Padrões de Reengenharia Auxiliados por Diretrizes de Qualidade de Software

Gizelle Sandrini Lemos
Depto. de Computação
Univ Federal de São Carlos
gizelle@dc.ufscar.br

Edson Luiz Recchia
UNICEP / Univ.
Anhembi-Morumbi
erecchia@terra.com.br

Rosângela Penteado
Depto. de Computação
UnivFederal de São Carlos
rosangel@dc.ufscar.br

Rosana T. V. Braga
ICMC / USP
Univ. de São Paulo
rtvb@icmc.usp.br

Resumo

Atualmente, a definição de padrões de reengenharia tem sido abordada por diversos autores pela necessidade da existência de diretrizes mais consistentes para guiar na realização desse processo. Aliada a isso, a preocupação com o resultado do processo, ou seja, a qualidade do produto gerado, contribui para a descrição da reengenharia sob a forma de padrões com o objetivo de torná-la mais clara para o engenheiro de software. Este artigo tem por objetivo descrever, na forma de sete clusters de padrões, o Processo de Reengenharia Orientada a Objetos (PRE/OO), que detalha a reengenharia de sistemas legados procedimentais para sistema orientados a objetos.

Abstract

The reengineering patterns definition has been approached by several authors due to the need for more consistent guidelines in this process. In addition, the concern with the result, that is, the quality of the final product, contributes to the description of the reengineering process in the pattern format, in order to make it available to the software engineering community. The goal of this paper is to describe, in the form of seven pattern clusters, the Object-oriented Reengineering Process (PRE/OO), which details the reengineering of procedural legacy systems into object-oriented systems.

1. Introdução

A reengenharia é a forma que muitas organizações estão buscando para manter/refazer seus softwares, livrando-se das manutenções difíceis e da degeneração de suas estruturas. Por este motivo, é importante que o resultado desse processo seja confiável. Desta forma, a garantia da qualidade também pode ser considerada como mais uma etapa da reengenharia.

O objetivo deste artigo é apresentar e documentar padrões para realizar a reengenharia de sistemas legados implementados em Delphi [1] de forma procedimental para sistemas orientados a objetos presentes na linguagem Object Pascal, utilizada no ambiente Delphi. Desta forma, foram elaborados vinte padrões que se encontram organizados em clusters relacionados às etapas de engenharia reversa e engenharia avante e, ainda, à qualidade de processo. Apesar dos padrões serem específicos para sistemas legados implementados em Delphi de forma procedimental, os mesmos podem ser utilizados, após algumas adaptações, em sistemas implementados em outras linguagens procedimentais.

Este trabalho está organizado da seguinte forma: na Seção 2 são citados os trabalhos que contribuíram para a elaboração dos padrões propostos. Já na Seção 3 os clusters e os

padrões que compõem o processo de reengenharia são apresentados, enquanto que na Seção 4 são apresentadas as considerações finais.

2. Trabalhos relacionados aos padrões do PRE/OO

A partir do estudo de diretrizes, métodos, linguagens e famílias de padrões existentes para conduzir a reengenharia de sistemas procedimentais para sistemas orientados a objetos, e da verificação da necessidade de que a reengenharia seja conduzida levando em conta o aspecto qualidade, foram compostos os padrões do PRE/OO, influenciados por seguintes trabalhos (Figura 1):

- Fusion/RE [8, 9]: o método aplicado de forma seqüencial foi utilizado em diversos estudos de caso para a realização da segmentação (reengenharia orientada a objetos sem mudança de linguagem) em sistemas legados procedimentais. No PRE/OO foram adaptados os seis passos que o compõe:
  - Revitalização da arquitetura do software legado;
  - Recuperação do Modelo de Análise do Sistema Atual (MASA);
  - Criação do Modelo de Análise do Sistema (MAS);
  - Mapeamento do MAS para o MASA;
  - Elaboração do Projeto Avante;
  - Segmentação do Sistema.
- UML (*Unified Modeling Language*) [6]: os diagramas UML foram utilizados para documentação dos produtos elaborados durante o processo;
- Processo Evolutivo [10]: este modelo de processo foi utilizado por permitir ao engenheiro de software a repetição de passos anteriormente realizados do processo como forma de refinar os produtos gerados;
- FaPRE/OO [11, 12]: padrões da Família de Padrões de Reengenharia foram adaptados para uso na reengenharia orientada a objetos de sistemas legados implementados em Delphi;
- SW-CMM (*Capability Maturity Model for Software*) [7]: as KPAs (key process areas) do Nível 2 de maturidade serviram como base para qualificação do processo de reengenharia. Propôs-se a realização de planejamento e acompanhamento para prover o mínimo de visibilidade com relação às necessidades do processo de reengenharia orientada a objetos de sistemas procedimentais. Essas práticas consideradas essenciais no processo de desenvolvimento de software não estavam explicitamente incluídas em processos de reengenharia, sendo uma das contribuições deste trabalho.

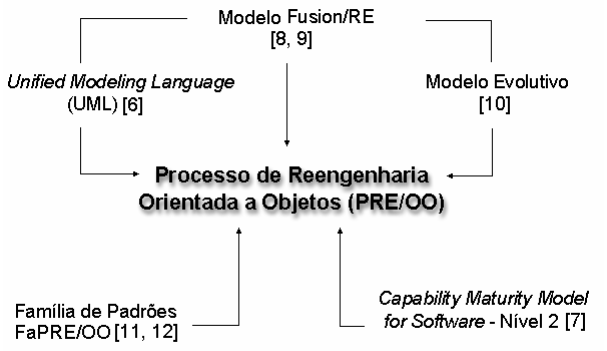

Figura 1. Trabalhos que originaram os padrões do PRE/OO [5]

3. PRE/OO - Processo de Reengenharia Orientada a Objetos

O Processo de Reengenharia Orientada a Objetos foi composto sob a forma de cinco clusters de padrões divididos como descrito a seguir e ilustrados pela Figura 2:

- Clusters 1 e 2: implementam a melhoria da qualidade no processo de reengenharia baseados nas KPAs do Nível 2 de maturidade do SW-CMM. O cluster 1 trata da preparação e do planejamento, sendo realizado antes do inicio do processo de reengenharia. O cluster 2 trata do acompanhamento da aplicação dos clusters 3 a 7;
- Clusters 3 a 5: os clusters do PRE/OO relacionados à engenharia reversa foram elaborados com base nos passos 1 a 3 do Fusion/RE, respectivamente. Porém, cada um desses passos pode ser realizado de forma evolutiva, como indicam as setas em torno das elipses na Figura 2 diferentemente do Fusion/RE, que é seqüencial linear. O conteúdo de alguns padrões foi influenciado, também, pela FaPRE/OO [11, 12];
- Clusters 6 e 7: correspondem à engenharia avante, elaborados com base na extensão do Fusion/RE [9] e no modelo de processo evolutivo [10].

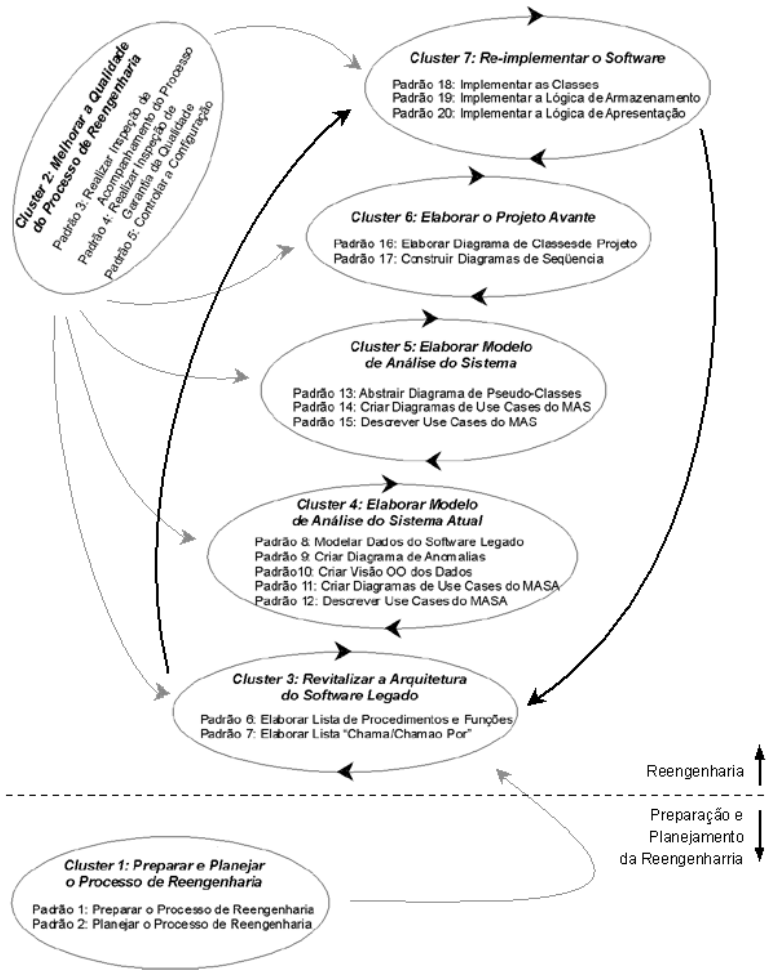

Figura 2. Estrutura de Clusters de Padrões do PRE/OO [5]

Cada padrão do PRE/OO é apresentado no seguinte formato: **Número, Nome, Intuito, Problema, Contexto, Solução, Exemplo, Usos Conhecidos, Padrões Relacionados** e **Produtos Obtidos**, formato derivado de [2, 3, 4].

O item **Exemplo** não foi descrito pelo fato de ocupar muito espaço no artigo, tornando-o excessivamente longo. A descrição desse item e a aplicação completa num estudo de caso dos padrões do PRE/OO podem ser encontradas on-line em [5]. O item **Usos Conhecidos** não consta da estrutura de cada padrão por conter informações comuns a todos, sendo, portanto, descrito a seguir.

Usos Conhecidos: utilização em um sistema legado que realizava o controle de vendas e estoque implementado em Delphi, porém sem características orientadas a objetos [5]. Os padrões foram utilizados ainda em um trabalho de iniciação científica que consistia na reengenharia de um sistema de vídeo-locadora[1] e em trabalho de reengenharia de um sistema de biblioteca ministrado na disciplina de engenharia de software do curso de graduação em ciência da computação da UFSCar.

CLUSTER 1 - Preparar e Planejar o Processo de Reengenharia.

1. Nome: Preparar o Processo de Reengenharia

Intuito: obter quais atividades serão realizadas durante o processo de reengenharia e quais produtos serão elaborados.

Problema: definir as atividades e os produtos de trabalho resultantes da aplicação do PRE/OO.

Contexto: a lista com os produtos de trabalho que serão elaborados pode ser obtida analisando-se os recursos disponíveis. Já as atividades que devem ser seguidas para a obtenção desses produtos dependem diretamente do que precisa ser realizado e da análise do processo descrito nos padrões seguintes (PRE/OO).

Solução: elaborar o documento Levantamento de Atividades, contendo as informações:

a) Projeto: contém o nome do software legado que será submetido à reengenharia;
b) Versão do Documento: consta do número adotado para diferenciar as versões do documento durante a evolução do processo de reengenharia e documentadas no controle de configuração, caso realizadas.
c) Data da Criação do Documento: data da criação da versão citada do documento. Como esse e outros campos serão utilizados em todos os modelos propostos, deve-se adotar o seguinte formato dd/mm/aaaa;
d) Responsável pela Criação do Documento: nome da pessoa responsável pela criação da versão correspondente ao documento;
e) Exame da Situação: deve conter a contextualização do software em relação ao domínio em que está inserido, contendo a descrição detalhada de suas funcionalidades para uso posterior como fonte de auxílio ao seu entendimento;
f) Itens Disponíveis: contém as fontes de informações disponíveis para utilização durante o processo de reengenharia. Em geral, consiste de: código-fonte, arquivos de dados e programa executável;

[1].Relatório de IC- PIBIC/CNPq, abril 2003 - UFSCar. Alunos: Raquel Murta e Hallen Fontana.

g) Produtos de Trabalho a Serem Elaborados: composto de produtos que contém resultados simples ou compostos desenvolvidos ao longo da reengenharia. O conteúdo desse campo varia conforme os Itens Disponíveis;
h) Atividades a Serem Realizadas: descrição de quais padrões são necessários à composição dos produtos de trabalho acima citados. Dependendo dos produtos já disponíveis, alguns padrões podem ser omitidos.

Produto Obtido: documento Levantamento de Atividades.

Padrões Relacionados: a aplicação desse padrão resulta na entrada para o padrão 2 (Planejar o Processo de Reengenharia).

2. Nome: Planejar o Processo de Reengenharia.

Intuito: planejar as atividades de reengenharia definidas com a aplicação do padrão Preparar Processo de Reengenharia.

Problema: estimar os tempos, recursos e itens de configuração relacionados ao projeto de reengenharia do software legado.

Contexto: esse padrão requer o planejamento das atividades, a partir do Documento Levantamento de Atividades obtido com a aplicação do padrão 1 (Preparar Processo de Reengenharia). Podem ser considerados os seguintes aspectos: é difícil estimar tempos para a realização do processo de reengenharia e dos passos nos quais esse se subdivide. Essa dificuldade pode ser minimizada com a adoção de métricas de tamanho e/ou com o auxílio da experiência do engenheiro de software.

Solução: elaborar o documento Plano para Realização da Reengenharia contendo as seguintes informações, referentes à aplicação de cada um dos clusters 3 a 7:

a) Cabeçalho: contém Projeto, Versão do Documento, Data da Criação do Documento e Responsável pela Criação do Documento têm as mesmas informações apresentadas no modelo Levantamento de Atividades, descrito no padrão anterior;
b) Tempo Disponível para o Processo: tempo disponível para o processo de reengenharia, expresso em dias e horas;
c) Datas Estimadas para Inicio e Finalização do Processo de Reengenharia: essas datas são obtidas através do somatório dos tempos necessários para realização de cada passo do processo de reengenharia, estimados pelo engenheiro de software;
d) Itens de Configuração: produtos de trabalho desenvolvidos durante o processo de reengenharia, para os quais é importante que seja realizado o controle de alterações;
e) Inspeções de Acompanhamento e Supervisão do Projeto: lista quantas, quando e por quem serão realizadas as inspeções de acompanhamento de projeto. Sugere-se a realização de uma inspeção de acompanhamento de projeto ao final de cada passo do processo de reengenharia;
f) Inspeções de Garantia da Qualidade: lista quantas serão, no total, as inspeções realizadas nos produtos de trabalho;
g) Produtos de Trabalho: para cada passo do processo de reengenharia, devem ser especificados quais os produtos de trabalho devem elaborados, por quem, a estimativa de tempo e qual a ferramenta de auxílio será usada para sua elaboração, além de informações sobre a necessidade de controle de configuração;
h) Tempo Necessário para a Conclusão do Passo: é o tempo obtido a partir do somatório das estimativas de tempo para a elaboração de cada um dos produtos de trabalho produzidos nesse determinado passo.

Esse documento deve ser preenchido considerando-se cada um dos passos do processo de reengenharia (clusters 3 a 7). Assim, esse plano estará completo somente quando todos os passos forem estimados.

Produto Obtido: Plano Para Realização da Reengenharia.

Padrões Relacionados: esse padrão deve ser aplicado, necessariamente, após o padrão 1, (Preparar o Processo Reengenharia), por sua saída ser utilizada aqui como entrada. Após o planejamento do processo, deve ser iniciado o processo de reengenharia efetivamente. Relaciona-se com o padrão 3, (Acompanhar o Progresso do Processo de Reengenharia - cluster 2).

CLUSTER 2 – Melhorar a Qualidade do Processo de Reengenharia

3. Nome: Acompanhar o Progresso do Processo de Reengenharia

Intuito: manter sempre atualizado o plano do processo de reengenharia.

Problema: contornar os possíveis desvios que o projeto de reengenharia do software legado possa vir a sofrer a partir das estimativas realizadas durante o seu planejamento.

Contexto: a inspeção de acompanhamento do processo é a verificação do progresso do projeto de reengenharia de um software legado com relação ao que foi planejado. Caso, durante a inspeção de acompanhamento do processo, conclua-se que o desenvolvimento do processo esteja defasado em relação ao que foi planejado, ações corretivas devem ser tomadas. Tais ações incluem a correção do Plano para Realização da Reengenharia, de modo que esse reflita a execução real, ou o replanejamento dos passos restantes. Esse padrão é aplicado com base no Plano para Realização da Reengenharia elaborado no padrão Planejar o Processo de Reengenharia e no andamento real do processo.

Pode-se encontrar dificuldades em definir o que deve ser considerado como desvio no Plano, visto o aspecto subjetivo a ser definido pelo engenheiro de software e, ainda, qual atitude deve ser tomada em caso de verificação de um desvio.

Solução: 3.1) Elaborar um documento denominado "Inspeção de Acompanhamento do Processo" contendo as informações:

a) Cabeçalho preenchido como nos documentos elaborados nos padrões anteriores;
b) Responsável pela Inspeção de Acompanhamento: nome da pessoa ou dos integrantes da equipe que realizou a inspeção de acompanhamento e supervisão de projeto;
c) Passo do PRE/OO: descreve em qual passo do processo de reengenharia que foi realizada a revisão de acompanhamento de projeto, ou seja, durante a aplicação de qual cluster de padrões. Por exemplo: Revitalização da Arquitetura do Software Legado, MASA, Projeto Avante, etc.
d) Número de Inspeção: contém o número seqüencial para identificação das inspeções de acompanhamento de projeto;
e) Itens Analisados: descreve os aspectos do Plano para Realização da Reengenharia que foram submetidos à inspeção. Podem ser divididos em sub-seções em que são analisados diferentes aspectos relacionados ao processo;
f) Resultados Obtidos: deve constar a comparação entre as estimativas e os resultados reais alcançados em termos de tempo gasto e produtos de trabalho elaborados;
g) Ações Corretivas Necessárias: descreve quais as ações corretivas foram utilizadas para correção ou adaptação do Plano para Realização da Reengenharia, no caso de desvio em relação ao planejado.

3.2) Para cada um dos passos do processo de reengenharia (clusters 3 a 7) deve-se realizar e documentar a inspeção de acompanhamento do progresso do projeto, bem como os ajuste feitos no Plano para Realização da Reengenharia, caso necessários.
3.3) Caso seja necessária a criação de uma nova versão do Plano para Realização da Reengenharia, deve-se realizar o controle de configuração.

Produto Obtido: documento de Inspeção do Acompanhamento do Processo.

Padrões Relacionados: esse padrão deve ser aplicado durante processo de reengenharia, preferencialmente após o fim de cada um dos passos (clusters 3 a 7) do PRE/OO.

4. Nome: Realizar Inspeção de Garantia da Qualidade

Intuito: garantir a qualidade dos produtos gerados durante o processo de reengenharia.

Problema: descobrir os erros em produtos de trabalho em relação ao que foi planejado, às especificações e/ou padrões propostos.

Contexto: cada produto de trabalho gerado ao longo do processo de reengenharia deve ser alvo da inspeção proposta com esse padrão. O engenheiro de software pode decidir sobre a realização das inspeções apenas em produtos críticos, de acordo com o porte do software submetido ao processo de reengenharia orientada a objetos.

Solução: elaborar um documento contendo as seguintes informações:

a) Cabeçalho preenchido como nos documentos elaborados nos padrões anteriores;
b) Item Alvo da Inspeção de Garantia da Qualidade: contém o nome e a versão do produto de trabalho inspecionado;
c) Responsável pela Inspeção de Garantia da Qualidade: contém o nome da pessoa ou dos integrantes da equipe responsável pela realização da inspeção;
d) Número de Inspeção: contém um número seqüencial para identificação de cada inspeção de garantia da qualidade;
e) Aspectos Analisados: descreve todos os aspectos analisados no produto de trabalho, a forma de análise e a situação em relação ao desejado;
f) Diferenças Encontradas: relata as diferenças entre os aspectos analisados e os resultados esperados;
g) Ações Corretivas Necessárias: descreve a proposta para corrigir as diferenças encontradas.

Esse padrão é aplicado de forma a garantir a qualidade dos produtos de trabalho gerados durante o processo de reengenharia, verificando se tais produtos refletem o que foi solicitado e se seguem os padrões propostos.

Produto Obtido: documento de Inspeção de Garantia da Qualidade.

Padrões Relacionados: relaciona-se com os clusters 3 a 7 do PRE/OO, de forma a inspecionar cada saída produzida a partir da aplicação de seus padrões.

5. Nome: Controlar a Configuração

Intuito: estabelecer e manter a integridade dos produtos de trabalho elaborados ao longo do processo de reengenharia.

Problema: controlar as alterações realizadas nos vários produtos de trabalho de forma a não permitir a ocorrência de inconsistências.

Contexto: a condução do processo de reengenharia de forma evolutiva torna indispensável a implementação desse padrão, como forma de controlar mudanças e versões dos produtos de trabalho construídos, fator que torna-se ainda mais crítico no caso do processo ser conduzido por mais de uma pessoa. Podem ser considerados os seguintes aspectos:

- A aplicação do padrão pode ser dificultada por não ser realizada de forma automatizada;
- O uso de ferramentas próprias para o Controle de Configuração e a criação de Baselines pode ser uma opção a aplicação desse padrão.

Solução: 5.1) Definir os itens de configuração, ou seja, os produtos de trabalho que terão suas versões controladas;

5.2) Documentar todas as alterações realizadas nos itens de configuração, descrevendo sua origem, quando e por quem foi feita a alteração (no caso de mais de uma pessoa estar associada ao processo) e em que versão do item resultou, com as informações:

a) Cabeçalho preenchido como nos documentos elaborados nos padrões anteriores;
b) Responsável pelo Controle de Configuração e Passo do PRE/OO;
c) Data de Criação: relata a data (dd/mm/aaaa) de criação da versão do item de configuração;
d) Nome do Documento: refere-se ao nome dado ao produto de trabalho que é alvo do controle de configuração;
e) Versão: contém a versão do item de configuração criada após a realização das alterações;
f) Origem: especifica a partir de qual processo ou inspeção foi originada a versão desse item de configuração.

5.3) Estabelecer uma baseline ao final de cada passo do processo de reengenharia para que se possa ter um maior controle do projeto em termos de gerenciamento de configuração, ou seja, um conjunto de produtos de trabalho inspecionados que servem de base para a continuação do projeto e que só podem ser alterados mediante controle documentado. A documentação para o gerenciamento das baselines deve conter os seguintes itens:

a) Cabeçalho: preenchido como nos documentos elaborados nos padrões anteriores;
b) Responsável pelo Controle de Configuração e Passo do PRE/OO;
c) Data de Criação da Baseline: relata a data (dd/mm/aaaa) em que os itens de configuração elaborados ao longo do passo do PRE/OO foram reunidos e formaram a baseline;
d) Descrição: contém a explicação sobre o conteúdo da baseline criada;
e) Itens de Configuração: lista o nome de cada item de configuração que compõe a baseline;
f) Versão: lista a versão do item de configuração que compõe a baseline;
g) Data de Criação: relata a data da criação da versão descrita do item de configuração.

Produtos Obtidos: Lista de Controle de Configuração e Documento com as Baselines do Projeto.

Padrões Relacionados: relaciona-se com todos os padrões dos clusters 3 a 7, de forma a controlar as alterações em todos os itens de configuração gerados a partir de sua aplicação.

CLUSTER 3 – Revitalizar a Arquitetura do Software Legado

6. Nome: Elaborar Lista de Procedimentos e Funções

Intuito: identificar todos os procedimentos e funções, definidos pelo programador, que serão transformados pelo processo de reengenharia.

Problema: destacar, a partir do código-fonte, somente os procedimentos e funções definidos pelo programador, ignorando os procedimentos e funções declaradas em bibliotecas ou

APIs (Application Programmable Interfaces) fornecidas pelo fabricante do ambiente de desenvolvimento Delphi.

Contexto: consideram-se apenas os procedimentos e as funções definidos pelo programador para realizar a engenharia reversa. A base para a análise dos procedimentos e funções é o código-fonte todas as units do software legado;

No Delphi, grande parte do código-fonte é executado direta ou indiretamente em resposta a eventos. Um evento é um tipo especial de propriedade que representa uma ocorrência em tempo de execução, geralmente uma ação do usuário (como o click do mouse sobre um botão). Podem ser considerados os seguintes aspectos:

- Podem-se localizar os eventos clicando-se sobre a aba Events do Object Inspector.
- A ilegibilidade dos nomes utilizados para descrever os procedimentos e funções pode dificultar o processo;
- A aplicação do padrão é facilitada pelo fato dos protótipos dos procedimentos e funções serem descritos na seção Interface das units (arquivos-fonte do Delphi com extensão ".PAS").

Solução: 6.1) Identificar o protótipo (cabeçalho) de todos os procedimentos e funções presentes no código das units do software legado;

6.2) Classificar cada procedimento ou função de acordo com os critérios a seguir:

- Ev – Evento: procedimentos originados em resposta à eventos disparados pelo sistema. Encontram-se declarados acima da palavra reservada public na interface da unit,
- Pr – Private: procedimentos ou funções visíveis apenas internamente a unit em que estão declarados. Encontram-se declarados abaixo da palavra reservada private na interface da unit,
- Pb – Public: procedimentos ou funções visíveis a outras units, além daquela em que se encontram declarados. Encontram-se declarados abaixo da palavra reservada public na interface da unit.

6.3) Documentar os passos acima, de acordo com os itens:

a) Cabeçalho: como nos documentos elaborados nos padrões anteriores;
b) Módulos do Software: deve conter todos os arquivos com extensão ".pas" que compõem o software legado;
c) Nomes dos Procedimentos e Funções: lista todos os procedimentos e funções que são extraídos a partir do código-fonte;
d) Classificação: lista a visibilidade do procedimento ou função em relação ao módulo em que está declarado e ao software. Utiliza a nomenclatura Ev, Pb ou Pv conforme descrito no Passo 2 da solução.

Obs.: Os três últimos campos acima descritos devem ser repetidos até que se esgotem todos os procedimentos e funções que compõem um módulo e todos os módulos que compõem o software legado.

Produto Obtido: Lista de Procedimentos e Funções.

Padrões Relacionados: esse padrão inicia o processo de reengenharia, sendo aplicado após a aplicação do cluster 1. Após a aplicação deste padrão, deve-se:

- realizar a inspeção de garantia da qualidade da Lista de Procedimentos e Funções: utilizar o padrão 4 (Realizar Inspeção de Garantia da Qualidade);
- realizar o controle de configuração: padrão 5 (Controlar a Configuração);
- continuar o processo de engenharia reversa: padrão 7 (Elaborar Lista "Chama/Chamado Por").

7. Nome: Elaborar Lista de "Chama/Chamado Por"

Intuito: obter entendimento da funcionalidade implementada em cada procedimento e função do software legado.

Problema: extrair, de cada procedimento e função presente na Lista de Procedimentos e Funções, a funcionalidade, os procedimentos e as funções que são por ele chamados e por quais outros procedimentos e funções ele é chamado.

Contexto: o código-fonte do software legado contém muitas regras de negócio e detalhes de implementação importantes para o entendimento da funcionalidade que podem passar despercebidos pelo engenheiro de software que está conduzindo a engenharia reversa.

- Um ponto negativo que o engenheiro de software pode encontrar para a aplicação desse padrão é o tempo que este consome;
- Por se tratar do estudo e documentação do código, que é um trabalho minucioso, o engenheiro de software pode optar por não aplicá-lo ou aplicá-lo apenas nas units mais importantes do software legado;
- Após a aplicação desse padrão, o entendimento acerca do software aumenta consideravelmente.

Solução: preencher a Lista "Chama/Chamado Por" para cada procedimento e função que conste da Lista de Procedimentos e Funções, com as seguintes seções:

a) Cabeçalho: como nos documentos elaborados nos padrões anteriores;
b) Módulo do Software: contém o arquivo com extensão ".PAS", o qual teve seu código-fonte utilizado na composição da lista. Elabora-se uma Lista "Chama/Chamado Por" para os arquivos com extensão ".PAS" que compõe o software;
c) Procedimento/Função: contém o nome do procedimento ou função com descrição sucinta de sua funcionalidade, que é obtida a partir do entendimento do código implementado no corpo do procedimento ou função, encontrado na seção implementation da unit em que esse se encontra declarado;
d) Chama: lista os procedimentos chamados por outros procedimentos e funções da Lista de Procedimentos e Funções, existentes no código do procedimento ou função analisado;
e) Chamado Por: lista os procedimentos ou funções que são chamados pelo procedimento listado no item Chama. De acordo com a classificação, podem ocorrer três situações:
    - Caso o procedimento ou função apresente a classificação Pb (Public), procedimentos e funções de outras units podem chamá-lo (considerar apenas aqueles que constem da Lista de Procedimentos e Funções), sendo a coluna "Chamada Por" completada somente quando todos os procedimentos ou funções foram classificados, ou seja, quando o processo de revitalização das demais units for concluído;
    - Caso o procedimento ou função apresente a classificação Pv (Private), apenas procedimentos e funções da unit em que esse se encontra podem chamá-lo (considerar apenas aquelas que constam da Lista de Procedimentos e Funções), portanto a coluna "Chamado Por" será completada ao término da revitalização da unit em que esse se encontra;
    - Caso o procedimento tenha a classificação Ev, a coluna "Chamado Por" será vazia, pelo fato de se tratar de um evento, o qual somente é disparado por algum agente externo ao software e nunca por outros procedimentos/funções.

Produto Obtido: Lista "Chama/Chamado Por".

Padrões Relacionados: esse padrão somente pode ser aplicado após ter sido aplicado o padrão 6 (Elaborar Lista de Procedimentos e Funções), pois a saída dele é utilizada como entrada para este padrão. Após a aplicação deste padrão, deve-se:

- realizar a inspeção de garantia da qualidade da Lista "Chama/Chamado Por": padrão 4 (Realizar Inspeção de Garantia da Qualidade);
- atualizar o controle de configuração: padrão 5 (Controlar a Configuração);
- inspecionar o andamento do processo em relação ao Plano para Realização da Reengenharia: padrão 3 (Acompanhar o Progresso do Processo de Reengenharia);
- continuar o processo de engenharia reversa: padrão 8 (Modelar Dados do Software Legado).

CLUSTER 4 – Elaborar Modelo de Análise do Sistema Atual

8. Nome: Modelar Dados do Software Legado

Intuito: construir Modelo Entidade Relacionamento (MER) correspondente à implementação atual dos dados do software legado a partir dos dados contidos no banco de dados relacional.

Problema: interpretar e abstrair os dados das tabelas software legado: nome de cada tabela, nomes dos campos de dados e tipo de cada campo. Essas informações geralmente encontram-se documentados sob a forma de scripts SQL (Structured Query Language) presentes nos arquivos de dados do software legado.

Contexto: caso os dados estejam documentados sob a forma de scripts SQL, é necessário que o engenheiro de software possua conhecimentos básicos sobre SQL;

Os dados podem, ainda, estar presentes num documento ou serem obtidas analisando-se a base de dados;

Como o banco de dados relacional não consegue representar o relacionamento entre as tabelas, é necessário analisar seus campos para obter tais informações;

As cardinalidades entre as entidades do MER nem sempre são triviais e, muitas vezes, só são obtidas a partir de investigação das regras de negócios embutidas no código ou percebidas através da execução do software legado.

Solução: 8.1) Montar, a partir dos arquivos de dados do software legado escritos sob a forma de scripts SQL, um documento com as seguintes seções:

a) Cabeçalho: preenchido como nos padrões anteriores;
b) Tabelas de Dados: contém as informações obtidas com a realização do passo 2 da Solução;
c) Campos de Dados: contém as informações obtidas realizando-se o passo 3 da Solução;
d) Chave Primária: contém o(s) campo(s) que identifica(m) unicamente cada registro da Tabela de Dados, conforme instruções do passo 4 da Solução;
e) Chaves Estrangeiras: campo(s) da Tabela de Dados que compõe(m) a Chave Primária de outra(s) Tabela(s) de Dados, o que pode ser verificado através da análise do campo Chave Primária da Lista de Tabelas e Chaves;

8.2) Preencher a coluna Tabela de Dados a partir da identificação do nome de cada tabela de dados associado às cláusulas CREATE TABLE dos scripts;

8.3) Identificar, abaixo de cada cláusula CREATE TABLE os nomes e tipos dos campos relacionados às tabelas de dados identificadas no passo anterior. Esses campos deverão ser inseridos na coluna Campos de Dados;

8.4) Identificar as cláusulas PRIMARY KEY nos scripts SQL dos arquivos de dados. Cada cláusula desse tipo corresponde à chave primária da tabela de dados identificada no Passo 2 da solução. Cada chave primária identificada deve ter seu nome inserido na coluna Chave Primária;

8.5) Identificar as chaves estrangeiras, ou seja, as Chaves Primárias de Tabelas de Dados declaradas como campos em outras tabelas, e inseri-las na coluna Chaves Estrangeiras.

8.6) Construir o MER. A partir das informações da Lista de Tabelas e Chaves. Cada Tabela de Dados dá origem a uma entidade do MER, cada Chave Estrangeira identifica os relacionamentos entre as diversas Tabelas de Dados representadas. As cardinalidades entre as entidades devem ser obtidas através das regras de negócios, extraídas do código-fonte, com a execução do software legado e/ou de entrevistas com usuários.

Produtos Obtidos: Lista de Tabelas e Chaves, Modelo Entidade Relacionamento.

Padrões Relacionados: após a aplicação do padrão, o engenheiro de software deve:

- realizar a inspeção de garantia da qualidade do MER: padrão 4 (Realizar Inspeção de Garantia da Qualidade);
- atualizar o controle de configuração: padrão 5 (Controlar a Configuração);
- prosseguir com a engenharia reversa: padrão 9 (Criar Lista de Anomalias).

9. Nome: Criar Lista de Anomalias

Intuito: obter, a partir do código-fonte, todas as anomalias relacionadas aos procedimentos que acessam as tabelas de dados identificadas.

Problema: registrar e classificar os procedimentos e funções que observam e/ou consultam mais de uma entidade.

Contexto: para a aplicação do padrão, todo o código-fonte deve ser percorrido;
Permite a posterior divisão dos procedimentos e funções de forma a eliminar as anomalias e permitir a migração para o paradigma orientado a objetos.

Solução: criar uma lista a partir dos procedimentos e funções da Lista de Procedimentos e Funções e a classificação de cada um quanto as anomalias em relação às tabelas de dados, obedecendo aos critérios apresentados Tabela 1.

Tabela 1. Critérios para classificações das anomalias em procedimentos e funções

Símbolo	Significado	Critério
o	Observador	Procedimento ou função que observa a tabela de dados
c	Construtor	Procedimento ou função que altera a tabela de dados
i	de Implementação	Procedimento ou função relacionado(a) à implementação
+	Mais de uma tabela de dados observada ou alterada	Associado aos símbolos (o) ou (c), indica que o procedimento/função observa e/ou altera 2 ou mais tabelas.

A Lista de Anomalias deve conter as seguintes seções:

a) Cabeçalho: preenchido conforme padrões anteriores;
b) Módulos do Software: contém cada arquivo com extensão ".pas" que compõe o software legado;
c) Procedimento ou Função: deve conter todos os procedimentos e as funções que compõem a Lista de Procedimentos e Funções;
d) Tabelas de Dados: descreve os nomes das tabelas de dados observadas e/ou alteradas dentro do procedimento ou função declarado na coluna anterior;

e) Critério de Acesso à(s) Tabela(s): classificação de acordo com o tipo de acesso que o procedimento ou função faz a cada Tabela de Dados;
f) Classificação da anomalia: caso o procedimento/função seja anômalo, ou seja, altere e/ou observe mais de uma Tabela de Dados simultaneamente, o tipo final de sua anomalia deve ser transcrito, obedecendo os critérios da Tabela 1.

Produto Obtido: Lista de Anomalias.

Padrões Relacionados: esse padrão somente pode ser aplicado após a aplicação dos padrões 6 e 8 (Elaborar Lista de Procedimentos e Funções e Modelar Dados do Software Legado), pelo fato das saídas destes serem utilizadas aqui como entradas. De forma a prosseguir a reengenharia, o engenheiro de software deve:

- realizar a inspeção de garantia da qualidade da Lista de Anomalias: padrão 4 (Realizar Inspeção de Garantia da Qualidade);
- atualizar o controle de configuração: padrão 5 (Controlar a Configuração);
- continuar o processo de engenharia reversa: padrão 10 (Criar Visão OO dos Dados).

10. Nome: Criar Visão OO dos Dados

Intuito: obter a visão orientada a objetos dos dados a partir do sistema legado procedimental.

Problema: transformar o modelo de dados construído de forma procedimental em uma visão orientada a objetos.

Contexto: pode-se utilizar ferramentas, como o Rational Rose, para modelar o diagrama elaborado com a aplicação desse padrão. Além disso, o engenheiro de software tem conhecimento dos conceitos da UML, para gerar modelos orientados a objetos a partir do MER.

Solução: 10.1) Considerar cada entidade do MER como uma pseudo-classe (estrutura de dados que pode vir a representar uma classe na implementação orientada a objetos do sistema);

10.2) Transportar os Campos de Dados descritos na Lista de Tabelas e Chaves para o Diagrama de Pseudo-Classes, que está sendo gerado, como atributos das pseudo-classes consideradas;

10.3) Analisar a Lista de Anomalias e, cada procedimento e função que tenha sido classificado como oc, c+, o+, o+c+, oc+ ou o+c, deve ser considerado como método, no Diagrama de Pseudo-Classes, de todas as pseudo-classes com as quais se relaciona (as quais modifica e/ou consulta);

10.4) Verificar as cardinalidades entre as entidades do Modelo Entidade-Relacionamento:

a) Para relacionamentos binários (entre duas entidades) e com cardinalidade igual a 0..1, 1..1, 0..N ou 1..N, deve-se transpor a cardinalidade para o Diagrama de Pseudo-Classes da mesma forma que esta se encontra no MER. Deve-se verificar também, o tipo do relacionamento, de forma a representá-lo como associação ou agregação (todo-parte), de acordo com a funcionalidade que representa. Os relacionamentos de agregação podem ser percebidos nas situações em que uma pseudo-classe faz o papel, dentro do contexto do software, de parte de outra pseudo-classe. Os relacionamentos que não forem dessa forma, devem ser mapeados como associação;
b) Para relacionamentos ternários (entre três entidades) e com cardinalidade N..N, deve-se verificar, no MER, se o resultado desse relacionamento pode ser representado como um link-atributo das pseudo-classes envolvidas.

Produto Obtido: Diagrama de Pseudo-Classes.

Padrões Relacionados: esse padrão somente pode ser aplicado após a aplicação dos padrões 8 e 9 (Modelar Dados do Software Legado e Elaborar Lista de Anomalias), pelo fato que as saídas deles são utilizadas como entradas para sua aplicação. Continuando o processo de engenharia reversa, o engenheiro de software deve:

- realizar a inspeção de garantia da qualidade no Diagrama de Pseudo-Classes: padrão 4 (Realizar Inspeção de Garantia da Qualidade);
- atualizar o controle de configuração: padrão 5 (Controlar a Configuração);
- continuar a engenharia reversa: padrão 11 (Criar Diagramas de Use Cases do MASA).

11. Nome: Criar Diagramas de Use Cases do MASA

Intuito: modelar as funcionalidades e o comportamento do software legado.

Problema: mapear o comportamento do software legado, de forma a completar o processo de engenharia reversa.

Solução: modelar os agentes externos que interagem com o sistema (atores) e os eventos (use cases) promovidos por eles:

11.1) Verificar, através de cada interface com o usuário, presente no software legado, quem dispara as interações com o ambiente externo. Essas interações podem ocorrer por meio de botões, menus, caixas de texto ou grades (grids). O responsável pelo disparo do evento deve ser considerado como ator do use case;

11.2) Verificar quais as interações ocorrem em cada interface. Cada interação origina um use case;

11.3) Nomear o use case:

a) Caso o evento que origina o use case seja tratado pelo ambiente Delphi através de componentes, não há código-fonte mapeado com o conteúdo do use case. Nesse caso, o nome do use case deve refletir sua funcionalidade;

b) Caso o evento que origina o use case seja tratado pelo programa, deve-se obter, através do código-fonte, o nome do procedimento que trata o evento descrito, o qual deve constar da Lista de Procedimentos e Funções. O nome do use case deve ser o nome do procedimento.

Produto Obtido: Diagramas de Use Cases do MASA.

Padrões Relacionados: esse padrão somente pode ser aplicado após a aplicação do padrão 6 (Elaborar Lista Procedimentos e Funções), pelo fato de sua saída ser utilizada aqui como entrada. Após a aplicação desse padrão, o engenheiro de software deve:

- realizar a inspeção de garantia da qualidade no Diagrama de Use Cases do MASA: padrão 4 (Realizar Inspeção de Garantia da Qualidade);
- atualizar o controle de configuração: padrão 5 (Controlar a Configuração);
- continuar a engenharia reversa: padrão 12 (Descrever os Use Cases do MASA).

12. Nome: Descrever Use Cases do MASA

Intuito: detalhar a documentação dos use cases do software legado.

Problema: documentar o curso normal e os cursos alternativos de execução dos use cases.

Contexto: a elaboração das descrições de cada use case pode variar em função de fatores específicos a cada software legado, como o conhecimento do engenheiro de software a respeito do domínio da aplicação, sua experiência em Delphi e o tempo disponível para a aplicação do padrão;

Apenas os use cases tratados em procedimentos implementados pelo programador devem ter suas descrições elaboradas. Os use cases tratados por componentes automáticos do Delphi devem ser desconsiderados.

Solução: o código-fonte das units (arquivos-fonte do Delphi com extensão ".PAS") e os Diagramas de Use Case formam a base para a elaboração das descrições dos use cases. Para cada use case deve ser obtida a descrição correspondente, seguindo a forma:

12.1) Estudar e documentar o código do procedimento responsável pela execução do use case, extraindo a seqüência das operações;

12.2) Verificar e documentar todas as ocorrências de cursos alternativos no código, com o objetivo de mapear todas as ramificações decorrentes do curso normal.

Produto Obtido: Descrições dos Use Cases do MASA.

Padrões Relacionados: esse padrão somente pode ser aplicado após a aplicação do padrão 11 (Criar Diagramas de Use Cases do MASA), pelo fato da saída deste ser utilizada aqui como entrada. Continuando o processo de reengenharia, o engenheiro de software deve:

- realizar a inspeção de garantia da qualidade nas Descrições dos Use Cases do MASA: padrão 4 (Realizar Inspeção de Garantia da Qualidade);
- atualizar o controle de configuração: padrão 5 (Controlar a Configuração);
- inspecionar o andamento do processo em relação ao Plano para Realização da Reengenharia: padrão 3 (Acompanhar o Progresso do Processo de Reengenharia);
- continuar a engenharia reversa: padrão 13 (Abstrair Diagrama de Pseudo-Classes).

CLUSTER 5 – Elaborar Modelo de Análise do Sistema.

13. Nome: Abstrair Diagrama de Pseudo-Classes

Intuito: criar a visão orientada a objetos do software legado.

Contexto: a análise dos relacionamentos modelados é, em muitos casos, subjetiva. Portanto, o conhecimento do engenheiro de software sobre orientação a objetos é indispensável;

No MAS, os procedimentos e funções anômalos do MASA são mapeados num número variável de métodos associados às classes modificadas e/ou consultadas, cabendo ao engenheiro de software a decisão sobre a modularização das funcionalidades implícitas a esses procedimentos e funções.

Solução: 13.1) Alterar os nomes das classes, caso necessário, para mnemônicos mais significativos;

13.2) Verificar a representatividade dos nomes de cada atributo das classes e sua utilidade no software legado;

13.3) Mapear as alterações realizadas nos passos 13.1 e 13.2 elaborando um documento com as seguintes seções:

a) Cabeçalho: preenchido como citado nos padrões anteriores;

b) Coluna MASA (Pseudo-Classes): contém a documentação dos itens presentes no código legado, antes da aplicação do padrão:
   - Nome: nome da pseudo-classe,
   - Atributos Modificados: nomes dos atributos antes de sua alteração no MAS,
   - Métodos sem Anomalias: nomes dos métodos classificados como (o) ou (c) antes de sua alteração no MAS;

c) Coluna MAS (Classes): contém os nomes das modificações e exclusões realizadas nas classes, atributos e métodos sem anomalias. Nessa coluna são preenchidos:
   - Nome: nome da classe,
   - Atributos Modificados: nomes dos atributos após sua alteração no MAS,

- Métodos sem Anomalias: nomes dos métodos classificados como (o) ou (c) após sua alteração no MAS,
- Novos Métodos: métodos criados no MAS, a partir da divisão da funcionalidade de métodos sem anomalias, para a melhoria da modularização e aumento do reuso do código.

13.4) Verificar os nomes dos procedimentos e funções sem anomalias, ou seja, classificados como (o) ou (c). Esses procedimentos e funções devem ter o nome verificado quanto à representatividade e, ainda, o código-fonte analisado quanto a possibilidade de serem "quebrados" em um ou mais métodos, de forma a facilitar o reuso;

13.5) Mapear as alterações realizadas nos métodos sem anomalias na lista parcialmente preenchida no passo 13.3;

13.6) Transformar os métodos anômalos em quantos métodos forem necessários de forma a dividi-los nas classes que observam/constróem. Por exemplo, um procedimento (oc) declarado nas classes A e B deve originar pelo menos dois métodos, um observador na classe A e um construtor na classe B. Outros métodos podem surgir, com o objetivo de reutilizar código e torná-lo mais estruturado, ou ainda, pode-se fazer uso dos métodos já existentes (criados a partir da aplicação do Passo 13.4), caso tenham a mesma funcionalidade;

13.7) Documentar as transformações realizadas nos métodos anômalos, conforme as seções:

a) Cabeçalho: preenchido como citado nos padrões anteriores;
b) Módulo do Software: lista a unit do software legado que contém o método anômalo;
c) Método Anômalo: contém o nome do método com anomalia;
d) Classif.: descreve a classificação de acordo com o tipo da anomalia do método: oc, c+, o+, o+c+, oc+ ou o+c;
e) Classes: contém o nome das classes a que serão associados os métodos após a eliminação das anomalias;
f) Métodos: nomes dos métodos resultantes da eliminação das anomalias

13.9) Identificar as funcionalidades do software legado tratadas por meio de funções e procedimentos de componentes Delphi de acesso direto às Tabelas de Dados;

13.10) Transformar essas funcionalidades em quantos métodos forem necessários para prover seu encapsulamento nas classes definidas. Esses novos métodos deverão constar apenas da coluna MAS da Lista elaborada a partir do passo 13.3;

13.11) Verificar e, caso necessário, substituir os tipos de relacionamentos entre as classes (associação, agregação e herança). Para isso, identifica-se a representatividade de cada relacionamento junto ao domínio da aplicação e com relação às regras da orientação a objetos nos relacionamentos existentes;

13.12) Alterar os nomes dos relacionamentos, caso necessário, com o objetivo de melhorar a representatividade dos mesmos.

Produtos Obtidos: Diagrama de Classes, Lista de Mapeamento MASA x MAS, Lista de Correspondência de Métodos Anômalos.

Padrões Relacionados: esse padrão somente pode ser aplicado após a aplicação do padrão 10 (Criar Visão OO dos Dados), pelo fato de sua saída ser utilizada aqui como entrada. Continuando o processo de engenharia reversa, o engenheiro de software deve:

- realizar a inspeção de garantia da qualidade no Diagrama de Classes: padrão 4 (Realizar Inspeção de Garantia da Qualidade);

- atualizar o controle de configuração: padrão 5 (Controlar a Configuração);
- continuar a engenharia reversa: padrão 14 (Criar Diagramas de Use Cases do MAS).

14. Nome: Criar Diagramas de Use Cases do MAS

Intuito: completar visão orientada a objetos do software legado.

Problema: refletir, no Diagrama de Use Cases do MASA, as modificações feitas no Diagrama de Classes, com a eliminação das anomalias, a reestruturação dos métodos existentes e a criação de novos métodos.

Contexto: o engenheiro de software deve analisar cada use case para definir a seqüência da realização das operações sem alterar a funcionalidade representada, mas de forma a refletir os métodos criados no MAS;

O engenheiro de software deve verificar a necessidade da criação de novos use cases a partir da divisão da funcionalidade dos métodos anômalos.

Solução: 14.1) Substituir, nos Diagramas de Use Cases do MASA, os nomes dos use cases pelos nomes que os métodos sem anomalias passaram a utilizar no MAS. Esses nomes são visualizados na Lista de Mapeamento MASA x MAS;

14.2) Substituir, nos Diagramas de Use Cases do MASA, os nomes dos use cases pelos nomes que os métodos criados a partir da eliminação das anomalias passaram a utilizar no MAS. Esses nomes são visualizados na Lista de Correspondência de Métodos Anômalos;

14.3) Re-especificar os Diagramas de Use Cases do MASA, mantendo os atores e atualizando os fluxos de entrada e saída dos use cases para que reflitam as alterações necessárias;

14.4) Elaborar a Lista de Mapeamento dos Use Cases para documentar as modificações efetuadas nos Diagramas de Use Cases do MASA. A lista deve conter as seções:

a) Projeto, Versão do Documento, Data da Criação do Documento e Responsável pela Criação do Documento são preenchidos como nos padrões anteriores;
b) Use Cases (MASA): contém o nome dos use cases descritos nos Diagramas de Use Cases elaborados no MASA;
c) Use Cases (MAS): contém o nome dos use cases após a abstração do MAS;
d) Classe: lista a classe a que pertence o método que trata o use case.

Produtos Obtidos: Diagramas de Use Cases do MAS e Lista de Mapeamento dos Use Cases.

Padrões Relacionados: esse padrão somente pode ser aplicado após a aplicação dos padrões 11 e 13 (Criar Diagrama Use Cases do MASA e Abstrair Diagrama de Pseudo-Classes), pois suas saídas são utilizadas aqui como entradas. Continuando o processo de engenharia reversa, o engenheiro de software deve:

- realizar a inspeção de garantia da qualidade no Diagrama de Use Cases do MAS: padrão 4 (Realizar Inspeção de Garantia da Qualidade);
- atualizar o controle de configuração: padrão 5 (Controlar a Configuração);
- continuar o processo de engenharia reversa: padrão 15 (Descrever Use Cases do MAS).

15. Nome: Descrever Use Cases do MAS

Intuito: re-especificar as descrições dos use cases alterados durante a abstração dos modelos elaborados.

Problema: atualizar as descrições dos use cases modificados após a abstração do Diagramas de Use Cases do MASA.

Solução: 15.1) Obter, na Lista de Mapeamento de Use Cases, os nomes dos métodos que originam os use cases do MAS e que tenham sofrido modificações, além da troca do nome para melhoria da representatividade;
15.2) Re-especificar as descrições dos use cases, para cada método encontrado no passo anterior, de forma que reflitam as modificações necessárias com relação à funcionalidade e acesso a classes;
15.3) Descrever os use cases que, no MASA, eram tratados por componentes do ambiente Delphi e que no MAS passaram a ser tratados por métodos das classes.
Produto Obtido: Descrições dos Use Cases do MAS.
Padrões Relacionados: esse padrão somente pode ser aplicado após a aplicação dos padrões 11 e 13 (Criar Diagrama Use Cases do MASA e Abstrair Diagrama de Pseudo-Classes) , pelo fato de suas saídas serem utilizadas aqui como entradas. Continuando o processo de reengenharia, o engenheiro de software deve:

- realizar a inspeção de garantia da qualidade nas Descrições dos Use Cases do MAS: padrão 4 (Realizar Inspeção de Garantia da Qualidade);
- atualizar o controle de configuração: padrão 5 (Controlar a Configuração);
- inspecionar o andamento do processo em relação ao Plano para Realização da Reengenharia: padrão 3 (Acompanhar o Progresso do Processo de Reengenharia);
- iniciar o processo de engenharia avante: padrão 16 (Elaborar Diagrama de Classes de Projeto).

CLUSTER 6 – Elaborar o Projeto Avante.

16. Nome: Elaborar Diagrama de Classes de Projeto
Intuito: modularizar a funcionalidade do software, de forma a separá-la do acesso aos dados.
Problema: criar, uma nova instância do Diagrama de Classes, com menor nível de abstração, para visualizar as modularizações necessárias à separação da lógica de negócios da lógica de armazenamento do software.
Contexto: separar o acesso a dados dos aspectos funcionais e da interface do software é difícil para um programador sem experiência em programação orientada a objetos.
Solução: 16.1) Derivar cada classe constante do Diagrama de Classes elaborado com a aplicação do padrão 13 (Abstrair Diagrama de Pseudo-Classes) ;
16.2) Criar um método para cada funcionalidade presente em relação à manipulação e ao acesso aos dados da(s) tabela(s) com as quais cada classe se relaciona;
16.3) Caso necessário, criar atributos que facilitem a manipulação dos dados;
16.4) Criar um Diagrama de Classes de Projeto para documentação das classes definidas nos Passos 16.1 a 16.3.
Produto Obtido: Diagrama de Classes de Projeto.
Padrões Relacionados: esse padrão utiliza como entrada, a saída da aplicação do padrão 13 (Abstrair Diagrama de Pseudo-Classes). Continuando a engenharia avante, o engenheiro de software deve:

- realizar a inspeção de garantia da qualidade no Diagrama de Classes de Projeto: padrão 4 (Realizar Inspeção de Garantia da Qualidade);
- atualizar o controle de configuração: padrão 5 (Controlar a Configuração);
- continuar a engenharia avante: padrão 17 (Construir Diagramas de Seqüência).

17. Nome: Construir Diagramas de Seqüência
Intuito: documentar a lógica de negócio do sistema legado.
Problema: descrever a lógica de negócio do sistema legado para facilitar sua futura re-implementação.
Contexto: representar a interação dos métodos obtidos no padrão 16, Elaborar Diagrama de Classes de Projeto, pode ser uma tarefa complexa;
Métodos já definidos a partir das Descrições de Use Cases viabilizam a solução desse problema.
Solução: construir um Diagrama de Seqüência, a partir de cada Descrição de Use Case, obtidas a partir da aplicação dos padrões 12 e 15 (Descrever Use Cases do MASA e Descrever Use Cases do MAS), respectivamente. Devem ser considerados os métodos disponíveis nas classes para a construção dos diagramas.
Produto Obtido: Diagramas de Seqüência do Sistema.
Padrões Relacionados: esse padrão utiliza como entradas, as saídas da aplicação dos padrões 12, 15 e 16 (Descrever Use Cases do MASA, Descrever Use Cases do MAS e Elaborar Diagrama de Classes de Projeto). Continuando a engenharia avante, o engenheiro de software deve:

- realizar a inspeção de garantia da qualidade nos Diagramas de Seqüência: padrão 4 (Realizar Inspeção de Garantia da Qualidade);
- atualizar o controle de configuração: padrão 5 (Controlar a Configuração);
- inspecionar o andamento do processo em relação ao Plano para Realização da Reengenharia: padrão 3 (Acompanhar o Progresso do Processo de Reengenharia);
- continuar a engenharia avante: padrão 18 (Implementar as Classes).

CLUSTER 7 – Re-implementar o Software.

18. Nome: Implementar as Classes
Intuito: codificar a lógica de negócio do sistema, previamente modelada durante a aplicação dos padrões anteriores.
Problema: definir os atributos, tratando o encapsulamento dos mesmos, além de implementar os métodos de acesso a esses atributos, de forma a tratar a lógica de negócio de cada objeto.
Contexto: as classes referentes à lógica de negócios do sistema descritas no Diagrama de Classes de Projeto resultante da aplicação do padrão 16 (Elaborar Diagrama de Classes de Projeto) formam a base para a aplicação desse padrão.
O sucesso da aplicação desse padrão depende, em grande parte, da qualidade da modelagem previamente elaborada, ou seja, da correta expressão da lógica de negócio nos produtos de trabalho criados com a aplicação dos padrões anteriores;
O engenheiro de software tem facilidade na expressão da lógica de negócio por meio de diagramas.
Solução: Para cada classe relacionada a lógica de negócio presente no Diagrama de Classes de Projeto, deve-se:
18.1) Criar a estrutura da classe com as seções destinadas aos atributos e métodos públicos, protegidos e/ou privados;
18.2) Criar o código fonte correspondente aos atributos simples e aos atributos originados de relacionamentos com outras classes, verificando sua visibilidade (em qual seção serão declarados) e tipo;

18.3) Criar o código-fonte para os métodos que tratam das funcionalidades do objeto. Para isso, deve-se verificar a Lista de Mapeamento MASA x MAS e a Lista de Correspondência de Métodos Anômalos, no sentido de buscar a origem do código-fonte legado que deverá ser re-implementado em cada método.

Produto Obtido: Código-fonte das classes relativas à lógica de negócios do software.

Padrões Relacionados: a saída da aplicação do padrão 16 (Elaborar Diagrama de Classes de Projeto), forma a base para a aplicação desse padrão, aplicado em conjunto com os demais padrões do *cluster*, de forma a completar, simultaneamente, a interface, o acesso a dados e as funcionalidades previstas no software. Em conjunto com a aplicação dos padrões, o engenheiro de software deve:

- realizar a inspeção de garantia da qualidade no código fonte das classes implementadas: padrão 4 (Realizar Inspeção de Garantia da Qualidade);
- atualizar o controle de configuração: padrão 5 (Controlar a Configuração).

19. Nome: Implementar a Lógica de Armazenamento

Intuito: definir a lógica de armazenamento do software.

Problema: implementar os objetos responsáveis pelo encapsulamento dos métodos de acesso aos dados.

Contexto: as classes que provêem a interface com o banco de dados, documentadas no Diagrama de Classes de Projeto construído a partir da aplicação do padrão 16 (Elaborar Diagrama de Classes de Projeto) são utilizadas como entrada.

A orientação a objetos permite que o código-fonte seja totalmente organizado, de forma que a hierarquia dos objetos é definida através de vários níveis de abstração;

Para aplicações multi-usuários, esse padrão deve contemplar a implementação, nos métodos, da manutenção da integridade dos dados, aspecto disponível em bancos de dados como Oracle e Microsoft SQL Server.

Solução: para cada classe relacionada a lógica de armazenamento dos dados, presente no Diagrama de Classes de Projeto, deve-se:

8.1) Criar a estrutura da classe, com as seções destinadas aos atributos e métodos públicos, protegidos e/ou privados;

8.2) Criar o código-fonte correspondente aos atributos simples e aos atributos originados de relacionamentos com outras classes, verificando sua visibilidade (em qual seção serão declarados) e tipo;

8.3) Criar o código-fonte para os métodos que tratam do acesso aos dados do objeto. Para isso, deve-se verificar a Lista de Mapeamento MASA x MAS e a Lista de Correspondência de Métodos Anômalos, no sentido de buscar a origem do código-fonte legado que deverá ser re-implementado em cada método.

Produto Obtido: código-fonte das classes relativas à lógica de armazenamento e manipulação de dados.

Padrões Relacionados: esse padrão relaciona-se com os padrões 16 e 18 (Elaborar Diagrama de Classes de Projeto e Implementar as Classes), por suas saídas serem utilizadas aqui como entrada. Continuando a engenharia avante, o engenheiro de software deve:

- realizar a inspeção de garantia da qualidade no código-fonte das classes relativas à lógica de armazenamento e manipulação de dados do software: padrão 4 (Realizar Inspeção de Garantia da Qualidade);
- atualizar o controle de configuração: padrão 5 (Controlar a Configuração);
- continuar a engenharia avante: padrão 6 (Implementar a Lógica de Apresentação).

20. Nome: Implementar a Lógica de Apresentação

Intuito: redefinir a interface do software sem a utilização de componentes de acesso direto ao banco de dados.

Problema: implementar a interface do software e a ligação aos métodos das classes anteriormente criadas.

Contexto: no Delphi, software implementado sem a utilização do paradigma orientado a objetos utilizam, para apresentar os dados na interface, componentes que manipulam diretamente o banco de dados, sem a interferência de procedimentos ou funções definidos pelo programador. Já, software implementado de acordo com o paradigma orientado a objetos, fazem uso de eventos que utilizam os métodos definidos nas classes implementadas com a aplicação dos padrões 18 e 19 (Implementar as Classes e Implementar a Lógica de Armazenamento do Software), de forma a prover a apresentação dos dados.

- Os componentes de manipulação de dados presentes no ambiente Delphi facilitam a implementação. Porém, estes somente podem ser utilizados no caso de acesso direto aos dados, o que fere o conceito de encapsulamento do paradigma orientado a objetos;
- Há componentes gráficos (campos de edição e grids) que apresentam dados fornecidos pelo programador, podendo ser utilizados para mostrar a saída de métodos das classes por este definidas;
- A codificação da interface orientada a objetos no ambiente Delphi é, em primeira instância, mais complexo que a programação procedimental, pelo fato do acesso aos dados ter que ser completamente implementado pelo programador de forma a apenas trocar informações com as interfaces. Porém, uma vez codificadas as interfaces básicas, estas podem ser reutilizadas, bem como as classes de auxílio à apresentação dos dados.

Solução: 20.1) Definir as interfaces do software de forma a prover todos os recursos disponíveis no legado utilizando somente componentes simples de apresentação de dados;
20.2) Implementar, a partir de eventos, o acesso aos métodos das classes codificadas durante a aplicação dos padrões 18 e 19.

Produtos Obtidos: Interfaces e código-fonte referente à lógica de apresentação do software.

Padrões Relacionados: os padrões 17, 18 e 19 (Construir Diagramas de Seqüência, Implementar as Classes e Implementar a Lógica de Armazenamento) são complementares a esse padrão pelo fato de suas saídas serem utilizadas aqui como entradas. Complementando o processo de engenharia avante, o engenheiro de software deve:

- realizar a inspeção de garantia da qualidade no código-fonte referente à lógica de apresentação do software: padrão 4 (Realizar Inspeção de Garantia da Qualidade);
- atualizar o controle de configuração: padrão 5 (Controlar a Configuração).

4. Considerações Finais

A forma de descrição do processo por meio de padrões facilita o aprendizado pelos engenheiros de software, bem como a aplicação repetitiva de partes isoladas do processo para prover refinamentos de forma a gerar um produto final com qualidade. Ressalta-se que os padrões existentes na FaPRE/OO, originalmente desenvolvidos para serem utilizados em sistemas legados implementados em Clipper foram instanciados para softwares implementados em Delphi, comprovando assim que o processo de reengenharia apresentado

pode ser utilizado, após algumas adaptações, independente da linguagem de programação do software legado.

Os padrões do PRE/OO foram elaborados para serem utilizados de forma evolutiva, garantindo um melhor resultado aos produtos resultantes dessa aplicação. Seus resultados, documentados com formato padronizado auxiliam os engenheiros de software no entendimento, manuseio e continuidade da reengenharia.

Qualquer sistema implementado no ambiente Delphi, do qual se tenha o código-fonte, tabelas de dados e programa executável, pode ser submetido aos padrões propostos. Como trabalho futuro pretende-se derivar os padrões propostos para sua aplicação em sistemas desenvolvidos em outros ambientes, como o Visual Basic. Além disso, pretende-se pesquisar a construção de ferramentas para automatização de alguns padrões que consistem de passos cabíveis de realização automática, como os padrões 6, 7 e 8 (Elaborar Lista de Procedimentos e Funções, Elaborar Lista de "Chama/Chamado Por" e Modelar Dados do Software Legado).

Outra possibilidade de trabalho futuro é a adaptação de outras KPAs do SW-CMM relacionadas ao processo de software, tornando-as novos padrões de qualidade do processo de reengenharia.

Agradecimentos

Agradecemos ao shepherd Tiago Lima Massoni pelas sugestões e acompanhamento dados a este trabalho.

Referências

[1] Cantu, M. **"Dominando o Delphi 6: a Bíblia"**, Makron Books. Isbn: 8534614083. 2001.

[2] Demeyer, S.; Ducasse, S.; Tichelaar, S. **"A Pattern Language for Reverse Engineering"**, 4th European Conference on Pattern Languages of Programming and Computing, Paul Dyson (Ed.), Germany. July, 1999.

[3] Demeyer, S.; Ducasse, S.; Nierstrasz, O. **"A Pattern Language for Reverse Engineering".** 5th European Conference on Pattern Languages of Programming and Computing. Pages 189-208. July 2000a.

[4] Demeyer, S.; Ducasse, S.; Nierstrasz, O. **"Tie Code and Questions: a Reengineering Pattern".** 5th European Conference on Pattern Languages of Programming and Computing. Pages 209-217. 2000b.

[5] Lemos, G. S. **"PRE/OO – Um Processo de Reengenharia Orientada a Objetos com Ênfase na Garantia da Qualidade"**, São Carlos-SP. Dissertação de Mestrado apresentada ao PPGCC-DC. Universidade Federal de São Carlos. Agosto/2002. Disponível em http://www.svconsultoria.com.br/pessoal/gizelle/

[6] OMG. **"Unified Modeling Language Specification"**. Object Management Group. Versão 1.3. Junho 1999.

[7] Paulk, M.; Weber, C.; Wise, C.; Withey, J. **"Key Practices of the Capability Maturity Model, Version 1.0".** Software Engineering Institute, CMU/SEI-91-TR-25. 1991.

[8] Penteado, R. A. D., **"Um Método para Engenharia Reversa Orientada a Objetos"**.

Tese de Doutorado em Física Computacional apresentada ao Instituto de Física de São Carlos, Universidade de São Paulo. 237 páginas. 1996.

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

[10] Pressman, R. S. **"Software Engineering: A Practitioner's Approach".** Fifth Edition. McGraw-Hill Higher Education. 2001.

[11] Recchia, E. L., **"Engenharia Reversa e Reengenharia Baseadas em Padrões"** Dissertação de Mestrado apresentada ao PPGCC-DC. Universidade Federal de São Carlos. Junho/2002.

[12] Recchia, E. L.; Penteado, R. **"Avaliação da Aplicabilidade da Linguagem de Padrões de Engenharia Reversa de Demeyer a Sistemas Legados Procedimentais".** Artigo apresentado em 2nd Latin American Conference on Pattern Languages of Programming – Software Pattern Applications. (SugarLoafPLoP–SPA), Itaipava-RJ, Agosto/2002.