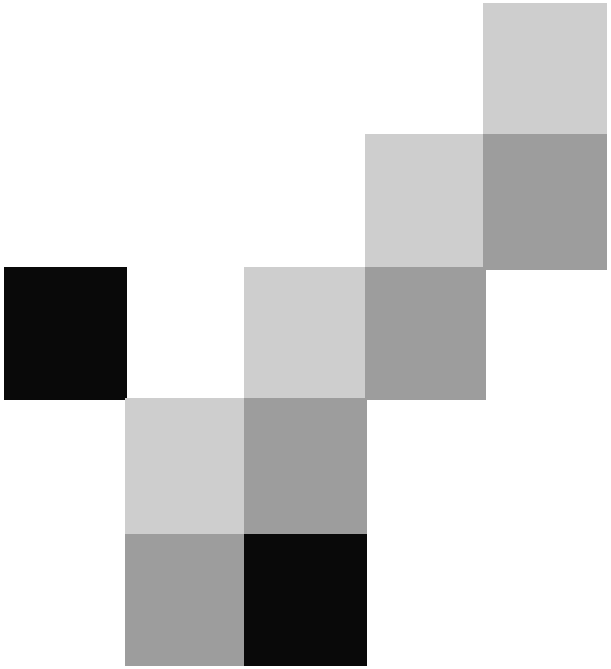


Especificação Formal de Software

Carlos Figueiredo,
Jorge Mack,
Luís Magalhães,
Vitor Pinto

Disciplina de Engenharia de Software

6 Dezembro 2002

Introdução

- Nos dias de correm é cada vez maior a dependência dos computadores e seus aplicativos.
- Torna-se assim crucial desenvolver aplicações de *software* mais complexas e precisas.
- Á medida que a complexidade aumenta, aumenta a probabilidade de se cometerem erros.

Introdução

- Na construção de um *software* deve-se garantir que este cumpre os requisitos pedidos.

- Engenharia de Software importante no desenvolvimento de software.

Introdução

- Então porquê a necessidade de usar métodos formais?

- Se todas as Engenharias recorrem a métodos formais (matemática), porque não a Engenharia de Software?!

Introdução

- Mas o que é um método formal em ES?

- Método de desenvolvimento de software através do qual se pode definir precisamente um sistema e desenvolver implementações garantidamente correctas em relação a sua definição.

Introdução

- Diferenças entre os métodos aplicados em ES.

- **Informal**
  - Pouca (ou nenhuma) regra pré-definida
- **Sistemático**
  - Conjunto de regras pré-estabelecidas
- **Rigoroso**
  - Regras de transformação estritas
  - Partes críticas desenvolvidas formalmente
- **Formal**
  - Verificação formal de todo o processo

Introdução

- Descrição do método Formal:
  - Descrição Matemática Precisa
    - Não ambígua
    - Permite verificar propriedades (ex. consistência)
    - Usualmente abstractas e concisas
  - É a base para
    - Documento contratual
    - Documentação do produto
    - Referência para etapas seguintes
  - Completa e coerente

Vantagens

- Permite uma compreensão alargada dos requisitos e do design.
- Especificações formais podem ser analisadas matematicamente. É possível provar que a implementação corresponde à especificação.

Vantagens

- Especificações formais podem ser utilizadas para identificação dos testes mais apropriados para o *software.*
- É possível animar uma especificação formal de modo a obter um protótipo do *software* (através de ferramentas de *software* apropriadas).

Desvantagens

- Porque que é que os métodos formais não são utilizados?

Desvantagens

- É difícil demonstrar as vantagens das especificações formais de uma forma objectiva.
- Muitos engenheiros de software não têm a experiência necessária em matemática discreta para especificações formais.

Desvantagens

- Os clientes não estão familiarizados com estas técnicas, por isso, não as financiam.
- Algumas classes de software são difíceis de especificar.
  - Sistemas interactivos
  - Sistemas concorrentes

Mitos

- Existem algumas crenças acerca das especificações formais que foram exageradas sendo mesmo consideradas mitos.

Mitos

1. Os Métodos formais podem garantir que o software seja perfeito.
   - Disparate – o modelo de especificação formal é um modelo do mundo real, pode incorporar erros ou omissões.
2. Os Métodos formais só são utilizados para prova do programa.
   - Também têm valor ao forçar cedo uma análise detalhada ao processo de desenvolvimento.

Mitos

3. Devido ao elevado custo, os métodos formais só são úteis para sistemas de segurança críticos.
   - Os custos de desenvolvimento de todas as classes de sistemas são reduzidos.
4. São necessários matemáticos treinados para a implementação dos métodos formais.
   - A matemática usada é simples, embora necessite de algum treino, como qualquer outro método.

Mitos

5. Os Métodos Formais aumentam o custo de desenvolvimento.
   - Aumentam os custos de especificação mas reduzem os custos de implementação.
6. Os Métodos Formais são inaceitáveis pelos utilizadores.
   - Podem ser traduzidos em linguagem natural.

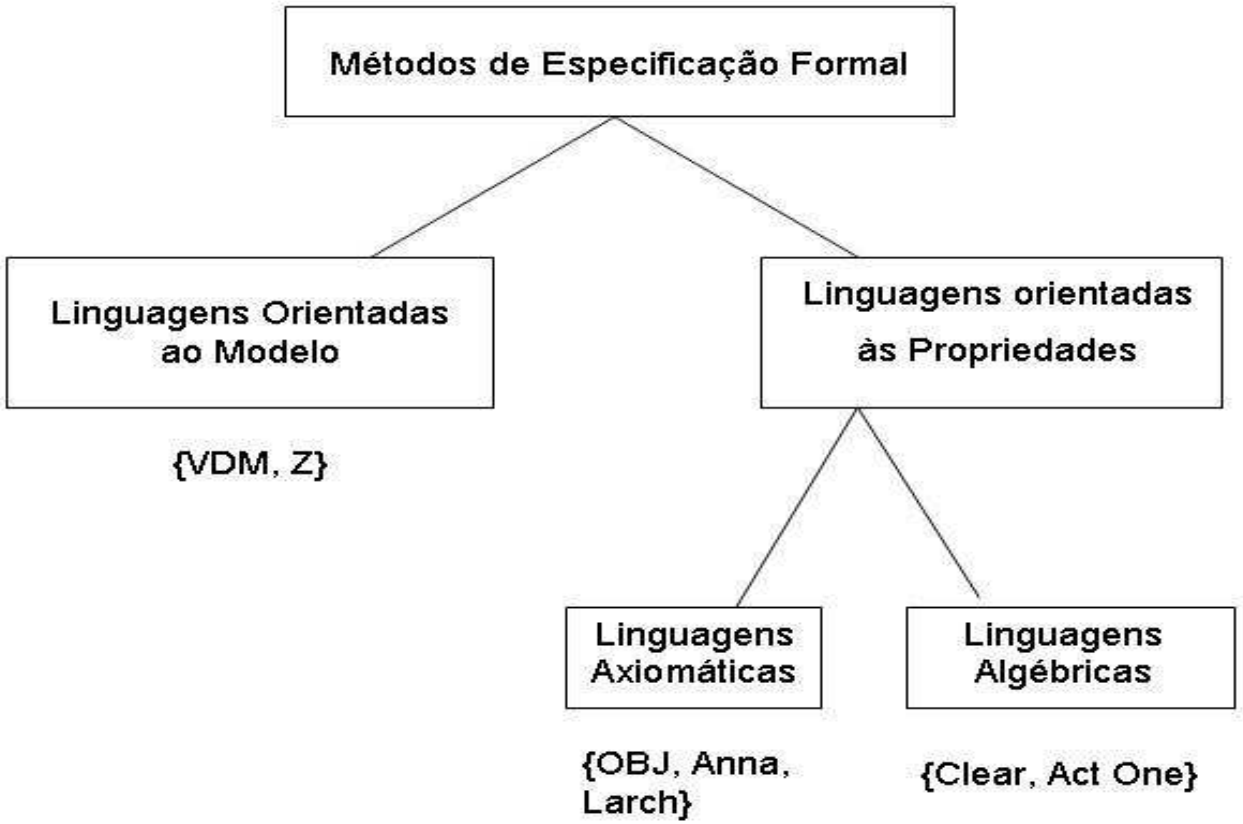
Mitos

7. Os Métodos Formais só são usados em sistemas triviais.
   - Existem publicados muitos exemplos de experiências com métodos formais para sistemas não triviais.

Veredicto

- Todas as razões para não usar especificações formais foram invalidadas.
- No entanto
  - As especificações formais são difíceis de aplicar em sistemas interactivos.
  - Se as técnicas de engenharia de software estão bem para quê mudar.

Linguagens de Especificação



Linguagens Orientadas ao Modelo

- usam modelos matemáticos construídos usando tipos de dados simples, p. ex.
  - listas,
  - caracteres,
  - números naturais,
- são ainda especificadas as operações que mudam o estado do modelo (máquina de estados).

Linguagens Orientadas ao Modelo (VDM)

- VDM (The Vienna Development Method)
- A especificação é constituída por:
  - Tipo de dados,
  - Operações permitidas.
- Exemplo: Reservas de um hotel
  - Tipo de dados: Listas (clientes, reservas)
  - Operações, p. ex.: Novo Cliente, Apagar Reserva

Linguagens Orientadas ao Modelo (Z -“Zed”)

- Muito Semelhante ao VDM.
- Estruturada por um conjunto de Esquemas.
- Esquema (variáveis, relações entre variáveis, operações).
- As operações provocam mudanças de estado.

Linguagens Baseadas em propriedades

- Especifica o comportamento do sistema de uma forma indirecta, através da especificação de um conjunto de propriedades que o sistema deve satisfazer.
- Axiomáticas: baseadas em lógica.
- Algébricas: usam axiomas na forma de equações.

Linguagens Axiomáticas

- Usa abstracções do sistema baseadas na lógica (p. ex.: primeira ordem)
- Algumas linguagens:
  - OBJ
  - Anna

Linguagens Axiomáticas (OBJ)

- Várias extensões (OBJ3, BOBJ)
- Consiste em:
  - Conjunto de Sentenças de um Sistema Lógico
  - Operações semânticas deduzidas a partir do Sistema Lógico
- Usa conceitos de Álgebra para definir características próprias (p. ex.: tratamento de excepções)

Linguagens Axiomáticas (Anna)

- Extensão da Linguagem Ada - **Ann**otated **A**da
- Consiste numa especificação por "comentários formais"
  - Anotações, especificações de elementos do programa (p. ex.: definição de objectos)
  - Texto Ada, especifico da linguagem (p. ex.: definição de um *package*)

Linguagens Algébricas

- Objectivo:
  - Simplificar notações. Usando axiomas na forma de equações mais simples.
  - Linguagens Algébricas enquadram-se mais com a especificação dos requisitos.
  - Recorre a classes de tipos para ilustrar melhor as propriedades dos objectos.
  - Divide-se em 4 módulos
    - Introdução
    - Descrição informal
    - Assinatura
    - axiomas

FIM