Padrões para o Processo de Engenharia Avante na Reengenharia Orientada a Objetos de Sistemas Legados Procedimentais

Edson Luiz Recchia
Univ. Anhembi Morumbi e UNICEP - São Carlos
erecchia@terra.com.br

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

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

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

Resumo

Padrões de reengenharia, incluindo engenharia reversa e engenharia avante são desenvolvidos por profissionais experientes, registrando como esses profissionais conduzem esses processos. Esses padrões têm como objetivo auxiliar engenheiros de software a conduzir esses processos em seus sistemas legados. A Família de Padrões de Reengenharia - FaPRE/OO - para a Reengenharia Orientada a Objetos de Sistemas Legados Procedimentais foi criada para conduzir esses processos a partir de sistemas legados procedimentais, tendo sistemas orientados a objetos como alvo. Foram apresentados no SugarLoafPLoP'2002 os padrões da FaPRE/OO para o processo de engenharia reversa. Neste trabalho, são apresentados os padrões para o processo de engenharia avante, ilustrando-se, passo a passo, o seu uso.

Abstract

Reengineering Patterns, including reverse engineering and forward engineering patterns, have been developed by experienced software engineers, in order to track how these processes are conducted in legacy systems. The goal of these patterns is to help software engineers to conduct these processes in their systems. The Family of Patterns for Object-Oriented Reengineering - FaPRE/OO - of legacy systems was developed to conduct these processes from procedural legacy systems to object-oriented target systems. At SugarLoafPLoP'2002, the patterns of FaPRE/OO for reverse engineering were presented. In this paper, forward engineering process patterns are presented step by step, illustrating their use.

1. Introdução

O processo de reengenharia é amplamente reconhecido como um dos desafios mais significativos para engenheiros de software. O problema é comum, afetando todos os tipos de organização; sério, pois uma falha na reengenharia pode dificultar os esforços da organização para manter-se competitiva; persistente, pois parece não haver razão para ficar confiante de que os novos sistemas não vão ser também os legados de amanhã (Stevens *et al*. [15]).

Dewar *et al*. [5] relatam que pretendiam entender como profissionais experientes conduziam a reengenharia de sistemas legados, a fim de desenvolver técnicas e materiais para transferir essa experiência. Em particular, queriam tratar o problema de sintetizar experiência com reengenharia de sistemas e de organizações, para ajudar engenheiros de software a adquiri-las, em paralelo.

Os padrões para o processo de reengenharia, especialmente para engenharia avante orientada a objetos de sistemas legados procedimentais, abrangem, particularmente, o domínio de sistemas de informação, bem como técnicas de condução desses processos.

A abordagem proposta para resolver esse problema é uma Família de Padrões de Reengenharia, denominada FaPRE/OO, contendo um conjunto de três *clusters* de padrões para o Processo de Engenharia Reversa [12,13] e um *cluster* de padrões para o Processo de Engenharia Avante.

Neste trabalho, mostramos como os padrões para o processo de engenharia avante da FaPRE/OO foram elaborados a partir de sistemas legados em Clipper [12,13,14] e Cobol [2], para sistemas alvo em Delphi [12,13,14] e Java [2]. Os padrões da FaPRE/OO foram aplicados a casos concretos de elaboração de orçamentos de obras da construção civil [12,13,14], controle de material em uma mineradora [2] e controle contábil [7].

Este trabalho está organizado da seguinte forma: na Seção 2, são introduzidos os padrões para o processo de engenharia avante, propostos pela Família de Padrões de Reengenharia (FaPRE/OO); na Seção 3, são apresentados os comentários finais.

2. Padrões para o Processo de Engenharia Avante da Família de Padrões de Reengenharia - FaPRE/OO

A FaPRE/OO é uma Família de Padrões de Reengenharia para gerar processos de engenharia reversa e de engenharia avante orientados a objetos, a partir de sistemas legados procedimentais. É composta de quatro *clusters,* cada um agrupando os padrões relacionados a situações similares da reengenharia, sendo três *clusters* para o processo de engenharia reversa e um para o processo de engenharia avante. A Figura 1 ilustra graficamente os *clusters* e os padrões existentes em cada um deles.

Cluster 1: Modelar os Dados do Legado: agrupa padrões que extraem informações a partir dos dados e do código fonte do sistema legado gerando o MER [12,13] - Modelo Entidade-Relacionamento (visão procedimental dos dados) e o MASA [9,10,11] - Modelo de Análise do Sistema Atual - Diagrama de Pseudo-Classes (visão orientada a objetos dos dados). Esses padrões conduzem as ações do engenheiro de software quando se tem o primeiro contato com um sistema de software. Fazem parte desse *cluster* os seguintes padrões: Iniciar Análise dos Dados; Definir Chaves; Identificar Relacionamentos; Criar Visão OO dos Dados.

Cluster 2: Modelar a Funcionalidade do Sistema: agrupa padrões para obter a funcionalidade do sistema, criando modelos que recuperem as regras de negócio da empresa, contidas no sistema legado. Esses padrões habilitam o engenheiro de software a obter um entendimento detalhado dos componentes (partes) do sistema de software, aprofundando, assim, sua compreensão sobre o sistema legado. Fazem parte desse cluster os seguintes padrões: Obter Cenários; Construir Diagramas de Use Cases; Elaborar a Descrição de Use Cases; Tratar Anomalias.

Cluster 3: Modelar o Sistema Orientado a Objetos: agrupa padrões para se obter o diagrama de classes e os diagramas de seqüência do sistema, através da interação dos produtos obtidos pelos padrões dos *clusters* anteriores. Esses padrões habilitam o engenheiro de software a obter o MAS [9,10,11] - Modelo de Analise do Sistema, sendo o modelo orientado a objetos a servir de suporte ao processo de engenharia avante. Fazem parte desse *cluster* os seguintes padrões: Definir as Classes; Definir Atributos; Analisar Hierarquias; Definir Métodos; Construir Diagramas de Seqüência.

Cluster 4: Gerar o Sistema Orientado a Objetos: agrupa padrões que completam o processo de reengenharia do sistema, transformando o sistema legado, do paradigma procedimental para o paradigma orientado a objetos. Fazem parte desse *cluster* os seguintes

padrões: Definir a Plataforma; Converter o Banco de Dados; Implementar os Métodos; Realizar Melhorias na Interface.

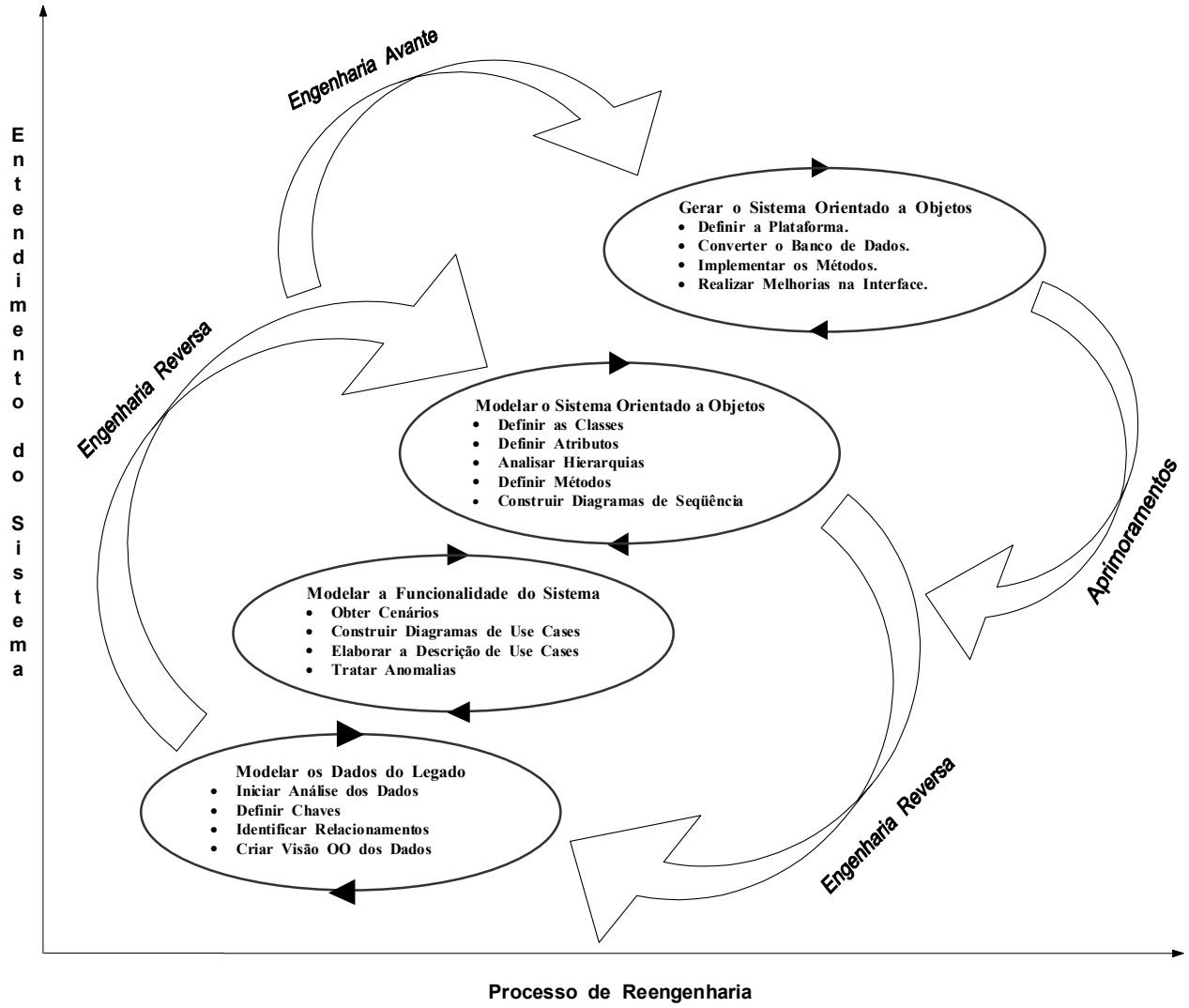

Figura 1: FaPRE/OO - Família de Padrões para Reengenharia Orientada a Objetos [12,13]

O leitor observará que muitos padrões terão como entrada a saída de algum padrão aplicado anteriormente, exigindo então sua aplicação seqüencial. No entanto, como se pode observar na Figura 1, o modelo é evolutivo, podendo-se, de qualquer padrão, avançar ou retroceder após a sua aplicação. A aplicação seqüencial somente será necessária durante o primeiro ciclo. Como exemplo, pode-se citar o caso de sistemas legados com centenas de arquivos de dados, divididos em módulos funcionais (compras, vendas, financeiro, estoque, entre outros.). Inicia-se a engenharia reversa a partir de um módulo qualquer, aplicando-se todos padrões apresentados na Figura 1, construindo-se assim todos os modelos envolvidos. Com isso, vai-se dominando paulatinamente o sistema, podendo-se incorporar outros módulos, num processo cíclico e progressivo.

Embora a FaPRE/OO seja composta de padrões para tratar tanto a engenharia reversa como a engenharia avante, somente os padrões referentes ao Processo de Engenharia Avante são considerados neste trabalho.

Os padrões de engenharia avante apresentados a seguir são descritos de acordo com o seguinte formato: Nome, Intuito, Contexto, Problema, Solução, Conseqüências, Usos Conhecidos e Padrões Relacionados. Esse formato foi adotado com base nos já existentes na literatura (Demeyer *et al*. [4], Stevens *et al*. [15]), de acordo com a experiência dos autores, além do fato de não existirem padrões de reengenharia orientados a objetos que abordem sistemas legados procedimentais.

O item correspondente à Solução a ser adotada é apresentado na forma de passos, sempre que necessário. O item Conseqüências (méritos e desvantagens da solução) não é apresentado, uma vez que deve conter a opinião de outros engenheiros de software, quando da utilização desses padrões em seus processos de reengenharia (os engenheiros de software ao aplicarem os padrões, deixarão registrados nesse item as suas experiências positivas e negativas).

Esta seção está organizada da seguinte forma: na subseção 2.1, são introduzidos os padrões para o processo de engenharia avante, correspondendo ao *cluster* Gerar o Sistema Orientado a Objetos, proposto pela Família de Padrões de Reengenharia (FaPRE/OO); e na subseção 2.2, são apresentados os comentários sobre Métodos Automáticos de Geração de Códigos OO, relativos à esse trabalho.

2.1 O *Cluster* Gerar o Sistema Orientado a Objetos

Agrupa padrões para o processo de engenharia avante de um sistema, transformando o sistema legado, do paradigma procedimental para o paradigma orientado a objetos.

2.1.1 Nome: Definir a Plataforma

Intuito:

Definir a plataforma de desenvolvimento do novo sistema conforme as exigências do Negócio e dos Objetivos Estratégicos da organização empresarial.

Contexto:

- O atual e o futuro ambiente operacional da organização empresarial são objetos de estudo que devem incluir as seguintes análises:
  (1) O quê de fato já existe;
  (2) Quais os atuais e os futuros recursos operacionais disponibilizados;
  (3) A cultura da organização;
  (4) O histórico de desenvolvimentos realizados;
  (5) O foco estratégico da organização;
  (6) Quais o processos de negócios que serão suportados.
- Avaliar as considerações anteriores juntamente com a área estratégica da organização, pois a mudança de um ambiente operacional envolve custos, tais como:
  (1) Treinamento dos usuários na nova tecnologia;
  (2) Investimentos em infra-estrutura que envolvem a solução adotada, podendo ser necessário adquirir: novos computadores servidores e estações de trabalho, roteadores e switches para interligação remota dessas estações. Também, pode

ocorrer a necessidade de reestruturar toda a rede de informática, efetuando-se, por exemplo, a mudança de cabos coaxiais para a tecnologia de par-trançado;

(3) Investimento em todo o software necessário para a implementação da solução escolhida, podendo ser necessário adquirir: um novo sistema operacional, tal como Novell, Windows 2000 Server; um novo Sistema Gerenciador de Banco de Dados (SGBD) Relacional; uma ferramenta de desenvolvimento, tal como Java, Delphi; softwares de apoio, tais como backup, correio interno.

(4) Investimento no desenvolvimento do aplicativo. Poder-se-á optar pela terceirização desses serviços, ou treinar toda a equipe técnica da empresa na nova tecnologia, considerando que essa mesma equipe deverá manter o atual sistema legado em operação durante toda a fase de desenvolvimento do novo aplicativo.

- O engenheiro de software tem à sua disposição cópias de demonstração de todos os pacotes, utilitários e ferramentas de suporte técnico.
- O engenheiro de software tem um bom conhecimento de técnicas de reuniões para se obter um consenso da área estratégica da organização.

Problema:

Organizações empresariais que desejam modernizar a Tecnologia da Informação usada como suporte às realizações dos seus Negócios, enfrentam dificuldades na condução desse processo.

Solução:

(a) Escolher o Ambiente Operacional.

Escolher o ambiente que dará suporte ao negócio da empresa, dentre as seguintes possibilidades:

Ambiente *Stand-alone*: caracterizado por uma aplicação mono-usuário. Esse ambiente utiliza os recursos de atualizações em *Cache* como transações.

Ambiente de Rede: caracterizado por uma aplicação multiusuário. Esse ambiente utiliza os recursos de atualização Cliente/Servidor como transações, podendo ser uma rede local (LAN), metropolitana (MAN) ou de grande abrangência (WAN).

Ambiente *Web*: caracterizado por uma aplicação multicamadas como transações. Essa arquitetura é composta de três camadas lógicas: camada de persistência (acesso ao SGBD); camada do servidor de aplicação (gerenciamento das regras de negócio); camada de apresentação (interface com cliente). Um protocolo de transporte é o elo entre as camadas de persistência, servidor de aplicação e de apresentação. Protocolos típicos são DCOM, CORBA, HTTP e soquetes TCP/IP. O elo entre o servidor de aplicação e a camada de persistência é usualmente feito por soquetes TCP/IP.

(b) Escolher o Sistema Gerenciador de Banco de Dados (SGBD) Relacional.

Escolher um sistema gerenciador de banco de dados relacional, dentre as várias opções existentes: Oracle, Sybase, SQL-Server, Informix, Ingres, InterBase, Progress.

(c) Escolher a Linguagem de Implementação e o Ambiente de Desenvolvimento.

Escolher uma linguagem de implementação, dentre as várias opções existentes: Java, Delphi, entre outras.

Usos Conhecidos:

Como exemplo de definição da plataforma de desenvolvimento do novo sistema conforme as exigências do Negócio e dos Objetivos Estratégicos da organização empresarial, considere a aplicação desse padrão ao caso real do sistema de elaboração de orçamentos de obras da construção civil - Sistema SIGO [12,13,14].

(a) Escolher o Ambiente Operacional.
Decide-se pelo **Ambiente de Rede.** A escolha desse ambiente operacional deve-se ao fato de aproveitar os investimentos em hardware já realizados pelas construtoras da região e escritórios de engenharia civil que utilizam a versão caracter do Sistema SIGO:

- Elas possuem uma rede local implantada com a tecnologia de par-trançado, bem como microcomputadores Pentium. Nenhuma dessas empresas trabalha com estações remotas de acesso ao Sistema SIGO.

(b) Escolher a Linguagem de Implementação e o Ambiente de Desenvolvimento.
Escolher uma linguagem de implementação, dentre as várias opções existentes: Ada, C++, Delphi, Java, entre outras.
Optou-se pelo ambiente de desenvolvimento Delphi [1] pelo fato de ser uma opção para implementar a funcionalidade de sistemas de informação no paradigma de orientação a objetos. Esse ambiente de desenvolvimento tem obtido grande aceitação no mercado.

(c) Escolher o Sistema Gerenciador de Banco de Dados (SGBD) Relacional.
Escolher um sistema gerenciador de banco de dados relacional, dentre as várias opções existentes: Oracle, Sybase, SQL-Server, Informix, Ingres, InterBase, Progress.
Optou-se pelo InterBase , por já acompanhar o ambiente de desenvolvimento Delphi e por estar consolidado no mercado.

Padrões Relacionados:
Pode-se considerar a aplicação, em paralelo, do padrão *Arquitetural Layer* [16,18].

2.1.2 Nome: Converter o Banco de Dados

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

Contexto:

- A grande quantidade de arquivos de dados e a geração não automática desses arquivos para tabelas de um banco de dados, torna-se viável o desenvolvimento de um processo para a realização da conversão desses arquivos de dados em tabelas de um SGDB..
- Têm-se as classes geradas, a partir dos arquivos de dados, viabilizando esse processo. Essas classes são obtidas quando da aplicação dos padrões Definir as Classes; Definir Atributos e Analisar Hierarquias; no processo de engenharia reversa. Logo, essas classes serão as tabelas a serem criadas no SGBD.
- O engenheiro de software tem um bom conhecimento da linguagem de implementação, para derivar os arquivos de dados do sistema legado para tabelas de banco de dados relacional.
- O engenheiro de software tem à sua disposição ferramentas de conversão de dados, as quais acompanham a linguagem de implementação e o SGBD a serem utilizados na reengenharia.

Problema:
Gerar um banco de dados, a partir dos arquivos de dados que contêm os dados do sistema legado.

Solução:

Usar a ferramenta adequada à linguagem de implementação e ao SGBD escolhidos, para converter os arquivos de dados do sistema legado em tabelas do banco de dados relacional escolhido.

Usos Conhecidos:

Usar as ferramentas InterBase Windows ISQL, SQL Explorer e Data Pump, para converter os arquivos de dados (.dbf's) do sistema legado SIGO em tabelas do SGBD InterBase, de acordo com os seguintes passos:

(a) Criar um Banco de Dados, com extensão .gdb, utilizando a ferramenta: InterBase Windows ISQL. Ex.: SIGO.GDB;

(b) Criar um *alias* (objeto lógico) para a pasta (objeto físico) SIGO.GDB, que está no diretório C:\ SIGO, utilizando a ferramenta:*SQL Explorer* . Ex.: *Alias* SIGO;

(c) Para cada classe definida no Diagrama de Classes do Sistema, obtido no Processo de Engenharia Reversa, criar a respectiva tabela no banco de dados, utilizando a ferramenta *SQL Explorer*. Ex.:

```
CREATE TABLE CLIENTES (
     CODIGO-CLIENTE    INTEGER       NOT NULL,
     NOME              VARCHAR(40)   NOT NULL,
     ENDERECO          VARCHAR(30),
     CODIGO-PAIS       INTEGER       NOT NULL   )
```

(d) Para cada tabela definida no *alias* SIGO, transportar o respectivo arquivo de dados (.dbf) do sistema legado para o banco de dados, utilizando a ferramenta *Data Pump*. Essa ferramenta apenas transporta os dados de um arquivo de dados (.dbf) para um banco de dados (.gdb), não realizando a adição de *flags*, por exemplo *NOT NULL*. Ex.: Transportar o arquivo de dados Cliente.dbf para a Tabela temporária Transporte;

(e) Como o *Data Pump* apenas transfere dados, importar, para a tabela CLIENTE, os dados da tabela TRANSPORTE, usando o comando *select* de uma declaração SQL, sendo que, nesse momento são aplicadas as regras constantes na tabela CLIENTE, por exemplo, *NOT NULL*:

```
INSERT  INTO CLIENTE ( CODIGO-CLIENTE, NOME, ENDERECO, CODIGO-PAIS )
     SELECT  CODIGO, NOME, ENDERECO, CODPAIS
     FROM    TRANSPORTE;
```

(f) Para cada tabela gerada no banco de dados, criar um gerador de códigos para a geração automática de códigos pelo SGBD ao se inserir um novo registro.
Considere que o arquivo CLIENTE.DBF possuía 5000 registros, então inicializa-se o gerador de código `GERAR-CODIGO-CLIENTE` com o valor 5001, por meio da seguinte declaração SQL:

```
CREATE GENERATOR GERAR-CODIGO-CLIENTE
     SET GENERATOR GERAR-CODIGO-CLIENTE TO 5001;
```

(g) Criar um *Trigger* para a geração automática de código pelo SGBD ao se inserir um novo registro. Neste caso, o *Trigger* é disparado automaticamente toda vez que um novo registro for incluído. Para a tabela CLIENTE, a seguinte declaração SQL deve ser construída:

```
CREATE  TRIGGER  SETAR-CODIGO-CLIENTE
     FOR     CLIENTE
     BEFORE  INSERT   POSITION 0   AS
     BEGIN
         new.CODIGO-CLIENTE  =  gen-id( GERAR-CODIGO-CLIENTE, 1 )
     END;
```

(h) Criar a Chave Primária da tabela. Para a tabela CLIENTE, a seguinte declaração SQL deve ser construída:

```
ALTER TABLE  CLIENTE
      ADD CONSTRAINT  CLIENTE-CHAVE-PRIMARIA
      PRIMARY KEY     ( CODIGO-CLIENTE );
```

(i) Criar a Chave Estrangeira da tabela, se existir. Para a tabela CLIENTE, a seguinte declaração SQL deve ser construída:

```
ALTER TABLE  CLIENTE
      ADD CONSTRAINT  CLIENTE-CHAVE-ESTRANGEIRA
      FOREIGN KEY     ( CODIGO-PAIS )
      REFERENCES      PAIS
            ON DELETE   CASCADE
            ON UPDATE   CASCADE;
```

(j) Criar os índices de acesso para a tabela, se existirem. Para a tabela CLIENTE, a seguinte declaração SQL deve ser construída, para a criação de um índice de acesso ordenado pelo atributo NOME:

```
CREATE UNIQUE ASCENDING INDEX  INDICE-CLIENTE-NOME
     ON CLIENTE ( NOME );
```

Padrões Relacionados:

Pode-se considerar a aplicação, em paralelo, do padrão Examinar o Banco de Dados [14].

2.1.3 Nome: Implementar os Métodos

Intuito:

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

Contexto:

- Transformar em programas a representação textual das Descrições de Use Cases (códigos fonte do legado) é difícil.
- O engenheiro de software tem à sua disposição todos os métodos obtidos na engenharia reversa por meio do padrão Definir Métodos.
- O engenheiro de software tem um bom conhecimento da linguagem de implementação para derivar os diagramas de seqüências, obtidos por meio do padrão Construir Diagramas de Seqüências, em linhas de código.

Problema:

Necessidade de converter códigos fonte do sistema legado em códigos orientados a objetos, na linguagem de implementação escolhida para a reengenharia.

Solução:

Escrever o código para cada um dos métodos obtido pelos padrões Definir Métodos e Construir Diagramas de Seqüência, na linguagem de implementação escolhida.

Deve-se considerar a aplicação, em paralelo, do padrão de projeto *Persistence Layer* [17] e dos padrões: Verificar as Invocações dos Métodos, Observar a Execução dos Componentes e Refazer para Entender [14].

Usos Conhecidos:

- A aplicação SIGO foi implementada para ser executada no Ambiente Multiusuário (rede local), que utiliza os recursos de atualizações Cliente/Servidor como transações.

- Exemplifica-se esse padrão com a implementação dos métodos `botãoGravarClick()` e `ClienteAfterPost()`, para a tabela CLIENTE. Para as demais tabelas o mecanismo é o mesmo. A Figura 2 mostra esses métodos implementados na ferramenta Delphi.
- Observe-se, na Figura 2, que a atualização dos dados ocorre quando o usuário pressiona um botão Gravar, acionando evento *OnClick* desse botão, o qual executa automaticamente o procedimento `botãoGravarClick()`.
- O procedimento `botãoGravarClick()` atualiza o *buffer* de memória do computador do usuário (estação *Cliente*) por meio do comando *Post*. Esse comando, por sua vez, ao ser executado aciona o evento *AfterPost* da *query* Cliente, que representa a tabela CLIENTE na estação *Cliente*. O evento *AfterPost* executa automaticamente o procedimento `ClienteAfterPost()`. Finalmente, esse procedimento efetua a atualização física dos dados no servidor de aplicações.

Padrões Relacionados:

- Padrão Verificar as Invocações dos Métodos. Objetivo: saber como uma classe está relacionada com outra verificando os parâmetros definidos nos métodos da interface da classe (assinatura do método). Vínculo ao padrão: obter o relacionamento entre classes.
- Padrão Observar a Execução dos Componentes. Objetivo: obter um entendimento detalhado do comportamento de uma parte do código, através da execução de seus componentes (encapsulamento). Vínculo ao padrão: é preciso obter o entendimento detalhado de uma parte encapsulada do código.
- Padrão Refazer para Entender. Objetivo: obter um melhor entendimento de uma parte específica do código fonte, refazendo-a. "Refazer para Entender" é o processo de modificar um sistema de software, de tal maneira que o comportamento externo do código não seja alterado, mas sua estrutura interna seja melhorada. Vínculo ao padrão: compreender um particular trecho de código que aparenta ser importante mas é muito difícil de analisá-lo completamente.

2.1.4 Nome: Realizar Melhorias na Interface

Intuito:

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

Contexto:

- Exige do engenheiro de software habilidade em conseguir transformar telas do ambiente caracter em telas do ambiente gráfico (GUI).
- Requer, do engenheiro de software, experiência no desenvolvimento de interfaces que:
  (1) Sejam consistentes;
  (2) Tenham atalhos para usuários experientes;
  (3) Tenham *Feedback* informativo;
  (4) Sejam organizada quanto aos diálogos de interação;
  (5) Previnam erros de usuários;
  (6) Possibilitem a reversibilidade de ações;
  (7) Reduzam a necessidade de memorização.

```
unit UnitCliente;

interface

type
  TFormCliente = class(TForm)
    Cliente: TIBQuery;
    •••
    botãoGravar: TButton;
    •••
    procedure botãoGravarClick(Sender: TObject);
    procedure ClienteAfterPost(DataSet: TDataSet);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

implementation

uses
  UnitConexao, •••

procedure TFormCliente.botãoGravarClick(Sender: TObject);
begin
  Cliente.Post;   // Atualiza o buffer de memória do computador: Cliente
end;

procedure TFormCliente.ClienteAfterPost(DataSet: TDataSet);
begin
  //
  // Executa um Refresh na Tabela CLIENTE
  //
  if Conexao.TransacaoGEST_ADM.InTransaction then
  begin
    try                                              // Atualiza fisicamente,
      Conexao.TransacaoGEST_ADM.CommitRetaining;  // computador: Servidor
    except
      ShowMessage('Outro Usuário Alterou ou Excluiu Esse Registro,
                   Pressione OK para Receber os Dados Atualizados do
                   Servidor' );
      Conexao.TransacaoGEST_ADM.RollbackRetaining;
    end;
  end;
end;

end.
```

Figura 2: Código Delphi que implementa as atualizações Cliente/Servidor como Transações

- O engenheiro de software tem à sua disposição todas as telas de tratamento da informação, do sistema legado.
- O engenheiro de software tem um bom conhecimento da linguagem de implementação para construir interfaces que viabilizem:
  (1) Facilidade de aprendizado;
  (2) Facilidade de utilização;
  (3) Ser intuitiva;
  (4) Diálogo simples e natural;
  (5) Velocidade na execução das tarefas;
  (6) Mensagens de erros consistentes;
  (7) Satisfação subjetiva.

Problema:

Existem diversas telas de acesso às funções do sistema legado, desenvolvidas para com interface baseada em caracter, totalmente peculiar às aplicações implementadas em Clipper, Cobol, entre outras. É, pois, necessário conceber uma nova interface, baseada em gráfico (GUI), que atenda também aos critérios de usabilidade, e que possa ser construída na linguagem de implementação escolhida para o processo de engenharia avante.

Solução:

Para cada tela do ambiente caracter, do sistema legado, gerar conjunto de interfaces na linguagem de implementação da reengenharia, que realizem melhorias por meio dos conceitos de usabilidade. Utilizar para a implementação as ferramentas de suporte que acompanham o ambiente de desenvolvimento escolhido.

Usos Conhecidos:

Apresenta-se na Figura 3 a tela do sistema legado Clipper quando de sua apresentação. Aplicando-se os conceitos de usabilidade, a Figura 4 exibe a tela de apresentação do sistema, após passar pelo processo de engenharia avante, utilizando a Família de Padrões de Reengenharia Orientada a Objetos.

Figura 3: Apresentação do Sistema Legado SIGO

Figura 4: Apresentação do Sistema SIGO após o Processo de Engenharia Avante

2.2 Métodos Automáticos de Geração de Códigos OO.

A geração do código fonte orientado a objetos, quando da aplicação do padrão Implementar os Métodos é realizada de forma tradicional, ou seja, não automatizada.

Neste trabalho não é utilizada a estratégia proposta por Jesus *et al.* [6] devido à restrição nela existente quanto à forma de programação do sistema Clipper que foi utilizado. Para que o sistema Draco-Puc fosse usado neste projeto seriam necessárias diversas modificações na sua gramática atualmente existente, o que demandaria tempo e esforço não viáveis no momento.

O sistema transformacional utilizado por Jesus *et al.* [6], não pôde ser aqui utilizado, tendo em vista a forma como o Draco-Puc converte um sistema legado Clipper para OO: Tranforma todo o conteúdo de Clipper para Java. Programas procedimentais desenvolvidos em Clipper que manipulassem arquivos de dados (.dbf's) com milhares de registros, tornavam-se extremamente lentos quando esses arquivos fossem acessados. Para melhorar o desempenho do sistema utilizava-se de artifícios de programação, de tal forma que eram criados arquivos secundários de apoio (persistentes e/ou temporários). Dessa forma, como o Draco-Puc converte integralmente o codigo legado Clipper em Java, as centenas/milhares de linhas de código de manipulação aos arquivos secundários, seriam também incluídas no código Java, bem como, os arquivos secundários seriam transformados em classes.

O trabalho aqui proposto visa eliminar todas as anomalias existentes no código legado. Para isso, os padrões dos *clusters* Modelar os Dados do Legado, Modelar a Funcionalidade do Sistema e Modelar o Sistema Orientado a Objetos, responsáveis pelo processo de engenharia reversa de um sistema, obtém o Modelo de Análise do Sistema (modelo orientado a objetos a servir de suporte ao processo de engenharia avante), eliminando-se todas as entidades temporárias encontradas durante o processo.

3. Comentários Finais

A realização do processo de reengenharia de sistemas legados é considerada como um desafio para os engenheiros de software, pois esse processo envolve muitos fatores de risco. Há, então, interesse em tornar os engenheiros de software especialistas nesse processo. Para isso, surgem os padrões de engenharia reversa e de engenharia avante com o objetivo de registrar as técnicas e mecanismos que os engenheiros de software experientes utilizam para conduzir esses processos.

Este trabalho apresentou os padrões do Processo de Engenharia Avante da FaPRE/OO, ou seja, padrões que auxiliam a realização da engenharia avante orientada a objetos de sistemas legados desenvolvidos de forma procedimental e implementados em linguagens como Clipper, Cobol, RPG II.

Um sistema, originalmente desenvolvido de forma procedimental e implementado na linguagem Clipper [12,13,14], foi submetido ao processo de reengenharia seguindo, passo a passo, todos os padrões conforme proposto pela FaPRE/OO. Assim, é realizada a engenharia reversa procedimental do sistema legado e, a partir dos resultados dessa atividade, efetua-se a engenharia reversa orientada a objetos do legado. Em outras palavras, na primeira fase obtém-se uma documentação procedimental e, na segunda fase, com base na anterior, constrói-se a documentação de análise orientada a objetos. A partir dos produtos da engenharia reversa o processo de engenharia avante é aplicado.

Outro sistema, originalmente desenvolvido de forma procedimental e implementado na linguagem Cobol, foi submetido ao processo de reengenharia usando esta Família[3]. Nesse trabalho a engenharia reversa é realizada diretamente, isto é, sem a necessidade do produto intermediário. A documentação de análise orientada a objetos é obtida diretamente do código procedimental a fim de identificar possíveis objetos. Assim, a FaPRE/OO dá plena cobertura para conduzir a engenharia reversa diretamente orientada a objetos a partir do sistema legado procedimental, sendo a ordem de aplicação dos vários padrões da Família, o diferencial das duas formas em que foram aplicados. Como o modelo do processo é evolutivo, isso fortalece o seu potencial em questão do domínio de sistemas de informação. A partir dos produtos da engenharia reversa, o processo de engenharia avante é aplicado.

Além disso, um outro sistema, desenvolvido na linguagem Delphi [8], foi submetido, com sucesso, ao processo de reengenharia seguindo, passo a passo, os padrões propostos na FaPRE/OO. Embora o ambiente de desenvolvimento deste trabalho tivesse sido Delphi, o qual viabiliza a construção de sistemas orientados a objetos, o sistema envolvido em um estudo de caso foi originalmente implementado sem os conceitos da orientação a objetos.

A FaPRE/OO tem as seguintes características:

- Possui quatro *clusters* de padrões claramente definidos, com regras para guiar a passagem de um padrão para outro;
- Cada padrão é dirigido a artefatos que devem ser produzidos;
- Tem uma forma cíclica e evolutiva de aplicação, podendo-se, de qualquer padrão, avançar ou retroceder à sua aplicação;
- O resultado produzido é baseado, principalmente, no sistema atual e os requisitos são mínimos: o sistema executável e o código fonte;
- É um processo global que incorpora estratégias específicas para os processos de engenharia reversa e de engenharia avante, como partes do processo de reengenharia.

Agradecimentos

Agradecemos ao Shepherd Vander Ramos Alves pelas sugestões e acompanhamento dado a este trabalho.

Referências Bibliográficas

[1] Borland Brasil, 2002. URL:http://www.borland.com.br/artigos/artigo_ccantu.htm. Consultado em 03/2002.

[2] Camargo, V., **"Reengenharia Orientada a Objetos de Sistemas COBOL com a Utilização de Padrões de Projetos e Servlets"**, São Carlos-SP, 2001. Dissertação de Mestrado. Universidade Federal de São Carlos.

[3] Camargo, V.; Recchia, E. L.; Penteado, R. – **"Aplicabilidade da Família de Padrões de Reengenharia FaPRE/OO na Engenharia Reversa Orientada a Objetos de Sistemas Legados COBOL"**, Artigo apresentado no The Second Latin American Conference on Pattern Languages of Programming – Software Pattern Applications. (SugarLoafPLoP–SPA), Itaipava-RJ, Agosto/2002.

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

[5] Dewar, R.; Lloyd, A.D.; Pooley, R.; Stevens, P. **"Identifying and Communicating Expertise in Systems Reengineering: a patterns approach".** IEEE Proceedings – Software, v.146, n.3, pp.145-152, 1999.

[6] Jesus, E.; Prado, A.F. - **"Reengenharia de Software para Plataformas Distribuídas Orientadas a Objetos"**, XIII Simpósio Brasileiro de Engenharia de Software - SBES'99, pg. 289-304.

[7] Kulk, E.; Camargo, V. V.; Masiero, P. C.; Penteado, R.; Germano, F., **"Reengenharia Orientada a Objetos de um Sistema Contábil Implementado em Cobol para Java"**, Documento de Trabalho, 2002. Universidade de São Paulo – SP.

[8] Lemos, G. S., **"PRE/OO - Um Processo de Reengenharia com Ênfase na Garantia da Qualidade"**, São Carlos-SP. Dissertação de Mestrado apresentada ao PPGCC-DC. Universidade Federal de São Carlos, em Agosto/2002.

[9] Penteado, R. A. D., **"Um Método para Engenharia Reversa Orientada a Objetos",** São Carlos, 1996. 237 p. Tese (Doutorado em Física Computacional) - Instituto de Física de São Carlos, Universidade de São Paulo.

[10] Penteado, R., Germano, F., Masiero, P. C., **"An Overall Process Based on Fusion to Reverse Engineering Legacy Code",** In: Working Conference Reverse Engineering, 3, 1996, Monterey-California. Anais. IEEE, p. 179-188.

[11] Penteado, R., Braga, R.T.V., Masiero, P.C., **"Improving the Quality Legacy Code by Reverse Engineering",** In: 4th International Conference on Information Systems Analysis and Synthesis, ISAS/98, Julho/1998, Orlando-Flórida.

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

[13] Recchia, E. L.; Penteado, R. – **"FaPRE/OO: Uma Família de Padrões para Reengenharia Orientada a Objetos de Sistemas Legados Procedimentais"**, Artigo apresentado no The Second Latin American Conference on Pattern Languages of Programming. (SugarLoafPLoP– Writers' Workshop), Itaipava-RJ, Agosto/2002.

[14] Recchia, E. L.; Penteado, R. – **Avaliação da Aplicabilidade da Linguagem de Padrões de Engenharia Reversa de Demeyer a Sistemas Legados Procedimentais**, Artigo apresentado no The Second Latin American Conference on Pattern Languages of Programming – Software Pattern Applications. (SugarLoafPLoP–SPA), Itaipava-RJ, Agosto/2002.

[15] Stevens, P.; Pooley, R. - **"Systems Reengineering Patterns"**, Proceedings ACM-SIGSOFT, 6th Internation Symposium on the Foundations of Software Engineering, p.17-23, 1998.

[16] Yoder, J.W.; Barcalow, J. - **"Architetural Patterns for Enabling Application Secutity"**, Conference on the Pattern Languages of Programs. PLoP'1997.

[17] Yoder, J.W.; Johnson, R.E; Wilson, Q.D. - **"Connecting Business Objects to Relational Databases"**, Conference on the Pattern Languages of Programs. PLoP'1998.

[18] http://www.openloop.com/softwareEngineering/patterns/architecturePattern/arch_Layers.htm Consultado em 05/2003.