Métodos de Desenvolvimento de Software (MDS) 2011/2012

Miguel Goulão
mgoul@fct.unl.pt
http://ctp.di.fct.unl.pt/~mgoul/

Aula 2

Introdução

Motivação

Para projectos de software complexos, programar sem primeiro compreender especificar o problema (**análise**) e estudar a solução (**desenho**) é como construir uma casa sem o projecto do arquitecto (plantas gerais e de detalhe) e do gabinete de engenharia (plantas da especialidade).

Motivação

- Construir "alicerces" sólidos para desenvolver software orientado pelos objectos
- Usar técnicas de **análise** e **desenho** para ajudar a produzir software orientado pelos objectos
  - **Análise** e **desenho** orientado pelos objectos são os companheiros naturais da **programação orientada pelos objectos** e a **persistência feita com bases de dados**
    - também podem ser usados com linguagens procedimentais

Objectivo

- Estudar uma abordagem de desenvolvimento de software orientada pelos objectos
- Linguagens usadas:

UML (***U**nified **M**odeling **L**anguage*)

- Linguagem gráfica que suporta vários tipos de modelos para especificar o domínio do problema e da solução

OCL (***O**bject **C**onstraint **L**anguage*)

- Linguagem formal utilizada para especificar rigorosamente restrições complementares aos modelos UML

O que o UML oferece

- Uma linguagem de modelação expressiva
  - para especificar, construir, visualizar e documentar sistemas de software
  - para construir diferentes tipos de modelos
- Conceitos fundamentais prontos a usar
  - mas com extensão e especialização disponíveis
- Uma base formal para compreender a linguagem
  - diagrama de classes como metamodelo
  - semântica é parte da documentação
  - OCL (Object Constraint Language)
- Conceitos de desenvolvimento de alto nível
  - padrões, componentes, frameworks
- Integração das melhores práticas

O que o UML **NÃO** oferece

- É intenção explícita dos autores do UML <u>não oferecer</u>:
  - um processo
  - uma ferramenta de modelação
  - heurísticas de modelação
  - uma linguagem de programação

Os modelos

- Diagrama de casos de utilização
- Diagramas de classes

Diagrama de estrutura

- Diagramas de sequência
- Diagramas de colaboração

Diagramas de interacção

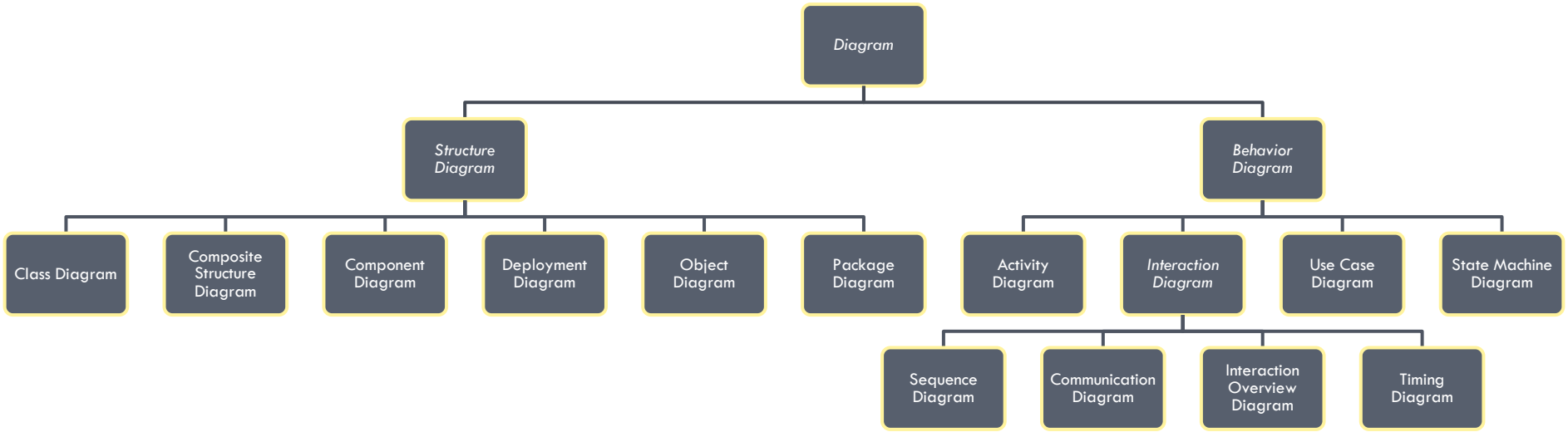
- Diagramas de estados
- Diagramas de actividades

Diagramas de comportamento

- Diagramas de componentes
- Diagramas de entrega

Diagramas de implementação

Diagramas UML



Enquadramento

Engenharia de Software: Criada para ...

- Melhorar a qualidade do software
- Aumentar a produtividade
- Contribuir para a satisfação profissional

Objectivos da Engenharia Software

- Modificabilidade
- Eficiência
- Fiabilidade
- Compreensibilidade

[Ross, Goodenough e Irvine, 1975]

A Engenharia de *software* deve ajudar-nos a desenvolver programas **modificáveis**, **eficientes**, **fiáveis** e **compreensíveis**.

Princípios de Engenharia Software

- Para alcançar os objectivos é preciso aplicar os princípios:
  - Abstracção
  - Ocultação
  - Modularização
  - Localização
  - Uniformização
  - Completação
  - Confirmação
  - Reutilização*

[Ross, Goodenough e Irvine, 1975], excepto *

Ciclo de vida do software (1)

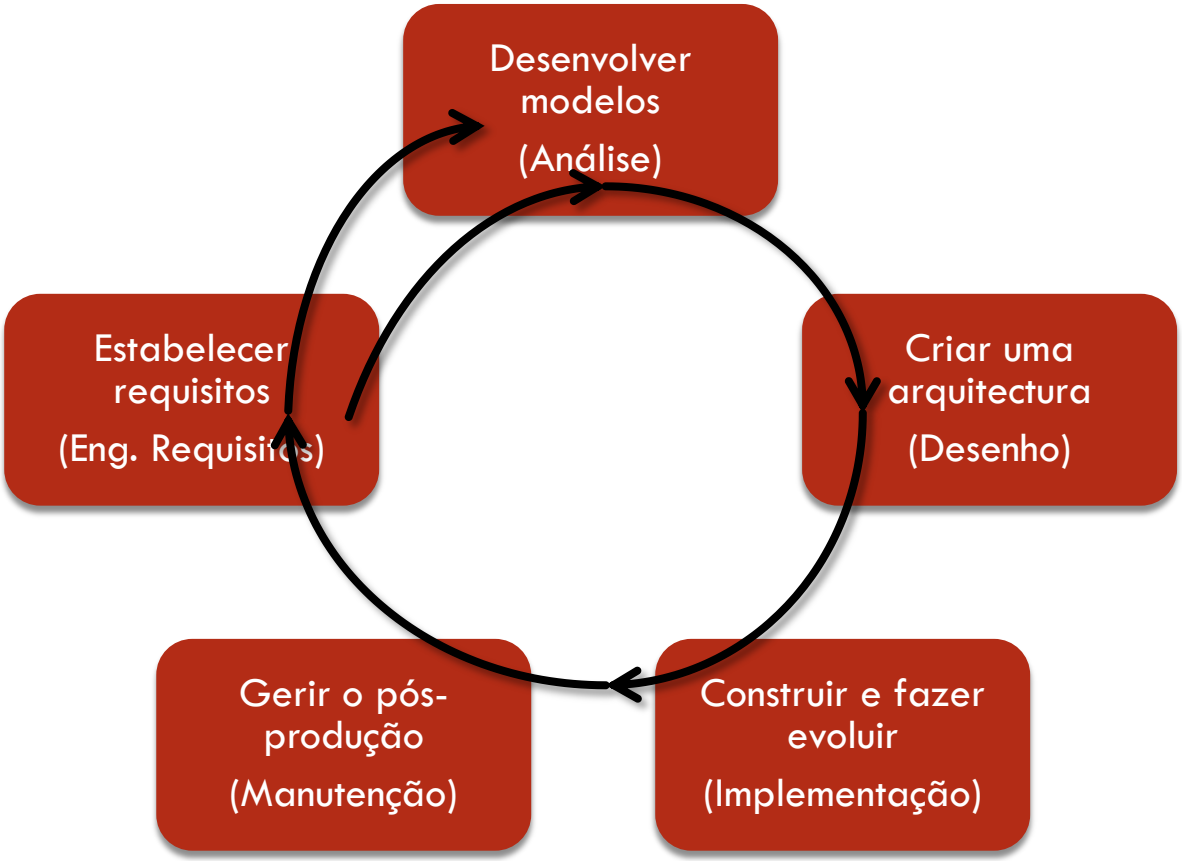
- Tudo começou com a programação!
  - Mas não basta ter uma óptima linguagem de programação para construir software de qualidade!
- É preciso:
  - **Gerir complexidade**, disciplinando o processo de desenvolvimento de software.
  - **Aumentar qualidade**, criando um modelo prévio, antes de atacar a programação.
  - **Satisfazer os clientes**, garantindo que compreendemos as suas necessidades.
- ... e já agora:
  - é preciso aumentarmos também a nossa satisfação pessoal.

Ciclo de vida do software (2)

- Para construir software precisamos:
  - Análise
  - Arquitectura
  - Desenho
  - Implementação
  - Teste
  - Manutenção

O ciclo de vida do *software* define uma sequência de tarefas necessárias para **desenvolver**, **usar** e **manter** os sistemas de *software*.

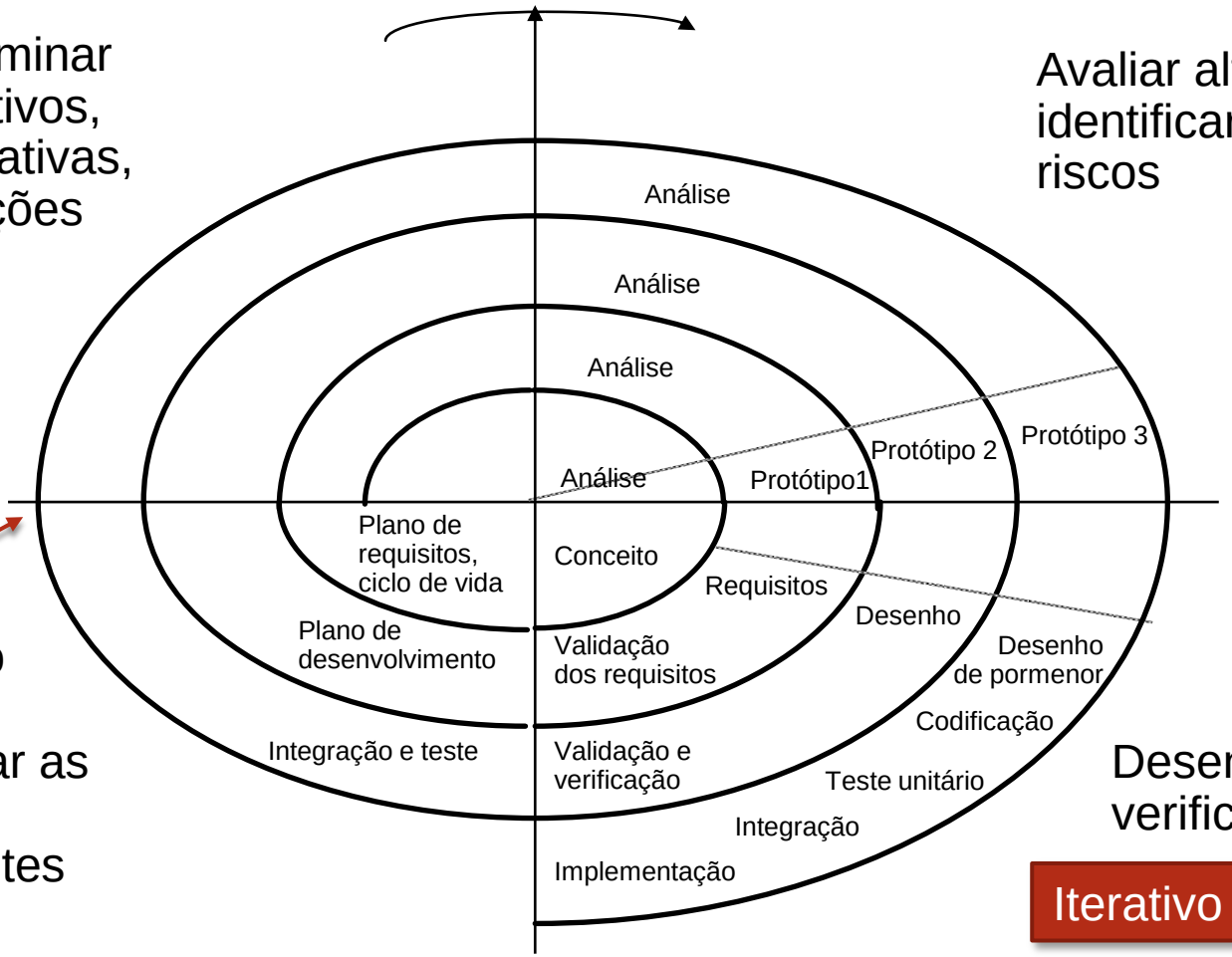
Ciclo de vida do *software* (3)



Ciclo de vida em espiral

Determinar objectivos, alternativas, restrições

Avaliar alternativas: identificar, assumir os riscos



Compromisso

Planear as fases seguintes

Desenvolver, verificar

Iterativo / incremental

Motivação para análise e desenho: Os modelos como abstracções da realidade

- É difícil compreender um sistema complexo
  - um só modelo não é suficiente;
  - são necessárias perspectivas diferentes, cada uma com o seu modelo;
  - cada modelo com diferentes níveis de granularidade.
- Bons modelos são necessários…
  - para tornar compreensíveis sistemas complexos
  - para visualizar aspectos essenciais de um sistema
  - para comunicação entre membros da equipa e com o cliente
  - para assegurar uma boa arquitectura

Desenvolvimento OO

- O mais importante na tecnologia dos objectos **NÃO É** ser uma nova forma de programar (face a outras que já existiam à data)!
- O mais importante é ser uma orientação para a forma de pensar abstractamente acerca dum problema, usando conceitos do mundo real e não conceitos informáticos…
  - Usando o conceito "objecto" ao longo de todo o ciclo de desenvolvimento.

Métodos e metodologias (1)

- **Método**: processo disciplinado para construir produtos de software utilizando um modelo.
- **Metodologia**: colecção organizada de regras, algoritmos, técnicas e ferramentas (integráveis); constituída por um conjunto de métodos que cooperam entre si.

Uma metodologia é indispensável:

- para construir software de qualidade;
- ajudar a comunicação entre todos os elementos da equipa, incluindo o cliente ou utilizador.

Métodos e metodologias (2)

- O tempo e o custo da transição e da adaptação é grande; (todos nós oferecemos um pouco de resistência às novidades!)
- Muitas vezes, falta a metodologia adequada para a organização;
- A automatização das actividades nem sempre está disponível;
- A falta de formação dos quadros técnicos é muitas vezes desesperante;
- Utilizar uma metodologia implica "gastar" mais tempo para aprendê-la, "gastar" mais tempo em documentação, etc;
- A qualidade das ferramentas não cresceu o suficiente.

Métodos de desenvolvimento e sua evolução

- Tipos de métodos:
  - Orientados pelos processos (funcionais)
  - Orientados pelos dados
  - (mistos)
  - Orientados pelos objectos (OOD)
  - Orientados pelos aspectos (AOSD)
  - Orientados pelos modelos (MDD)

Análise

- Objectivos da análise:
  - Compreender o problema, determinando a sua essência;
  - Modelar o problema **independentemente da tecnologia** utilizada na sua implementação.
- A análise constrói um modelo ideal que satisfaz os requisitos do utilizador.

A análise diz o **QUE** deve ser implementado.

Métodos de análise OO

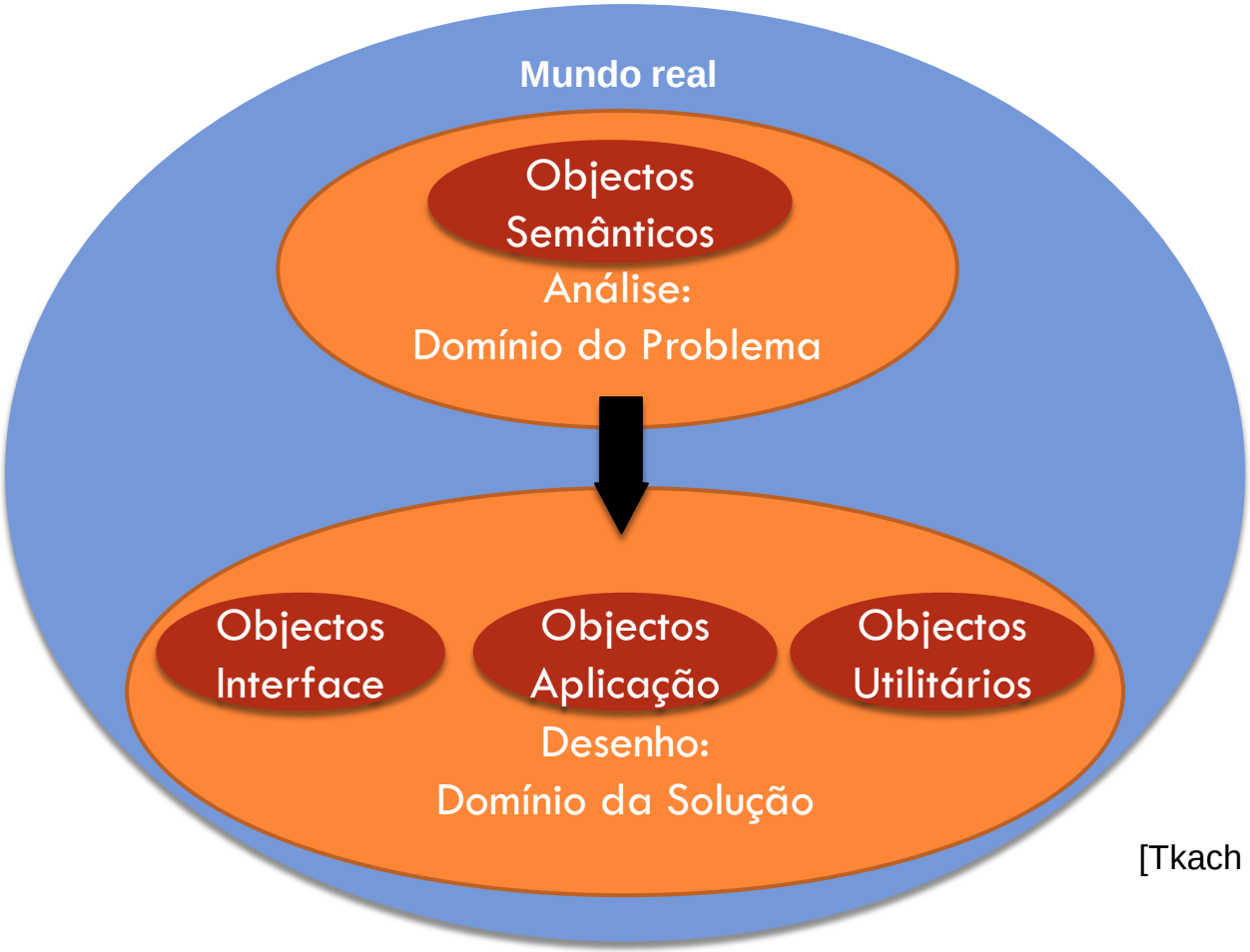
- Visão estática:
  - representada pelo modelo de objectos
  - suportado por um diagrama de entidades e associações estendido
- Visão dinâmica:
  - representada pelo modelo dinâmico
  - suportado por diagramas de transição de estados, diagramas de sequência de mensagens

Desenho

- O desenho transforma o modelo ideal da análise num modelo real.
- Para isso, precisamos de tomar em consideração as características do ambiente de implementação.
- O desenho tem por objectivo modelar o sistema determinando "como" implementar o que foi idealizado durante a análise.

O desenho diz **COMO** o modelo da análise deve ser implementado.

Análise e desenho em perspectiva



[Tkach e Puttick, 1996]

Bibliografia

- Ross, D.T.; Goodenough, J.B.; Irvine, C.A. "*Software Engineering: Process, Principles, and Goals*", *Computer* , vol.8, no.5, pp.17-27, May 1975
  doi: 10.1109/C-M.1975.218952
  URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1649428&isnumber=34586
- Tkach, D.; Puttick, R. "Object Technology in Application Development", Prentice-Hall, 1996, ISBN: 0201498332.