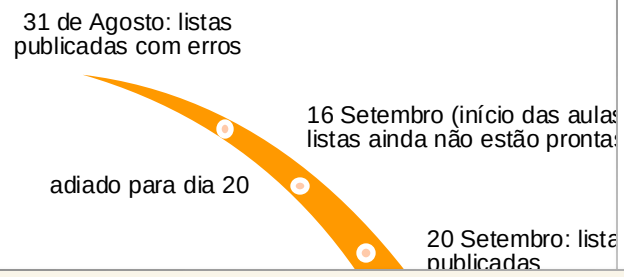
Modelos do Domínio

Colocação de professores em 2004



ECONOMIA

Governo desiste de sistema informático da Compta e avança para colocação manual de professores

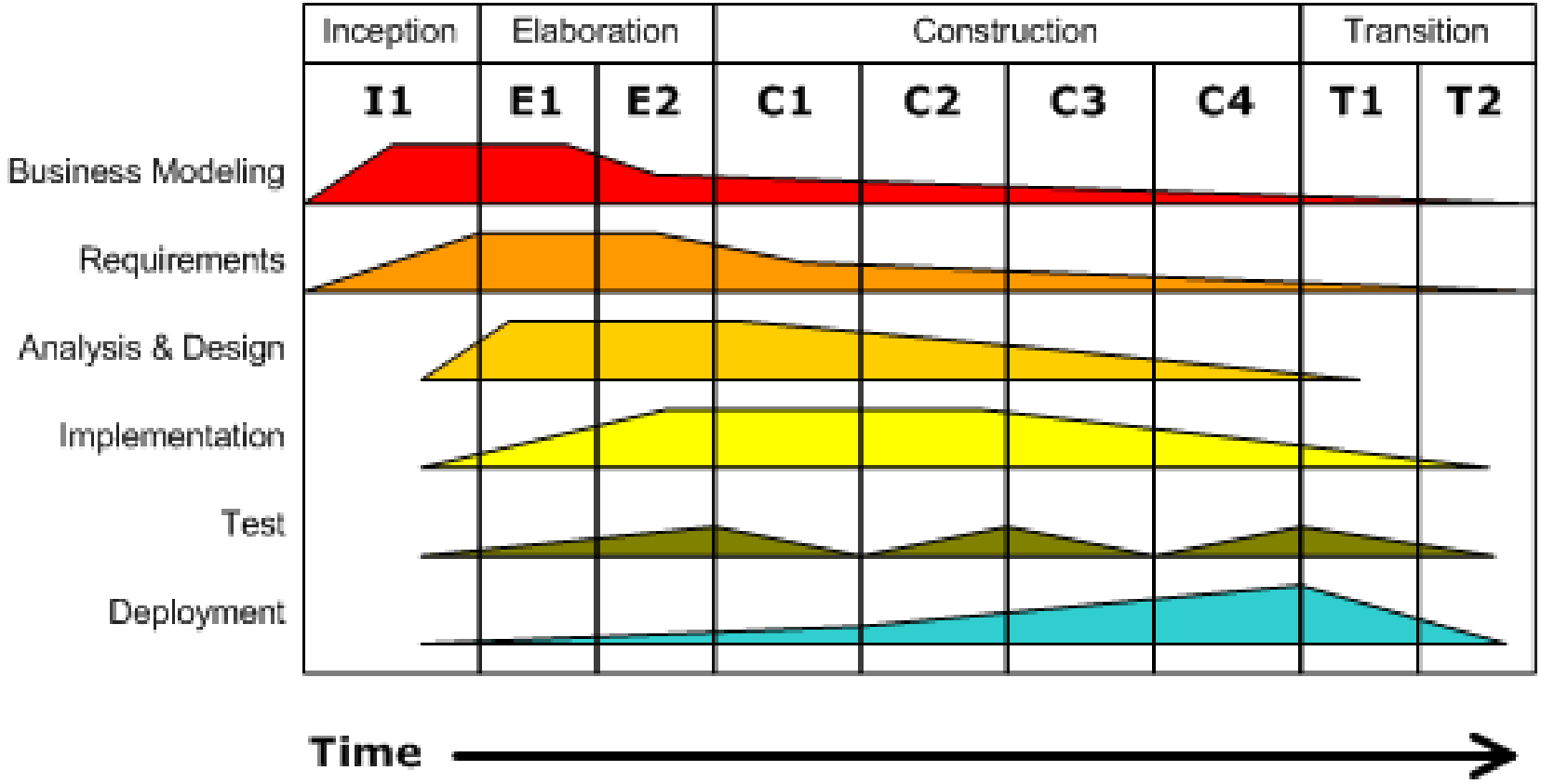
A empresa, que "foi publicamente apontada como principal responsável pelos incidentes no processo de colocação dos professores", recorda que iniciou o seu trabalho em Setembro de 2003, "na sequência de um concurso público em que foram relevantes factores como o preço, os prazos e os conteúdos das propostas".

Porém, "nos meses que se seguiram, e ao longo de todo o processo, (a Compta) foi confrontada múltiplas vezes com alterações das normas, da sua interpretação e critérios aplicáveis, por parte, nomeadamente, da Direcção Geral de Recursos Humanos da Educação" e que "os prazos estabelecidos tornaram-se por isso impossíveis de cumprir".

cerca de 70 mil docentes, tem em mãos 70 processos a decorrer no tribunal administrativo e 1056 recursos, hierárquicos ou de outra

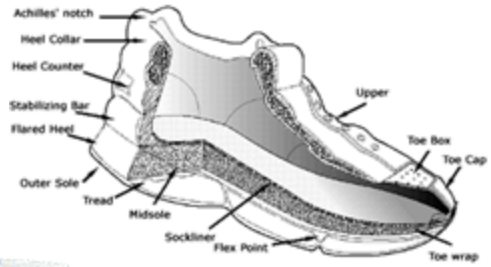
Fases do ciclo de vida do desenvolvimento

- (Rational) Unified Process (RUP)



Modelos do Domínio

- Os “produtos” da Engenharia Informática são produzidos nas mais variadas áreas (domínios) de negócio e da sociedade em geral:
  - Medicina, Calçado, Energia, Aero-espacial, Automóvel, Segurança, Telecomunicações, Gestão, etc., etc., etc.



- Em cada projeto, torna-se necessária uma forma de capturar as informações relevantes sobre o *domínio* do projeto.

Modelos do Domínio

> *"A domain model captures the most important types of objects in the context of the business. The domain model represents the 'things' that exist or events that transpire in the business environment."*
>
> (I. Jacobsen)

- O Modelo de Domínio captura as **Entidades** (*coisas*) do problema e os **Relacionamentos** entre elas
  - Captura o vocabulário do domínio **do problema** – fornece um glossário de termos
  - Fornece uma *framework* conceptual para raciocinar sobre **o problema** – ajuda a pensar
- É uma visão estática **do problema** – permite representar regras de negócio invariantes no tempo
- **É a base para a análise de requisitos**

Um exemplo

> *"A domain model captures the most important types of objects in the context of the business. The domain model represents the 'things' that exist or events that transpire in the business environment."*
>
> (I. Jacobsen)

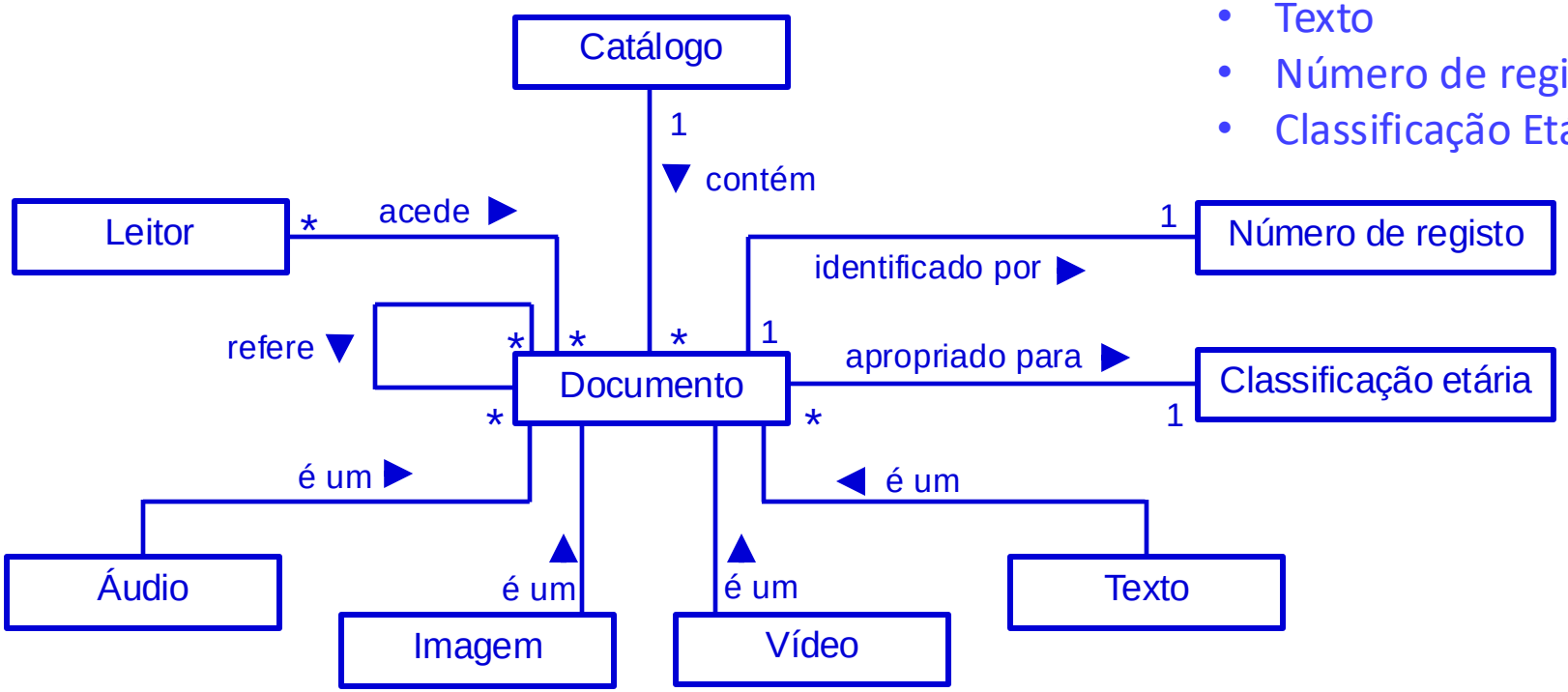
O sistema a desenvolver deverá gerir catálogos. Um catálogo contém documentos que são acedidos por leitores. Um documento pode ser do tipo áudio, imagem, vídeo ou texto e tem sempre um número de registo e uma classificação etária. Cada documento pode referir outros documentos.

Um exemplo

O sistema a desenvolver deverá gerir catálogos. Um catálogo contém documentos que são acedidos por leitores. Um documento pode ser do tipo áudio, imagem, vídeo ou texto e tem sempre um número de registo e uma classificação etária. Cada documento pode referir outros documentos.

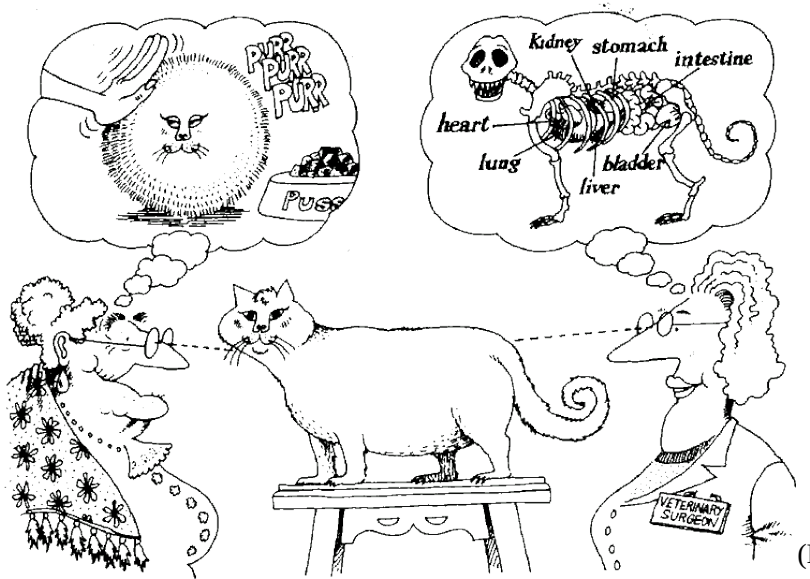
Entidades:

- Catálogo
- Documento
- Leitor
- Audio
- Imagem
- Vídeo
- Texto
- Número de registo
- Classificação Etária



Modelos de Domínio são…

- Simplificações da realidade — representações abstractas, efectuadas de um determinado ponto de vista
  - **Abstracção**:
    - O processo de remover informação de uma descrição para ficarem apenas os aspectos relevantes
- Úteis para descrever e analisar os problemas que queremos resolver



(Booch)

Porquê modelar?

Vantagens da utilização de modelos

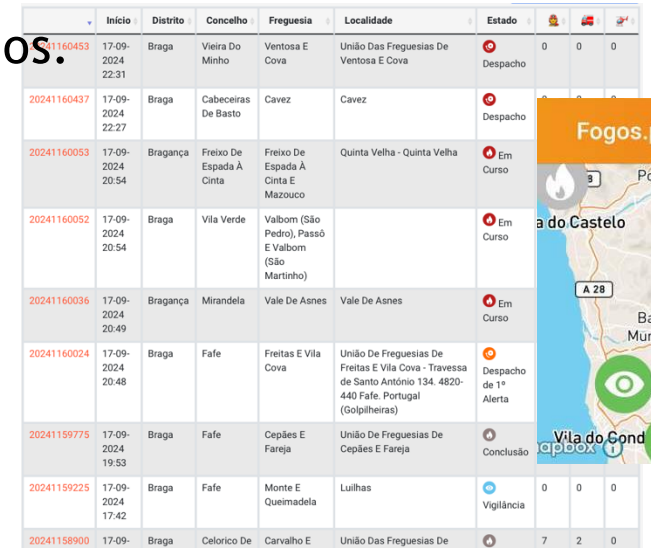
- Auxiliam a **compreender** a realidade
- Ajudam a **comunicar** ideias de forma simplificada.
- Ajudam a **documentar** as decisões tomadas durante o desenvolvimento.

- Modelos com uma semântica precisa (rigorosa) suportam **automação.**

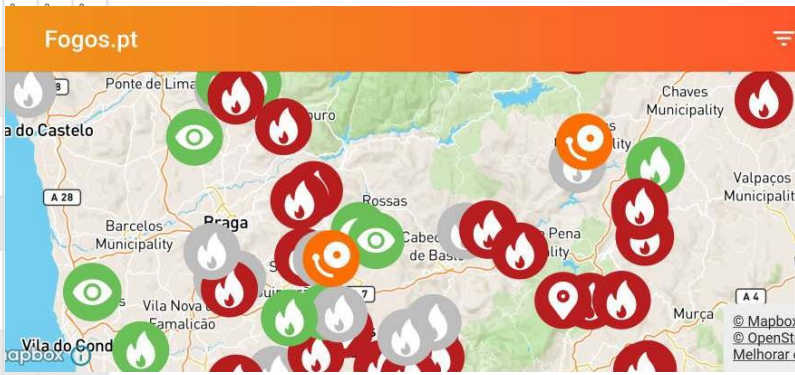
Porquê modelar?

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.



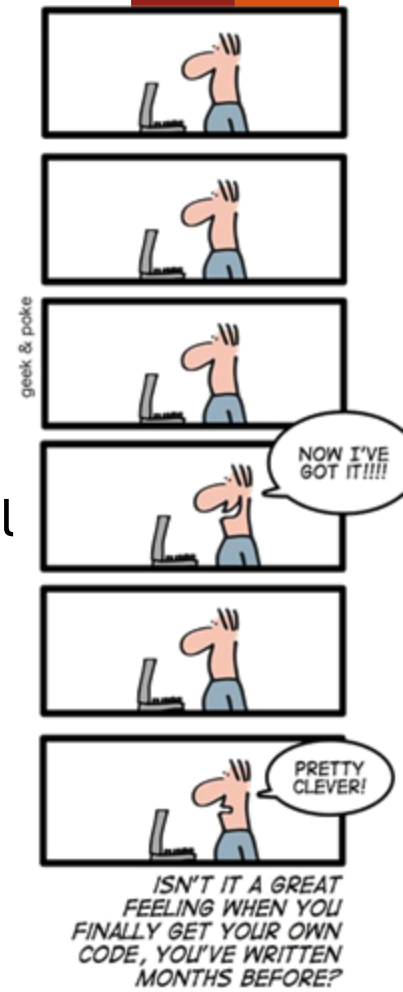
	Início	Distrito	Concelho	Freguesia	Localidade	Estado			
20241160453	17-09-2024 22:31	Braga	Vieira Do Minho	Ventosa E Cova	União Das Freguesias De Ventosa E Cova	Despacho	0	0	0
20241160437	17-09-2024 22:27	Braga	Cabeceiras De Basto	Cavez	Cavez	Despacho			
20241160053	17-09-2024 20:54	Bragança	Freixo De Espada À Cinta	Freixo De Espada À Cinta E Mazouco	Quinta Velha - Quinta Velha	Em Curso			
20241160052	17-09-2024 20:54	Braga	Vila Verde	Valbom (São Pedro), Passô E Valbom (São Martinho)		Em Curso			
20241160036	17-09-2024 20:49	Bragança	Mirandela	Vale De Asnes	Vale De Asnes	Em Curso			
20241160024	17-09-2024 20:48	Braga	Fafe	Freitas E Vila Cova	União De Freguesias De Freitas E Vila Cova - Travessa de Santo António 134. 4820-440 Fafe. Portugal (Golpilheiras)	Despacho de 1º Alerta			
20241159775	17-09-2024 19:53	Braga	Fafe	Cepães E Fareja	União De Freguesias De Cepães E Fareja	Conclusão			
20241159225	17-09-2024 17:42	Braga	Fafe	Monte E Queimadela	Luilhas	Vigilância	0	0	0
20241158900	17-09-	Braga	Celorico De	Carvalho E	União Das Freguesias De		7	2	0



Porquê modelar?

Vantagens da utilização de modelos

- Ajudam a documentar as decisões tomadas durante a análise e desenvolvimento.
  - Os modelos desenvolvidos constituem uma base documental para a compreensão do processo de desenvolvimento
  - "*Thinking made public*".
- Modelos com uma semântica precisa (rigorosa) permitem automação
  - Da produção de código
  - Da análise / teste
  - …



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**?
  - É possível passar, de forma (semi-)automática, dos modelos para o código.
  - Código produzido é de melhor qualidade.

Modelos de Domínio

- O Modelo de Domínio representa uma visão do problema –
  - **Não inclui** o sistema (software/solução) a desenvolver
- O Modelo de Domínio apresenta uma visão estática
  - Não representa fluxos de dados
- As entidades no Modelo de Domínio são **apenas** candidatas a serem classes na solução
  - Não pensem na solução, pensem no problema!
- As entidades no modelo de domínio podem ter atributos, mas devem ser de tipos simples (números, strings, etc.) e **nunca** outras entidades
  - Na dúvida, optar por entidades e relacionamentos, em vez de atributos

Algumas notas sobre entidades

- Entidades no modelo de domínio correspondem a “substantivos” na descrição

- Algumas regras para ponderar rejeição de entidades (a partir dos substantivos)
  - É sinónimo de outra entidade?
    - E.g. documento vs. ficheiro
  - Está fora do âmbito da análise?
    - E.g. “O *Sistema a desenvolver* deverá gerir…”
  - Refere-se a relações entre outras entidades?
    - “referir outros documentos” vs. “referir *um conjunto de* outros documentos”
  - É fruto do estilo de escrita?
    - “Um documento pode ser *do tipo* áudio, imagem, vídeo ou texto ”

Algumas notas sobre relações

- Relações no modelo de domínio correspondem a “verbos” na descrição
  - Não correspondem a ações no tempo mas a relações persistentes
  - Tal como para as entidades, são apenas candidatas a existirem na arquitetura da solução.

- Relação “é um(a)”
  - Explicita relação de tipagem
  - Representação posterior em OO
    - Herança – classe / subclasse
    - Realização – classe / interface
    - Atributo na classe

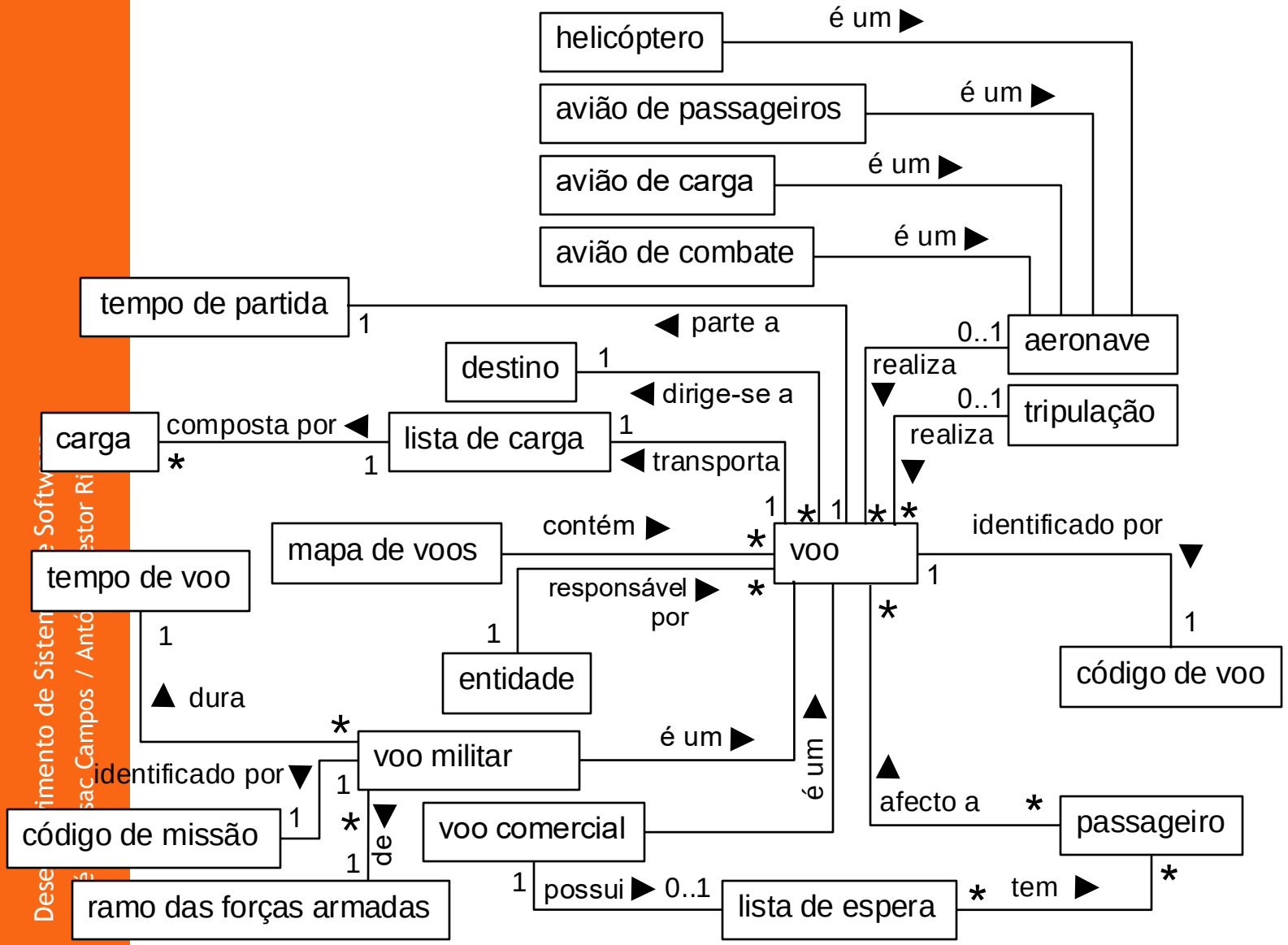
Num Mapa de Voos, cada voo é identificado por um código de voo, tem uma entidade responsável, um conjunto de passageiros afetos a si (caso seja um voo comercial, poderá ter, ou não, uma eventual lista de espera de passageiros), um conjunto de cargas a embarcar (definida numa lista de carga de produtos), um destino, e um tempo de partida. Uma aeronave específica capaz de realizar tal voo e uma tripulação, ser-lhe-ão posteriormente associadas também.

Um voo comercial é o mais usual mas existem também voos militares. Esses, têm a si associados a seguinte informação adicional: tempo de voo, ramo das forcas armadas e código de missão para comunicação (ex.º DELTA77).

As aeronaves são de diversos tipos: helicópteros, aviões de passageiros, aviões de carga, ou aviões de combate.

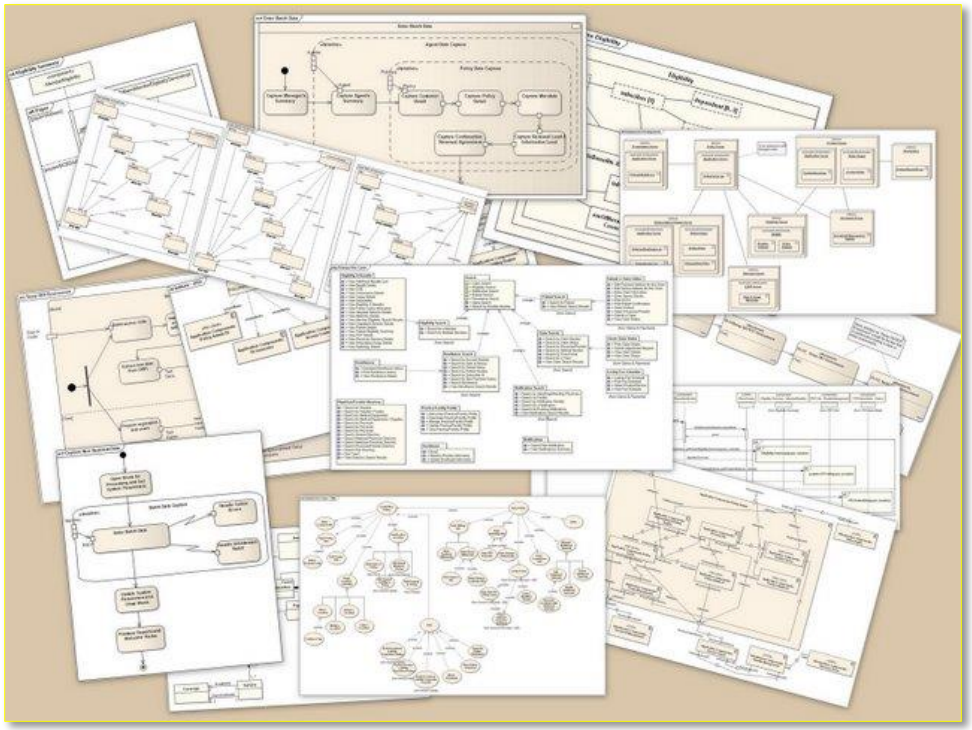
têm a si associados a seguinte informação adicional: tempo de voo, ramo das forcas armadas e código de missão para comunicação (ex.º DELTA77).

As aeronaves são de diversos tipos: helicópteros, aviões de passageiros, aviões de carga, ou aviões de combate.



Que linguagem de modelação? UML!

- 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).



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
  - 2007-2017 – UML 2.1.2 a UML 2.5.1
  - 2019 - ISO/IEC 19501 revisto e confirmado

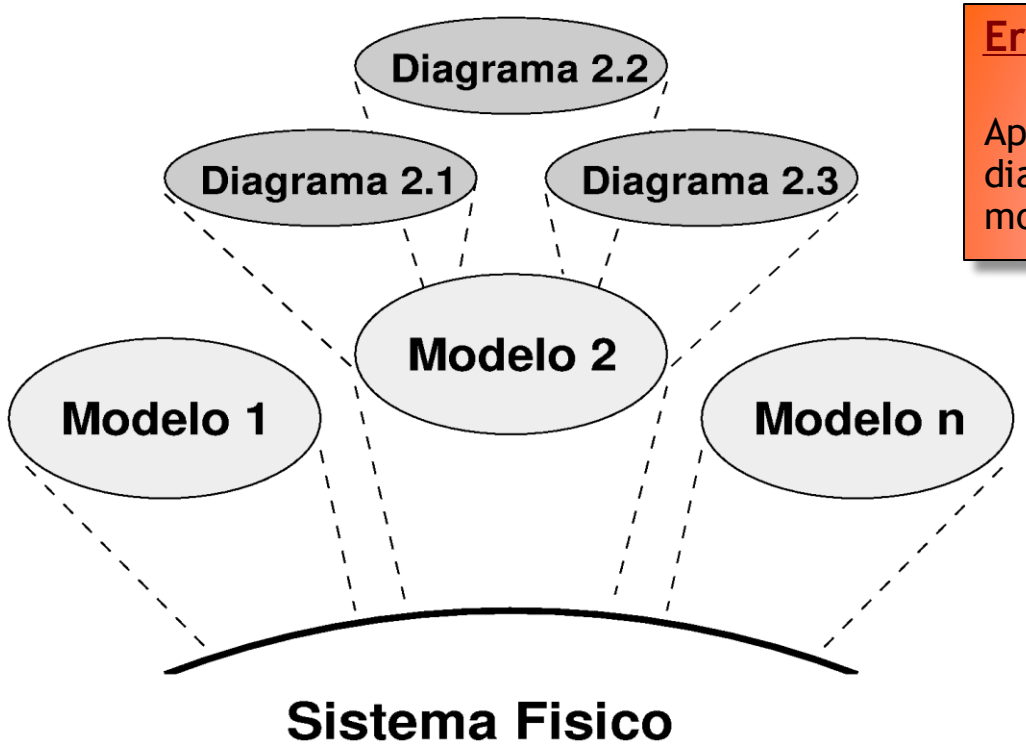
OBJECT MANAGEMENT GROUP
www.omg.org

Relevância da UML

- **Comunicação**: uma linguagem universal para visualizar **sistemas complexos** de software, facilita comunicação entre partes técnicas e não técnicas.
- **Documentação**: auxilia na compreensão e manutenção do software ao longo do tempo.
- **Design e Análise**: ajudam a tomar decisões informadas sobre a estrutura e o comportamento do sistema.
- **Desenvolvimento Orientado a Modelos**: suporta abordagens de desenvolvimento orientado a modelos, melhorando a consistência e reduzindo a diferença entre o design e a implementação.
- **Sistemas Legados**: facilita a compreensão e atualização de sistemas legados, através da engenharia reversa dos modelos a partir do código.
- **Normas da Indústria**: exigida ou recomendada em algumas indústrias, especialmente em sistemas críticos, como aeroespacial e saúde.

Modelos vs. diagramas

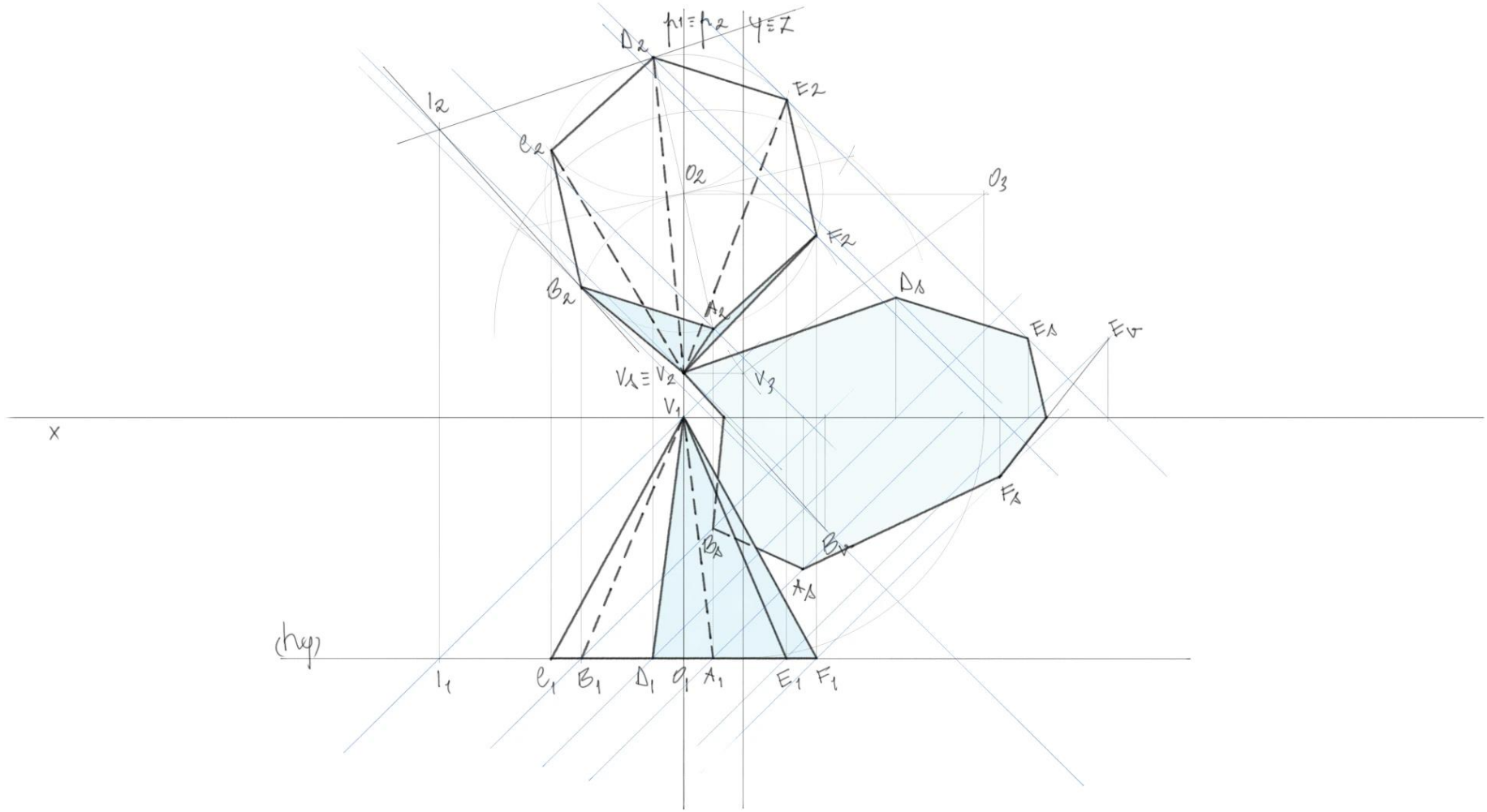
- Na UML:
  - Utilizam-se vários modelos para representar uma mesma realidade (os modelos não são a realidade)
  - Utilizam-se vários diagramas para representar um modelo (os diagramas não são os modelos)



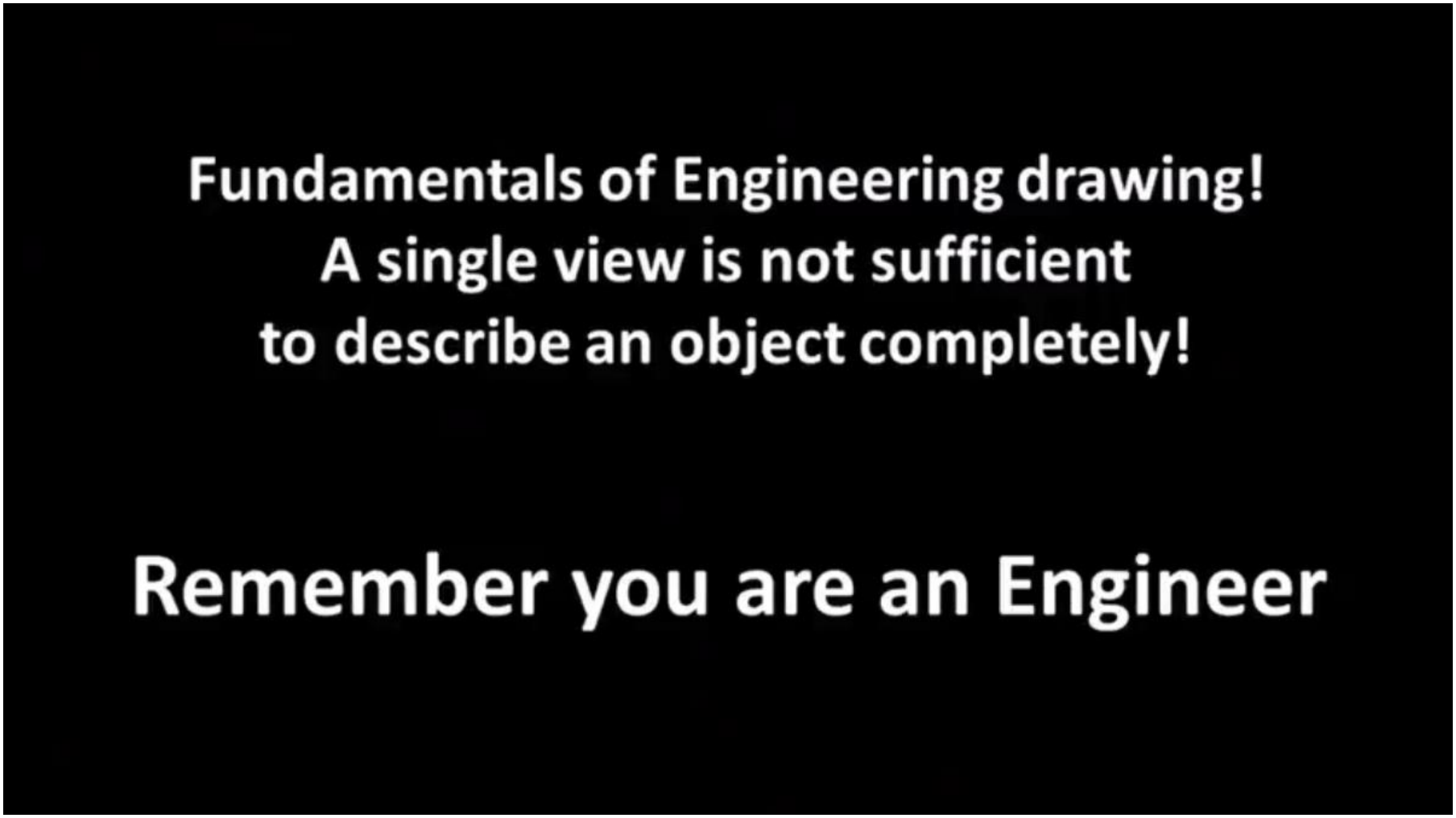
Erro de Principiante!

Apagar "coisas" dos diagramas e deixá-las no modelo.

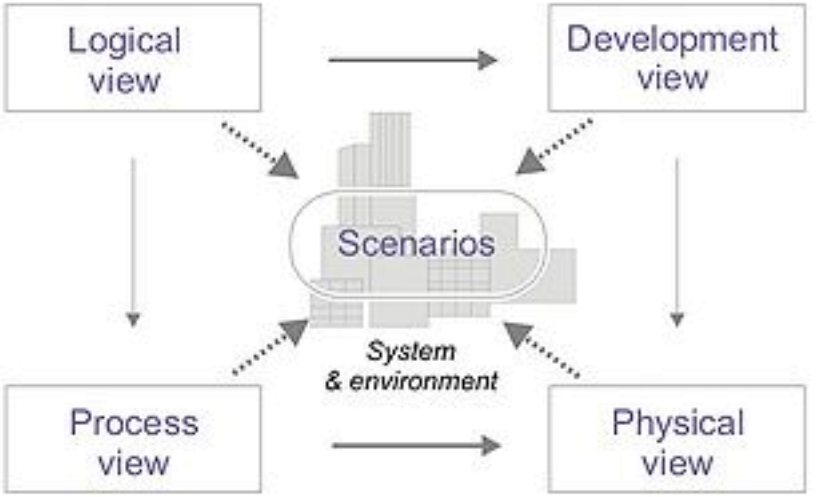
Multiplas vistas 1



Multiplas vistas 2



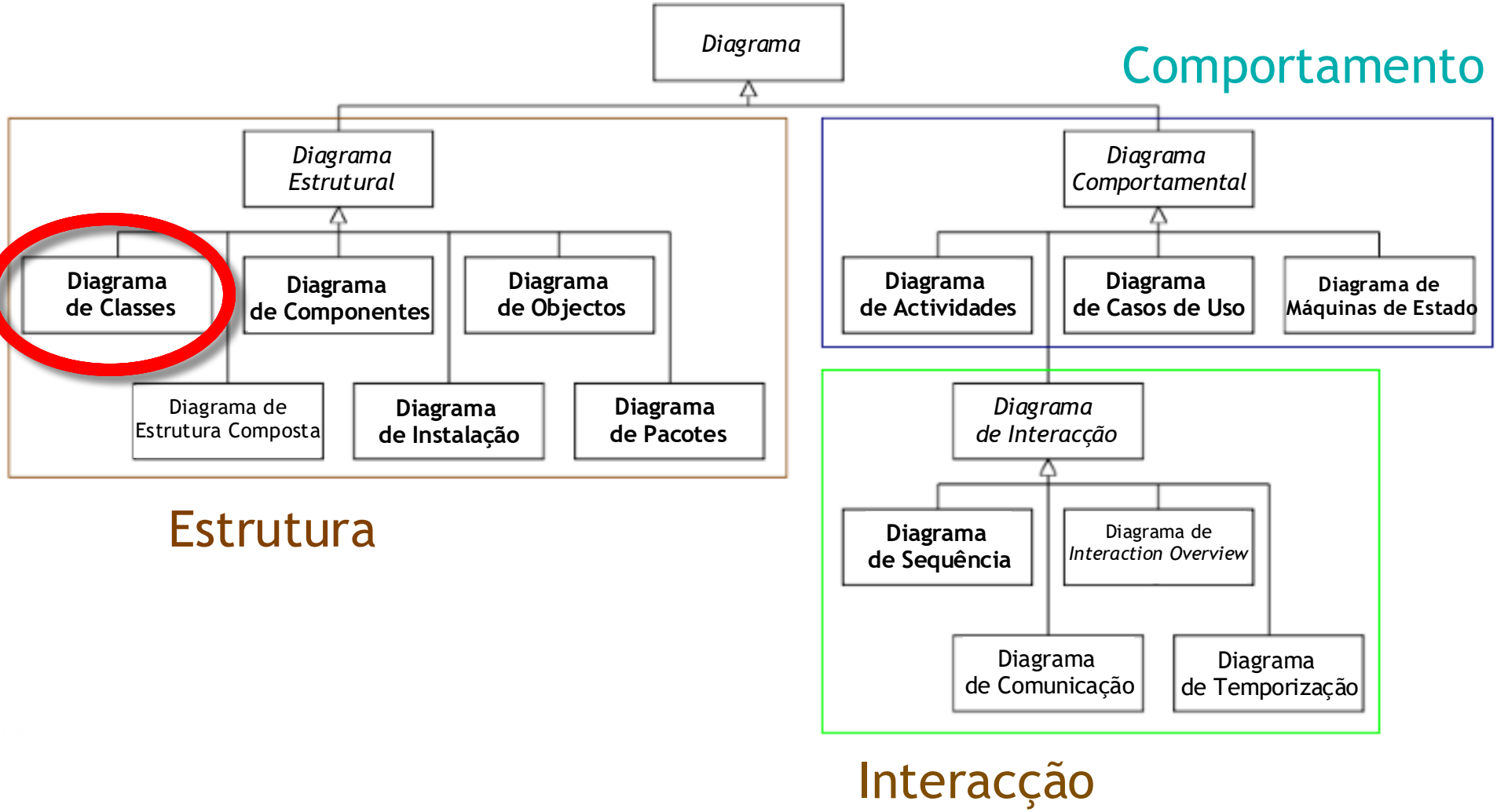
UML – Arquitectura 4+1



(Kruchten, 1995)

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

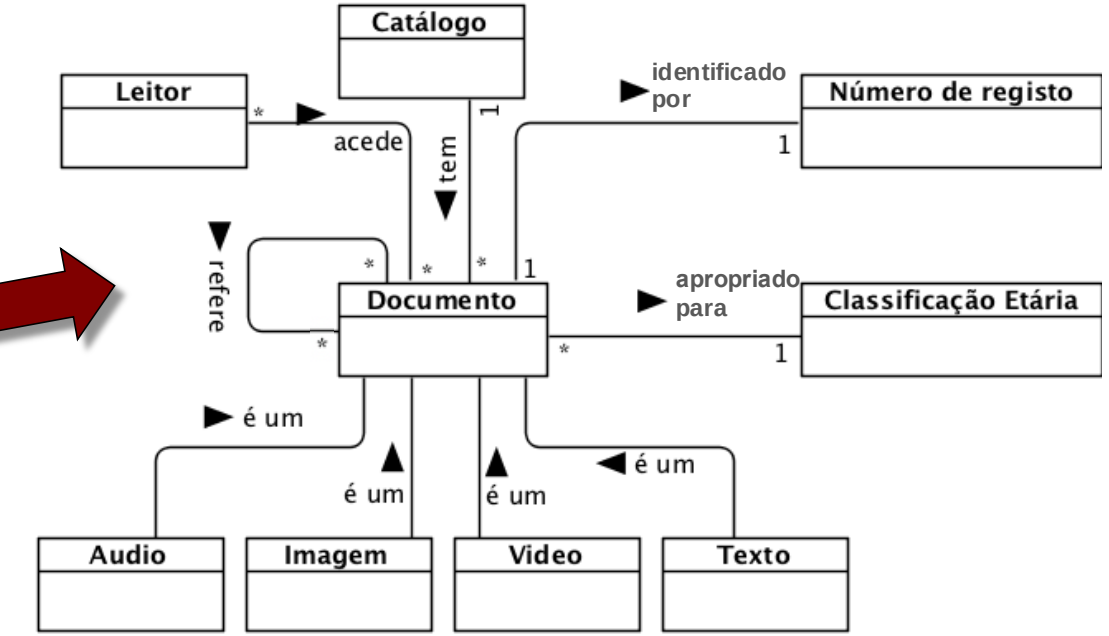
Diagramas da UML 2.x



Algumas notas sobre a notação

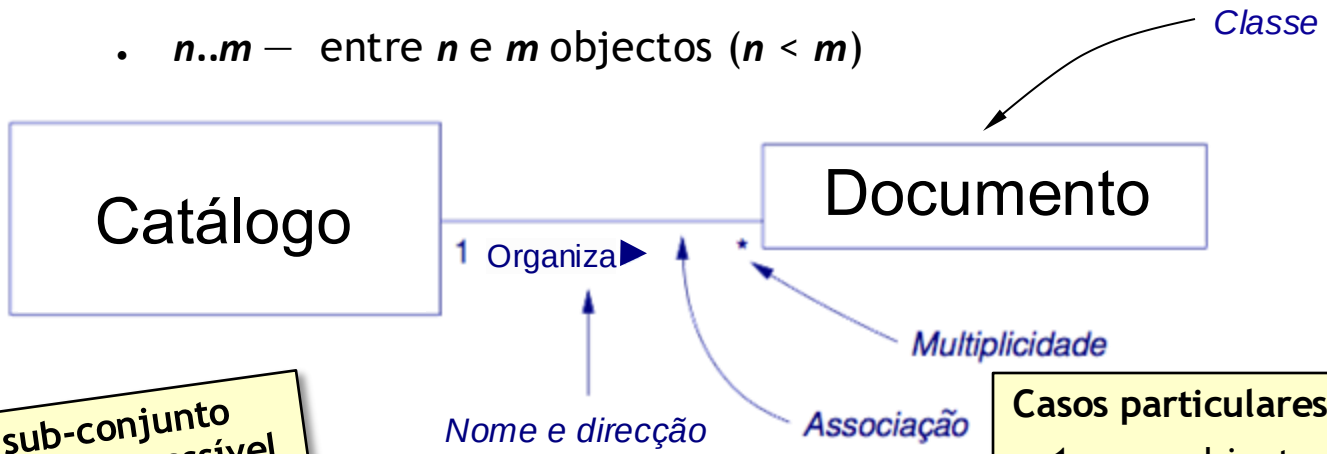
- Representamos os **Modelos de Domínio** em **diagramas de classe** da UML
- Utilizamos apenas **um sub-conjunto da notação**
  - Entidades – representadas por classes
  - Relacionamentos – representados por associações (com nomes)

Diagrama de Classe UML



Relações entre Entidades

- Associações entre classes
- Utilizamos duas decorações:
  - **nome** (com direcção) — descreve a natureza da relação
  - **multiplicidade** — quantos objectos participam na associação:
    - ∗ — zero ou mais objectos
    - ***n*** — ***n*** objectos (***n*** >= 1)
    - ***n..m*** — entre ***n*** e ***m*** objectos (***n*** < ***m***)

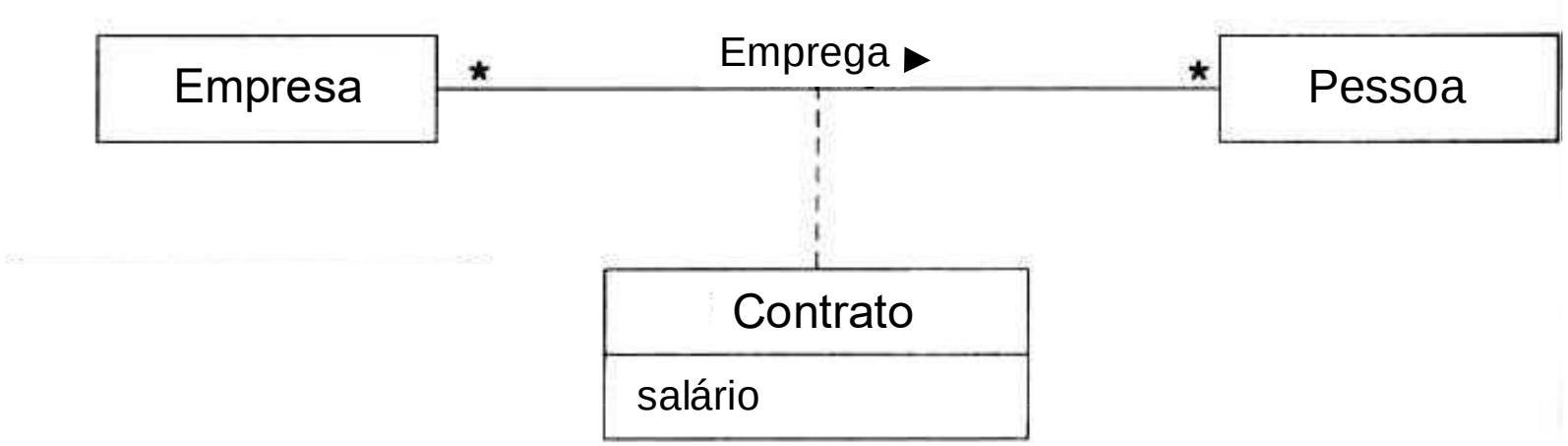


Um sub-conjunto da notação possível para os Diagramas de Classe da UML!

Casos particulares:

- 1 — um objecto = objecto obrigatório
- 0..1 — zero ou um objectos = objecto opcional
- ***n***..* — ***n*** ou mais objectos

Classes de Associação

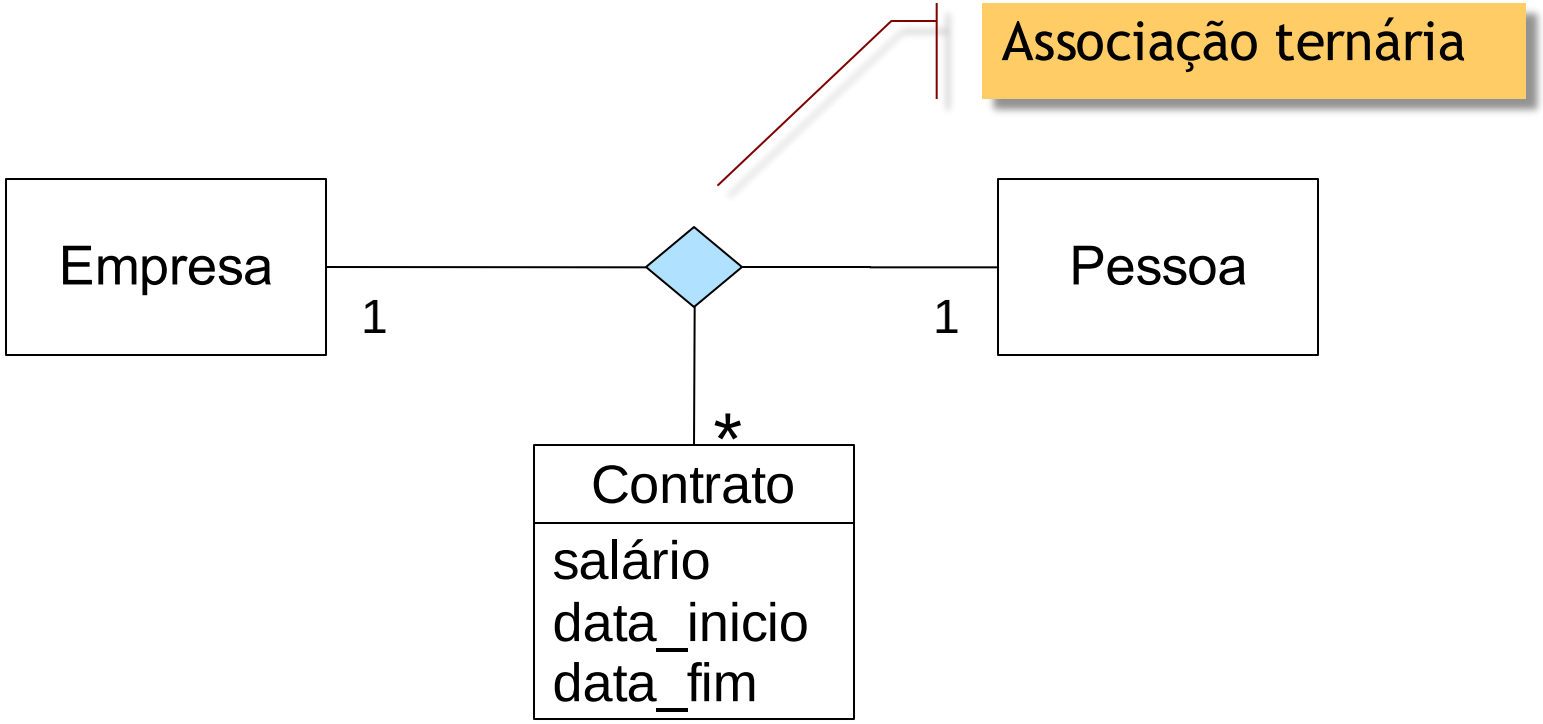


- A relação entre cada Empresa e cada um dos seus funcionários é caracterizada por um contrato.
- Cada Pessoa, pode ter estado contratada por várias Empresas e para cada uma há um contrato diferente.
- O Contrato não é característica da Empresa, nem da Pessoa, mas da relação entre ambas.

mas… **Dois contratos diferentes com a mesma empresa?!**

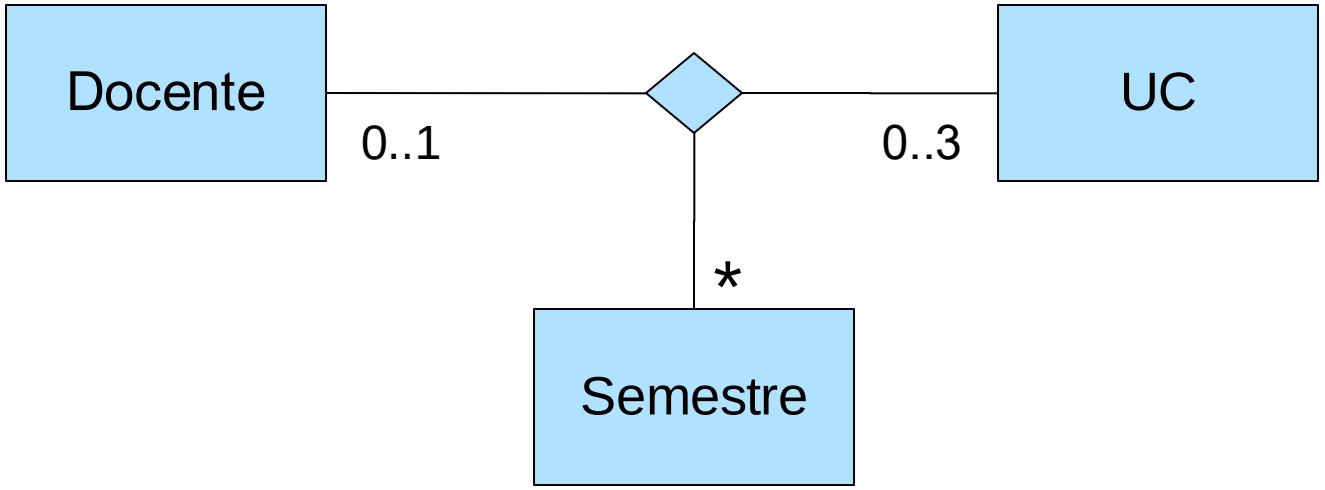
Associações n-árias

- A UML não se restringe a associações binárias:



- Multiplicidades indicam quantos objectos existem para uma dada combinação de objectos das outras classes.
- Navegabilidade, agregação e qualificação **não são** permitidos.

Associações n-árias – outro exemplo



- Cada docente pode leccionar num semestre, no máximo, três UCs.
- Uma multiplicidade de zero invalida a combinação de objectos(!)
  - Não é possível ter uma associação entre uma UC e um Semestre sem indicar o Docente
  - Não é possível dizer que um Docente dá aulas num Semestre sem indicar, pelo menos, uma UC

Primeira etapa do projecto…

- Construir um Modelo de Domínio
  - Perceber o domínio do problema!
  - Identificar Entidades e Relacionamentos entre elas
  - Usamos versão simplificada de diagramas de classe para os representar

- Próxima etapa
  - Modelar os requisitos do sistema