Testes de Unidade para Programação Orientada a Aspectos em Delphi

Eduardo Mafra

Prof. Marcel Hugo, M.Eng. - Orientador

Roteiro da Apresentação

- Introdução
- Testes e Testes de Unidade
- Programação Orientada a Aspectos (POA)
- DUnit
- Desenvolvimento da ferramenta
- Resultados e discussão
- Conclusão
- Extensões

Introdução

- Qualidade valorizada e conduz a uma evolução natural
- Tema central da Engenharia de Software
- Testes como fator chave para qualidade de software

Introdução

- Surgimento da Orientação a Objetos
- Surgimento da Programação Orientada a Aspectos (POA)
- Testes de Unidade para qualidade da POA

Introdução

- Objetivo geral do trabalho
  - Elaborar uma ferramenta que auxilie o desenvolvedor a gerar código na linguagem Object Pascal para testar os aspectos e as classes aspectadas

Testes

- Basicamente existem duas maneiras de construir testes:
  - Método da caixa branca
  - Método da caixa preta

Testes

- Caixa Branca
  - Todos caminhos lógicos são exercitados pelo menos uma vez
  - Exercita todas as decisões lógicas verdadeiras e falsas
  - Executa todos os laços em suas fronteiras
  - Exercita as estruturas de dados internas

Testes

- Caixa Preta
  - Funções incorretas ou ausentes
  - Erros de interface
  - Erros nas estruturas de dados
  - Erros de desempenho
  - Erros de inicialização ou término

Testes

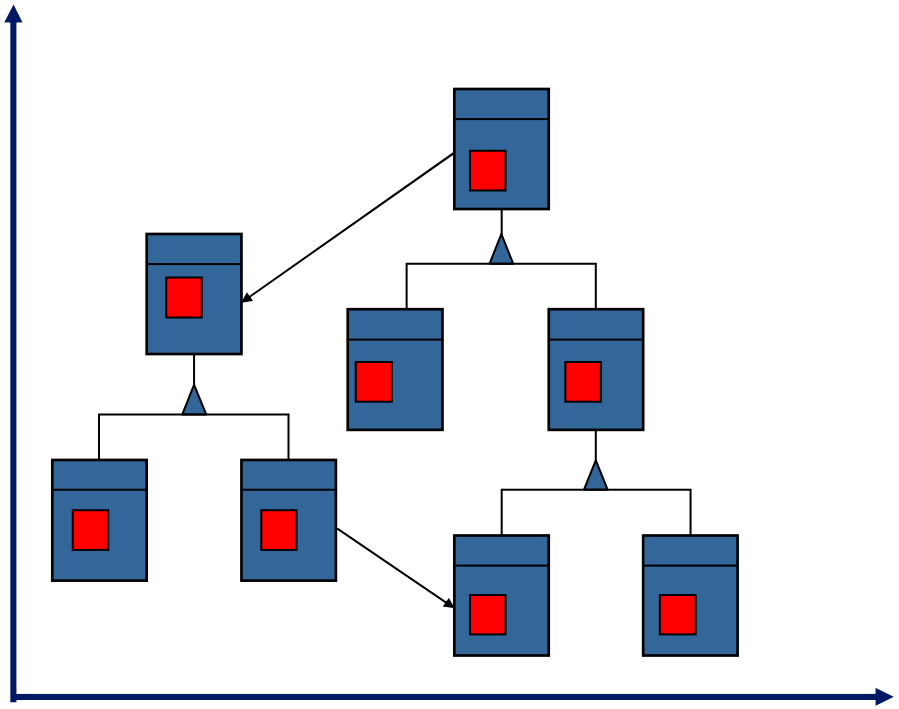
- Testes de Unidade
  - Baseado nos testes de Caixa Branca
  - Testa a menor unidade do projeto
  - Testes de unidade descobrem erros tais como:
    - Precedência matemática incorreta
    - Operações em modo misto
    - Inicialização incorreta
    - Erro de precisão

POA

- Separação de Interesses
- Interesses devem ser focados e trabalhados separadamente
- Reduz a complexidade no desenvolvimento
- Menor ocorrência de erros

POA

- Interesses transversais

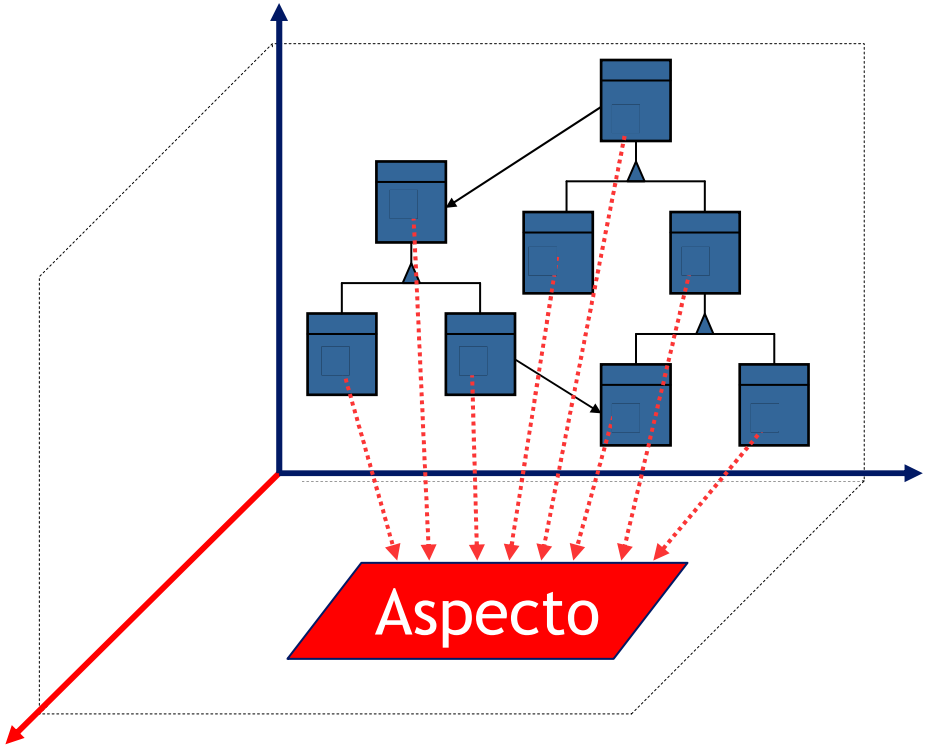


POA

- Encapsula os interesses transversais
- Reduz o acoplamento entre as partes
- Reusabilidade e manutenibilidade

POA

- Implementação encapsulada em uma única unidade: aspecto



POA

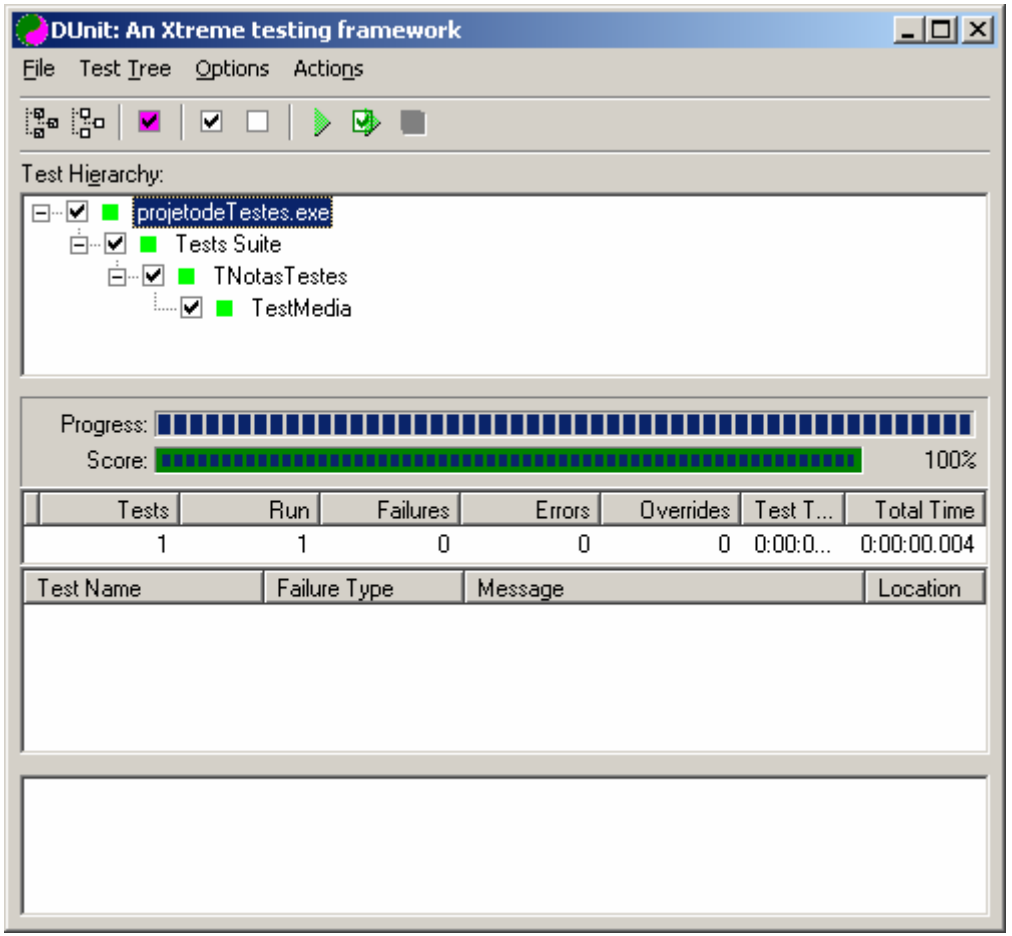
- Testes de Unidade para POA
  - Basicamente são testes nos aspectos
  - Testa-se os aspectos separadamente
  - Testes nas classes aspectadas
  - As classes aspectadas tornam-se mais complexas

DUnit

- *Framework* de classes para Delphi
- Suporte para testes de unidade para Delphi
- Não automatiza testes
- Baseado no JUnit

DUnit

```
CheckEquals(10, FNotas.Media);
```



AOPDelphi – Principais Requisitos

- Deve prover suporte a testes de unidade para POA em Delphi
- Programas de testes devem ser escritos em linguagem própria
- Integrar a ferramenta criada no ambiente desenvolvido por Oliveira(2006)

AOPDelphi – Principais Requisitos

- Deve receber como entrada uma ou mais rotinas de testes
- Gerar código em Object Pascal
- Possuir um ambiente para codificação dos testes

Especificação da Linguagem de Testes

- BNF

```
<letra>::= [a-zA-Z]
<digito>::= [0-9]
<programa>::= Test IDENTIFICADOR ; <corpo> End.
<corpo>::= <uses> <declaration> <setup> <teardown> <teste>
<uses>::= Uses <lista_uses>; | ε
<lista_uses>::= IDENTIFICADOR <lista_uses2>
<lista_uses2>::= , <lista_uses> | ε
<declaration>::= Declaration <declaration1> End; | ε
<declaration1>::= <declaration2> : IDENTIFICADOR ; <declaration4>
<declaration2>::= IDENTIFICADOR <declaration3>
<declaration3>::= , <declaration2> | ε
<declaration4>::= <declaration1> | ε
<setup>::= <codigo_setup> ; | ε
<codigo_setup>::= Setup (<codigo>)* EndSetup
<teardown>::= <codigo_teardown> ; | ε
<codigo_teardown>::= Teardown (<codigo>)* EndTeardown
<teste>::= <teste2> | ε
<teste2>::= Test IDENTIFICADOR ; <codigo_teste> ; <lista_teste>
<codigo_teste>::= BeginTest (<codigo>)* EndTest
<lista_teste>::= <teste2> | ε
<codigo>::= CODIGO
```

Especificação da Ferramenta

- Técnicas e ferramentas utilizadas
  - *Unified Modeling Language* (UML)
  - Enterprise Architect

Diagrama de Casos de Uso

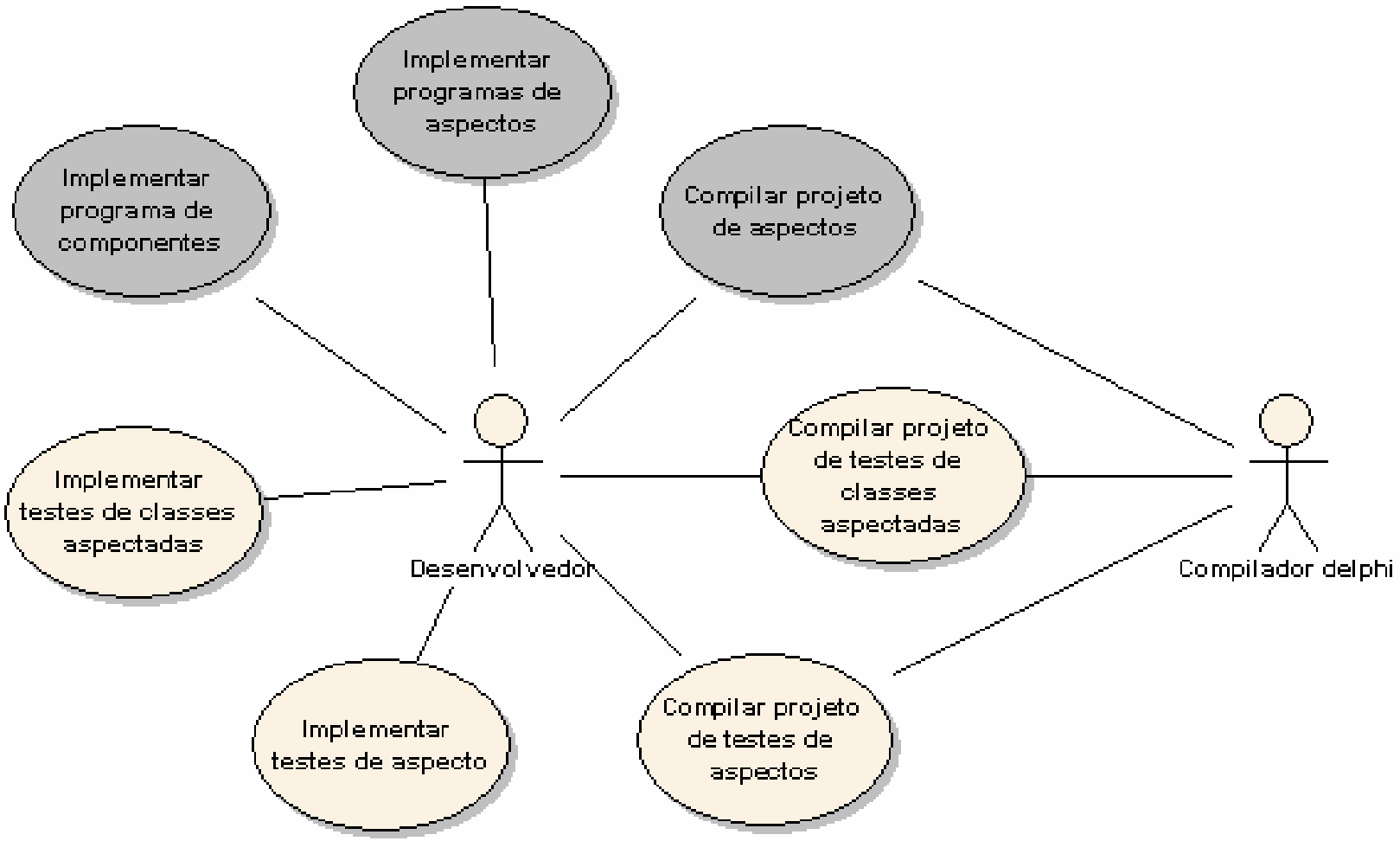


Diagrama de Atividades

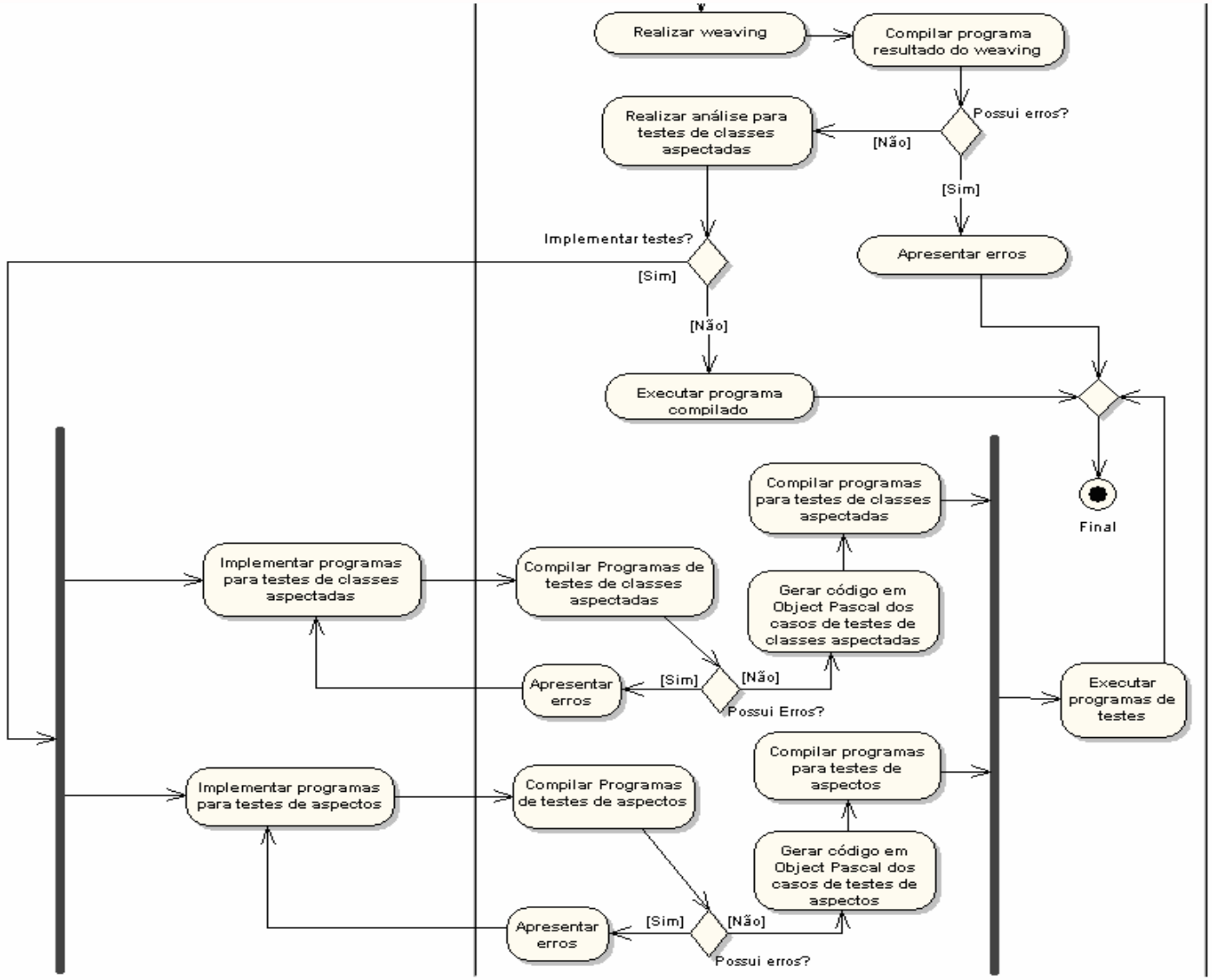
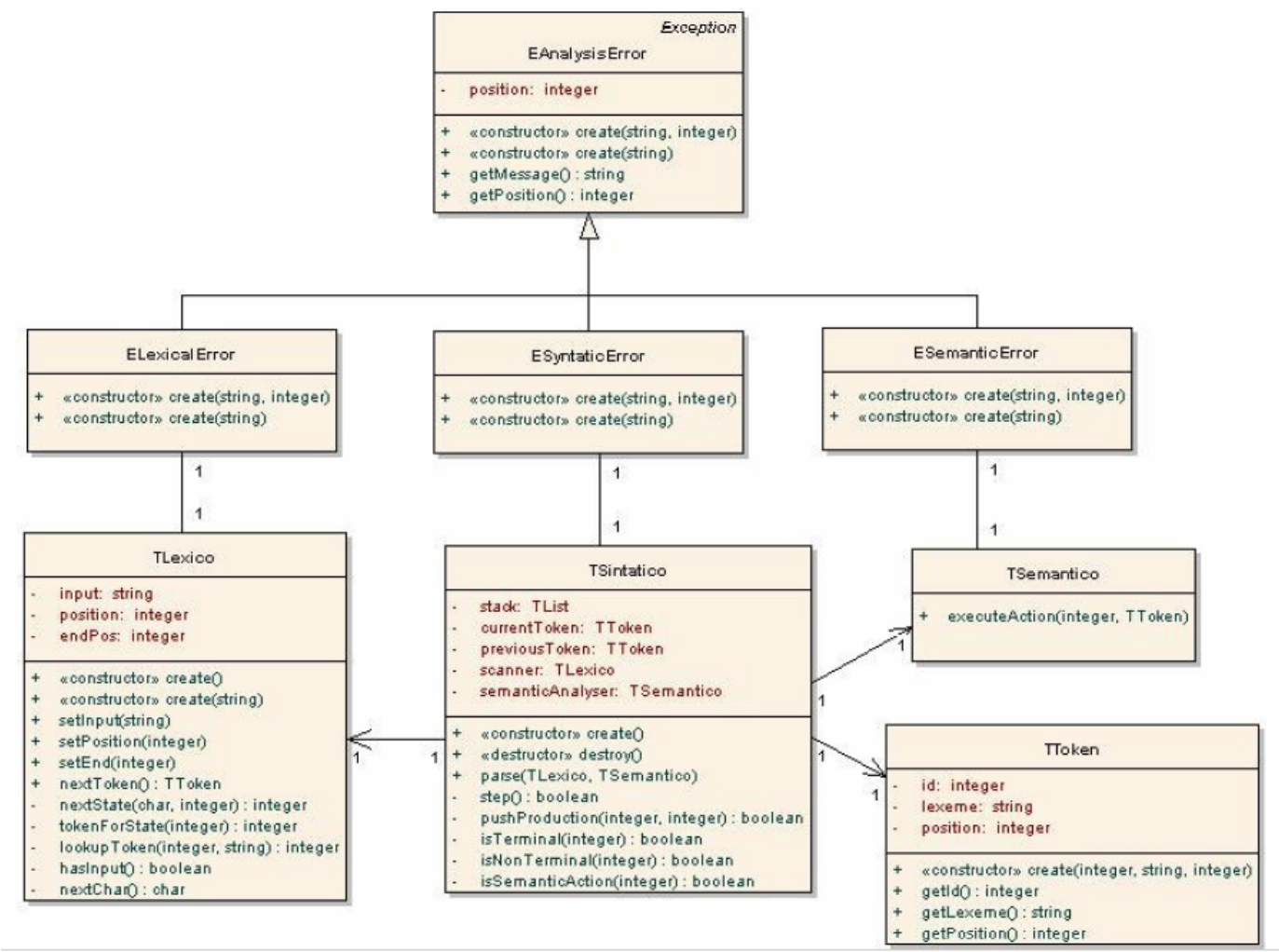
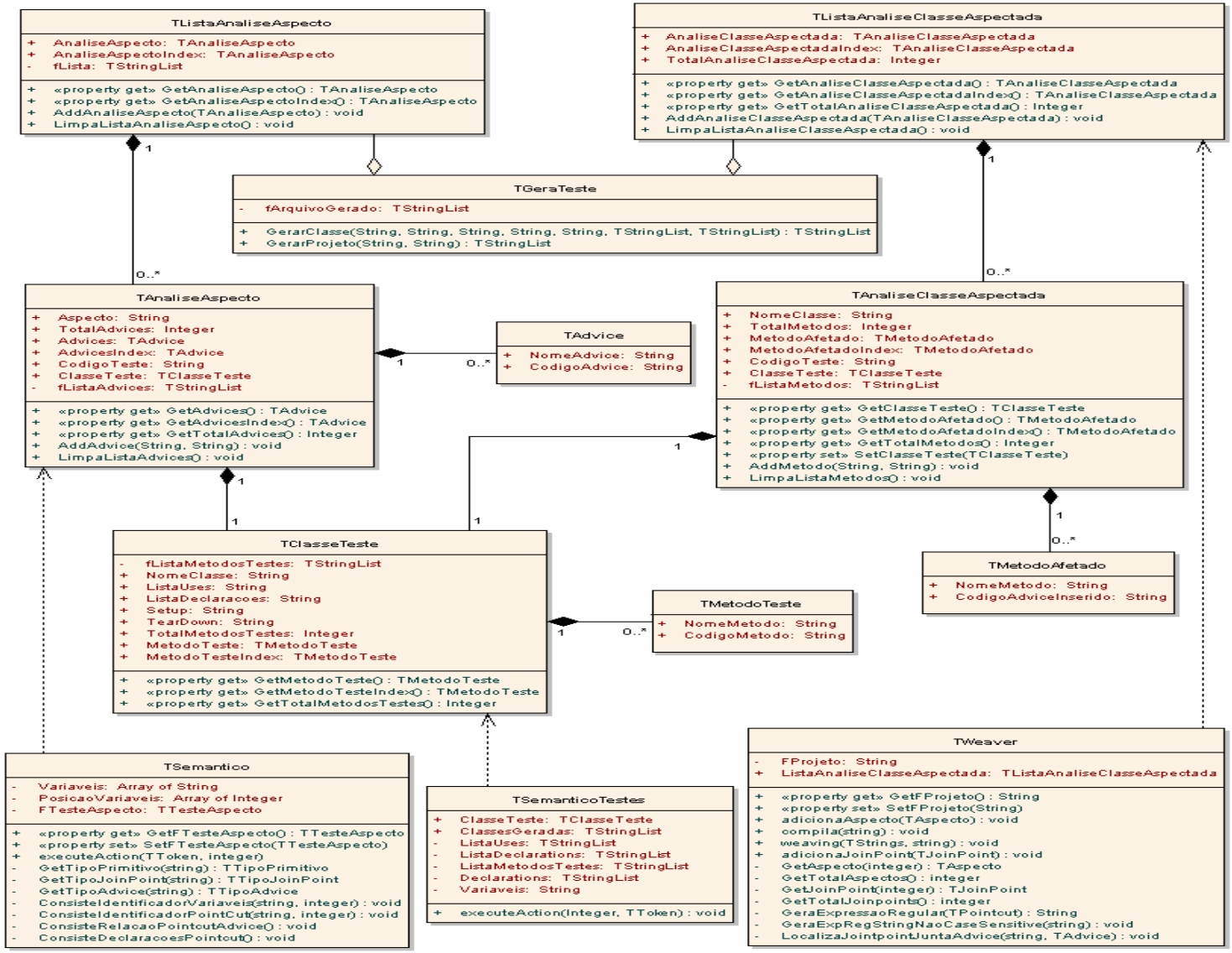


Diagrama de Classes



Classes geradas pelo GALS

Diagrama de Classes

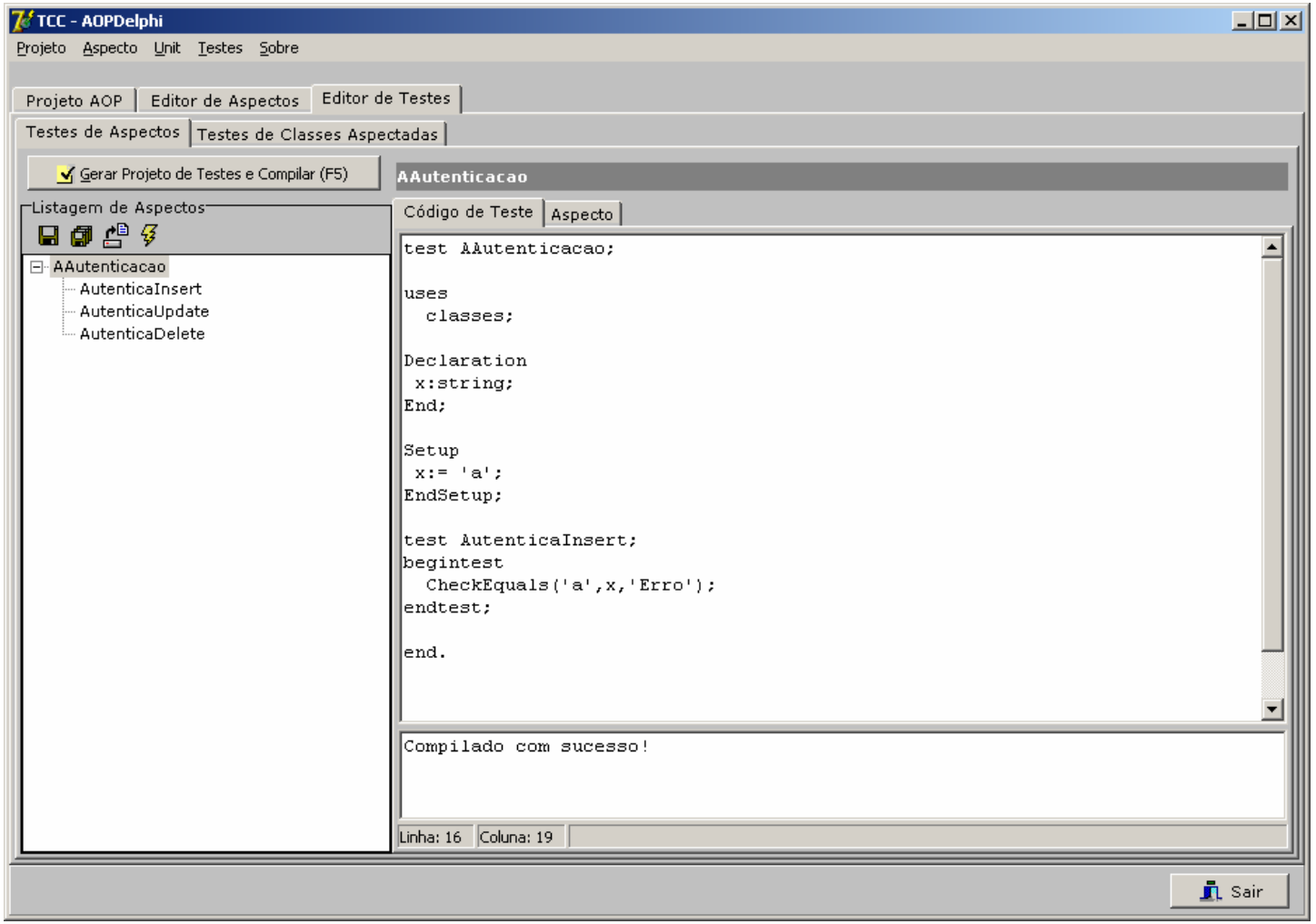


Implementação

- Ferramentas utilizadas
  - Linguagem Object Pascal – Ambiente Borland Delphi 7
  - GALS

Operacionalidade

- Apresentação



Operacionalidade

- Aspecto de Autenticação

```
AAutenticacao = Aspect
Implementation
 AutenticaInsert : Pointcut = (* *.*SQLInsert(*));
 AutenticaUpdate : Pointcut = (* *.*SQLUpdate(*));
 AutenticaDelete : Pointcut = (* *.*SQLDelete(*));

 Advice AutenticaInsert: Before;
  begin
    With DMTaskMan.qryAutentica do
    begin
      Close;
      Sql.Clear;
      Sql.Add('Select count(*) from DIREITO_USUARIO');
      Sql.Add('Where IDUSUARIO = :IDUSUARIO and OPERACAO = :OPERACAO');
      Sql.Add(' and TABELA = :TABELA');
      ParamByName('IDUSUARIO').AsInteger := giCodUsuario;
      ParamByName('OPERACAO').AsInteger := 1;  //1=Insert
      ParamByName('TABELA').AsString := Self.NomeTabela;
      Open;
      if  Fields[0].AsInteger = 0 then
          Raise Exception.Create('Você não tem permissão nesta atividade.');
    end;
  EndAdvice;
```

Operacionalidade

- Teste do Aspecto de Autenticação

```
Test AAutenticacao;

uses
  Classes, SysUtils, UDataModule;

Setup
  DMTaskMan:= TDMTaskMan.Create(Nil);
EndSetup;

TearDown
  DMTaskMan.Free;
EndTearDown;

Test AutenticaInsert;
Begintest
    With DMTaskMan.qryAutentica do
    begin
      Close;
      Sql.Clear;
      Sql.Add('Select count(*) from DIREITO_USUARIO');
      Sql.Add('Where IDUSUARIO = :IDUSUARIO and OPERACAO = :OPERACAO');
      Sql.Add(' and TABELA = :TABELA');
      ParamByName('IDUSUARIO').AsInteger := 1;
      ParamByName('OPERACAO').AsInteger := 1;  //1=Insert
      ParamByName('TABELA').AsString := 'CLIENTE';
      Open;
      if  Fields[0].AsInteger = 0 then
          Raise Exception.Create('Você não tem permissão para essa atividade. ' + Chr(13) + 'Entre
                                  em contato com o seu superior.');
    end;
Endtest;
```

Operacionalidade

- Classe Object Pascal (1/2)

```
Unit AAutenticacao;
Interface
Uses
  TestFramework, Classes,SysUtils,UDataModule;
Type
  TAAutenticacao = Class(TTestCase)
  Private

  Protected
    Procedure SetUp; Override;
    Procedure TearDown; Override;
  Published
    Procedure AutenticaInsert;
    Procedure AutenticaUpdate;
    Procedure AutenticaDelete;
  End;
Implementation

Procedure TAAutenticacao.SetUp;
Begin
   DMTaskMan:= TDMTaskMan.Create(Nil);
  Inherited;
End;
```

Operacionalidade

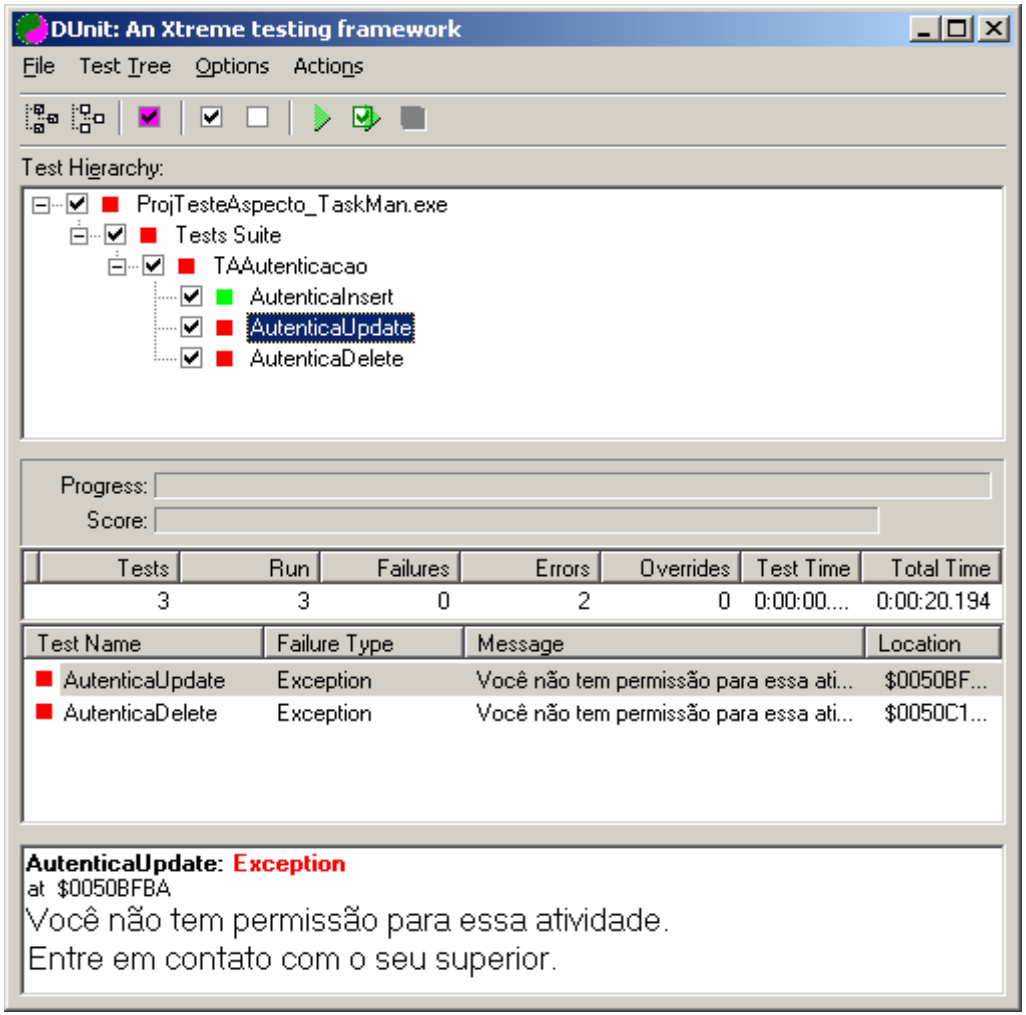
- Classe Object Pascal (2/2)

```
Procedure TAAutenticacao.TearDown;
Begin
  DMTaskMan.Free;
  Inherited;
End;

Procedure TAAutenticacao.AutenticaInsert;
Begin
    With DMTaskMan.qryAutentica do
    begin
      Close;
      Sql.Clear;
      Sql.Add('Select count(*) from DIREITO_USUARIO');
      Sql.Add('Where IDUSUARIO = :IDUSUARIO and OPERACAO = :OPERACAO');
      Sql.Add(' and TABELA = :TABELA');
      ParamByName('IDUSUARIO').AsInteger := 1;
      ParamByName('OPERACAO').AsInteger := 1;  //1=Insert, 2=Update, 3=Delete
      ParamByName('TABELA').AsString := 'CLIENTE';
      Open;
      if  Fields[0].AsInteger = 0 then
          Raise Exception.Create('Você não tem permissão para essa atividade. '
                          + Chr(13) + 'Entre em contato com o seu superior.');
    end;
End;
```

Operacionalidade

- Resultado exibido pelo DUnit



Resultados e Discussão

- Apresentou bons resultados no sentido de prover suporte a testes unitários para POA em Delphi
- Limitação da linguagem de testes
- Referência bibliográfica sobre testes unitários em POA precária

Conclusão

- Propicia uma visão geral sobre testes de software e Programação Orientada a Aspectos
- Importância do surgimento de ferramentas para prover qualidade para implementações com POA
- Importância do GALS
- Objetivos foram atingidos

Extensões

- Integrar a ferramenta à IDE do Delphi
- Melhorias na linguagem de testes
- Integração dos testes de aspectos com testes de classes