Ferramenta para aplicação do padrão Data Access Object (DAO) em sistemas desenvolvidos com a linguagem de programação Delphi

Marcelo Sardagna

Orientador
Prof. Adilson Vahldick

Roteiro da apresentação

- Introdução
- Fundamentação teórica
- Desenvolvimento
- Resultados e Discussão
- Conclusão
- Limitações e Extensões

Introdução

- Padrões de Projetos
  - ✓ Reutilização de projetos e arquiteturas bem sucedidas
  - ✓ Expressam técnicas testadas e aprovadas ficando mais acessíveis para os desenvolvedores de novos sistemas
- Motivação
  - ✓ Aplicar padrões em sistemas já desenvolvidos
  - ✓ Centralização de comandos SQL em classes

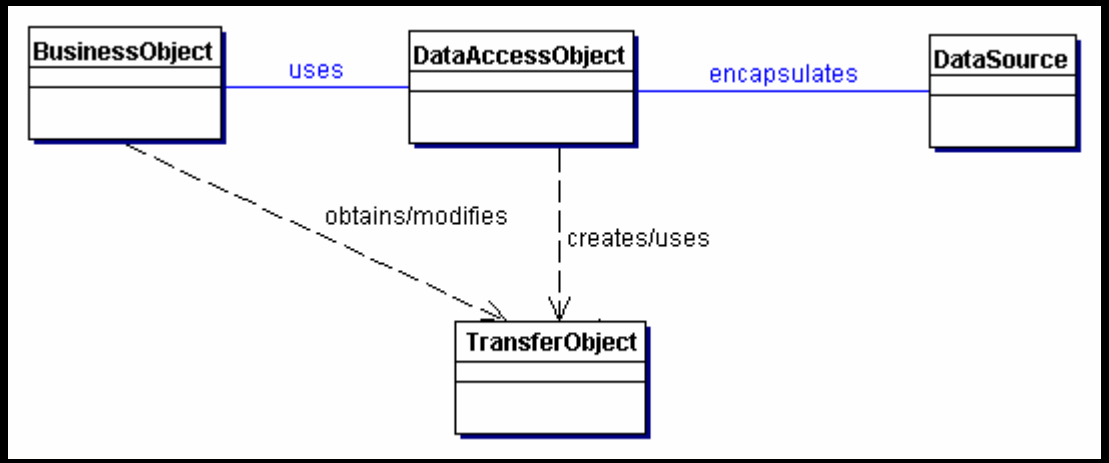
Introdução

- Objetivos do trabalho
  - ✓ Analisar arquivos fontes DFM e PAS
  - ✓ Encontrar e retirar comandos SQL dos arquivos fontes analisados
  - ✓ Criar classes e métodos DAO
  - ✓ Substituir os comandos SQL encontrados por métodos da classe DAO

Fundamentação Teórica

- Aplicação de Padrões
  - Benefícios
    - ✓Reutilização de soluções e código fonte
    - ✓Aplica-se uma terminologia comum
    - ✓Resolvem problemas reais
    - ✓Reutilização de idéias

Fundamentação Teórica - DAO



- *BusinessObject* - Objeto que requer o acesso aos dados
- *DataSource* - Origem dos dados
- *TransferObject* - Objeto de transferência usado como um portador de dados
- *DataAccessObject* – Objeto que abstrai a execução do acesso dos dados

Fundamentação Teórica

- Data Access Object
  - Vantagens
    - ✓abstração do acesso a dados da camada de negócios
    - ✓Acesso a base de dados transparente
    - ✓Maior produtividade
    - ✓Facilidade em futuras manutenções

Fundamentação Teórica
Componente de acesso `TSQLQuery`

- Porque o componente **`TSQLQuery`**
  - ✓ Confiabilidade
  - ✓ Componente nativo no Delphi 7.0
  - ✓ Portabilidade com a maioria dos bancos de dados utilizados
- **`TSQLQuery`**
  - ✓ Utilizado para executar comandos SQL,
  - ✓ Retorna resultados de um *select* ou executa comandos de *insert, update* e *delete*

Fundamentação Teórica
Componente de acesso `TSQLQuery`

- Propriedade e métodos

`Open`	Executa-se este método para comandos de *select* para que retorne os resultados da base de dados que está conectado
`ExecSQL`	Utilizado para comandos de *insert*, *update* e *delete*. Este método insere um registro no banco de dados a que está conectado
`Add`	Método utilizado para atribuir expressões SQL em tempo de execução
`ParamByName`	Método que atribui valores aos parâmetros
`SQL`	Propriedade para especificar o comando SQL
`Text`	Propriedade com a mesma funcionalidade do método Add

Trabalhos Correlatos

- Reutilização de Soluções com *Patterns* e *Frameworks* na Camada de Negócio
  - exemplificar o uso de *patterns* e *frameworks*;
  - criar um *framework* com os *patterns* selecionados;
  - desenvolver um material que ajudará os arquitetos de sistemas na tomada de decisão no desenvolvimento de um projeto
- Ferramenta para Extração e Documentação de Rotinas de Projetos de Sistemas de Informação
  - gerar uma biblioteca contendo todas as *procedures* e *functions*
  - identificar, através das análises léxica e sintática do código fonte, toda a manipulação feita com as tabelas do banco de dados
  - implementar um mecanismo para que o desenvolvedor possa consultar e emitir relatórios de toda documentação gerada

Desenvolvimento

- Cenários
- Diagrama de classes
- Geração de código
- Análise dos arquivos DFM
- Análise dos arquivos PAS
- Substituição dos Comandos SQL
- Estudo de caso

Desenvolvimento - Cenários

UC01 – Aplicar Padrão DAO

Sumário: O usuário deseja aplicar o padrão DAO em um projeto de Delphi.

Ator primário: Usuário.

Fluxo principal:

1. O usuário informa o banco de dados do projeto a ser analisado.
2. O usuário informa o caminho onde serão salvas as novas classes DAO.
3. O usuário informa o projeto a ser aplicado o padrão.
4. O usuário informa o caminho a serem salvos os novos arquivos de código fontes.
5. O usuário solicita à ferramenta para iniciar o processo.
6. A ferramenta cria uma classe DAO para cada tabela no banco de dados com os métodos Inserir, Alterar, Excluir e SelecionarTodos.
7. A ferramenta armazena os comandos SQL que foram especificados nas classes DAO.
8. A ferramenta analisa os arquivos DFM e PAS e retira os comandos SQL encontrados.
9. A ferramenta armazena os comandos SQL retirados dos arquivos fontes.
10. A ferramenta apresenta os comandos SQL encontrados nos fontes DFM e PAS.
11. O usuário solicita a ferramenta para proceder com a aplicação do padrão.
12. A ferramenta aplica o padrão e salva os novos arquivos fontes no local escolhido.

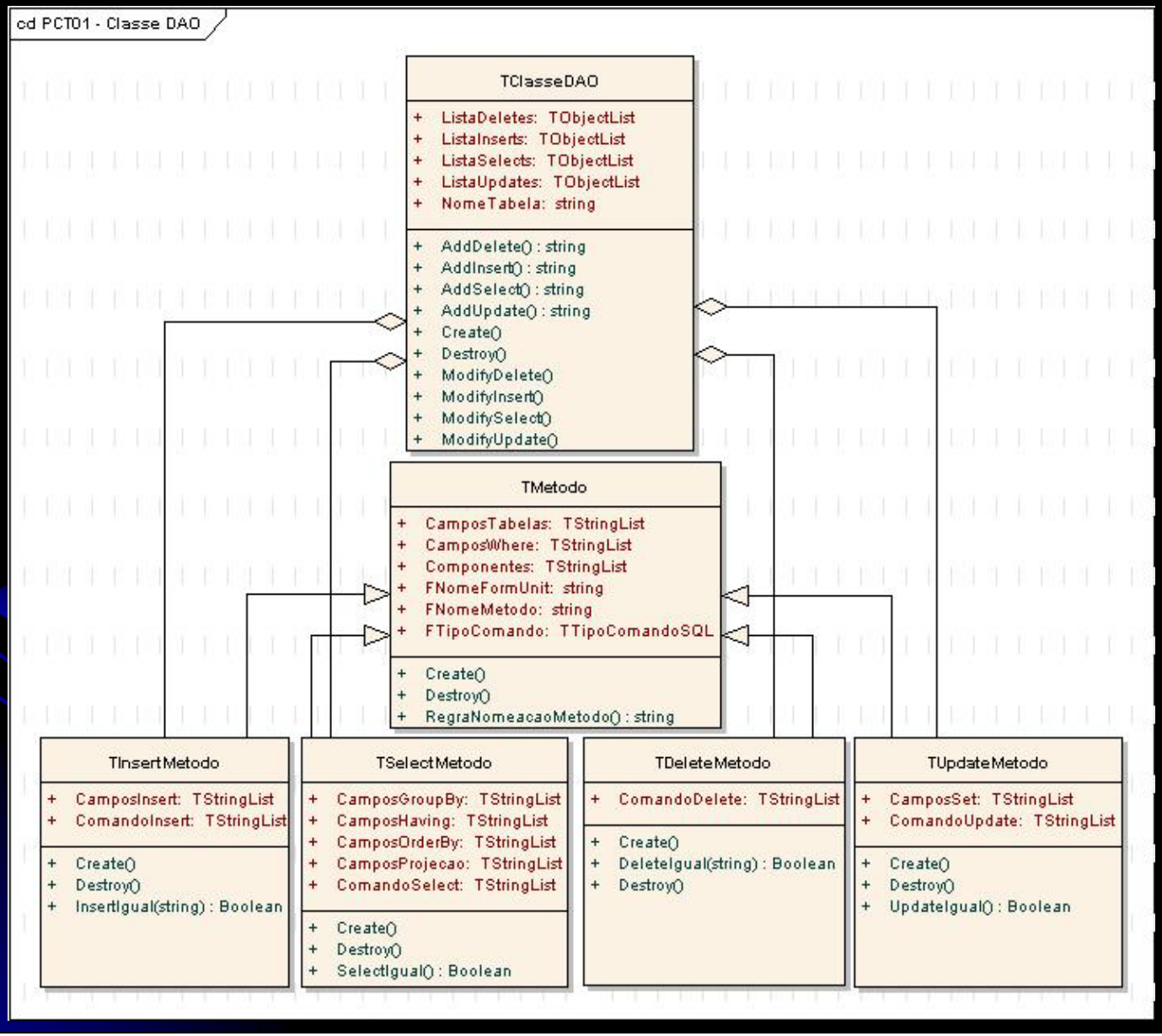
Fluxo alternativo: Alteração do nome dos métodos sugeridos.

a) Depois do passo 10 há possibilidade de alterar o nome dos métodos a serem criados.

Fluxo de exceção: erro léxico, sintático ou semântico.

a) No passo 9 se houver erro na análise léxica, sintática ou semântica dos comandos SQL, os mesmos são apresentados em tela, sendo que a análise continua ate não mais encontrar-los.

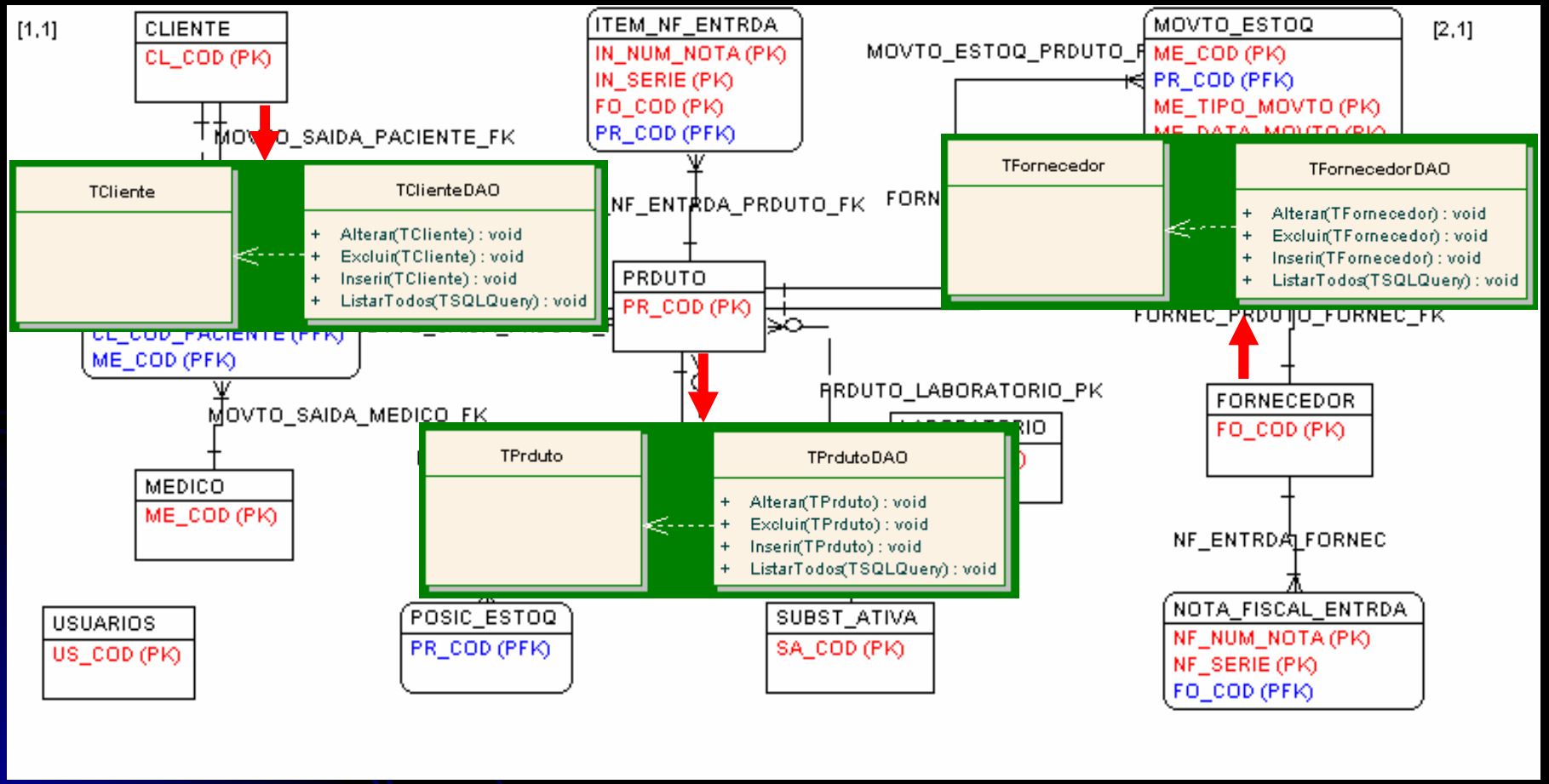
Desenvolvimento – Diagrama de Classes



Desenvolvimento
Geração de Código

- Base para gerar código
  - DFM
  - PAS
  - Tabelas
- Onde queremos chegar
  - Classes destinos
- Como fazer

Geração de Código
Interpretação Banco de dados



Geração de Código
Interpretação Banco de dados

```
procedure TFORNECEDORDAO.Alterar(FORNECEDOR : TFORNECEDOR);
var
  FQuery : TSQLQuery;
begin

procedure TFORNECEDORDAO.Excluir(FORNECEDOR : TFORNECEDOR);
var
  FQuery : TSQLQuery;
begin
  FQuery := TSQLQuery.Create(Nil);
  try
    FQuery.SQLConnection := Conecta;
    FQuery.SQL.Clear;
    FQuery.SQL.Add('Delete From FORNECEDOR');
    FQuery.SQL.Add('Where FO_COD = :FO_COD');
    FQuery.ParambyName('FO_COD').AsInteger := FORNECEDOR.FFO_COD;
    FQuery.ExecSQL;
  finally
    FreeAndNil(FQuery);
  end;
end;

    FQuery.ParambyName('FO_BAIRRO').AsString := FORNECEDOR.FFO_BAIRRO;
    FQuery.ParambyName('FO_UF').AsString := FORNECEDOR.FFO_UF;
    FQuery.ExecSQL;
  finally
    FreeAndNil(FQuery);
  end;
end;
```

Geração de Código
Interpretação Banco de dados

- Método ListarTodos

```
140 function TFORNECEDORDAO.ListarTodos(FQuery : TSQLQuery) : TSQLQuery;
    begin
      FQuery.SQL.Clear;
      FQuery.SQLConnection := Conecta;
      FQuery.SQL.Add('Select * From FORNECEDOR');
      Result := FQuery;
    end;
```

Geração de Código
Select extraído de DFM ou PAS

- Método Selecionar
  - Diferença na passagem de parâmetros

```
function TFORNECEDORDAO.SelecionarFornecedor2(FQuery : TSQLQuery;
  Fo_Cod : Variant) : TSQLQuery;
begin
  FQuery.SQLConnection := Conecta;
  FQuery.SQL.Clear;
  FQuery.SQL.Add('Select *');
  FQuery.SQL.Add('From Fornecedor');
  FQuery.SQL.Add('Where Fo_Cod= :Fo_Cod');
  FQuery.ParamByName('FO_COD').AsString := Fo_Cod;
  Result := FQuery;
end;
```

Geração de Código
Select com mais de uma tabela

```
procedure TFrmCadProdutos.CarregaProdutoMemoria;
begin
  Query.SQL.Clear;
  Query.SQL.Add('Select Fo.Fo_Cod, Fo.Fo_Dsc, Fo.Fo_CNPJ');
  Query.SQL.Add('From Fornec_Prduto Fp, Fornecedor Fo');
  Query.SQL.Add('Where Fp.Fo_Cod = Fo.Fo_Cod');
  Query.SQL.Add('  And Fp.Pr_Cod = :Pr_Cod');
  Query.ParamByName('PR_COD').AsInteger := edCodPrduto.AsInteger;
```

```
type
  TFORNEC_PRDUTOFORNECEDORDAO = class
  private
    Conecta : TSQLConnection;
  public
    constructor Create(Conexao : TSQLConnection);
    function SelecionarFornec_Prduto14(FQuery : TSQLQuery;
                                       Pr_Cod : Variant) : TSQLQuery;
  end;

implementation

constructor TFORNEC_PRDUTOFORNECEDORDAO.Create(Conexao: TSQLConnection);
begin
  inherited Create;
  Conecta := Conexao;
end;

function TFORNEC_PRDUTOFORNECEDORDAO.SelecionarFornec_Prduto14(FQuery : TSQLQuery;
  Pr_Cod : Variant) : TSQLQuery;
begin
  FQuery.SQLConnection := Conecta;
  FQuery.SQL.Clear;
  FQuery.SQL.Add('Select Fo.Fo_Cod, Fo.Fo_Dsc, Fo.Fo_CNPJ');
  FQuery.SQL.Add('From Fornec_Prduto Fp, Fornecedor Fo');
  FQuery.SQL.Add('Where Fp.Fo_Cod = Fo.Fo_Cod ');
  FQuery.SQL.Add('  And Fp.Pr_Cod = :Pr_Cod');
  FQuery.ParamByName('PR_COD').AsString := Pr_Cod;
  Result := FQuery;
end;
```

Geração de Código
Análise de DFM e PAS

```
procedure TFrmCadFornec.ExcluirFornecedor;
begin
  if MessageDlg('Você tem certeza que deseja excluir este fornecedor?',
                                    mtConfirmation, [mbYes, mbNo], 0) = mrYes then
    begin
      Transacao.TransactionID:= 1;
      Transacao.IsolationLevel:=  xilReadCommitted;
      DM.Conecta.StartTransaction(Transacao);
      try
        with QueryExcluirFornecedor do
          begin
            SQL.Add('Delete From Fornecedor');
            SQL.Add('Where Fo_Cod = :Fo_Cod');
            ParamByName('FO_COD').AsInteger := edCodFornec.AsInteger;
            ExecSQL;
          end;
        DM.Conecta
      except
        On E: Exce
          begin
            DM.Con
            Messag
            Abort;
          end;
      end;

      LimpaCampos;
      HabBotoes(Fa
    end;
end;
```

```
object QueryBuscaFornec: TSQLQuery
    MaxBlobSize = -1
    Params = <>
    SQL.Strings = (
      'Select Max(Fo_Cod) + 1 Maximo From Fornecedor')
    Left = 206
    Top = 128
  end
  object QueryInsereFornec: TSQLQuery
    .
    .
    SQL.Strings = (
      'Insert Into Fornecedor'
      '(Fo_Cod,  Fo_Dsc, Fo_Dsc_Red, Fo_CNPJ, Fo_Inscr_Est,'
                                                  Fo_UF)'
                                                  :CNPJ, :Fo_Inscr_Est,'
                                                  irro, :Fo_UF)')
```

```
procedure TFORNECEDORDAO.Excluir(FORNECEDOR : TFORNECEDOR);
var
  FQuery : TSQLQuery;
begin
  FQuery := TSQLQuery.Create(Nil);
  try
    FQuery.SQLConnection := Conecta;
    FQuery.SQL.Clear;
    FQuery.SQL.Add('Delete From FORNECEDOR');
    FQuery.SQL.Add('Where FO_COD = :FO_COD');
    FQuery.ParambyName('FO_COD').AsInteger := FORNECEDOR.FFO_COD;
    FQuery.ExecSQL;
  finally
    FreeAndNil(FQuery);
  end;
end;
```

```
function TFORNECEDORDAO.SelecionarFornecedor4(FQuery : TSQLQuery) : TSQLQuery;
begin
  FQuery.SQLConnection := Conecta;
  FQuery.SQL.Clear;
  FQuery.SQL.Add('Select Max(Fo_Cod)+1 Maximo');
  FQuery.SQL.Add('From Fornecedor');
  Result := FQuery;
end;
```

Desenvolvimento
Geração de Código

FORNECEDORDAO

```
type
  TFORNECEDOR = class
  private
```

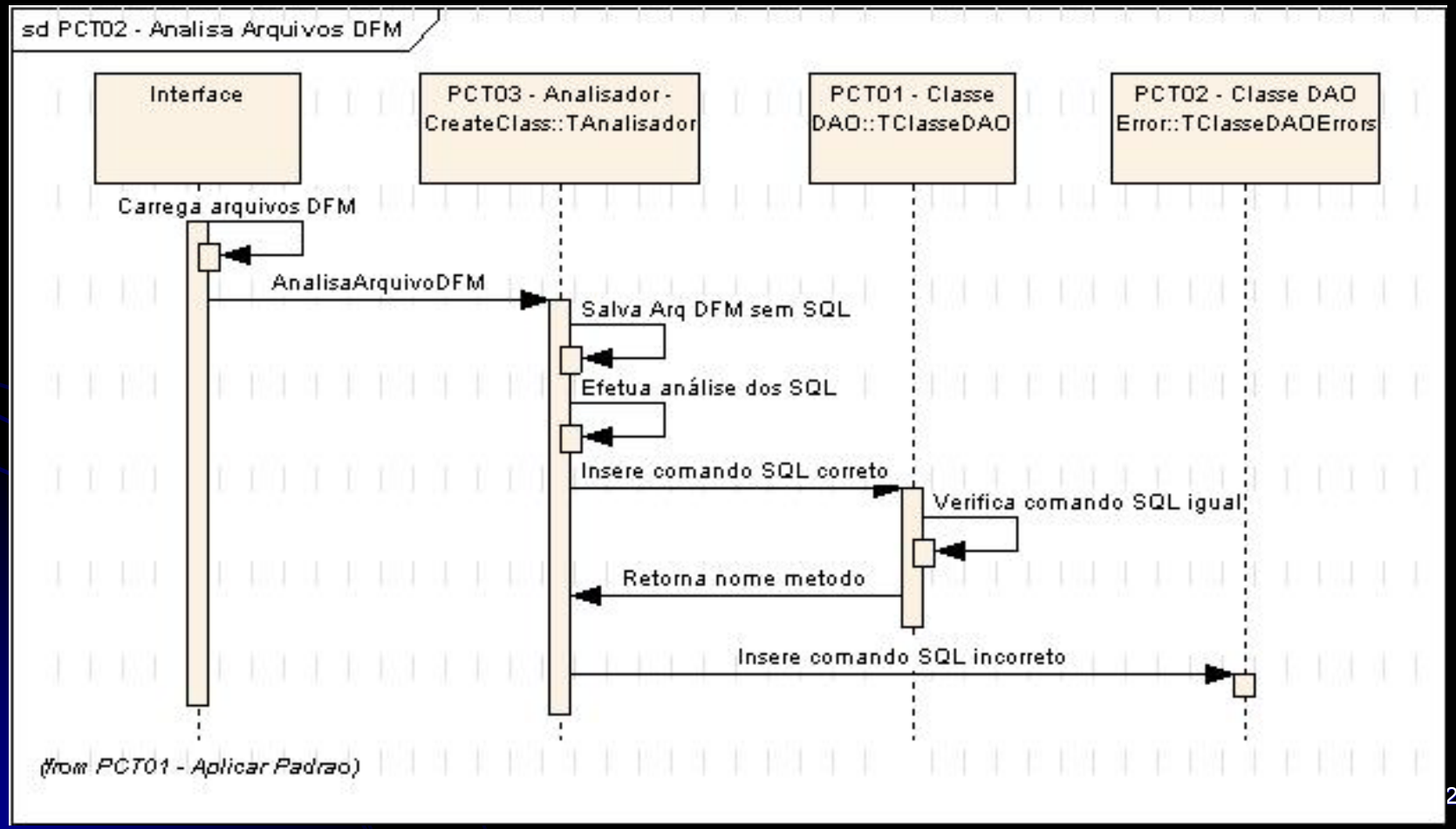
```
type
  TFORNECEDORDAO = class
  private
    Conecta : TSQLConnection;
  public
    constructor Create(Conexao : TSQLConnection);
    procedure Inserir(FORNECEDOR : TFORNECEDOR);
    procedure Alterar(FORNECEDOR : TFORNECEDOR);
    procedure Excluir(FORNECEDOR : TFORNECEDOR);
    function ListarTodos(FQuery : TSQLQuery) : TSQLQuery;
    procedure AlterarFornecedor1(Fornecedor : TFORNECEDOR);
    function SelecionarFornecedor2(FQuery : TSQLQuery; Fo_Cod : Variant) : TSQLQuery;
    function SelecionarFornecedor4(FQuery : TSQLQuery) : TSQLQuery;
  end;

implementation
```

```
    property FO_CEP : String Read FFO_CEP Write FFO_CEP ;
    property FO_CIDADE : String Read FFO_CIDADE Write FFO_CIDADE ;
    property FO_BAIRRO : String Read FFO_BAIRRO Write FFO_BAIRRO ;
    property FO_UF : String Read FFO_UF Write FFO_UF ;
  end;
```

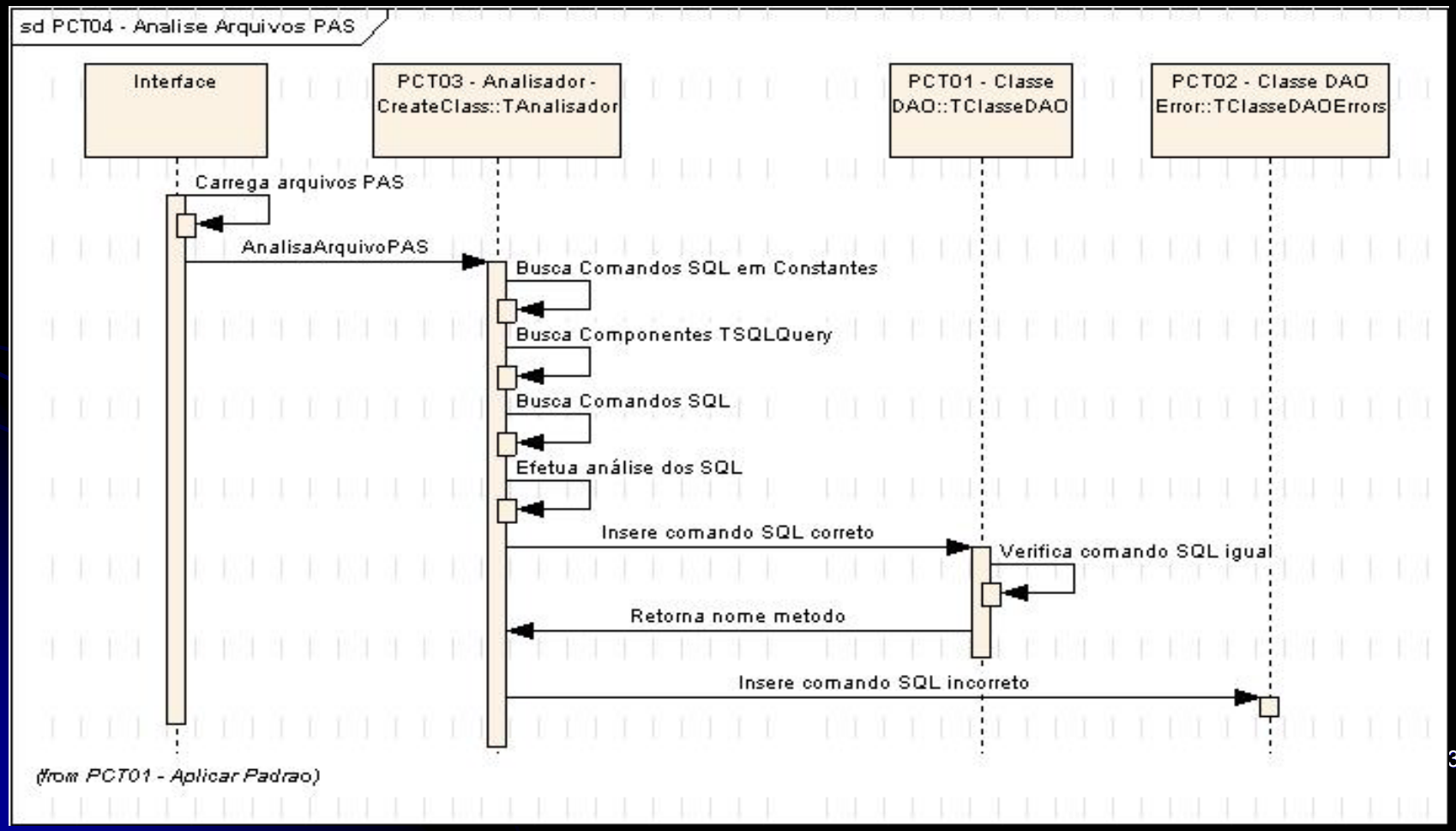
Desenvolvimento
Análise dos arquivos DFM

Diagrama de seqüência

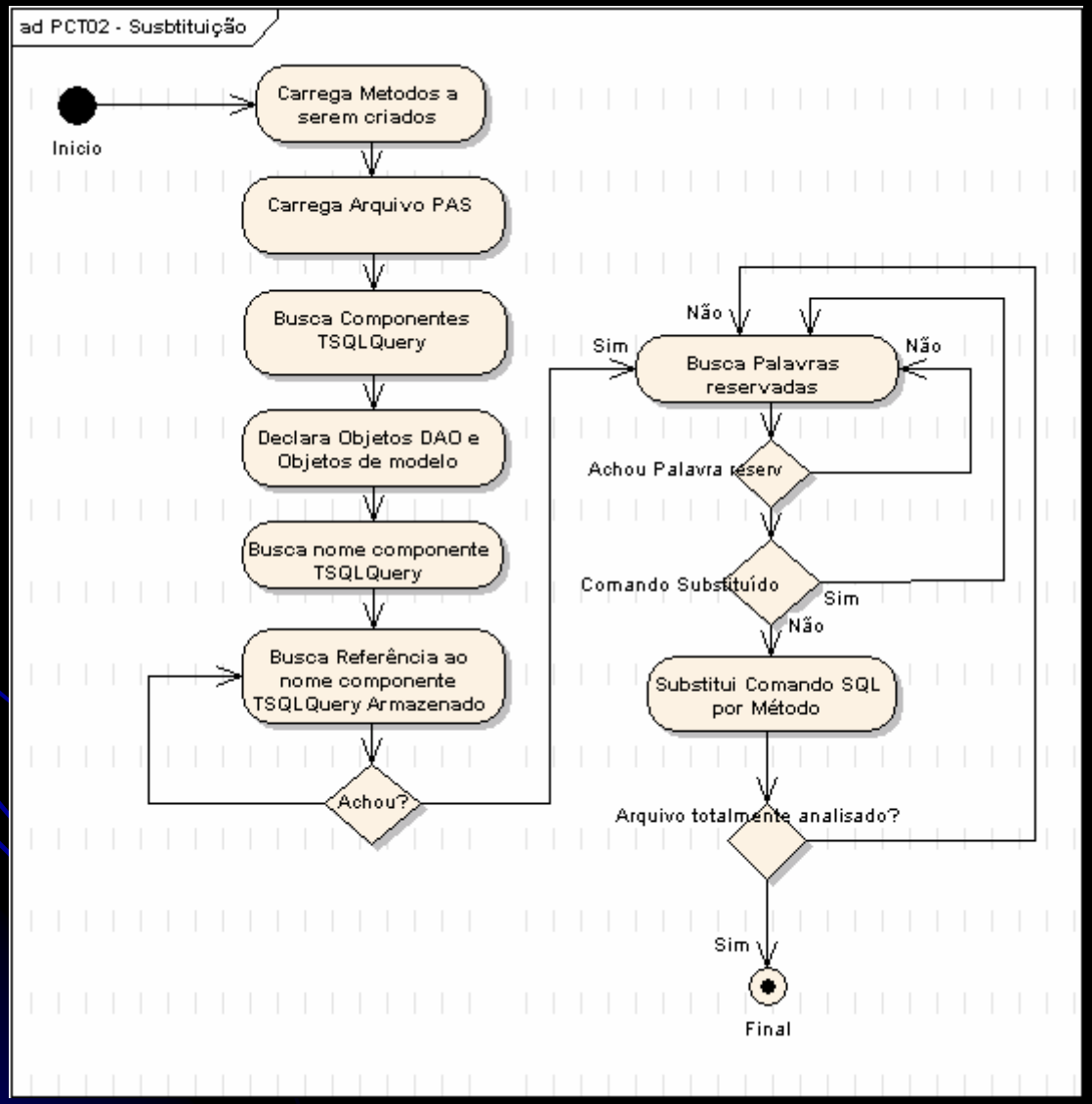


Desenvolvimento
Análise dos arquivos PAS

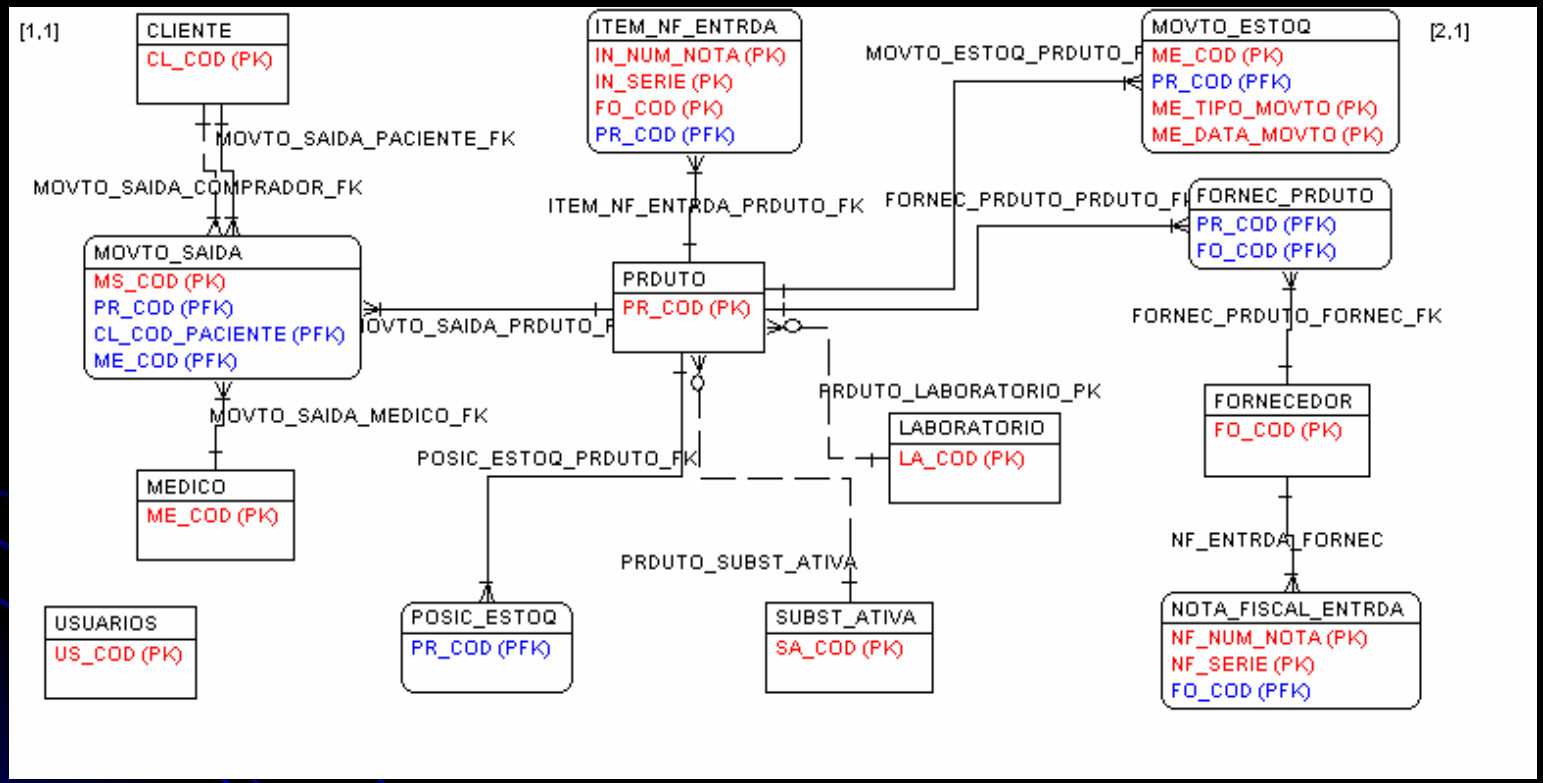
Diagrama de seqüência



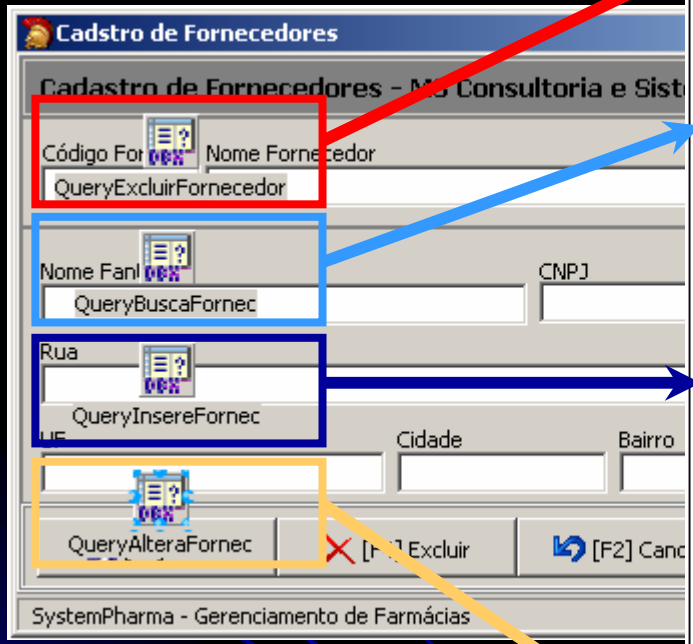
Desenvolvimento
Substituição dos comandos por métodos



Estudo de Caso



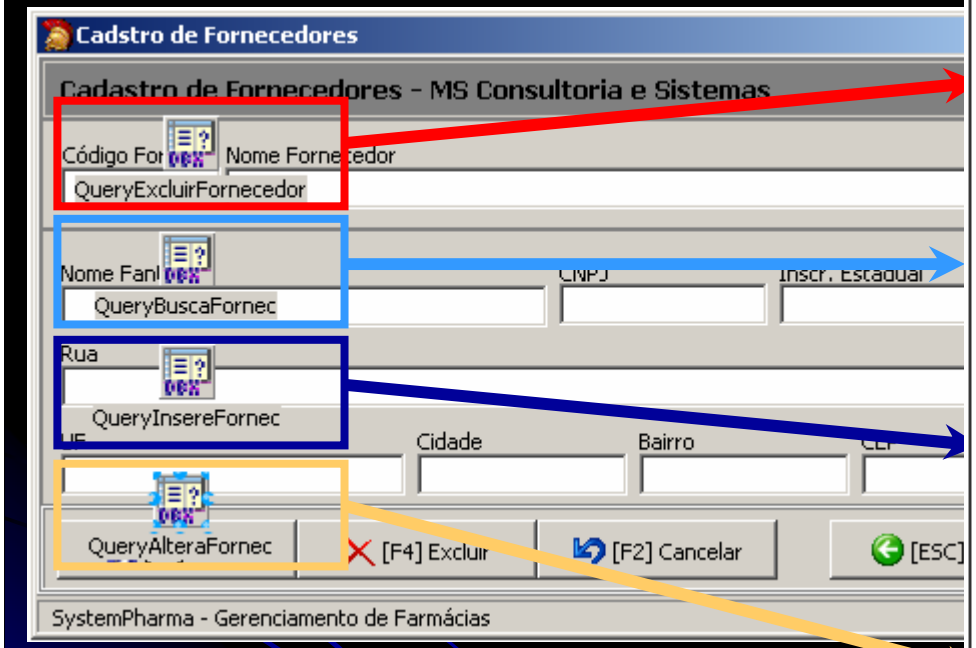
Estudo de Caso




```
inherited FrmCadFornec: TFrmCadFornec
  .
  object QueryExcluirFornecedor: TSQLQuery
    MaxBlobSize = -1
    Params = <>
    Left = 96
    Top = 128
  end
  object QueryBuscaFornec: TSQLQuery
    MaxBlobSize = -1
    .
    SQL.Strings = (
      'Select Max(Fo_Cod) + 1 Maximo From Fornecedor')
    Left = 206
    Top = 128
  end
  object QueryInsereFornec: TSQLQuery
    MaxBlobSize = -1
    .
    .
    SQL.Strings = (
      'Insert Into Fornecedor'
      '(Fo_Cod,  Fo_Dsc, Fo_Dsc_Red, Fo_CNPJ, Fo_Inscr_Est,'
      ' Fo_Rua, Fo_CEP, Fo_Cidade, Fo_Bairro, Fo_UF)'
      'Values'
      '(:Fo_Cod,  :Fo_Dsc, :Fo_Dsc_Red, :Fo_CNPJ, :Fo_Inscr_Est,'
      ' :Fo_Rua, :Fo_CEP, :Fo_Cidade, :Fo_Bairro, :Fo_UF)')
    Left = 320
    Top = 128
  end
  object QueryAlteraFornec: TSQLQuery
    MaxBlobSize = -1
    .
    .
    SQL.Strings = (
      'Update Fornecedor'
      'Set Fo_Dsc       = :Fo_Dsc,'
      '    Fo_Dsc_Red   = :Fo_Dsc_Red,'
      '    Fo_CNPJ      = :Fo_CNPJ,'
      '    Fo_Inscr_Est = :Fo_Inscr_Est,'
      '    Fo_Rua       = :Fo_Rua,'
      '    Fo_CEP       = :Fo_CEP,'
      '    Fo_Cidade    = :Fo_Cidade,'
      '    Fo_Bairro    = :Fo_Bairro,'
      '    Fo_UF        = :Fo_UF'
      'Where Fo_Cod     = :Fo_Cod')
    Left = 439
    Top = 128
  end
End
```

Estudo de Caso

DFM após análise da ferramenta




```
inherited FrmCadFornec: TFrmCadFornec
  ...
  object QueryExcluirFornecedor: TSQLQuery
    MaxBlobSize = -1
    Params = <>
    Left = 96
    Top = 128
  end
  object QueryBuscaFornec: TSQLQuery
    MaxBlobSize = -1
    Params = <>
    Left = 206
    Top = 128
  end
  object QueryInsereFornec: TSQLQuery
    MaxBlobSize = -1
    Params = <>
    Left = 320
    Top = 128
  end
  object QueryAlteraFornec: TSQLQuery
    MaxBlobSize = -1
    Params = <>
    Left = 439
    Top = 128
  end
End
```

Estudo de Caso

Arquivo fontes PAS antes de ser analisado pela ferramenta

```
procedure TFrmCadProdutos.edCodFornecExit(Sender: TObject);
var
  QueryAux : TSQLQuery;
begin
  inherited;
  QueryAux := TSQLQuery.Create(nil);
  try
    if edCodFornec.AsInteger > 0 then
      begin
        HabilitaBotoesFornec(True, True);
        btnSalvaFornec.SetFocus;
        QueryAux.SQLConnection := DM.Conecta;
        QueryAux.SQL.Add('Select * From Fornecedor');
        QueryAux.SQL.Add('Where Fo_Cod = :Fo_Cod');
        QueryAux.ParamByName('FO_COD').AsInteger := edCodFornec.AsInteger;
        QueryAux.Open;
        if not QueryAux.IsEmpty then
          begin

      QUERYAUX := FORNECEDORDAO.SELECIONARFORNECEDOR2(QUERYAUX,  edCodFornec.AsInteger);

          end
        else
          begin
            MessageDlg('Fornecedor não encontrado.', mtError, [mbOK], 0);
            edCodFornec.Clear;
            edNomeFornec.Clear;
            edCodFornec.SetFocus;
            HabilitaBotoesFornec(False, False);
          end;
      end
    else
      edNomeFornec.Clear;
  finally
    QueryAux.Free;
  end;
end;
```

Estudo de Caso
Classe gerada pela ferramenta

```
type
  TFORNECEDORDAO = class
  private
    Conecta : TSQLConnection;
  public
    constructor Create(Conexao : TSQLConnection);
    procedure Inserir(FORNECEDOR : TFORNECEDOR);
    procedure Alterar(FORNECEDOR : TFORNECEDOR);
    procedure Excluir(FORNECEDOR : TFORNECEDOR);
    function ListarTodos(FQuery : TSQLQuery) : TSQLQuery;
    procedure AlterarFornecedor1(Fornecedor : TFORNECEDOR);
    function SelecionarFornecedor2(FQuery : TSQLQuery; Fo_Cod : Variant) : TSQLQuery;
    function SelecionarFornecedor4(FQuery : TSQLQuery) : TSQLQuery;
  end;

implementation
```

Resultado e Discussões

- Todos os comandos SQL retirados,
- Foram criadas classes e métodos DAO,
- Substituição dos comandos por métodos DAO.

Comparação			
	Ferramenta para Aplicação do Padrão Data Access Object em Sistemas Desenvolvidos na Linguagem de Programação Delphi	**Ferramenta para Extração e Documentação de Rotinas de Projetos de Sistemas de Informação**	**Reutilização de Soluções com *Patterns* e *Frameworks* na Camada de Negócio**
Análise léxica, sintática e semântica	X	X	
Aplicação de Padrões	X		X
Análise dos arquivos fontes DFM e PAS	X	X	

Conclusão

- Os objetivos propostos neste trabalho foram alcançados.
  - A ferramenta consegue analisar os arquivos fontes DFM e PAS
  - Retira as expressões SQL dos mesmos
  - Cria classes DAO para cada tabela ou grupo de tabelas com as expressões retiradas dos arquivos fontes DFM e PAS
  - Substitui os comandos encontrados por métodos da classe DAO

Limitações e Extensões

- Limitações
  - ✓ Gera classe DAO apenas para banco de dados Interbase,
  - ✓ Não analisa cláusulas de *SubSelect*, *Union* e *Having*
  - ✓ Projetos com componentes de acesso a base TSQLQuery,
  - ✓ Busca apenas por componentes declarados no próprio Form,
  - ✓ Componentes não podem ter o mesmo nome
- Extensões
  - ✓ Implementar e usar um motor de *templates* para criação das classes DAO;
  - ✓ Possibilitar que os comandos SQL incorretos sejam alterados e validados;
  - ✓ Alterar a gramática para que seja possível analisar as cláusulas de *SubSelect*, *Union* e *Having*;
  - ✓ Possibilitar para que salve os comandos encontrados e os nomes dos métodos gerados em arquivo XML.