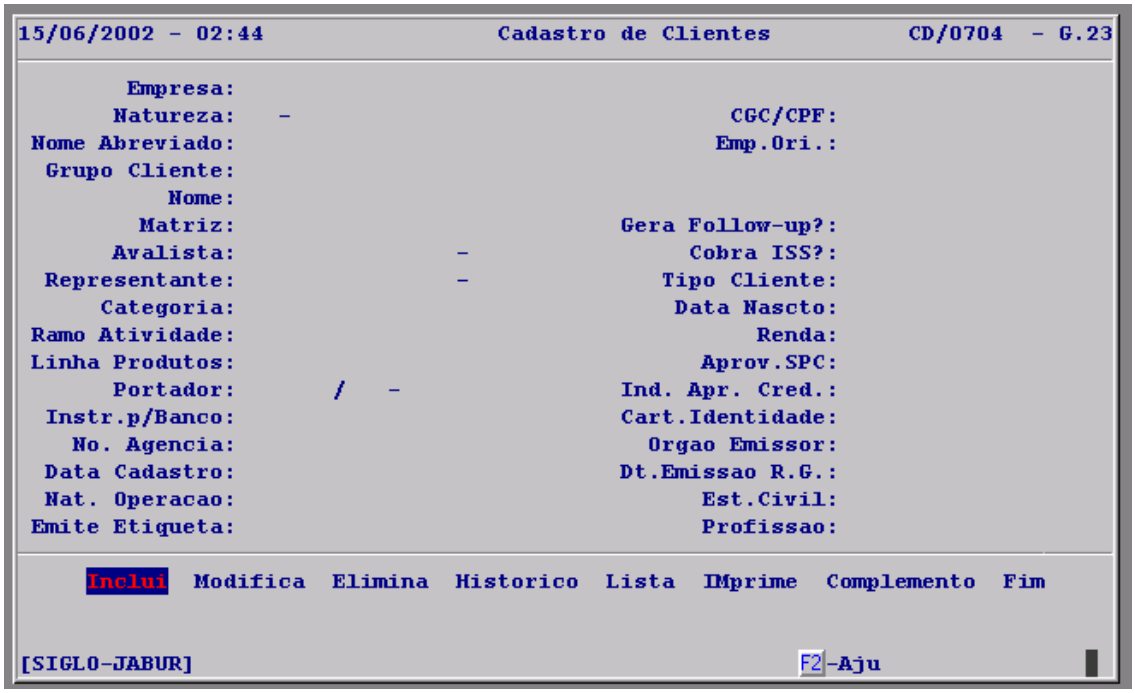


Ilustração 28 - Tela de Cadastro de Clientes

Para cada operação de sistema implementada o engenheiro de software deve executá-lo e especificar o respectivo diagrama de seqüência. A Ilustração 30 apresenta o diagrama de seqüência gerado por esta atividade e representa a funcionalidade Inclui . Nesta análise foi verificado que a funcionalidade Inclui utiliza dois procedimentos para sua execução: o pesquisar por cpfcnpf e o gravar cliente.

O modelo de seqüência gerado validado com a documentação do processo de engenharia reversa. Neste caso a validação constatou uma consistência da documentação pois os procedimentos implementados na funcionalidade Inclui são suportados pela classe Cliente através dos métodos Pesq_CpfCnpj e Criar_Cliente (Ilustração 29).

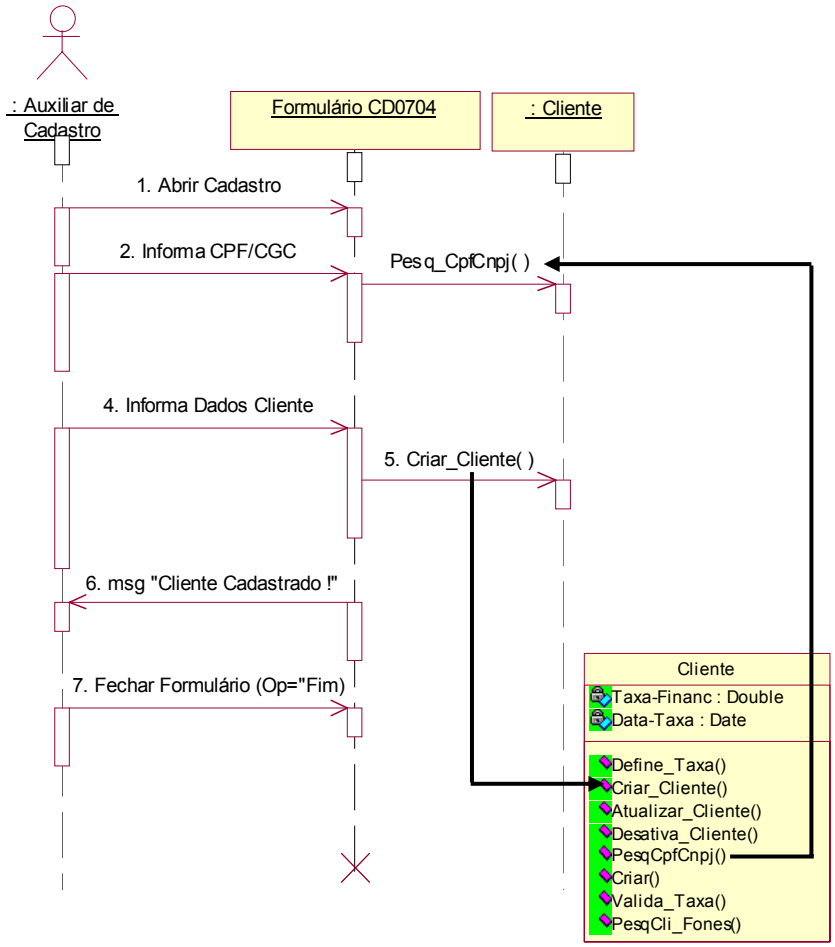


Ilustração 29 - Validação entre Diagrama de Seqüência e Classes Abstraidas da Engenharia Reversa

A Ilustração 30 apresenta a análise e validação da funcionalidade Agendar Visitas. Também neste caso a validação garantiu a consistência entre as classes especificadas na engenharia reversa e a funcionalidade analisada. Este consistência é demonstrada na Ilustração relacionando os procedimentos implementados na funcionalidade com os métodos especificados nas classes.

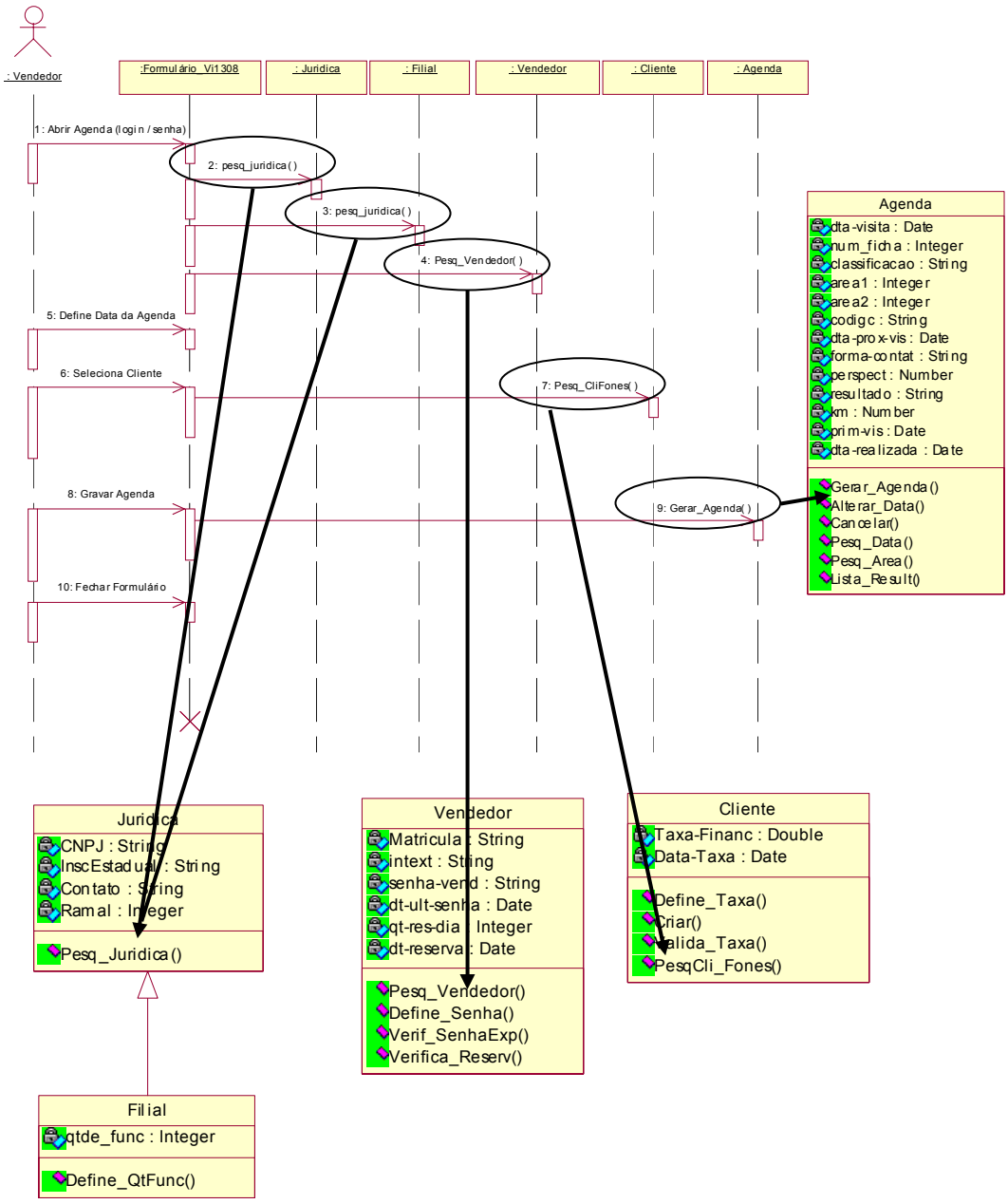


Ilustração 30 - Validação da Funcionalidade Agendar Visitas

Um exemplo de verificação da inconsistência entre a documentação gerada com o sistema atual é apresentada na Ilustração 31 Neste caso a funcionalidade Histórico, executada a partir de Cadastrar Cliente (Ilustração 28), foi executada e analisada gerando um diagrama de seqüência.

O diagrama de seqüência gerado representa que a execução da funcionalidade Histórico utiliza um procedimento para Listar os Históricos. Ao validarmos o diagrama de seqüência com as classes especificadas na documentação da engenharia reversa foi verificado que nenhuma classe possuía método(s) para suporta tal funcionalidade. Nesta caso uma alteração (atualização) na documentação da engenharia reversa é gerada para eliminar esta inconsistência.

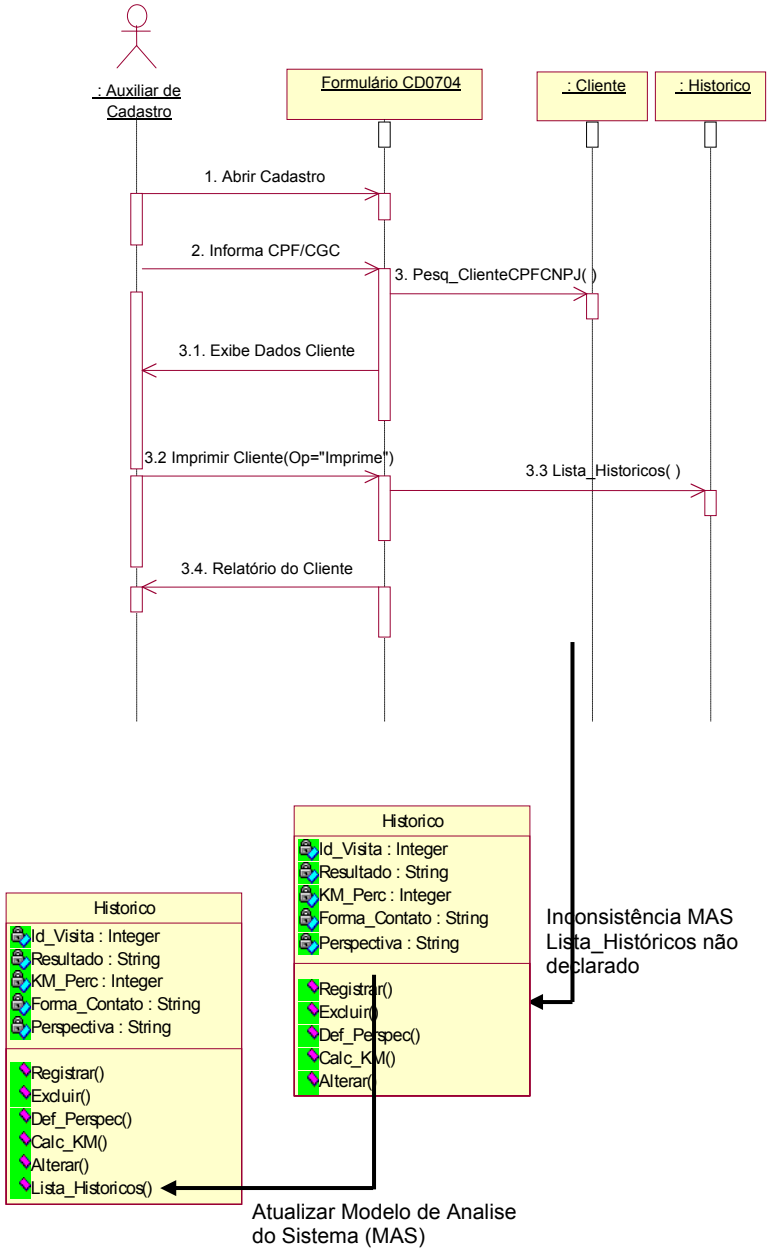


Ilustração 31 - Validação da Funcionalidade Histórico

4.8. Considerações Finais

A aplicação do método de engenharia reversa FUSION/RE em um sistema legado procedimental foi apresentada neste capítulo. Inicialmente foram detalhadas informações sobre o sistema contendo métricas e suas características técnicas: tamanho em linhas de código (loc), número de telas e estrutura de dados, ambiente de programação utilizado para o desenvolvimento, etc.

Após especificado o sistema as seções seguintes apresentaram a aplicação do método FUSION/RE, foram detalhados os passos e a documentação gerada em cada atividade da engenharia reversa, as variações de procedimentos quanto ao método original FUSION/RE também foram especificadas.

Finalizando o capítulo a última seção propõe a execução de um novo passo não previsto no método FUSION/RE. Este passo tem como objetivo validar se documentação gerada no processo de engenharia reversa suporta as funcionalidades do sistema atual. Passo este importante para verificar se todas as aplicações do sistema foram analisadas e recuperadas pelo processo de engenharia reversa.

5. CONCLUSÃO

A maioria dos sistemas que atualmente suportam aplicações comerciais, são sistemas que apresentam um elevado tempo de utilização e foram desenvolvidos sem a utilização correta dos conceitos de engenharia de software ocasionando projetos mal elaborados e problemas na documentação (falta ou desatualização).

Estes sistemas, denominados Sistemas Legados, consomem um grande esforço na atividade de manutenção mas não são abandonados por atenderem aos requisitos dos usuários ou suportarem tarefas importantes.

Para auxiliar o entendimento e recuperar a documentação destes sistemas e conseqüentemente facilitar a atividade de manutenção vários métodos de engenharia reversa foram propostos, entre eles o FUSION/RE que apresenta como vantagem a recuperação do entendimento e documentação do sistema dentro do paradigma orientado a objetos.

Este trabalho demonstra a aplicação do método FUSION/RE em um sistema comercial escolhido como estudo de caso. Durante a aplicação prática do método FUSION/RE no estudo de caso surgiram procedimentos variantes da proposta original do método que foram especificados.

Outra contribuição apresentada é a aplicação de um novo passo ao término do processo de engenharia reversa. Este passo definido como Validação do Modelo de Análise do Sistema tem como finalidade validar se a documentação gerada no processo de engenharia reversa suporta as funcionalidades do sistema atual. Esta validação foi especificada na seção 4.7 e tem como base à análise das funcionalidades do sistema através de sua execução com as classes, atributos e procedimentos modelados na documentação final.

Como contribuições futuras a este trabalho podem ser desenvolvidas pesquisas para o desenvolvimento de ferramentas que automatizem o processo de identificação dos procedimentos, uma vez que esta atividade foi realizada manualmente.

Pode-se descrever como outra contribuição importante para complementação desta pesquisa um estudo uma abordagem formal para o passo que realizou a validação do modelo gerado com o sistema atual.

Atividades de Verificação, Validação e Testes (VV&T) também podem ser futuramente pesquisadas e aplicadas durante os passos do método apresentado neste trabalho.

REFERÊNCIAS

BABIKER, E. et al. A model for Reengineering Legacy Expert Systems to Object-Oriented Architecutre. Expert Systems with Applications. Califórnia, USA, v.12, n.3, p.363-371,1997.

BARROS, P. UML: Linguagem de Modelagem Unificada - Português. Disponível em http://www.uml.com.br/arquivos/tutoriais/Apostila%20UML.doc. Acesso em: 04/03/2003.

BENEDUSI, P. Improving reverse engineering models with test-case related knowledge. Information and Software Technology. Portici, Italy, n.38, p.711-718, 1996.

BOOCH, G. Object-oriented analysis and design with applications, 2ed, Massachusetts: Addison-Wesley,1997.

BRAGA, R.T.V. Padrões de Software a Partir da Engenharia Reversa de Sistemas Legados. 1998. Dissertação. (Mestrado em Ciências de Computação e Matemática Computacional) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos.

CAGNIN, M. I.; PENTEADO, R. Avaliação das vantagens quanto à facilidade de manutenção e expansão de sistemas legados sujeitos à engenharia reversa e segmentação. Anais do XX Congresso Nacional da Sociedade Brasileira de Computação, Curitiba, p.32, 2000.

CHIKOFSKY, E. J. ; CROSS, J. H. Reverse engineering and design recovery: a taxonomy. IEEE Software. v.7, p13 – 17, jan.1990.

COLEMAN, D. et al. Desenvolvimento Orientado a Objetos: O Método Fusion. Rio de Janeiro: Campus, 1994.

CORDEIRO, Marco A. Manutenibilidade de Software. Celepar – Companhia de Informática do Paraná, Curitiba. Disponível em: http://www.pr.gov.br/celepar/batebyte/Internet–anteriores/2000/bb102/manutenibilidade.htm. Acesso em: 20/04/2003.

GALL, H., KLÖSCH R. Capsule Oriented Reverse Engineering for Software Reuse. European Software Engineering Conference, ESEC'93. Garmisch-Partenkirchen, Germany, p.418-33, 1993.

GALL, H., KLÖSCH R. Program Transformation to enhance the Reuse Potential of Procedural Software. ACM Symposium on Applied Computing, SAC'94Phoenix, Arizona,p. 99-104, 1994.

GALL, H., KLÖSCH R., MITTERMEIR R. Object-Oriented Re-Architecturing. Rockville, International Conference on Software Engineering and Knowledge Engineering. Maryland, USA, p. 334-341,1995.

GLASS, Robert L. The “Maintenance-first” Software Era. The Journal of Systems and Software. Bloomington, USA, p.171-174,2000.

IEEE Computer Society Techinal Council on Software Engineering. Disponível em http://www.tcse.org/revengr/taxonomy.html. Acesso em 13/03/2003

JABUR INFORMÁTICA Página sobre Gestão Empresarial – Sistema Siglo®, Londrina. Disponível em http:///www.jaburinfo.com.br/gestao.asp. Acesso em 16/10/2003.

JACOBSON, I., ERICSON, M., JACOBSON, A. The Object Advantage - Business Process Reengineering with Object Technology. 1ed. Addison-Wesley, 1994.

JACOBSON, I.; LINDSTRÖM F. Re-engineering of Old Systems to an Object Oriented Architecture. Conference proceedings on Object-oriented programming systems, languages, and applications. Phoenix, Arizona, p.340-350, 1991.

LEE, B. Y.; KIM W. A Knowledge-Based Maintenance of Legacy Systems: METASOFT. Expert Systems With Applications, Seoul, Korea, v.12, n.4, p.483-496, 1997.

LEE, H.; YOO C. A Form Driven Object-Oriented Reverse Engineering Methodology. Information Systems. Seoul, Korea, v.25, n.3, p.235-259, 2000.

MÜLLER, H. A.; et. al. Reverse enginner – A Roadmap. Proceedings of the 22th International Conference on Software Engineering (ICSE-00). Limerick, Ireland, p. 47-60, jun.2000.

PENTEADO, R. A.D. Um Método de Engenharia Reversa Orientado a Objetos. Tese (Doutorado em Ciências de Computação e Matemática Computacional) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 1996.

PENTEADO,R.; MASIERO,P.C., CAGNIN,M.I. An Experiment of Legacy Code Segmentation to Improve Maintainability. Proceedings of the 3rd European Conference on Software Maintenance Reengineering (CSMR’99) – IEEE. Amsterdan, p.111-119, 1999.

PRESSMAN, R.S. Engenharia de Software. São Paulo, Makron Books: 1995.

QUINAIA, M. A. SANCHES, R. , São Carlos, Relatório Técnico. Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 1999.

ROSENBERG, H. L.; HYATT E. L. Hybrid Re-Engineering. National Aeronautics and Space Administration. Disponível em http://satc.gsfc.nasa.gov/support/ISRE_JAN97/Rengart6.html. Acesso em 02/05/2003.

RUMBAUGH, J. et al. Object-Oriented Modeling and Design. Prentice Hall, 1991.

SCHNEIDEWIND, N. F. The State of Software Maintenance. 1987. IEEE Transaction on Software Engineering. Piscataway, NJ , v.13, n.3, p.303-310, 1987.

SILVA, P.P.; et. al. CAPPLES – A Capacity Planning and Performance Analysis Method for the Migration of Legacy System

SNEED, M. Harry Planning the Reengineering of Legacy Systems, Munich, Germany, 1995. IEEE Software, p.24-34.

SOMMERVILLE, I. Software Engineering, New York, USA, 1995. Ed. Addison Wesley, 5a.ed.

TILLEY, S. A Reverse-Engineering Environment Framework, Pittsburgh, PA, 1998. Technical Report, CMU/SEI-98-TR-005, ESC-TR-98-005, Software Engineering Institute, Reengineering Center, , disponível em: www.sei.cmu.edu/publications/documents/98.reports/98tr005/98tr005title.htm. Acesso em 24/06/2003.

VOSS, G. Object-Oriented Programming: An Introduction. 1991, Osborne Publishing, 584 p.

WINBLAD, A. L., EDW ARDS, S. D., KING, D. R. Software Orientado ao Objeto. São Paulo, 1993. Makron Books.

WIRFS-BROCK, R., et al. Desingning object-oriented software. 1990. Englewood Cliffs. Prentice-Hall, 341p.

WU, B., et al. Legacy Systems Migration – A Method and its Tool-kit Framework Hong Kong, 1997. Proceedings of the APSEC'97/ICSC'97, p. 312-320.