Protótipo de um *Weaver* para Programação Orientada a Aspectos em Delphi

Edmar Soares de Oliveira

Prof. Marcel Hugo, M.Eng. - Orientador

Roteiro da Apresentação

- Introdução
- Objetivos do trabalho
- Programação Orientada a Aspectos (POA)
- AOPDelphi
- Resultados e discussão
- Conclusão
- Extensões

Introdução

- Evolução da Engenharia de Software buscando qualidade no desenvolvimento
- Surgimento da Orientação a Objetos (OO)
- Crescimento dos sistemas e aumento de sua complexidade

Introdução

- Surgimento da Programação Orientada a Aspectos
- Falta de ferramentas de suporte a POA para Delphi
- AOPDelphi

Objetivos do Trabalho

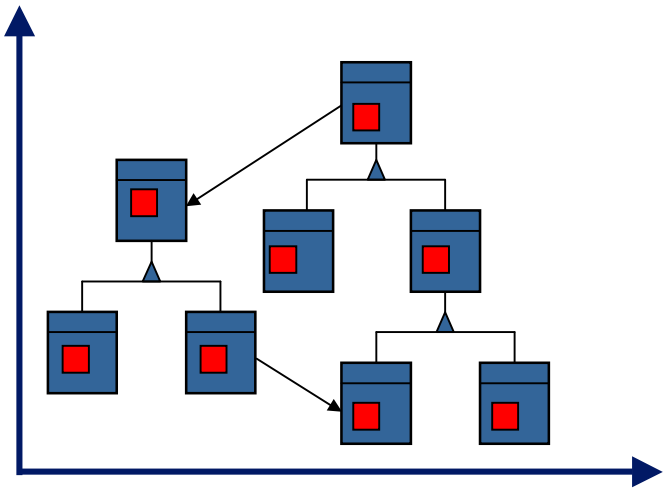
- Desenvolver um *weaver* para prover suporte a POA na linguagem Delphi
  - Criar uma linguagem para programação dos aspectos
  - Realizar análises léxica e sintática dos programas de aspectos
  - Permitir que o *weaver* receba como entrada arquivos de projeto do Delphi e programas de aspecto na linguagem criada
  - Disponibilizar um ambiente para codificação dos programas de aspectos

POA – Separação de Interesses

- *Separation of Concerns*
- Todo software lida com diferentes interesses
- Interesses devem ser focados e trabalhados separadamente
- Reduz a complexidade no desenvolvimento
- Menor probabilidade de ocorrência de erros

POA – Conceitos Básicos

- Interesses Transversais *(Crosscutting Concerns)*
  - Entrelaçamento de código *(code tangling)*
  - Espalhamento de código *(spreading code)*



POA – Conceitos Básicos

- Tecnologia para melhorar a separação de interesses
- Encapsula interesses transversais em módulos fisicamente separados do restante do código
- Reduz o acoplamento entre as partes
- Reusabilidade e manutenibilidade

POA – Conceitos Básicos

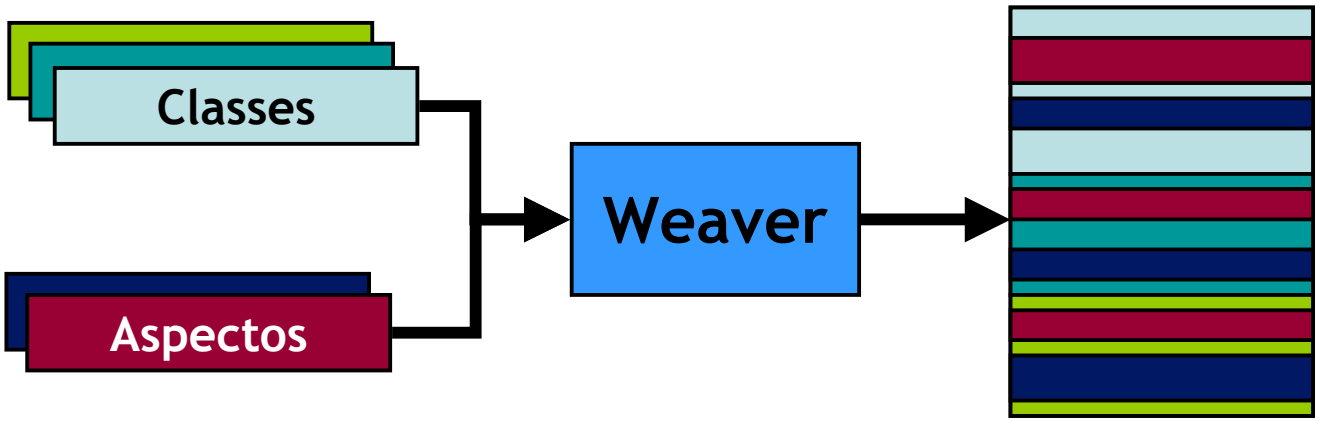
"O objetivo final da POA é reduzir a complexidade do desenvolvimento de software, permitindo que as diversas preocupações possam ser componentizadas, independentemente de sua natureza, funcional ou não funcional."

Resende e Silva (2005, p. 9)

POA – Elementos

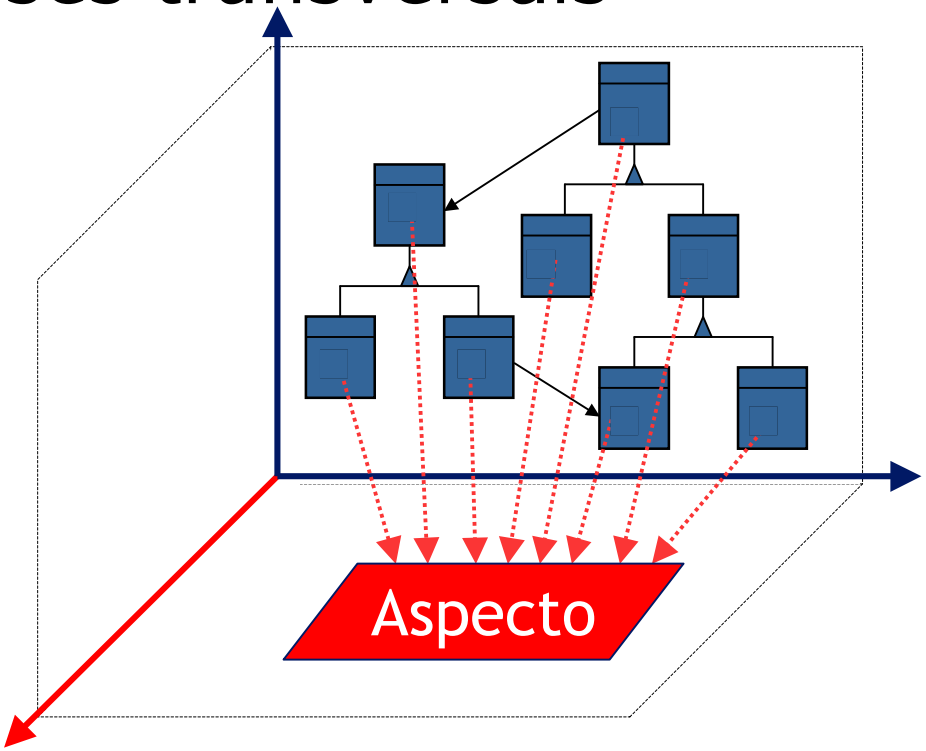
- Linguagem de componentes
- Linguagem de aspectos
- Programa de componentes
- Programa de aspectos
- *Weaver*

POA – *Weaver*



POA – Aspecto

- Unidade de modularidade para interesses transversais
- Permite a separação e composição dos interesses transversais



POA

- *Joinpoint* (pontos de junção) – programa de componente
- *Pointcut* (conjuntos de junção) – programa de aspecto
- *Advice* – programa de aspecto

AOPDelphi – Principais Requisitos

- Prover suporte a POA para linguagem Delphi
- Programas de aspectos devem ser escritos em linguagem própria

AOPDelphi – Principais Requisitos

- O *weaver* deve receber como entrada um ou mais programas de aspectos e um arquivo de projeto do Delphi
  - Projeto Delphi não pode conter erros de compilação
  - Projeto Delphi deve ser implementado utilizando orientação a objetos
  - Realizar análises léxica e sintática dos programas de aspectos

AOPDelphi – Principais Requisitos

- Programas gerados pelo *weaver* devem mesclar as funcionalidades dos programas fornecidos como entrada
- Possuir um ambiente para implementação dos programas de aspectos

Especificação da Linguagem de Aspectos

- Técnicas e ferramentas utilizadas
  - Definições regulares
  - *Backus-Naur Form* (BNF)
  - GALS

Especificação da Linguagem de Aspectos

- Definições regulares

```
IDENTIFICADOR : (<letra> | "_") (<letra> | <digito> | "_")*

CURINGA_IDENTIFICADOR: ("*" (<letra> | <digito> | "_")+)
                         | ((<letra> |"_") (<letra> | <digito> | "_")* "*"
                           (<letra> | <digito> | "_")* (<letra> |"_" | <digito>))
                         | ((<letra> |"_") (<letra> | <digito> | "_")* "*")
CODIGO : .
```

Especificação da Linguagem de Aspectos

- BNF

```
<AOPDelphi> ::= <define_aspecto> <declara_variaveis>
                <implementation>
                End.

<define_aspecto> ::= IDENTIFICADOR = Aspect

<letra> ::= [a-zA-Z]

<digito> ::= [0-9]

<declara_variaveis> ::= Var <lista_variaveis>
                          | ε

<lista_variaveis> ::= <lista_identificador> : <tipos_primitivos>;
                       <lista_variaveis_linha>

<tipos_primitivos>::=  string   |  integer  |  real   |   double
                     | extended | currency  | boolean | variant
                     | word     | byte      | char
```

Especificação da Linguagem de Aspectos

- BNF

```
<lista_identificador> ::= IDENTIFICADOR <lista_identificador_linha>

<lista_identificador_linha> ::= , IDENTIFICADOR <lista_identificador_linha> | ε

<lista_variaveis_linha> ::= <lista_variaveis>  | ε

<implementation> ::= Implementation <declara_pointcuts> <declara_advices> | ε

<declara_pointcuts>::= IDENTIFICADOR : Pointcut = ( <joinpoint> ) ;
                       <lista_pointcut>
<joinpoint>::= <excecao> | <metodo>

<excecao>::= Exception <indentificador_excecao>

<metodo>::= <tipo_metodo> <id_classe_metodo> . <id_classe_metodo> ( <parametros> )

<tipo_metodo>::= Procedure  | Function | Constructor
                | Destructor | <curinga>

<id_classe_metodo>::= IDENTIFICADOR  |  <curinga>   | CURINGA_IDENTIFICADOR
```

Especificação da Linguagem de Aspectos

- BNF

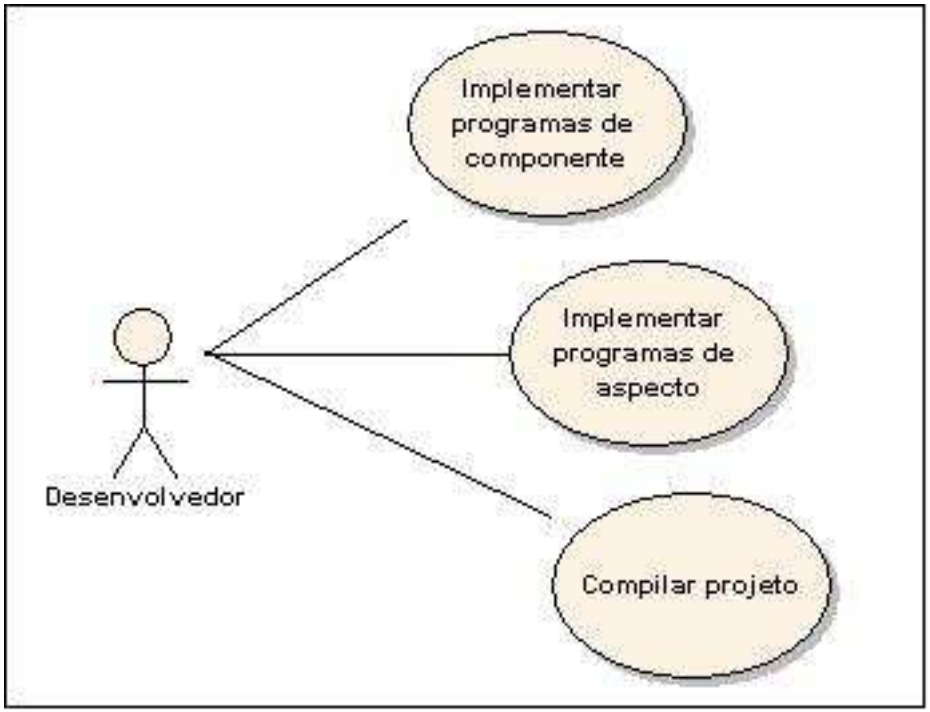
```
<parametros>::= <tipos_primitivos> <parametros_linha>  |  <curinga>

<parametros_linha>::= , <tipos_primitivos> <parametros_linha>  |  ε

<lista_pointcut>::= <declara_pointcuts>  |  ε

<declara_advices>::= Advice IDENTIFICADOR : <tipo_advice> ;
                     <codigo_advice> ;
                     <lista_advice>

<tipo_advice>::= Before  |  After

<lista_advice>::= <declara_advices>  |  ε

<indentificador_excecao>::= IDENTIFICADOR | <curinga>

<curinga>  ::= *

<codigo_advice> ::= Begin (<codigo>)* EndAdvice

<codigo> ::= CODIGO
```

Especificação da Ferramenta

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

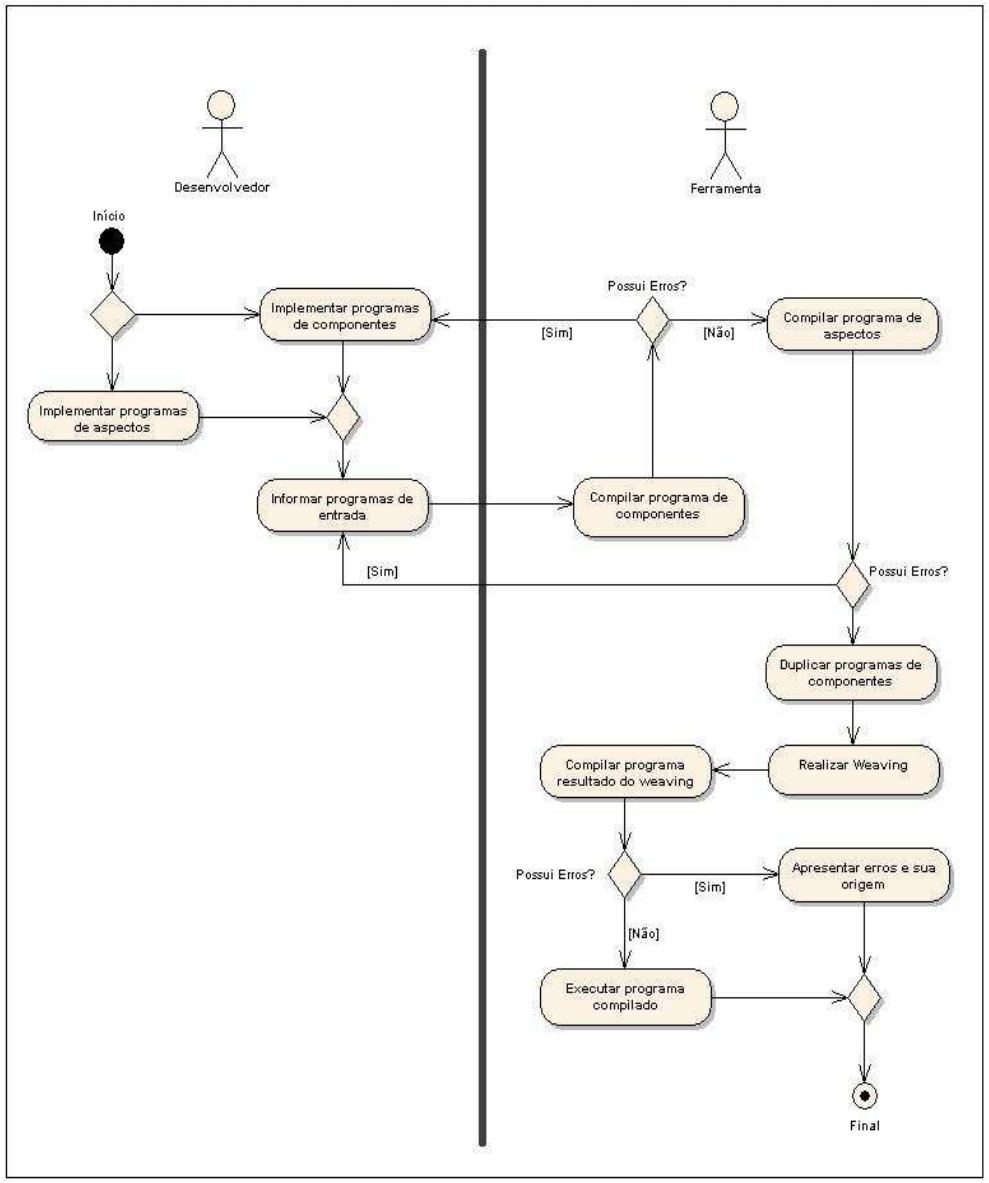
Especificação da Ferramenta

- Diagrama de Casos de Uso



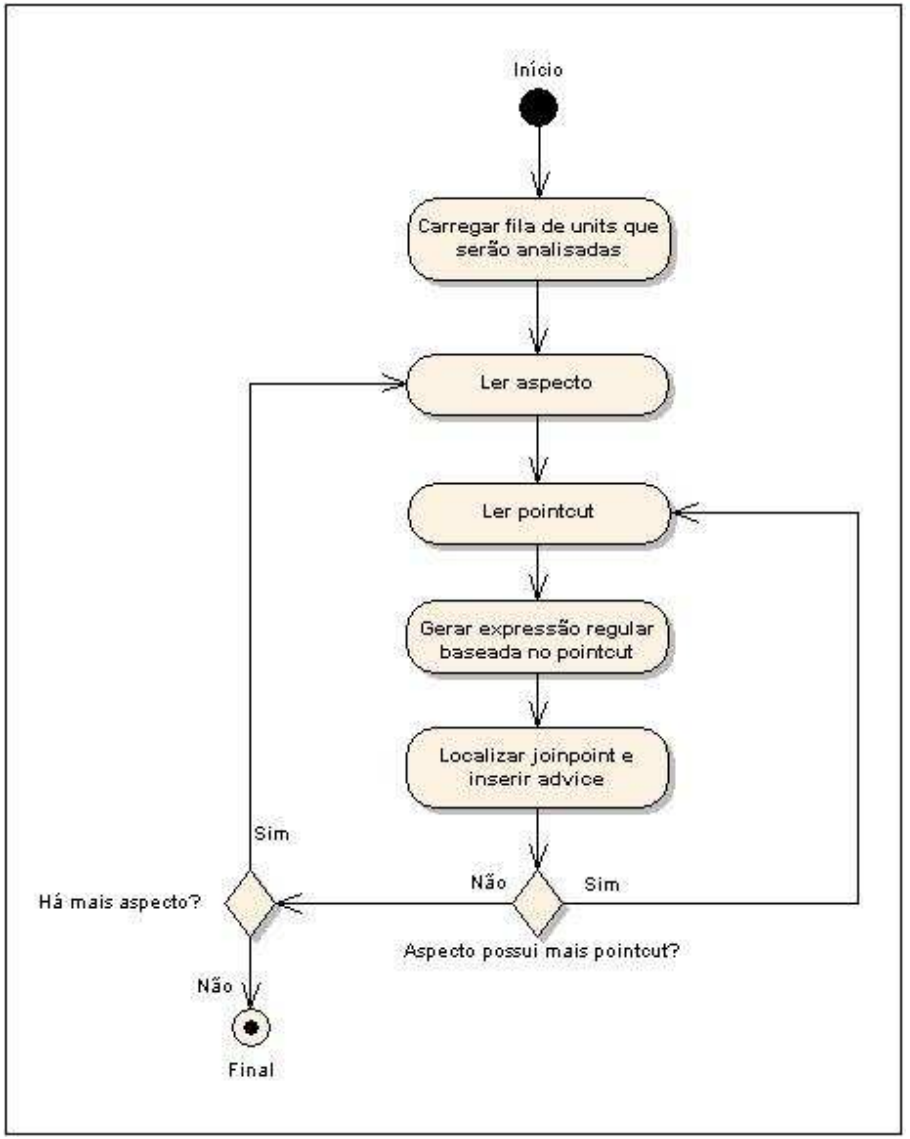
Especificação da Ferramenta

- Diagrama de Atividades



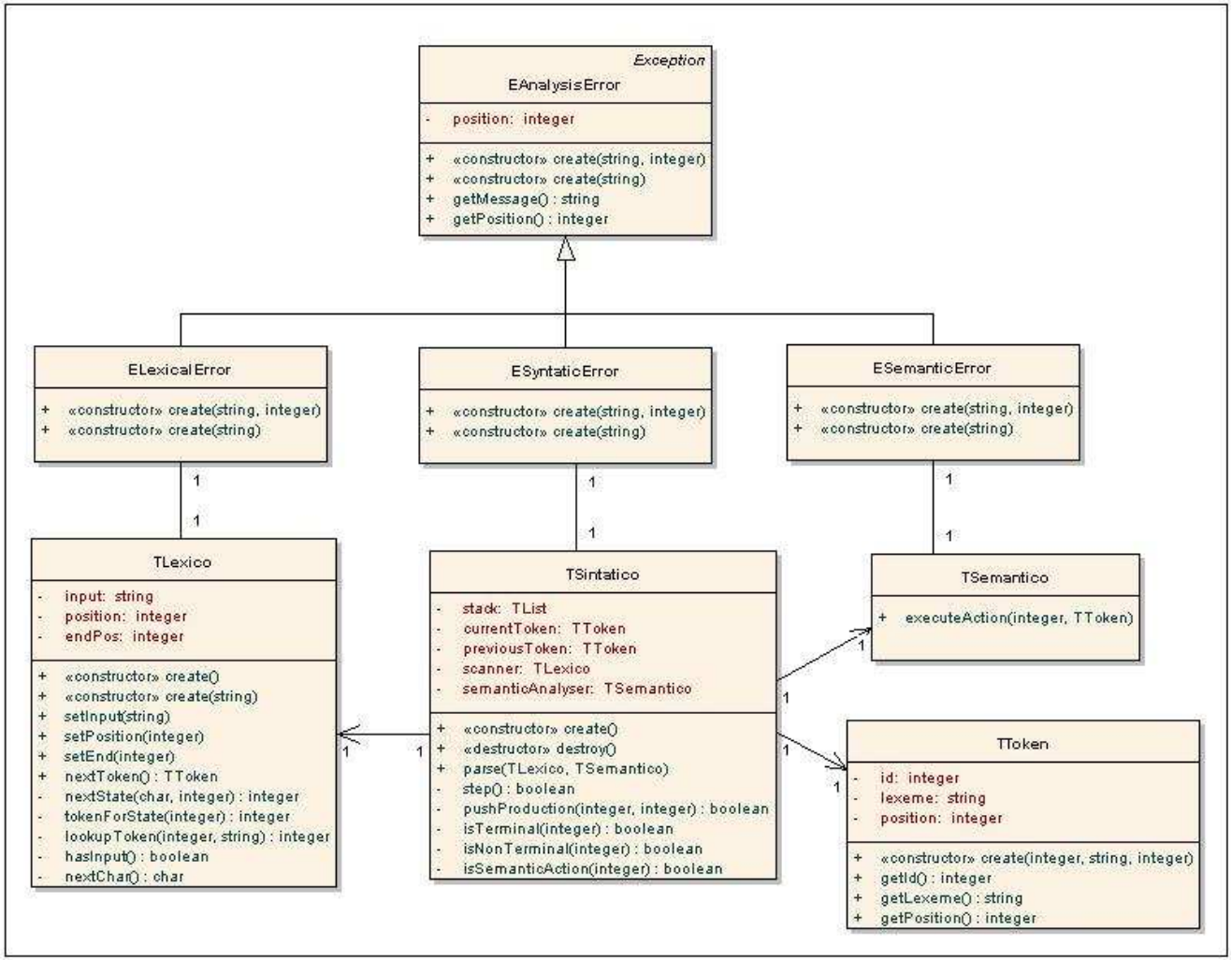
Especificação da Ferramenta

- Diagrama de Atividades
  - Atividade *Weaving*



Especificação da Ferramenta

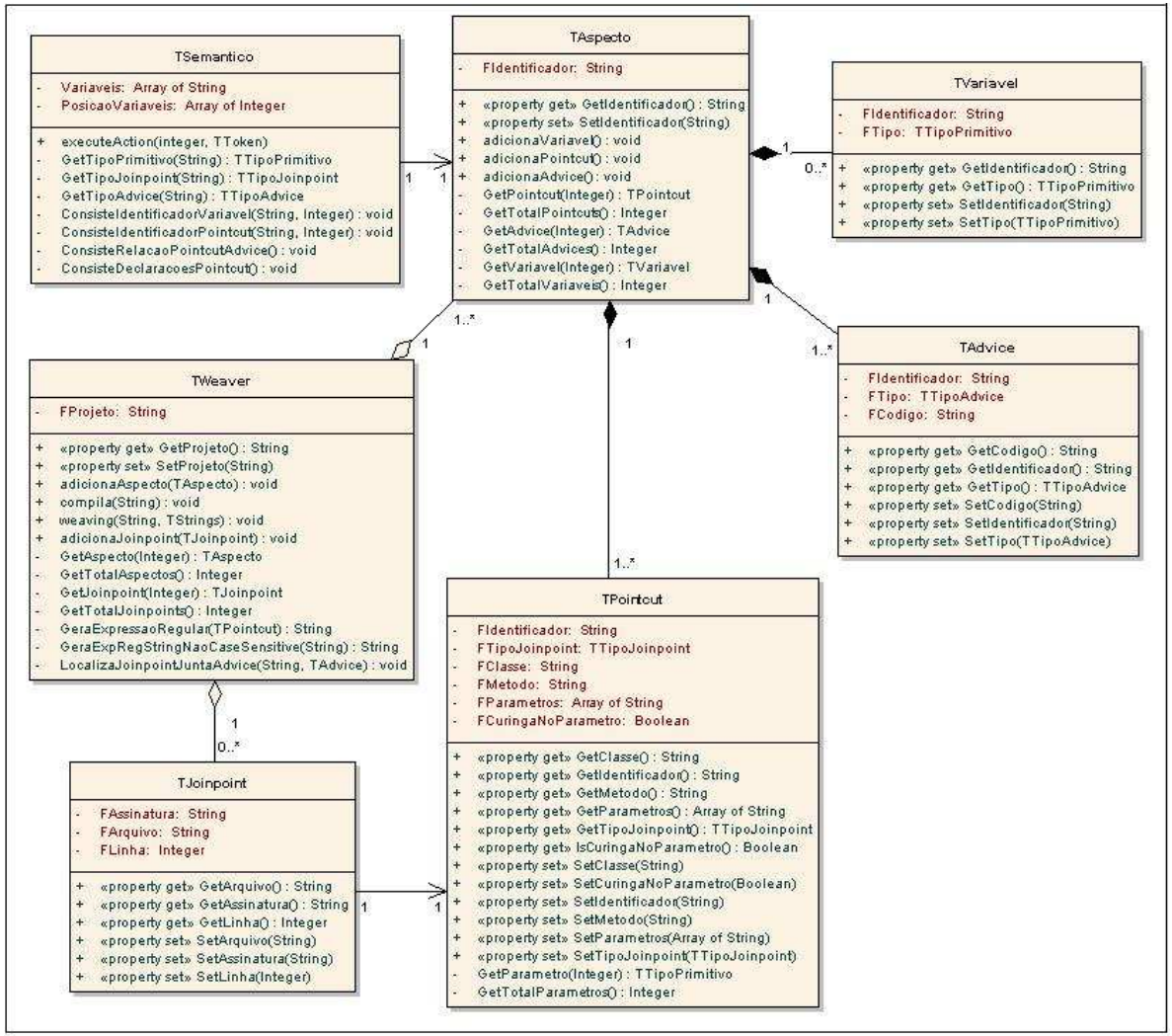
- Diagrama de Classes



Classes geradas pelo GALS

Especificação da Ferramenta

- Diagrama de Classes



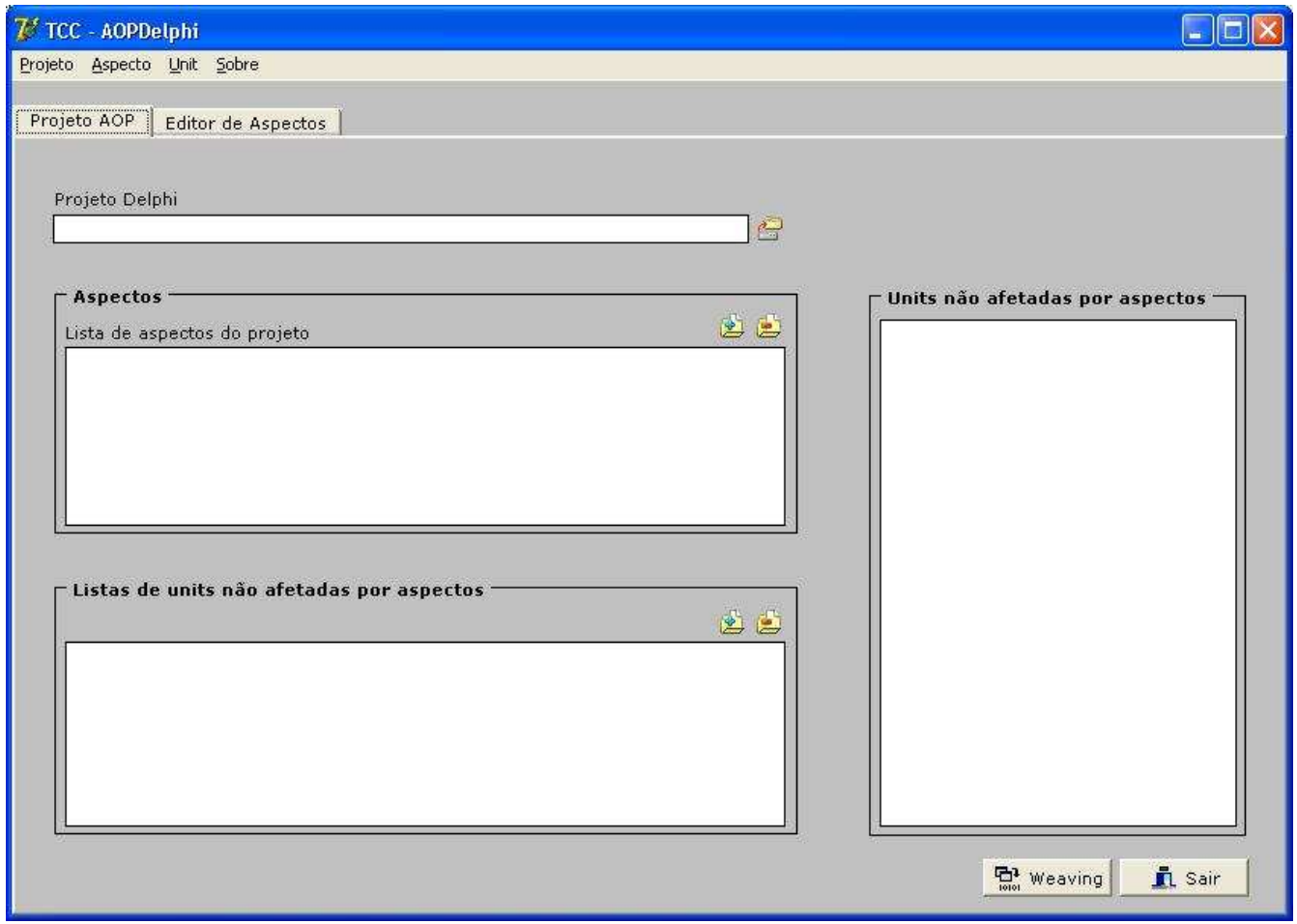
Classes da ferramenta

AOPDelphi – Implementação

- Ferramentas utilizadas
  - Linguagem Delphi – Ambiente Borland Delphi 7
  - Biblioteca RegExp Studio
  - GALS

AOPDelphi – Operacionalidade

- Apresentação



AOPDelphi – Operacionalidade

- Aspecto de log

```
ALog = Aspect
Implementation
  logInsert : Pointcut = (* *.SQLInsert(*));
  logUpdate : Pointcut = (* *.SQLUpdate(*));
  logDelete : Pointcut = (* *.SQLDelete(*));

  Advice logInsert: After;
  begin
    With DMTaskMan.qryLog do
    begin
      ParamByName('LO_IDUSUARIO').AsInteger := giCodUsuario;
      ParamByName('LO_TABELA').AsString := Self.NomeTabela;
      ParamByName('LO_OPERACAO').AsInteger := 1; //1=Insert, 2=Update, 3=Delete
      ParamByName('LO_DATAHORA').AsDateTime := Now;
      ParamByName('LO_SQL').AsString := DMTaskMan.SQLExec.SQL.Text;
      ExecSQL;
    end;
  EndAdvice;
```

AOPDelphi – Operacionalidade

- Aspecto de log

```
Advice logUpdate: After;
  begin
    With DMTaskMan.qryLog do
    Begin
      ParamByName('LO_IDUSUARIO').AsInteger := giCodUsuario;
      ParamByName('LO_TABELA').AsString := Self.NomeTabela;
      ParamByName('LO_OPERACAO').AsInteger := 2; //1=Insert, 2=Update, 3=Delete
      ParamByName('LO_DATAHORA').AsDateTime := Now;
      ParamByName('LO_SQL').AsString := DMTaskMan.SQLExec.SQL.Text;
      ExecSQL;
    end;
  EndAdvice;

  Advice logDelete: After;
  begin
    With DMTaskMan.qryLog do
    begin
      ParamByName('LO_IDUSUARIO').AsInteger := giCodUsuario;
      ParamByName('LO_TABELA').AsString := Self.NomeTabela;
      ParamByName('LO_OPERACAO').AsInteger := 3; //1=Insert, 2=Update, 3=Delete
      ParamByName('LO_DATAHORA').AsDateTime := Now;
      ParamByName('LO_SQL').AsString := DMTaskMan.SQLExec.SQL.Text;
      ExecSQL;
    end;
  EndAdvice;
End.
```

AOPDelphi – Operacionalidade

- Método SQLInsert antes do *weaving*

```
Function TEncaminhaTarefa.SQLInsert:Integer;
begin
  With DMTaskMan.SQLQuery do
    begin
      Close;
      Sql.Clear;
      Sql.Add('Insert into ENCAMINHAMENTO Values(:EN_ID_ENCAM, :EN_CLIENTE,');
      Sql.Add(' :EN_USUARIO, :EN_DATA_CRIACAO, :EN_DATA_CONCLUSAO, :EN_STATUS)');
      ParamByName('EN_ID_ENCAM').AsInteger := flGetProximoCodigo;
      Result:= ParamByName('EN_ID_ENCAM').AsInteger;
      ParamByName('EN_CLIENTE').AsInteger := uaClientes[cbCliente.ItemIndex];
      ParamByName('EN_USUARIO').AsInteger := uaUsuarios[cbUsuario.ItemIndex];
      ParamByName('EN_DATA_CRIACAO').AsDate := edtDataCriacao.Date;
      ParamByName('EN_DATA_CONCLUSAO').Value := Null;
      ParamByName('EN_STATUS').AsString := 'P'; // Pendente
      ExecSQL;
    end;
End;
```

AOPDelphi – Operacionalidade

- Método SQLInsert após o *weaving*

```
With DMTaskMan.SQLQuery do
   begin
     Close;
     Sql.Clear;
     Sql.Add('Insert into ENCAMINHAMENTO Values(:EN_ID_ENCAM, :EN_CLIENTE,');
     Sql.Add(' :EN_USUARIO, :EN_DATA_CRIACAO, :EN_DATA_CONCLUSAO, :EN_STATUS)');
     ParamByName('EN_ID_ENCAM').AsInteger := flGetProximoCodigo;
     Result:= ParamByName('EN_ID_ENCAM').AsInteger;
     ParamByName('EN_CLIENTE').AsInteger := uaClientes[cbCliente.ItemIndex];
     ParamByName('EN_USUARIO').AsInteger := uaUsuarios[cbUsuario.ItemIndex];
     ParamByName('EN_DATA_CRIACAO').AsDate := edtDataCriacao.Date;
     ParamByName('EN_DATA_CONCLUSAO').Value := Null;
     ParamByName('EN_STATUS').AsString := 'P'; // Pendente
     ExecSQL;
   end;
   // Código inserido por AOPDelphi – Início
   // [TEncaminhaTarefa.SQLInsert:Integer] Afetado pelo aspecto Alog – Advice logInsert(After)
   With DMTaskMan.qryLog do
   begin
     ParamByName('LO_IDUSUARIO').AsInteger := giCodUsuario;
     ParamByName('LO_TABELA').AsString := Self.NomeTabela;
     ParamByName('LO_OPERACAO').AsInteger := 1; //1=Insert, 2=Update, 3=Delete
     ParamByName('LO_DATAHORA').AsDateTime := Now;
     ParamByName('LO_SQL').AsString := DMTaskMan.SQLExec.SQL.Text;
     ExecSQL;
   end;
   // Código inserido por AOPDelphi - Fim
end;
```

Resultados e Discussão

- Apresentou bons resultados no sentido de prover suporte a POA para Delphi
- Limitações da linguagem de aspecto
- Facilidade e confiabilidade

Conclusão

- Propicia um conhecimento sobre a tecnologia de POA
- Softwares mais flexíveis. Maior manutenibilidade e reusabilidade
- Importância do surgimento de novas ferramentas de apoio na área de POA
- Importância do GALS e RegExp Studio
- Objetivos foram atingidos

Extensões

- Integrar as funcionalidades do AOPDelphi como opções de menu na IDE do Delphi
- Melhoria na linguagem de aspectos
  - Permitir declaração de variáveis nos *advices*
  - Implementação de procedimentos e funções locais nos *advices*