



# Desenvolvimento de Sistemas Software

## Aula Teórica 3: Introdução à UML

# Porquê modelar?

## Os modelos são

- Simplificações da realidade – representações abstractas de um sistema, efectuadas de um ponto de vista específico
  - **Abstracção:**
    - O processo de remover informação de uma descrição para ficarem apenas os aspectos relevantes
    - Mecanismo poderoso para lidar com a complexidade
  - Úteis para descrever e analisar os problemas e as soluções que queremos desenvolver
  - Os modelos são simplificações da realidade.



# Porquê modelar?

## Problemas com a utilização de modelos

- Mais uma “linguagem” a aprender.  
Isso acarreta custos, quer monetários (para as organizações), quer cognitivos (para os indivíduos).
- Modelos apresentam uma **visão idealizada da realidade**.
  - Existe o risco de durante o processo de modelação nos esquecermos que os modelos são representações da realidade e não a realidade.
  - É necessário encontrar as abstracções adequadas para modelar todos os aspectos relevantes.
- A fase de modelação **atrasa a produção de código** (pseudo-problema!)
  - Espera-se, no entanto, que o código produzido seja de melhor qualidade (assim como o próprio sistema desenvolvido).
  - Uma abordagem iterativa e incremental soluciona este problema. Por outro lado, é já possível passar, de forma semi-automática, dos modelos para o código (Model-Driven Engineering!).
  - Regra dos 5/6 – 1/6 (análise e concepção vs. Codificação).

Atenção à “analysis paralysis”!

# Porquê modelar?

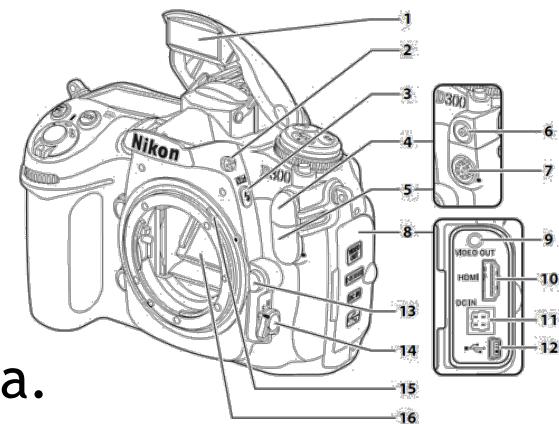
## Vantagens da utilização de modelos

- Auxiliam a **compreender** a realidade
- Ajudam a **comunicar** ideias de forma simplificada.
- Ajudam a **documentar** as decisões tomadas durante o desenvolvimento.
- Modelos com uma semântica precisa (**rigorosa**) permitem **análise rigorosa**

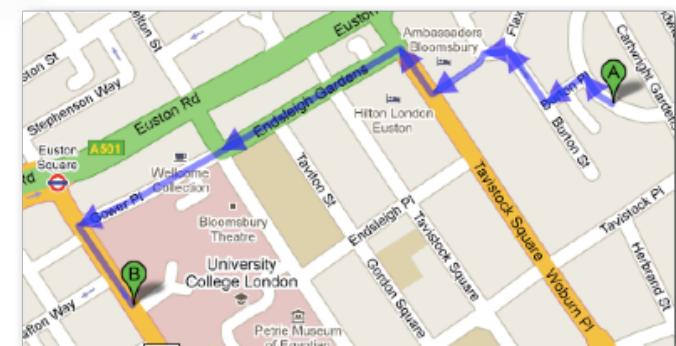
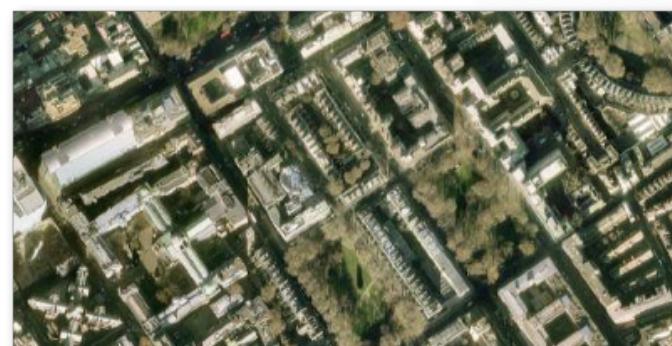
# Porquê modelar?

## Vantagens da utilização de modelos

- Auxiliam a **compreender** a realidade.
  - Sendo **abstracções** da realidade, os modelos permitem descrever o que é considerado essencial num dado contexto, escondendo detalhes desnecessários/irrelevantes nesse contexto.



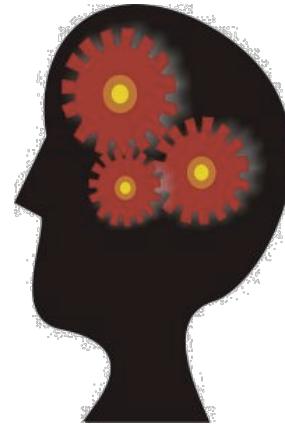
- Ajudam a **comunicar** ideias de forma simplificada.
  - Sendo **simplificações** da realidade, permitem comunicar apenas os aspectos pretendidos.



# Porquê modelar?

## Vantagens da utilização de modelos

- Ajudam a documentar as decisões tomadas durante o desenvolvimento.
  - Os modelos desenvolvidos constituem uma base documental para a descrição do processo de desenvolvimento
  - “*Thinking made public*”.



- Modelos com uma semântica precisa (rigorosa) permitem análise rigorosa
  - Validação - “estamos a construir o sistema certo?”
  - Verificação - “estamos a construir o sistema bem?”



# Tipos de modelos

- Quanto ao objectivo:

- Preditivos

Utilizados para prever o comportamento de um sistema.

- Normativos

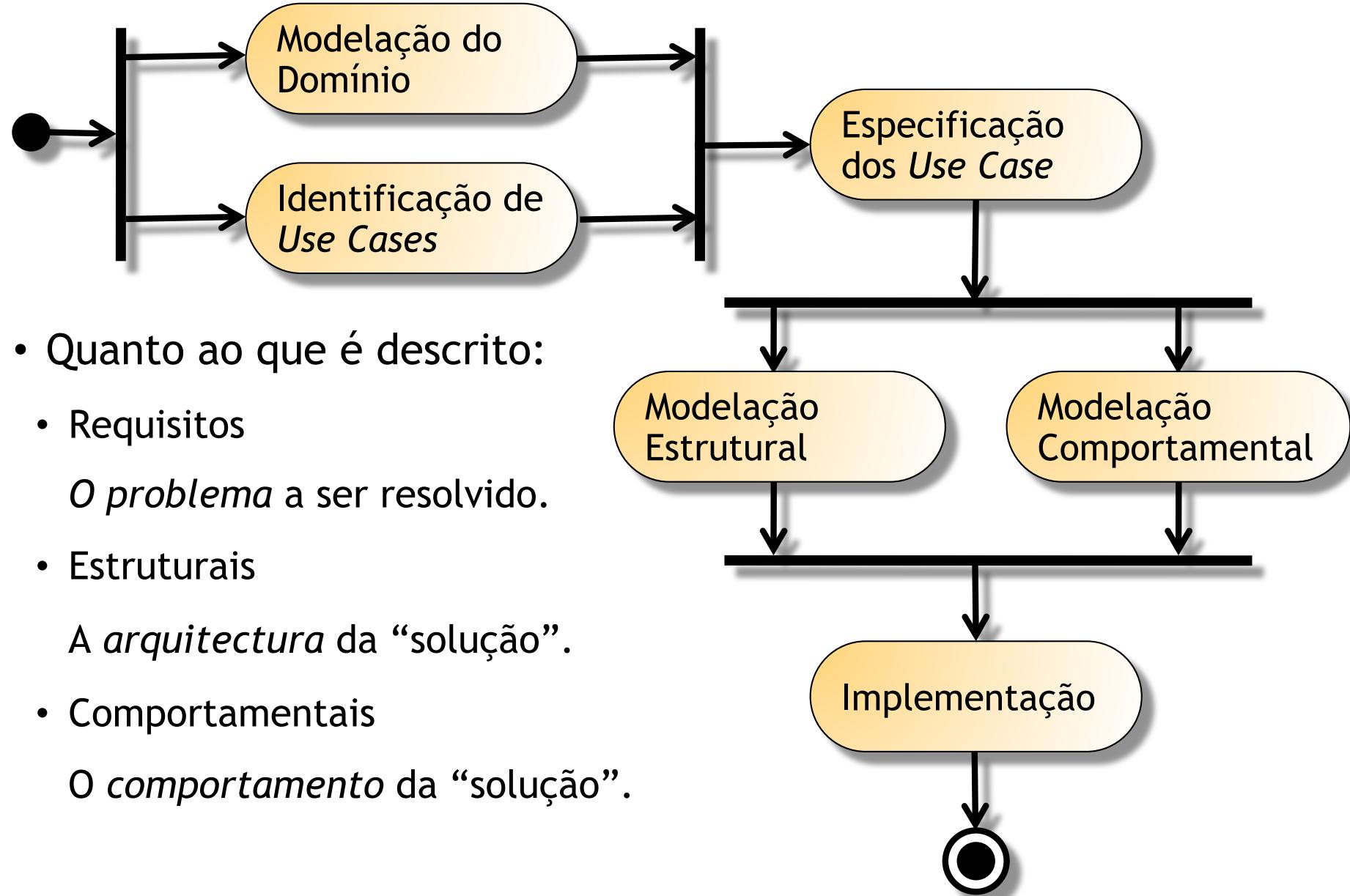
Utilizados para definir comportamentos adequados do sistema.

- Descritivos

Utilizados para descrever a estrutura e comportamento do sistema.

De uma forma geral, os modelos que vamos usar serão **descritivos**.

# Tipos de Modelos



# UML - Unified Modelling Language

(Booch, Jacobson & Rumbaugh)

- a UML foi pensada para o **desenvolvimento de sistemas orientados aos objectos**, mas é independente das linguagens de programação a utilizar
  - permite explorar o paradigma OO
  - decisão sobre tecnologia a utilizar pode ser adiada (Java? .NET?)
  - cf. riscos tecnológicos
- a UML possibilita o **trabalho a diferentes níveis de abstracção**
  - adapta-se às várias fases de desenvolvimento
  - facilita comunicação e análise
  - cf. riscos de requisitos
- a UML não é uma linguagem, mas **uma família de linguagens gráficas** para modelar e construir sistemas software
  - inclui modelos para as diferentes fases do desenvolvimento



# UML - Unified Modelling Language

- a UML não é um processo de desenvolvimento de software, mas pode ser utilizado com diferentes processos
  - é possível escolher o processo mais adequado à equipa / projecto
- a UML é uma *norma* mantida pelo OMG (Object Management Group)
  - não vai acabar *amanhã*
  - existe um esforço de formalização da linguagem
  - cf. riscos tecnológicos
- a UML é suportada por **ferramentas**
  - *Rational Rose (IBM), Together (Borland), Visual Paradigm, Poseidon, etc., etc.*
  - ... *mas também papel e lápis!!*



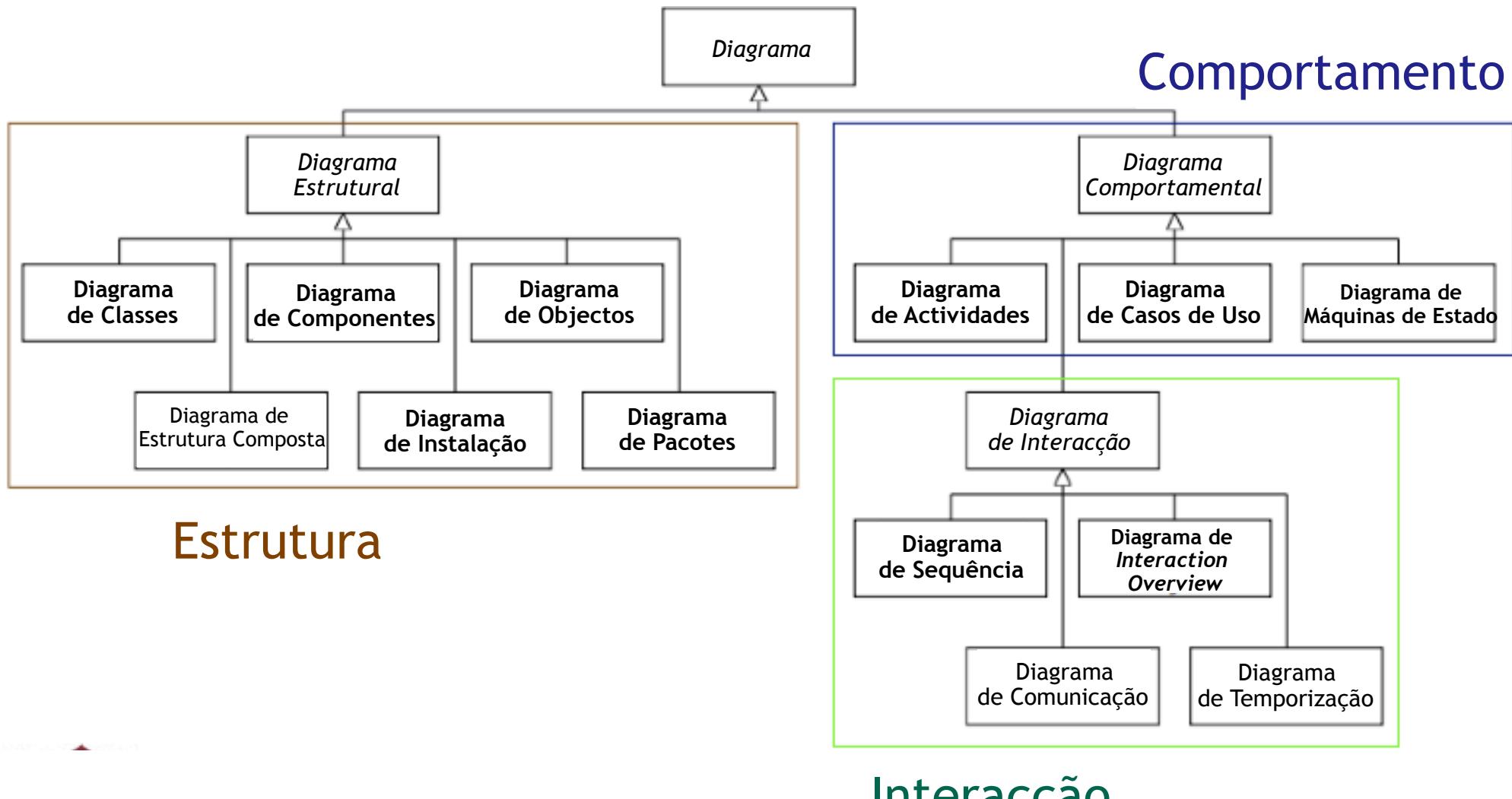
Visual Paradigm  
for **UML**

# Breve história da UML

- Anos 60
  - Simula 67 é a primeira linguagem orientada aos objectos;
- Anos 70
  - Aparece o Smalltalk;
- Anos 80
  - as linguagens orientadas aos objectos (OO) tornam-se utilizáveis: Smalltalk estabiliza; surgem o Objective C, C++, Eiffel, CLOS, etc.
  - Finais dos anos 80 - surgem as primeiras metodologias de modelação OO
- Anos 90
  - existem dezenas de metodologias de modelação OO: Shlaer/Mellor, Coad/Yourdon, Booch, OMT (Rumbaugh), OOSE (Jacobson), etc.
  - meados dos anos 90 - começam as tentativas de unificação dos métodos
  - 1994 - Rumbaugh junta-se a Booch na Rational Software Corporation
  - 1995 - Booch e Rumbaugh apresentam a versão 0.8 do Unified Method (viria depois a mudar de nome para Unified Modelling Language); Jacobson junta-se a Booch e Rumbaugh
  - 1996 - o OMG (Object Management Group) pede propostas para uma norma de modelação OO
  - Setembro, 1997 - a Rational, em conjunto com outras empresas (HP, IBM, Oracle, SAP, Unisys, ...), submete a UML 1.0 ao OMG como proposta de norma (existiram outras propostas)
  - Novembro, 1997 - a UML é aprovado como norma OO pelo OMG; o OMG assume a responsabilidade do desenvolvimento da UML
- Sec XXI
  - 2005 - a UML 1.4.2 é aceite como norma ISO (ISO/IEC 19501); o OMG adopta a UML 2.0
  - 2007-2011 - UML 2.1.1 a UML 2.4.1
  - 2012 - UML 2.5b1

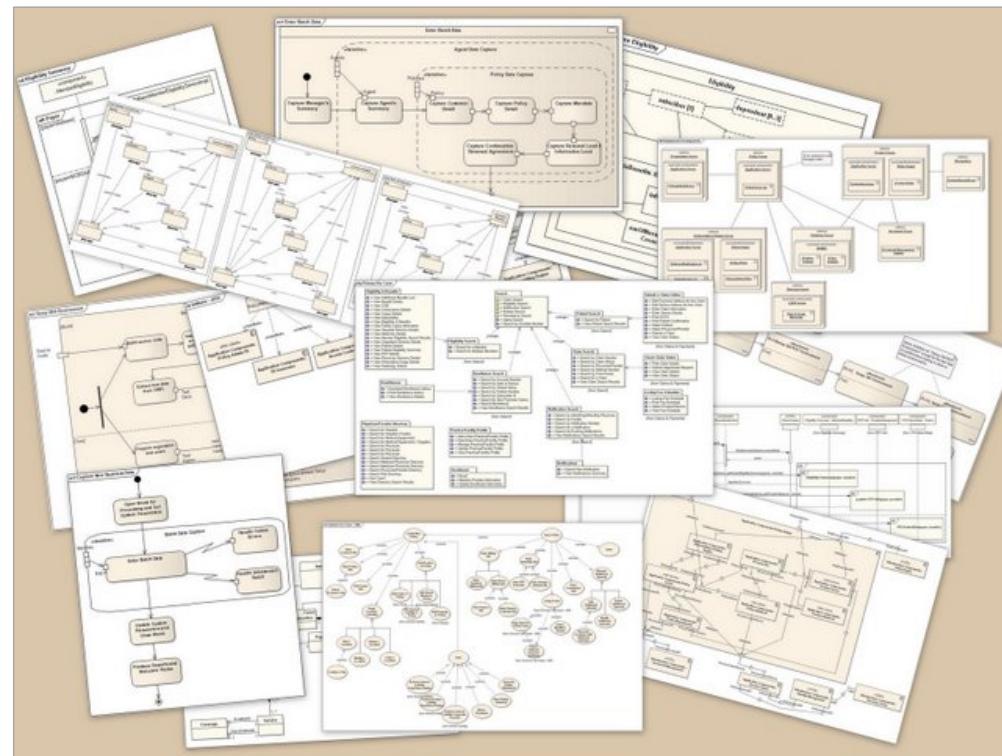
# Diagramas da UML 2.x

- Em UML, para descrever modelos, utilizámos diagramas...

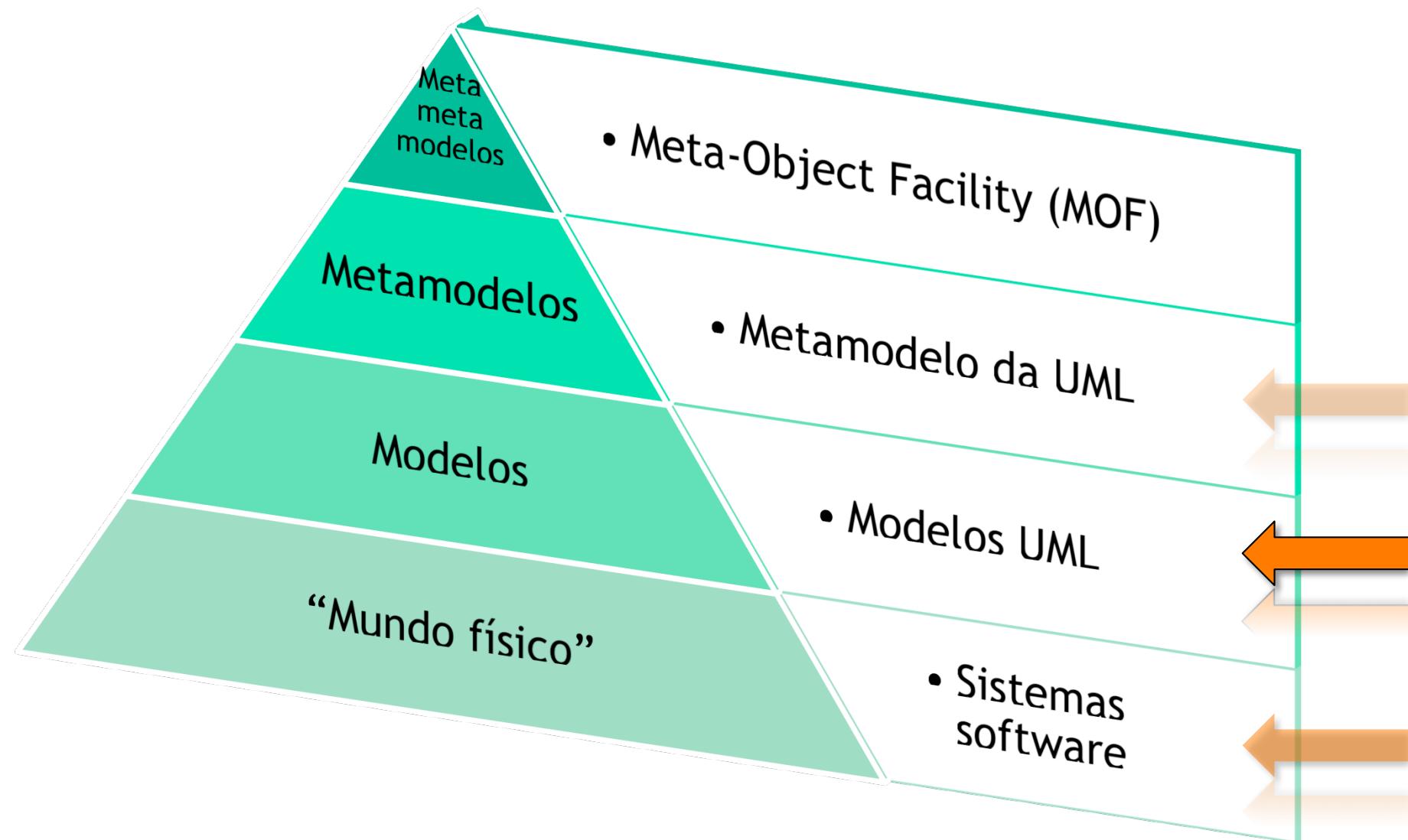


# UML - que linguagem?

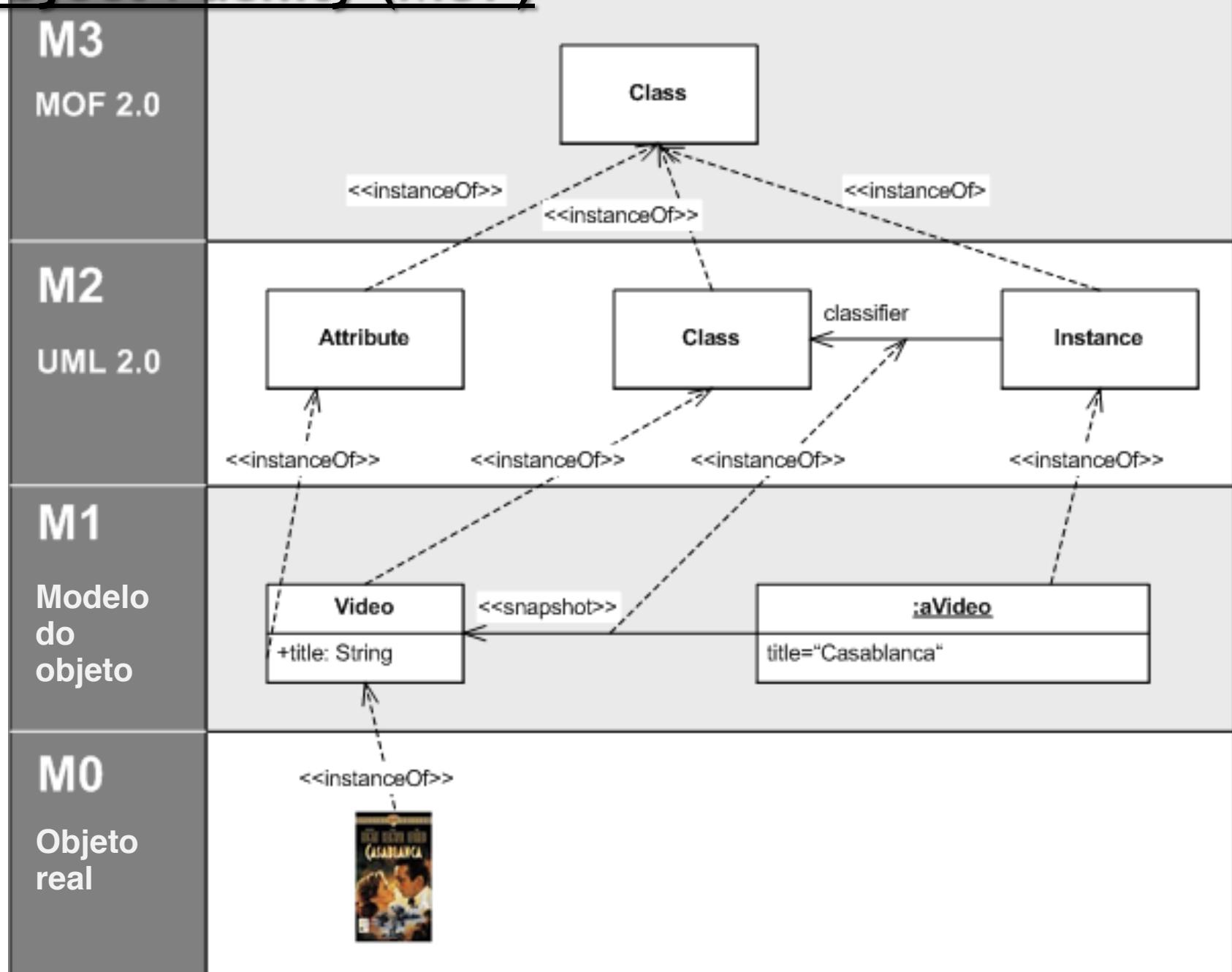
- Uma linguagem de modelação tem:
  - léxico – regras que definem quais os elementos válidos da linguagem;
  - sintaxe – regras que definem quais as combinações válidas dos elementos;
  - semântica – regras que definem o significado dos modelos legais.
- A linguagem UML é diagramática – modelos são expressos com diagramas (+ texto).



# UML - que linguagem?

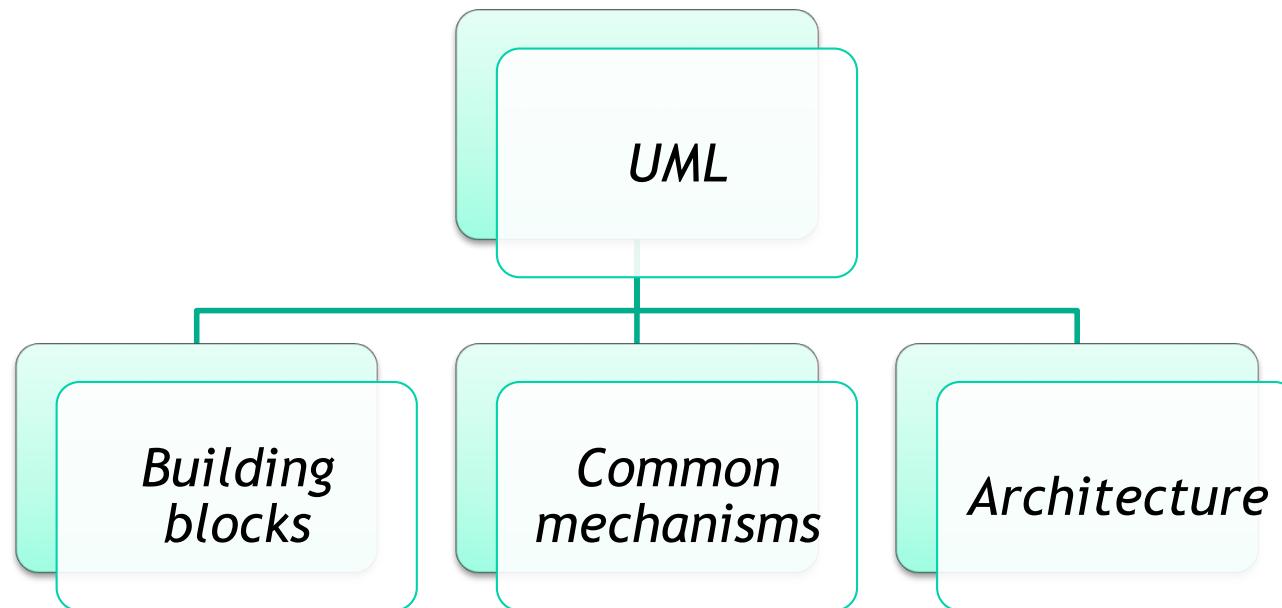


# Meta-Object Facility (MOF)



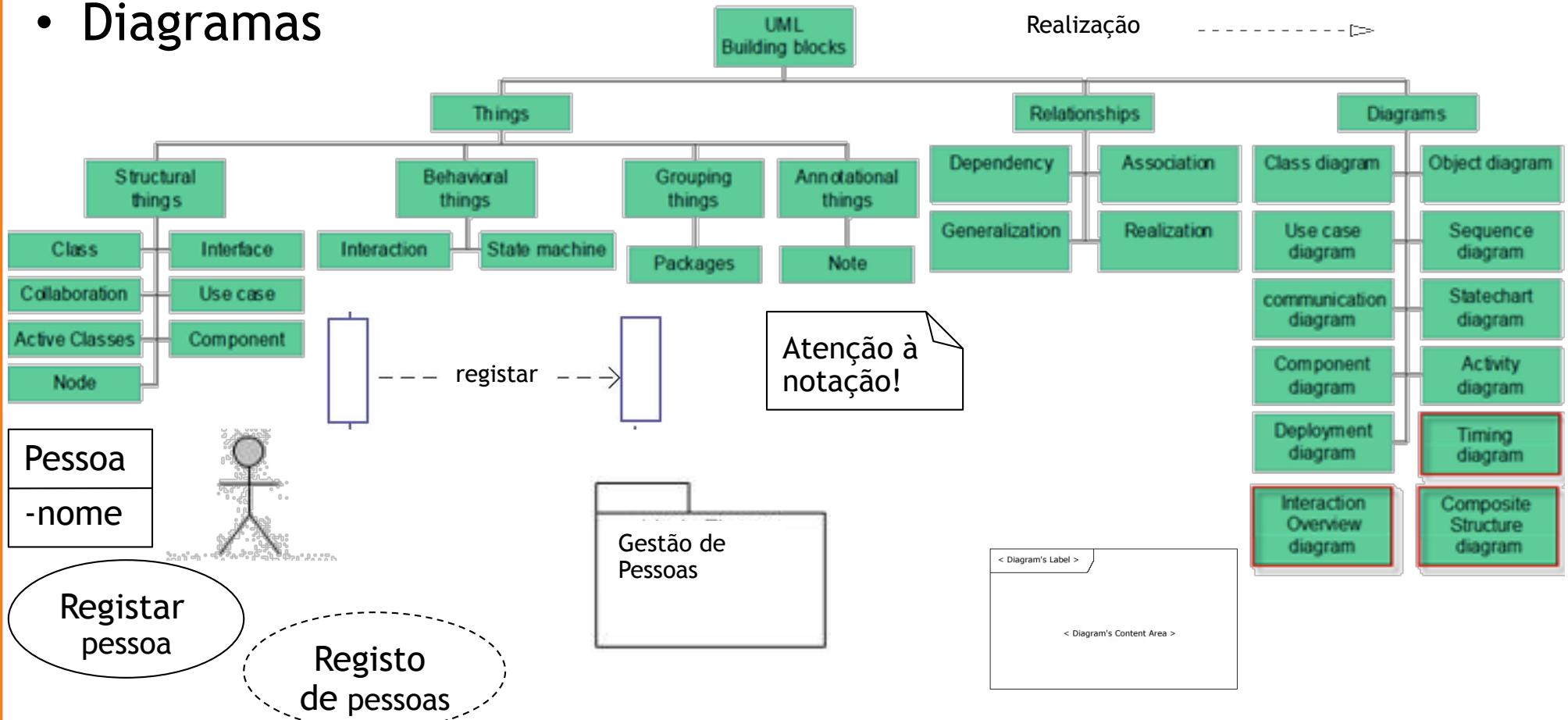
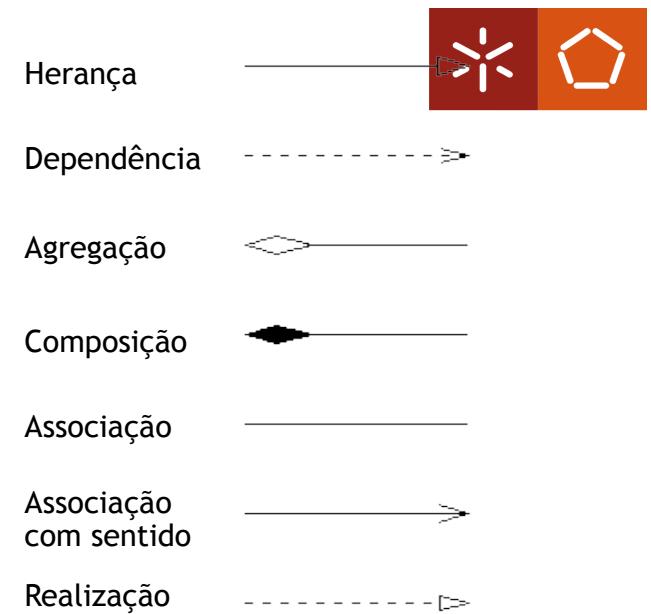
# Modelo Conceptual da UML

- *Building blocks* - elementos base da modelação
- *Common mechanisms* - modos comuns de atingir determinados objetivos
- *Architecture* - a visão que a UML tem da arquitectura dos sistemas



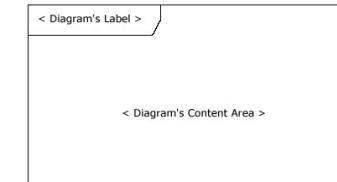
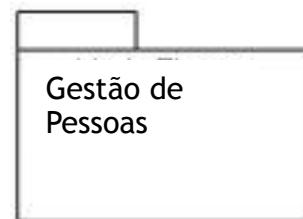
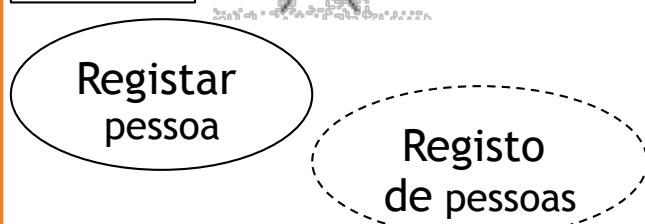
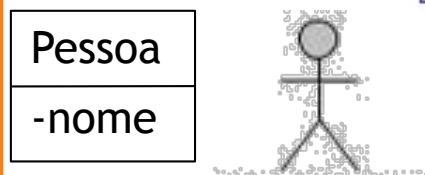
# UML building blocks

- Coisas (*Things*)
- Relacionamentos
- Diagramas

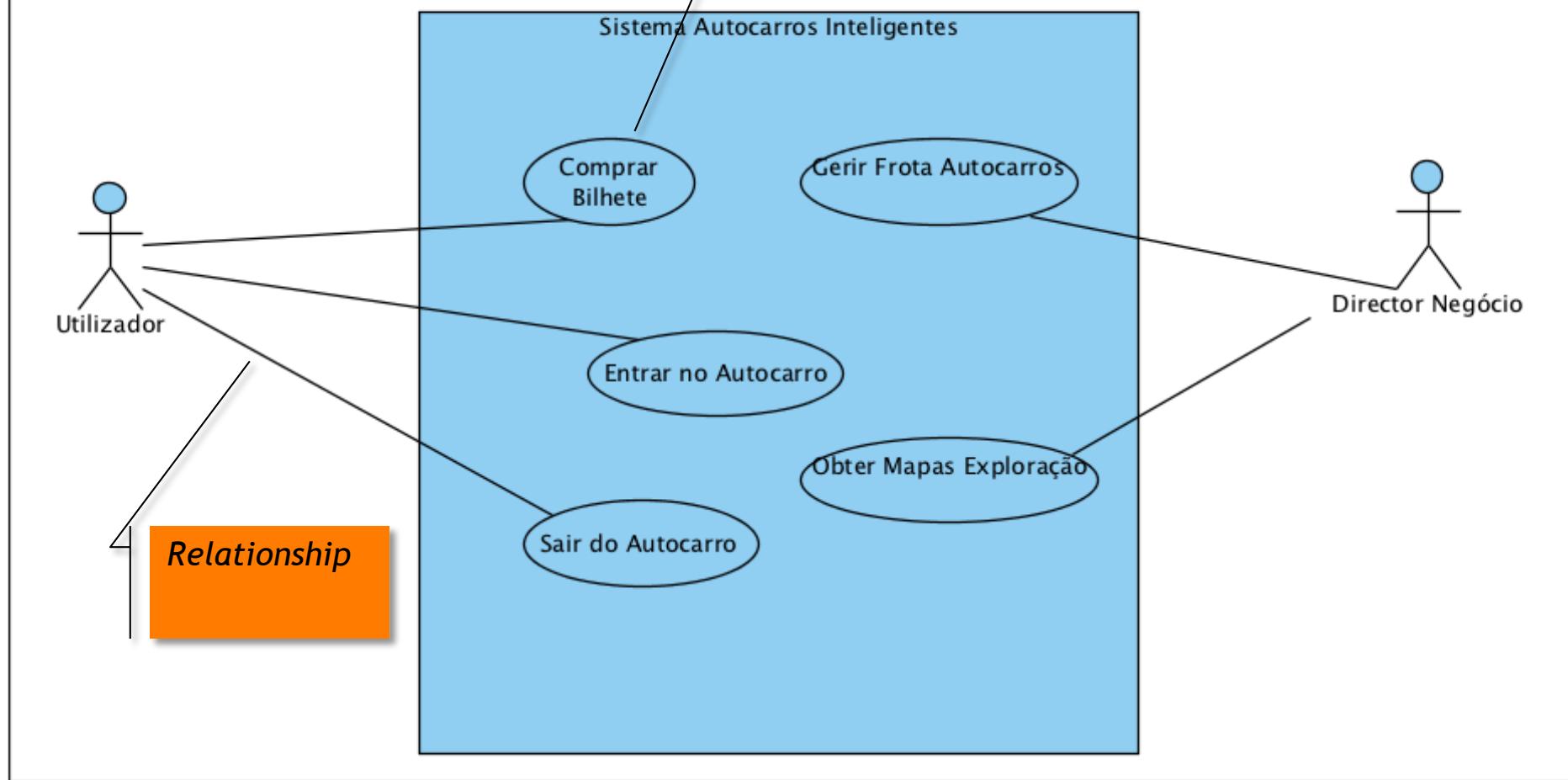


--- registrar --->

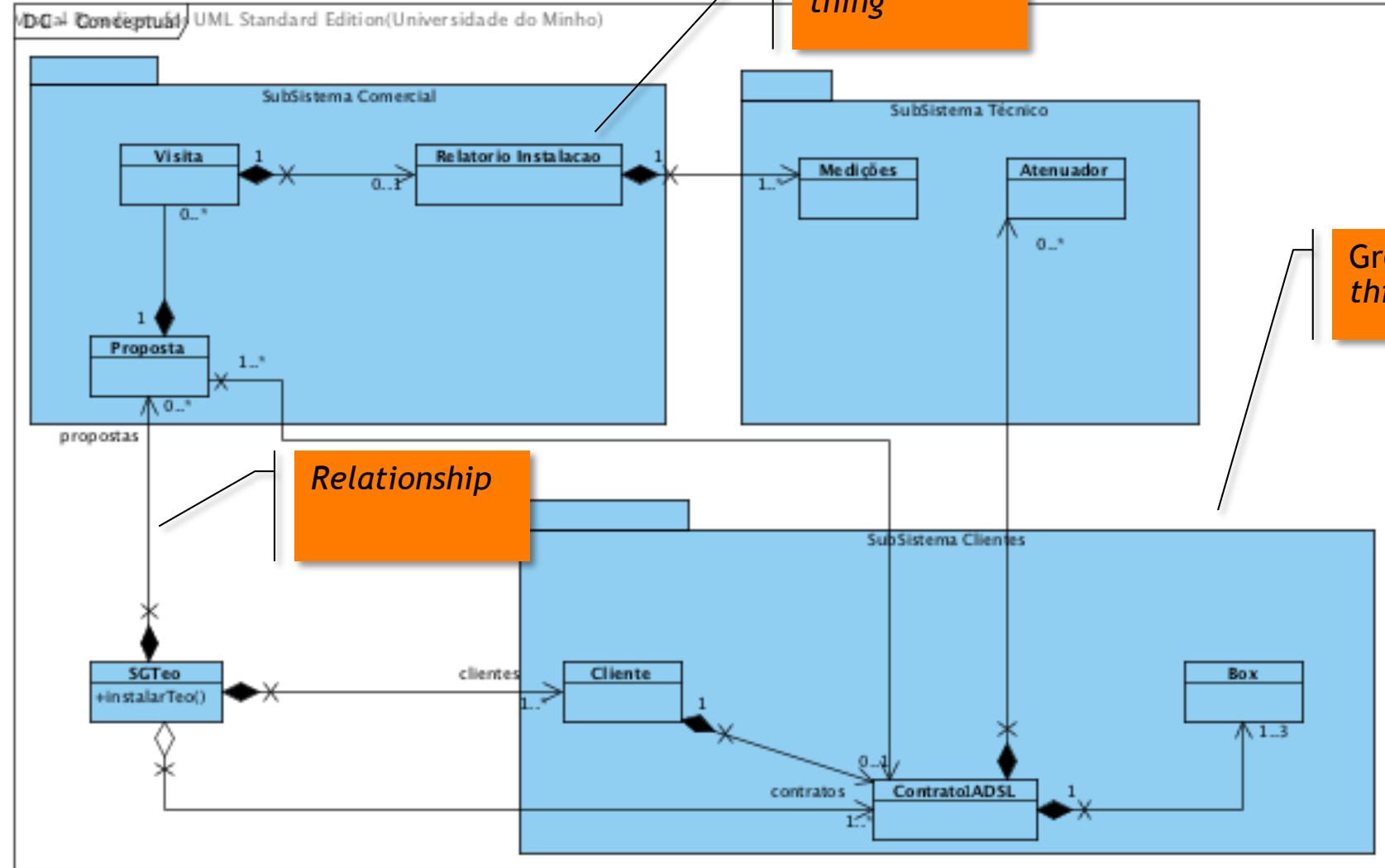
Atenção à  
notação!



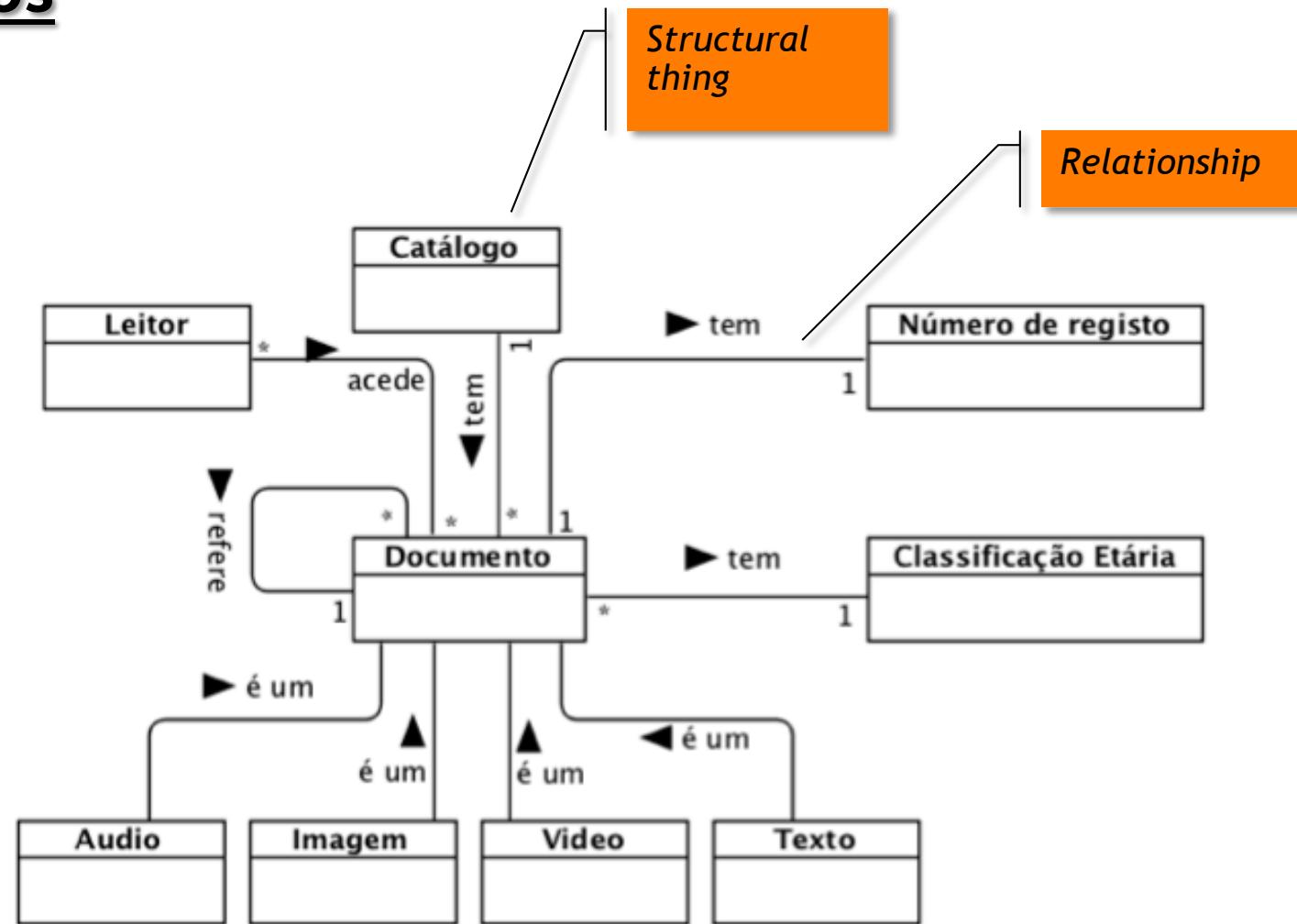
# Exemplos

  
Diagrama  
Structural  
thing  
Diagrama de Contexto

# Exemplos

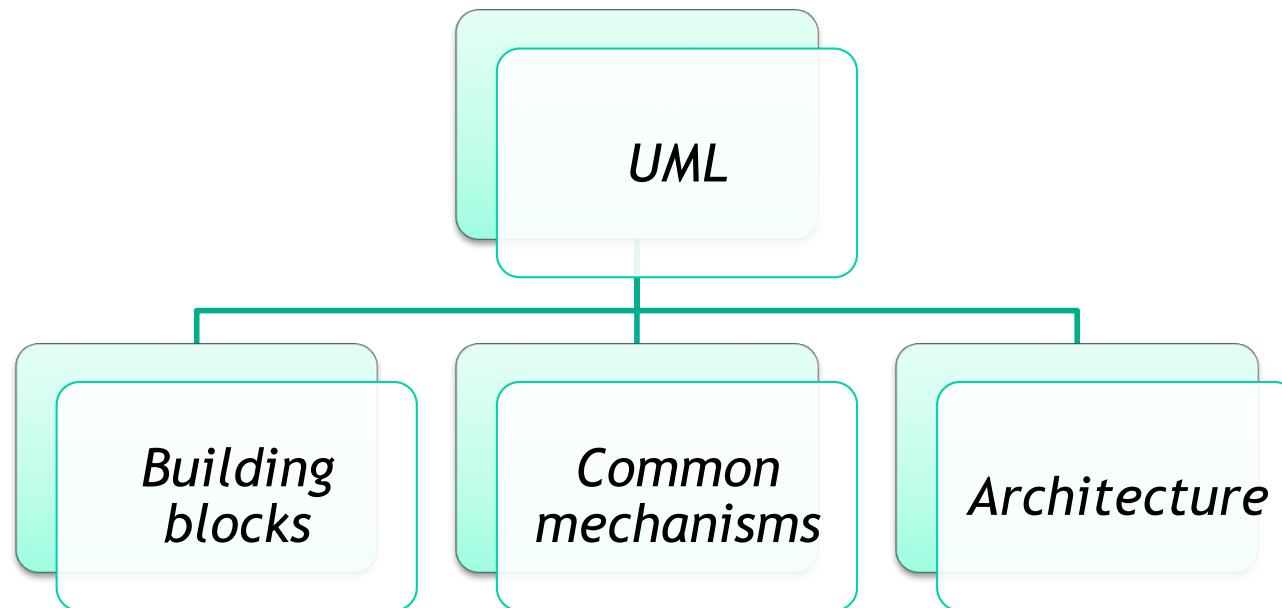


# Exemplos



# Modelo Conceptual da UML

- *Building blocks* - elementos base da modelação
- *Common mechanisms* - modos comuns de atingir determinados objetivos
- *Architecture* - a visão que a UML tem da arquitectura dos sistemas

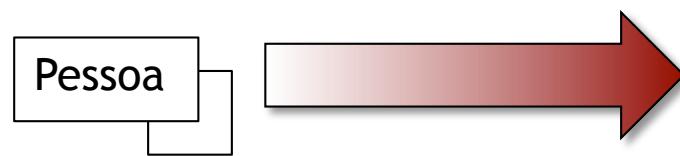


# UML common mechanisms

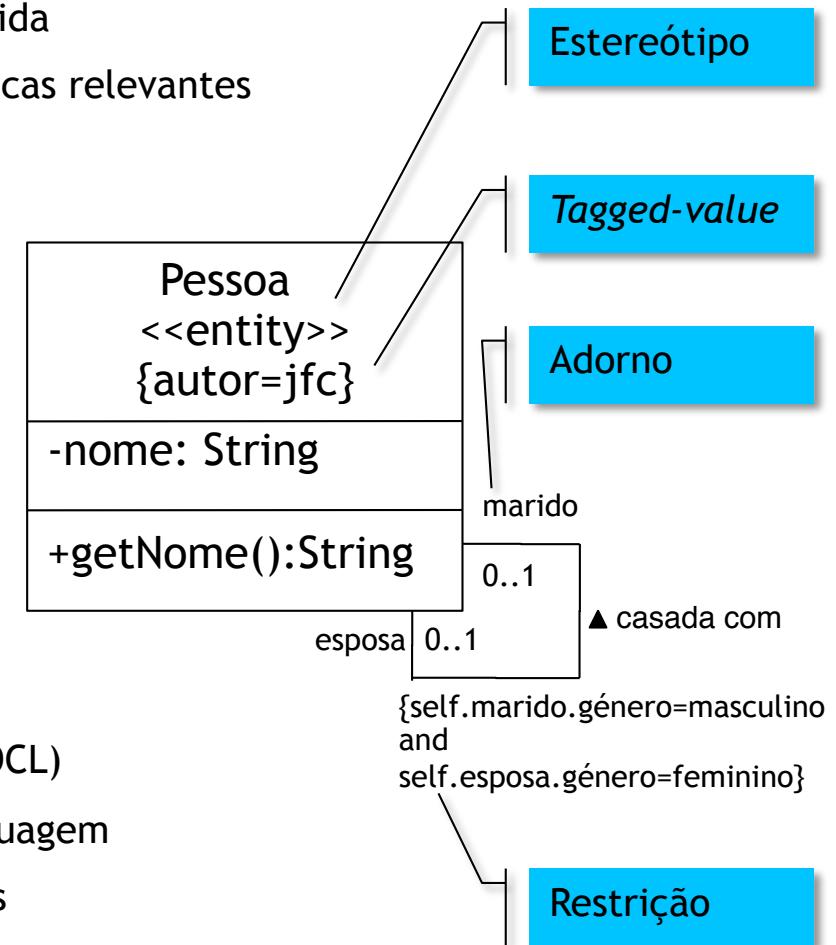
- Especificações
  - Descrições textuais da semântica de um elemento
- Adornos(*Adornments*)
  - Permitem controlar a quantidade de informação fornecida
  - Só devem ser mostrados os que apresentam características relevantes

## Erro de Principiante!

“death by diagrams”: modelos com diagramas a mais e especificação a menos...

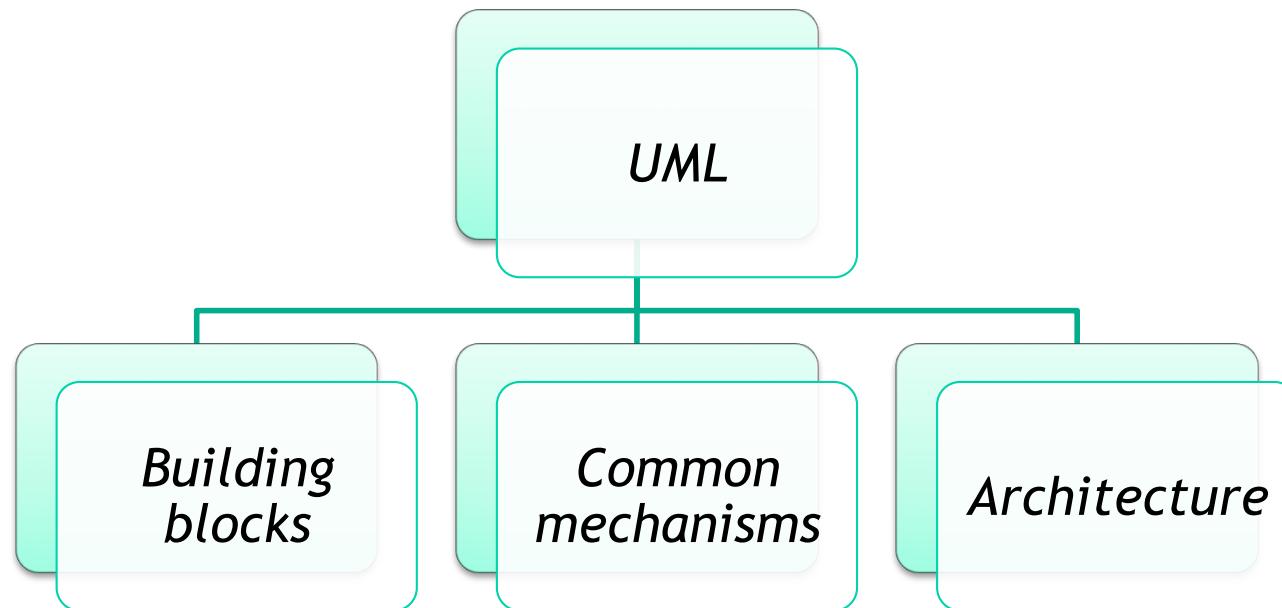


- Divisões comuns
  - *Classifiers* e instâncias
  - Interfaces e implementações
- Mecanismos de extensão
  - Restrições - propriedades para especificar semântica (OCL)
  - Estereótipos - permitem extender o vocabulário da linguagem
  - *Tagged-values* - permitem adicionar novas propriedades



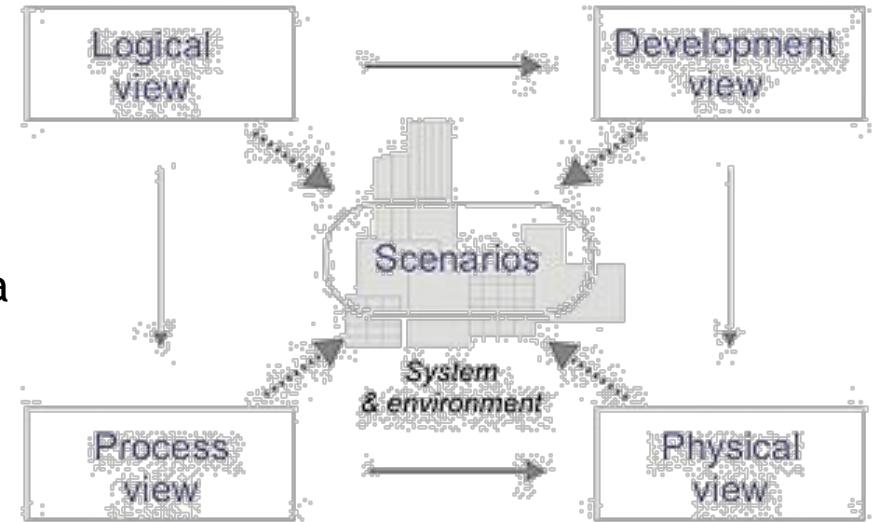
# Modelo Conceptual da UML

- *Building blocks* - elementos base da modelação
- *Common mechanisms* - modos comuns de atingir determinados objetivos
- *Architecture* - a visão que a UML tem da arquitectura dos sistemas



# UML - Arquitectura 4+1

- **Vista lógica**
  - Captura o vocabulário do domínio do problema
  - Mostra como os objectos e classes implementam o comportamento pretendido
- **Vista do processo**
  - Semelhante, mas focada nos processos internos (comportamento em *runtime*)
- **Vista da implementação/desenvolvimento**
  - modela os ficheiros e componentes que constituem a implementação do sistema (captura de dependências, gestão de configurações)
- **Vista da instalação/física**
  - modela a instalação dos componentes em nodos físicos
- **Cenários / Vista de Casos de uso**
  - Captura os requisitos do sistema



(Kruchten, 1995)



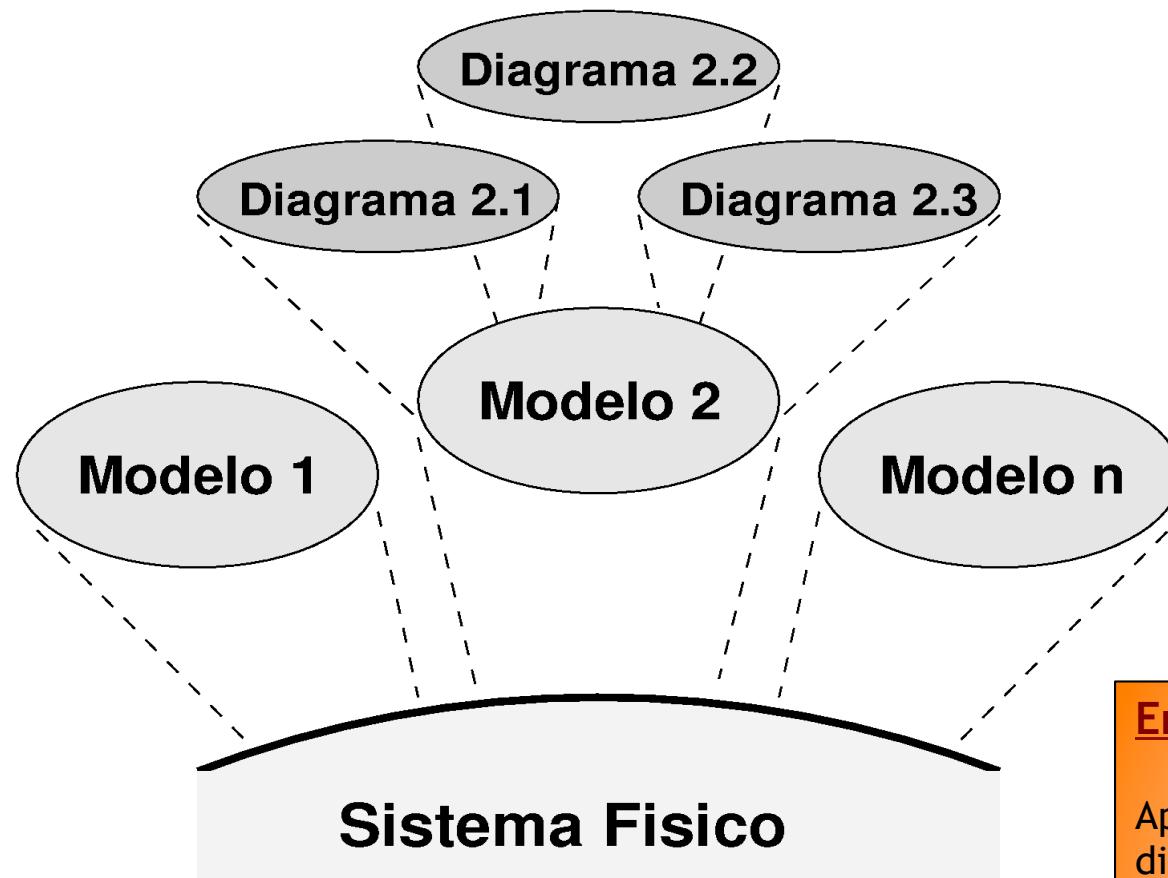
## Modelos vs. diagramas

- A UML suporta as cinco vistas anteriores com diferentes tipos de modelos.
- Cada modelo é expresso com um dado tipo de diagrama
- Na UML:
  - Utilizam-se vários modelos para representar uma mesma realidade (os modelos não são a realidade)
  - Utilizam-se vários diagramas para representar um modelo (os diagramas não são os modelos)



# Modelos vs. diagramas

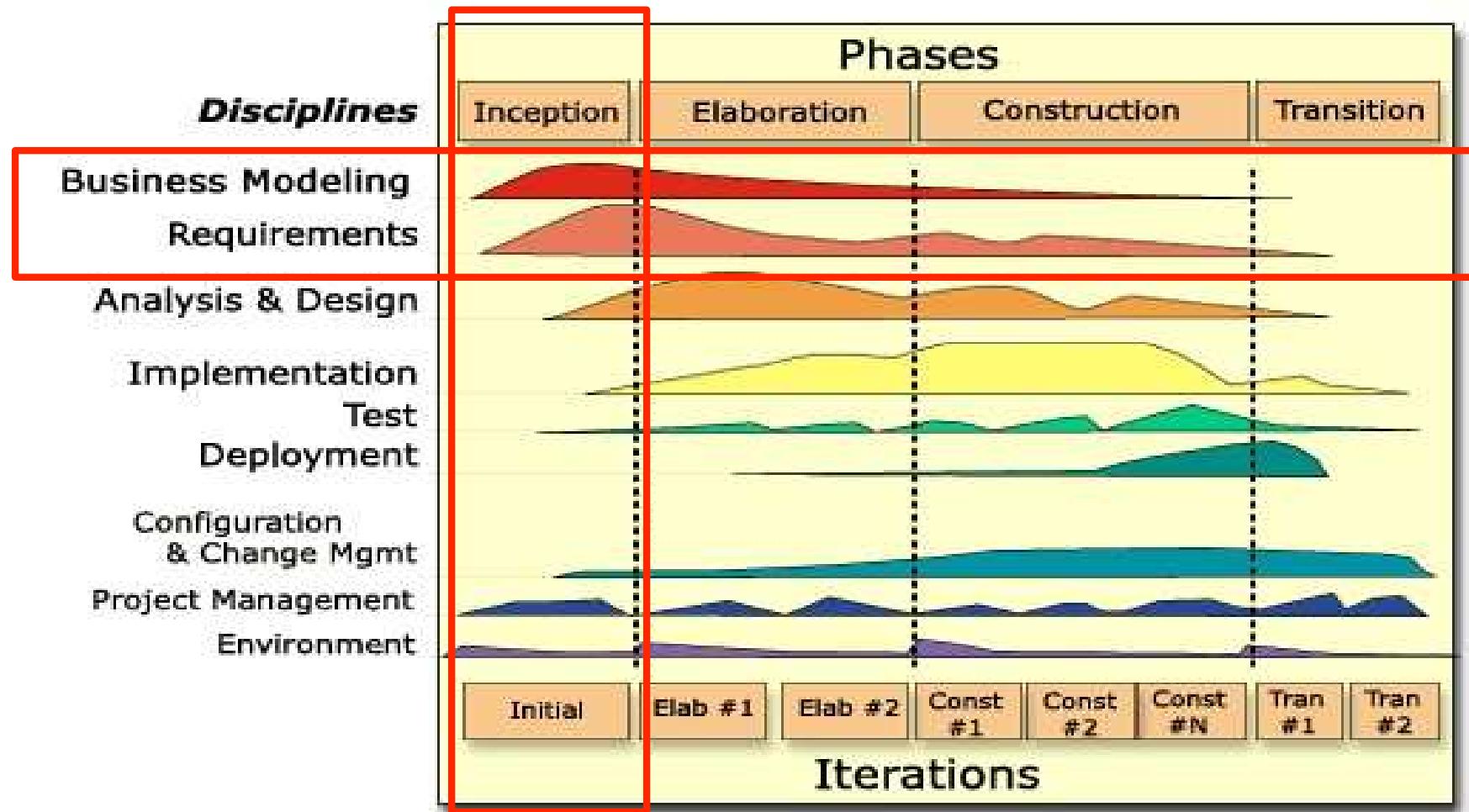
Os diagramas não são os modelos! / Os modelos não são o sistema!



**Erro de Principiante!**

Apagar “coisas” dos  
diagramas e deixá-las no  
modelo.

# O que diz o UP...



# Introdução à UML

## Sumário:

- Vantagens da utilização de modelos
- Problemas com a utilização de modelos
- UML
  - Breve história da UML
  - Meta-modelo da UML