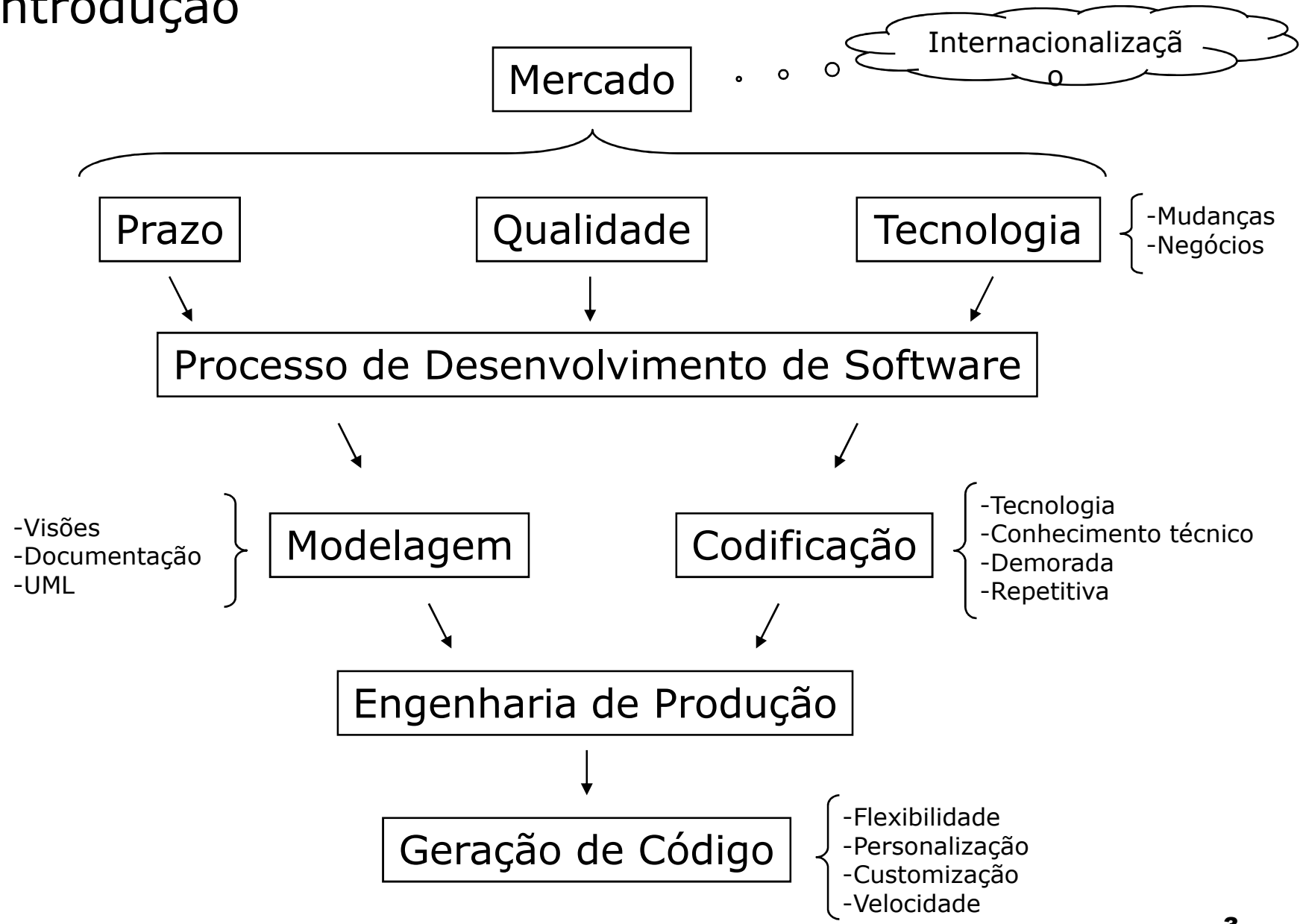
FERRAMENTA PARA GERAÇÃO DE CÓDIGO A PARTIR DA ESPECIALIZAÇÃO DO DIAGRAMA DE CLASSES

FURB – Universidade Regional de Blumenau
Sistemas de Informação
Acadêmico: Alexandro Deschamps
Orientador : Everaldo Artur Grahl

Seqüência da Apresentação

- Introdução
- Objetivo
- UML (*Unified Modeling Language)*
- Modelo e Diagrama de Classes
- Especialização
- Geração de Código
- Diagrama de Atividades
- Implementação
- Resultados e Discussão
- Conclusões
- Extensões
- Apresentação da Ferramenta

Introdução


Mercado
Internacionalizaçã
o
Prazo
Qualidade
Tecnologia
-Mudanças
-Negócios
Processo de Desenvolvimento de Software
-Visões
-Documentação
-UML
Modelagem
Codificação
-Tecnologia
-Conhecimento técnico
-Demorada
-Repetitiva
Engenharia de Produção
Geração de Código
-Flexibilidade
-Personalização
-Customização
-Velocidade

Objetivo

- Desenvolver uma ferramenta de auxílio ao processo de desenvolvimento que permita aos desenvolvedores de software o enriquecimento das informações contidas em diagramas de classes UML (*Unified Modeling Language*) para geração de código.

Diagramas de Classes

+

Especialização

=

Repositório Rico em Informações

UML (*Unified Modeling Language*)

- Linguagem para modelagem de sistemas
- Independente tanto de linguagens de programação quanto de processos de desenvolvimento.
- Aprovada como padrão em 1997 pela OMG (*Object Management Group).*
- Principais contribuintes: Grady Booch, James Rumbaugh e Ivar Jacobson, normalmente conhecidos como “os três amigos”.
- Sua última versão é a 2.0 e possui 13 diagramas.
- Através dos diagramas pode-se obter diferentes visões de um software.

Modelo de Classes

- Evolui durante as iterações do desenvolvimento do software, a medida que o software é desenvolvido o modelo de classes é incrementado com novos detalhes, passando por três níveis de abstração (BEZERRA, 2003, p. 96). Esses níveis são:
  - Domínio
  - Especificação
  - Implementação
- Composição
  - Descrição Textual
  - Diagrama de Classes

Diagrama de Classes - Definição

- É considerado a essência da UML, resultado de uma combinação de diagramas propostos pela OMT (*Object Modeling Technique*), Booch e vários outros métodos (FURLAN, 1998, p. 91).

- Importantes não só para visualização, a especificação e a documentação de modelos estruturais, mas também para a construção de sistemas executáveis por intermédio da Engenharia de Produção (BOOCH, 2000, p. 104).

- De todos os diagramas da UML, esse é o mais rico em termos de notação (BEZERRA, 2003, p. 97).

Diagrama de Classes - Composição

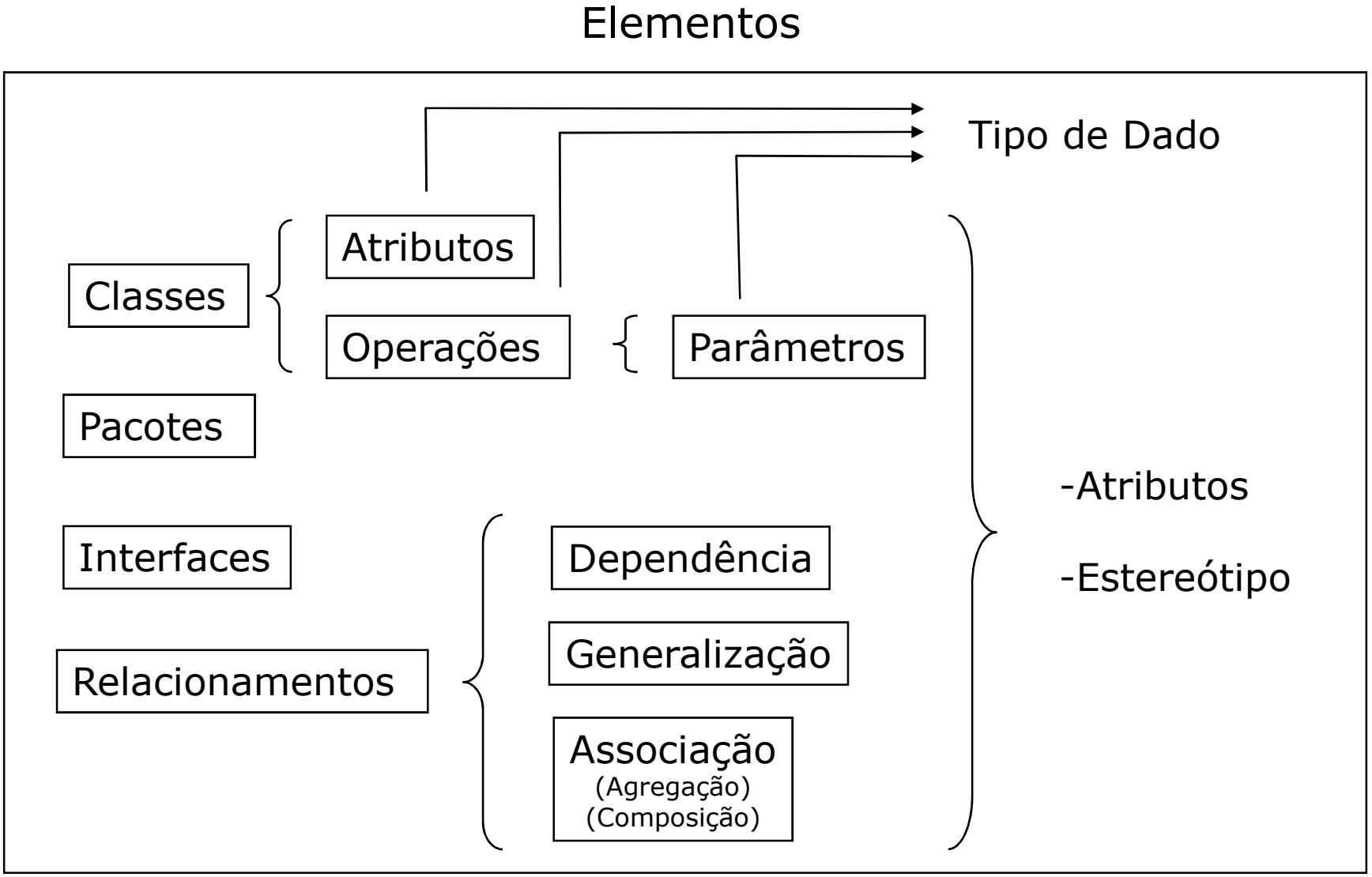
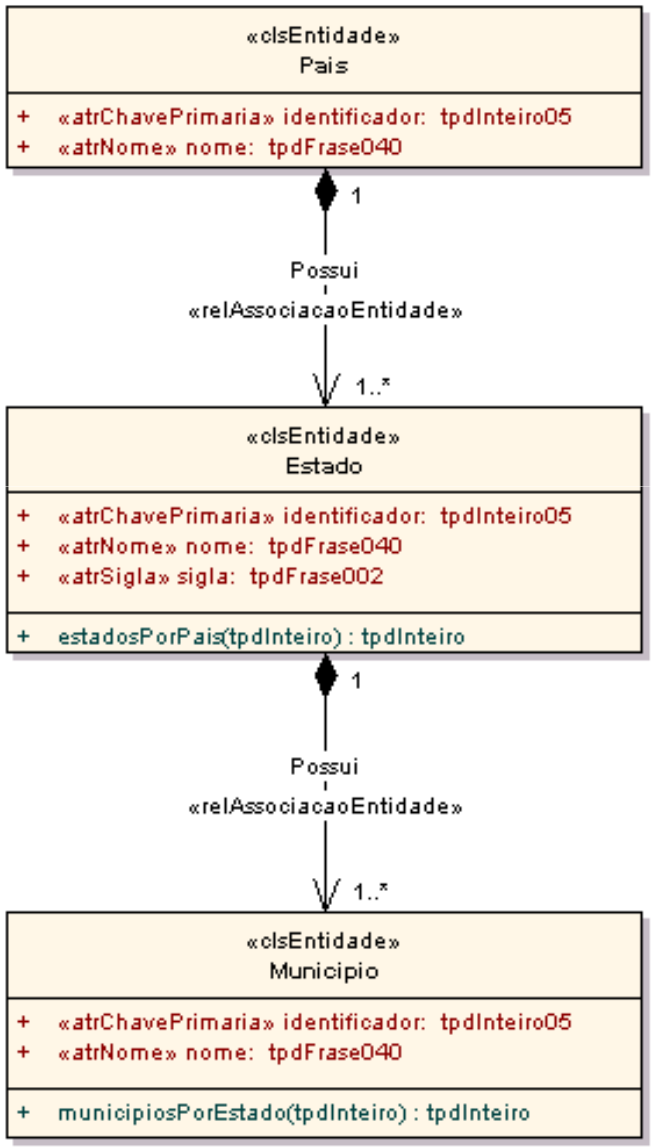


Diagrama de Classes - Exemplo



Especialização

- Arquitetura
- Idioma
- Estrutura de Especialização

Arquitetura

- A arquitetura associa as capacidades do sistema identificadas na especificação de requisitos com os componentes do sistema que irão implementá-las (PFLEEGER, 2004, p. 164).

- Exemplos:
  - Bancos de Dados
  - Linguagens de Programação
  - Bibliotecas de Desenvolvimento

- Considerações
  - Semelhanças e Diferenças
  - Herança (Ex: Delphi CLX e Delphi VCL, Bibliotecas de Desenvolvimento)

Idiomas

- Muitas das informações utilizadas em um sistema são dependentes de idioma.

- Problemas

  - Sua utilização não é contemplada pelos modelos e diagramas.
  - Forte ligação com o código fonte.
  - Armazenamento (Arquivos de Tradução).

- Considerações

  - Ligação com os elementos do diagrama de classes.
    - Rótulos
    - Títulos
    - Mensagens

  - Internacionalização

Estrutura de Especialização - Definição

- Objetivo: Agregar novas características ou propriedades aos elementos do diagrama de classes, visando o enriquecimento das informações para geração de código.

- Categorias de Especialização: Tem o objetivo de representar as diferentes categorias dos elementos do diagrama de classes. As categorias podem ser definidas através da Engenharia de Software ou conforme a necessidade do processo de desenvolvimento

  - Exemplos de categorias

Elemento Classe
Entidade
Controle
Fronteira

Elemento Atributo
Identificação
Localização
Legais

Estrutura de Especialização - Composição

- As categorias de especialização são compostas por:
  - Nome – Identificação da especialização para o usuário
  - Estereótipo – Ligação da especialização com os elementos do diagrama de classes.
  - Propriedades – Características da especialização, possuem um tipo de dado e uma visibilidade

Tipos de Dado:

- Frase
- Texto
- Número Inteiro
- Número Decimal
- Lógico
- Data
- Hora
- Lista Seleção
- Lista de Múltipla Seleção

Visibilidades:

- Genérica
- Arquitetura
- Idioma
- Arquitetura e Idioma

Estrutura de Especialização - Visibilidades

- Genérica: Indica que a propriedade é completamente independente de arquitetura e idioma, proporciona uma considerável reutilização.
- Arquitetura: Indica que a propriedade pertence a somente uma arquitetura.
- Idioma: Indica que a propriedade pertence a somente um idioma.
- Arquitetura e Idioma: Indica que a propriedade pertence a uma arquitetura e a um idioma.

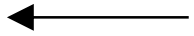
Estrutura de Especialização - Herança

- A estrutura de especialização segue o modelo de herança semelhante ao utilizado pela orientação a objetos, a utilização do modelo de herança permite a criação de categorias de especialização descendentes, sendo que as mesmas herdam todas as propriedades da sua categoria de especialização ancestral.

- As propriedades de especialização herdadas de uma categoria de especialização ancestral podem ser sobrescritas para a categoria de especialização descendente através da atribuição de um novo valor para a mesma.

- Exemplo (Elemento Classe):

Nome	Classe Genérica
Estereótipo	clsGenerica
Propriedades	Papel = “” Prefixo = “”

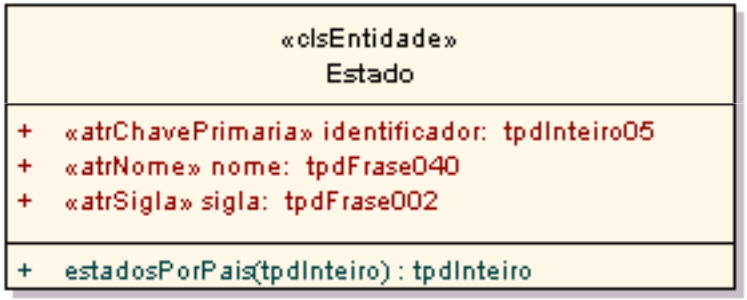


Nome	Classe de Entidade
Estereótipo	clsEntidade
Propriedades	Papel = “” **Prefixo = “Etd”** Descrição Singular = “” Descrição Plural = “” Nome Físico = “” Nome Lógico = “”

Ligação dos Elementos com as Especializações

- A ligação dos elementos dos diagramas de classes importados com as categorias de especialização ocorre através do "estereótipo". O estereótipo é utilizado para estender o significado de um determinado elemento de um diagrama (BEZERRA, 2003, p. 41).

Elemento Classe



+

Atributos do Elemento (UML)

Name = "Estado"
Visibility = "public"
Abstract = false
Root = true
Final = false
Active = false

+

Especialização do Elemento

Papel = ""
Prefixo = "Etd"
Descrição Singular = "Estado"
Descrição Plural = "Estados"
Nome Físico = "TB_EST"
Nome Lógico = "Estado"

=

Elemento Especializado

Geração de Código

- Ao contrário das soluções disponíveis no mercado, o sistema não trabalha com a utilização de *scripts* ou *templates* para implementação das rotinas de geração, as rotinas com a lógica para geração de código são construídas através de *plugins*, módulos DLL (*Dynamic Link Library*) compilados através da plataforma Microsoft .NET.

- Diferenciais

  - Utilização das principais linguagens de programação (C#, JAVA, Delphi, Visual Basic) na implementação dos *plugins*, dispensando o estudo de novas tecnologias ou *scripts.*

  - Aumento da velocidade de processamento na geração de código.

  - Uso da orientação a objetos e dos recursos da plataforma Microsoft.NET na implementação.

  - Proporciona maior flexibilidade e personalização para o processo de geração de código, visando atender as particularidades dos diferentes processos de desenvolvimento de software das organizações

Geração de Código

- Cada *plugin* de geração está associado a uma arquitetura.

- A interface para acoplamento de *plugins* proposta pelo sistema disponibiliza para os *plugins* o diagrama de classes e suas especializações conforme apresentado na figura abaixo:

```
public interface IQuelleControlUMLClassModelGeneration
{
    bool ExecuteGeneration(IQuelleControlUMLClassModel classModel);
}
```

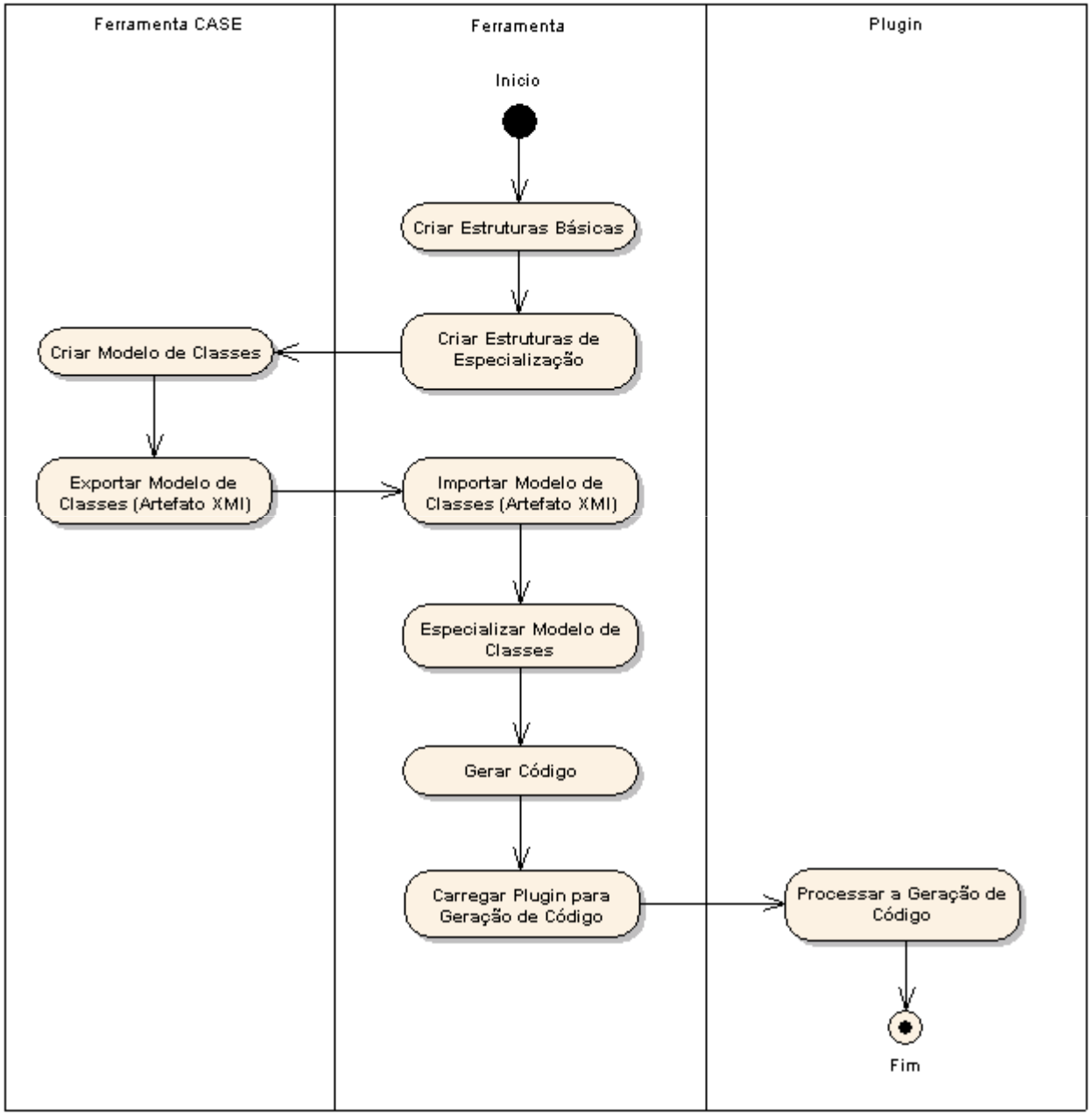


Diagrama de Atividades

Implementação

- Ferramentas
  - Enterprise Architect – Versão 4.51
  - Plataforma Microsoft .NET – Versão 2.0 Beta
  - Visual C# – Versão 2.0 Beta
  - Ápice .NET Custom Library – Versão 1.0 Beta
  - Firebird – Versão 1.5
- Tecnologias
  - UML (*Unified Modeling Language*) – Versão 2.0
  - XML (*Extensible Markup Language*) – Versão 1.0
  - XMI (*XML Metadata Interchange*) – Versão 1.2
- Engenharia de Software / Metodologias
  - Orientação a Objetos
  - Divisão em camadas (persistência, lógica e apresentação)

Resultados e Discussão

- Tempo X Tecnologias / Ferramentas
- Fundamentação Teórica X Implementação
- Volume Final X Exemplos
- Trabalhos Correlatos
- Autores
- Falta de documentação sobre Geração de Código

Conclusões

- O objetivo do trabalho foi atingido.
  - Requisitos
    - Importação através de artefatos XMI (*XML Metadata Interchange*).
    - Especialização dos diagramas de classes por arquitetura e idioma.
    - *Interface para acoplamento de Plugins*
    - Rotinas de Geração
  - Processo de desenvolvimento
    - Redução do tempo de desenvolvimento
    - Aumento da qualidade
    - Aumento da Padronização
    - Mudanças tecnológicas
    - Foco maior na lógica de negócio

Extensões

- Entrada
  - Importação através de formatos proprietários das ferramentas CASE (*Computer Aided Software Engineering*).
  - Importação através de outras versões do XMI (*XML Metadata Interchange).*
- Saída
  - Criação de *plugins* para geração de código.
  - Criação de *plugins* de documentação.
- Especialização
  - Criação de regras de negócio.
  - Criação de fórmulas.
  - Criação de propriedades compostas.

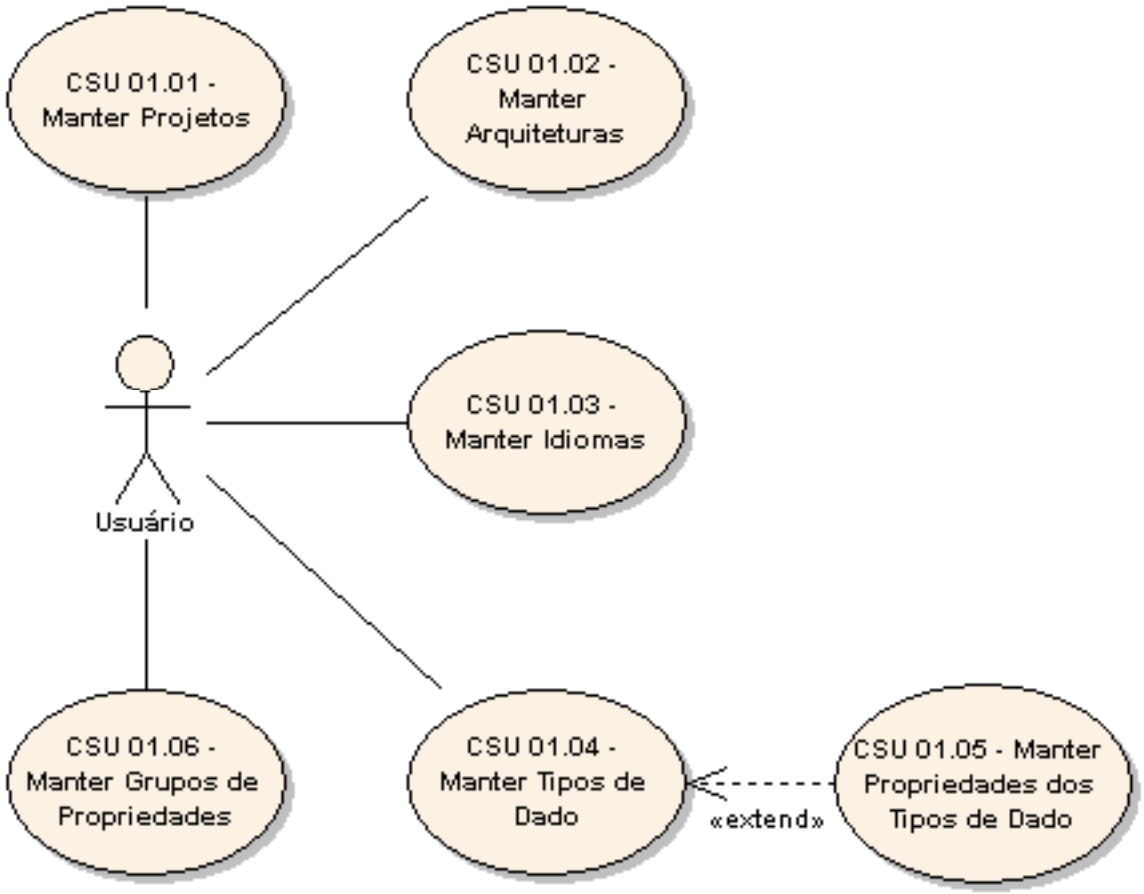
Conclusões Pessoais

- Conceitos X Prática
  - Graduação
  - Mercado
- Motivação
  - Gostar do que faz
  - FURB/GENE – Projeto de Extensão
- Mudanças
  - Pessoais
  - Profissionais
- Final X Inicio
  - Conclusão da graduação
  - Projeto Incubado
- Mercado
  - Geração de Código (Diferencial competitivo / Benefícios)
- Aprendizado

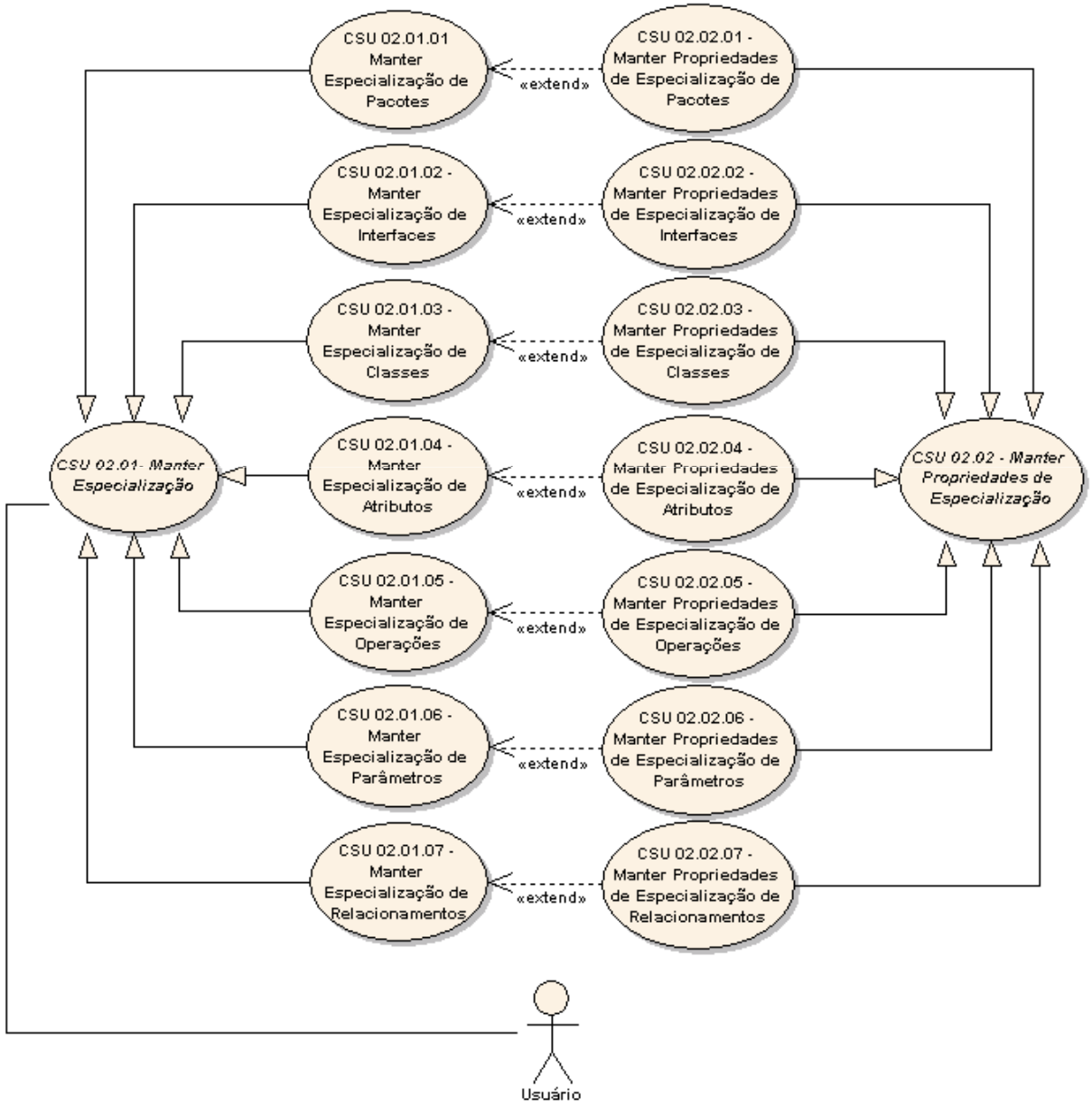
Apresentação da Ferramenta

- Objetivos:
  - Demonstrar a operacionalidade da ferramenta através dos casos de uso.
  - Demonstrar na prática os conceitos apresentados.
  - Gerar código para Java e PCL (*PHP Custom Library*) utilizando um único diagrama de classes.
  - Demonstrar o refinamento da geração de código.

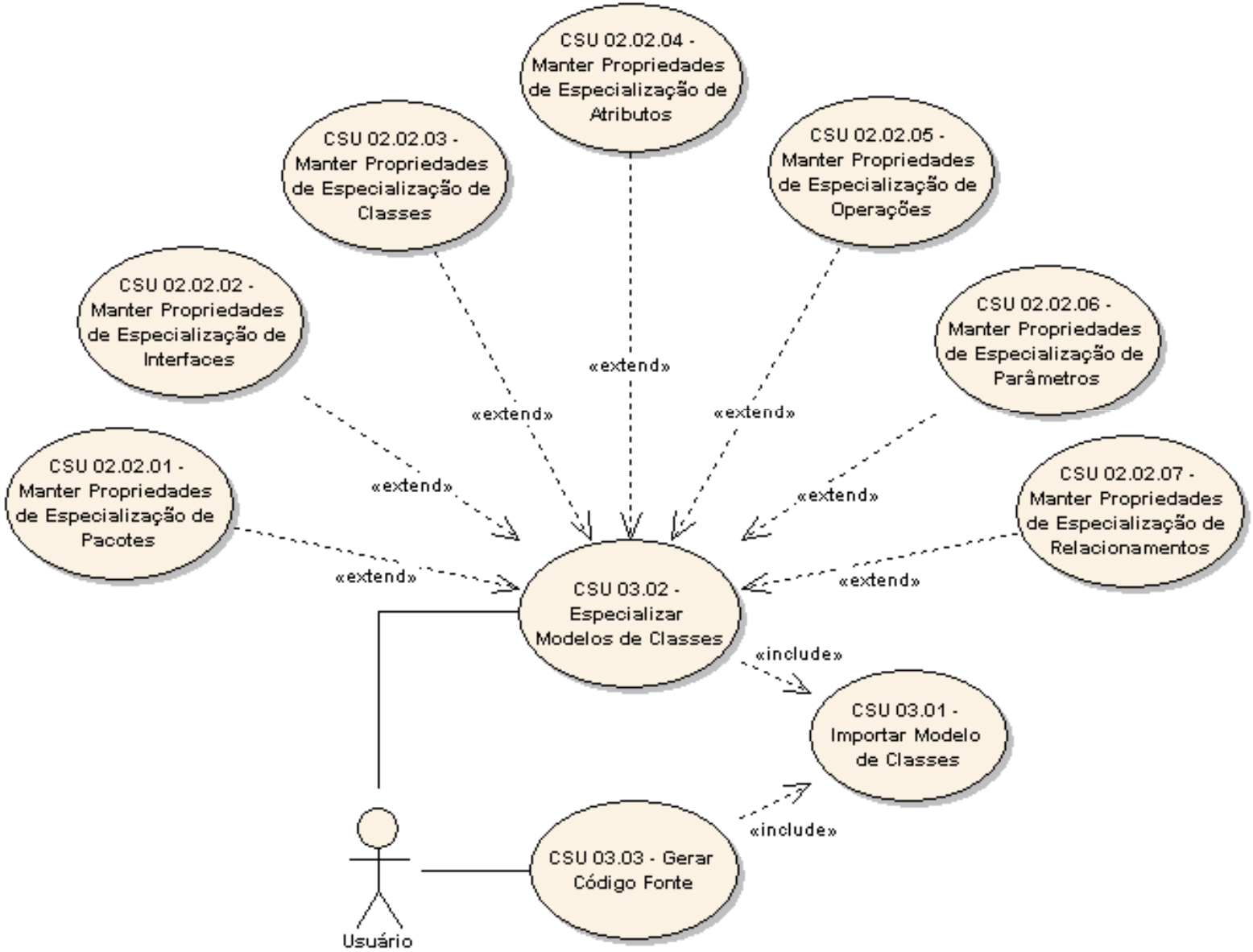
Caso de Uso – Estruturas Básicas



Caso de Uso – Especializações



Caso de Uso – Modelo de Classes



Agradecimento

Agradeço a todos pela atenção!

Sua presença foi muito importante para mim!

Alexandro Deschamps
alexandro@apicesoft.com