

# Modelação e Design

## 01: Apresentação

Leonor Melo

leonor@isec.pt

1

### Sumário

- Apresentação
- Apresentação da FUC
- Noção de Modelo
- Introdução ao UML

## Docentes

- Leonor Melo
  - responsável pela Unidade Curricular
  - teóricas + algumas práticas diurno + práticas pós-laboral
  - leonor@isec.pt
- Ivo Gonçalves
  - algumas práticas diurno
  - ivo.goncalves@isec.pt

01 Apresentação

3

3

## Objetivos

- Estabelecer contacto com a linguagem de modelação UML
- Utilizar o UML como ferramenta para a modelação de sistemas
- Adquirir conhecimentos em metodologias orientadas a objectos
- Compreender a evolução de paradigmas de análise e construção de software
- Desenvolver capacidades para analisar e projectar sistemas complexos

01 Apresentação

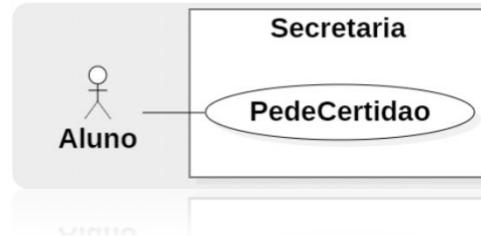
4

4

2

## Programa - 1

1. Introdução ao UML
2. Casos de Uso
  - Serve para refletir e definir os requisitos que o sistema deve cumprir.
  - Que utilizadores o vão usar e que funcionalidades pretendem



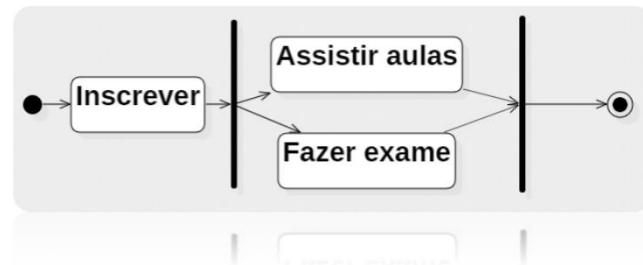
01 Apresentação

5

5

## Programa - 2

3. Diagramas de Atividade
  - modelar os processos (processos de negócio ou processos de software). Mecanismo de controlo do fluxo de execução e do fluxo de dados



01 Apresentação

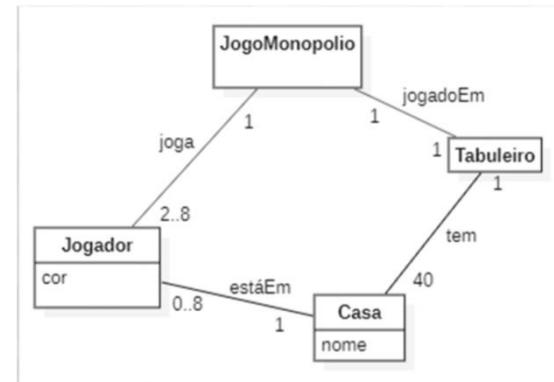
6

6

3

## Programa - 3

3. Modelo do domínio
  - ferramenta de comunicação e análise do problema
  - dicionário visual do domínio do problema



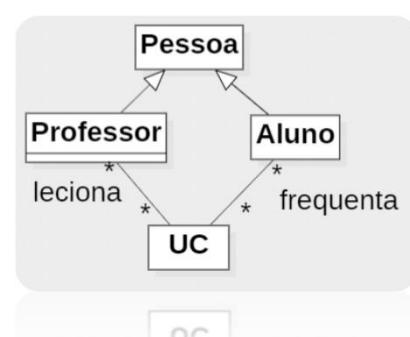
01 Apresentação

7

7

## Programa - 4

4. Modelação Orientada a Objetos
5. Diagramas de Classes
  - teve origem no desenvolvimento orientado a objetos. Baseado nos conceitos de classe , generalização e associação



01 Apresentação

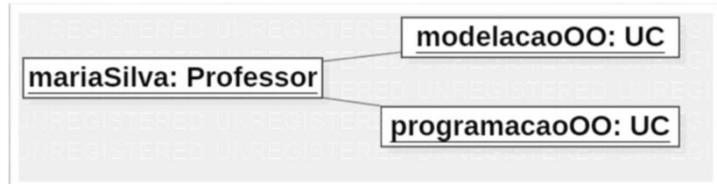
8

8

## Programa - 5

## 6. Diagramas de Objetos

- retrato concreto do estado do sistema em determinado instante



01 Apresentação

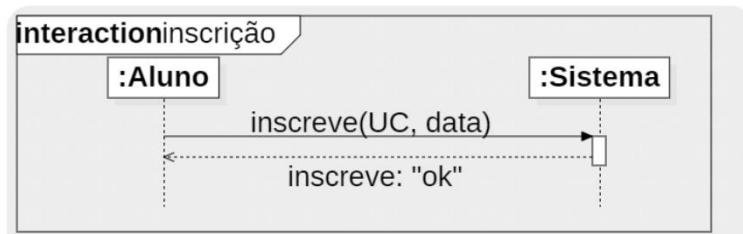
9

9

## Programa - 6

## 7. Diagramas de Sequencia

- Descreve as interações entre objetos de forma a realiza determinada tarefa. O foco é a ordem cronológica das mensagens trocadas



01 Apresentação

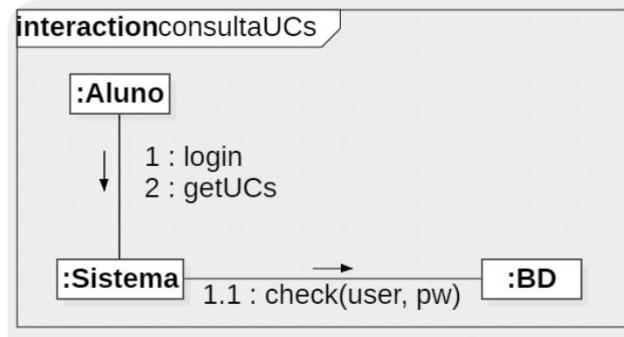
10

10

## Programa - 7

## 8. Diagramas de Comunicação

- Também descreve as interações entre objetos mas o foco é quem comunica com quem



01 Apresentação

11

11

## Programa - 8

## 9. Diagramas de Componentes

- Um componente é uma unidade de execução independente, que fornece serviços ou usa serviços de outros componentes. Podemos modelar duas perspetivas

- blackbox: que representa a especificação do componente
- whitebox: que representa a implementação do componente



01 Apresentação

12

12

## Programa - 9

### 10. Diagramas de Instalação

- Representa a forma como o software é atribuído ao hardware (servidor, desktop,...) e ambientes de execução (sistema operativo, web server,...), e o modo como as partes comunicam



01 Apresentação

13

13

## Programa - 10

### 11. Padrões de Software

### 12. Metodologias de desenvolvimento de software

01 Apresentação

14

14

## Bibliografia - 1

- UML distilled, Martin Fowler, 2004, Addison Wesley, 3<sup>a</sup> edição
- UML: guia do usuário, Grady Booch, Ivar Jacobson, James Rumbaugh, 2006, Editora Campus
- Applying UML and patterns : an introduction to object-oriented analysis and design and iterative development, Craig Larman, 2007, Prentice-Hall, 3<sup>a</sup> edição
- Software Engineering, Ian Sommerville, 2011, Pearson, 9<sup>a</sup> edição

01 Apresentação

15

15

## Bibliografia - 2

- UML, Metodologias e Ferramentas CASE, Alberto Silva, Carlos Videira, 2008
- Fundamental de UML, Mauro Nunes, Henrique O'Neil, 2004, FCA
- Slides das aulas teóricas

01 Apresentação

16

16

## Metodologias de ensino / aprendizagem

- Aulas teóricas
  - 2 aulas (de 1h) por semana
  - apresentação dos conceitos
  - ilustração dos conceitos através de exemplos
- Aulas práticas
  - 1 aulas (de 2h) por semana
  - aplicação dos conceitos através de exercícios guiados
  - eventual exposição de conceitos adicionais
- Desenvolvimento de um Trabalho Prático

01 Apresentação

17

17

## Avaliação

- Exame escrito - 10 valores
- Trabalho prático - 10 valores
  - fase 1: (25%)
  - fase 2: (25%)
  - fase 3: (50%)
  - defesa/feedback após cada fase entregue

01 Apresentação

18

18

## Alunos Erasmus

- Alunos Erasmus?
  - enviar mail para leonor@isec.pt

01 Apresentação

19

19

## Motivação

- Um modelo realça os aspectos importantes do software através de uma notação clara e tão simples quanto possível, e omite detalhes irrelevantes
- Diferentes tipos de modelos são usados para comunicar diferentes aspectos do que queremos construir.
  - Cada modelo apresenta apenas parte da informação e deve ser facilmente interpretada por quem o vai ler
- Os vários modelos devem ser consistentes entre si

01 Apresentação

20

20

## Sistemas

- Os modelos permitem descrever sistemas de forma eficiente
- Um sistema é algo constituído por componentes que estão relacionadas entre si de tal forma que podem ser percecionadas - do ponto de vista de determinada tarefa ou propósito - como uma unidade, e assim serem distintas do ambiente que as rodeia:
  - exemplo: um carro, um avião, um lago, uma floresta, um sistema informático...

01 Apresentação

21

21

## Propriedades dos modelos - 1

- Um modelo é uma representação de um sistema reduzida ao essencial
  - aspectos específicos e individuais são removidos deixando apenas as facetas fundamentais e caracterizadoras do sistema
- Normalmente o sistema é descrito não apenas por um modelo mas por um conjunto de "vistas" que em conjunto produzem uma imagem coerente e completa

01 Apresentação

22

22

## Propriedades dos modelos - 2

- Um modelo deve:
  - Omitir detalhes irrelevantes
  - Usar uma notação simples e intuitiva
  - Ser rigoroso
  - Ajudar a prever características não triviais do sistema
  - Ser mais simples de criar que o sistema

01 Apresentação

23

23

## Modelos podem ser usados para - 1

- Modelos como rascunho:
  - usados para comunicar de forma simples certas ideias ou aspetos parciais do sistema. Usados para clarificar raciocínios ou simplificar a discussão com outros membros da equipa
- Modelos como "plantas":
  - usados para sistematizar a informação e estabelecer a base a partir da qual se podem começar a desenvolver e implementar. Normalmente parciais mas com detalhe suficiente para permitir simulações, implementação, ou exploração de uma das suas sub-partes

01 Apresentação

24

24

Modelos  
podem ser  
usados para - 2

- Modelos para desenvolvimento automático de software:
  - Modelos especificados de forma tão precisa que permitem a construção automática de código a partir deles. Uma área ainda em investigação com os seus defensores e oponentes

01 Apresentação

25

25

UML - 1

- Unified Modeling Language
  - linguagem de modelação muito usada para a modelação de conceitos orientados a objetos (e não só)
- Permite representar vários aspectos de um sistema de sw
  - requisitos
  - estruturas de dados
  - fluxos de informação
  - fluxo de dados

01 Apresentação

26

26

## UML - 2

- Não está ligada a nenhuma linguagem de programação nem metodologia de desenvolvimento de software
  - mas é particularmente útil para abordagens iterativas e incrementais

01 Apresentação

27

27

## Perspetivas do modelo



01 Apresentação

28

28

## Perspetiva de caso de uso

- Perspetiva de caso de uso
  - Descreve as funcionalidades do sistema do ponto de vista do mundo exterior.
  - É necessária para descrever o que o sistema deve fazer.
  - Todas as outras perspetivas dependem desta
  - Inclui entre outros Diagramas de Caso de Uso e Especificação de Caso de Uso

01 Apresentação

29

29

## Perspetiva de processo

- Descreve os processos dentro do sistema
  - Particularmente útil para visualizar o que deve acontecer no sistema
  - Inclui Diagramas de Atividade

01 Apresentação

30

30

15

## Perspetiva lógica

- Descreve de forma abstrata as várias partes do sistema
  - Usada para modelar as partes que constituem o sistema e como é que elas interagem
  - Inclui os diagramas de classe, de objeto, de máquinas de estado e diagramas de interação

01 Apresentação

31

31

## Perspetiva de desenvolvimento

- Descreve como as partes do sistema estão organizadas em módulos e componentes
  - Util para gerir as camadas da arquitetura do sistema
  - Esta vista inclui diagramas de componentes

01 Apresentação

32

32

## Perspetiva física

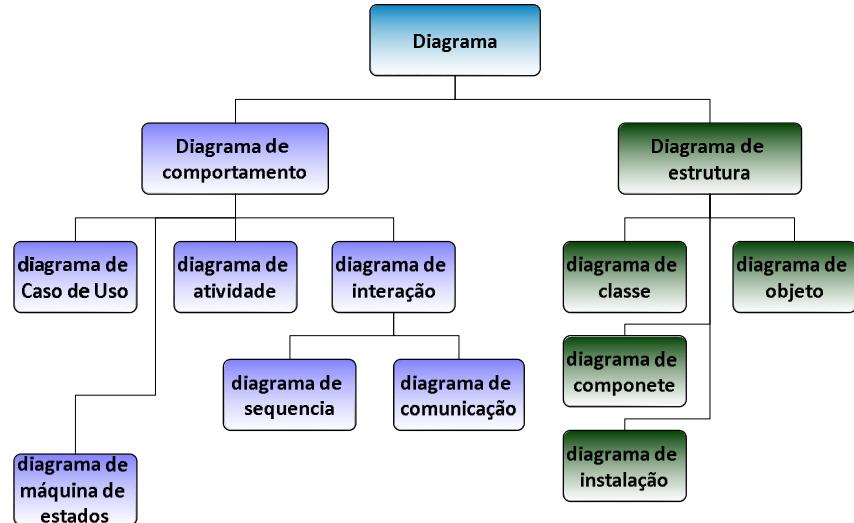
- Descreve como o design do sistema, descrito nas perspetivas anteriores, se deve mapear num sistema real
  - Inclui os diagramas de instalação

01 Apresentação

33

33

## Diagramas UML



01 Apresentação

34

34

# Modelação e Design

## 03: Diagrama de Casos de usos

Leonor Melo

leonor@isec.pt

1

### Sumário

- Notação: Sistema, Atores e Casos de Uso
- Relação entre:
  - Ator e caso de uso
    - associação
  - Atores
    - generalização
  - Casos de uso
    - include
    - generalização
    - extend

03 Diagrama de Casos de Uso

2

## Casos de uso

- O caso de uso, da perspetiva do cliente, representa uma utilização do sistema completa:
  - Contém alguma interação com o sistema
  - Contém algum resultado dessa interação
    - Deve fornecer algum benefício mensurável ao utilizador ou sistema externo
    - Deve ter associado um critério claro de sucesso/insucesso que possa ser verificado

03 Diagrama de Casos de Uso

3

## Diagramas de caso de uso

- Diagramas de caso de uso servem para modelar a interação dos atores com os sistema
- Fornece perspetiva rápida de:
  - Principais funcionalidades fornecidas pelo sistema
  - Tipos de utilizadores
  - Relações entre funcionalidades
  - Relações entre tipos de utilizadores

03 Diagrama de Casos de Uso

4

## Representação UML do caso de uso

- Mais usual:
  - Representar o caso de uso por uma elipse
  - O nome do caso de uso aparece dentro da elipse
- Também possível:
  - elipse + nome logo abaixo
  - retângulo com o nome do caso de uso + pequena elipse no canto



03 Diagrama de Casos de Uso

5

## Atores UML

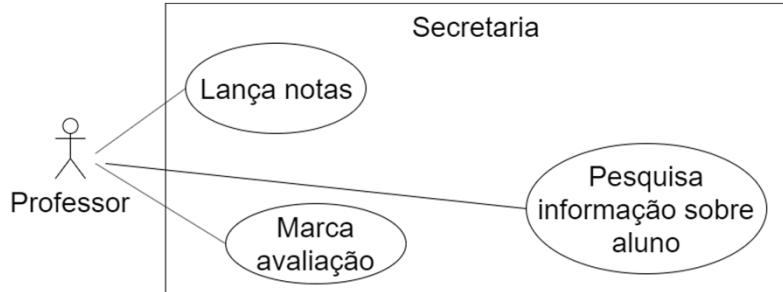


- Mais usual:
  - um boneco (stick person) + nome logo abaixo
- Também possível:
  - uma caixa estereotipada (<<ator>>) + nome abaixo do esteriótipo
- A representação é a mesma seja um ator primário ou secundário, humano ou não

03 Diagrama de Casos de Uso

6

## Fronteiras do sistema



- Os casos de uso são (geralmente) agrupados dentro de um retângulo que representa os limites do sistema
- Os atores estão fora dos limites do sistema

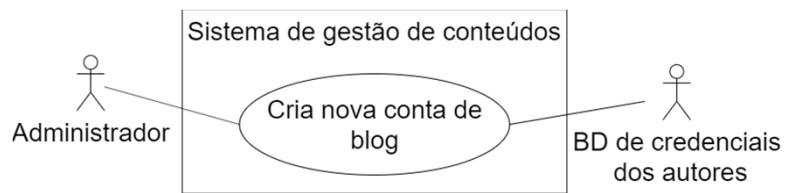
03 Diagrama de Casos de Uso

7

## Associações ou linhas de comunicação - 1

### • Requisito A1:

- O sistema de gestão de conteúdos irá permitir ao administrador criar uma nova conta de blog, uma vez tendo sido verificados os dados pessoais do novo blogger usando a base de dados de credenciais dos autores



03 Diagrama de Casos de Uso

8

## Associações ou linhas de comunicação - 2



- A associação significa:
  - o ator comunica com o sistema
  - o ator usa (ou é usado por) determinada funcionalidade
- Cada caso de uso tem de ter
  - pelo menos uma associação com um ator;
- Cada ator tem de ter
  - pelo menos uma associação com um caso de uso

03 Diagrama de Casos de Uso

9

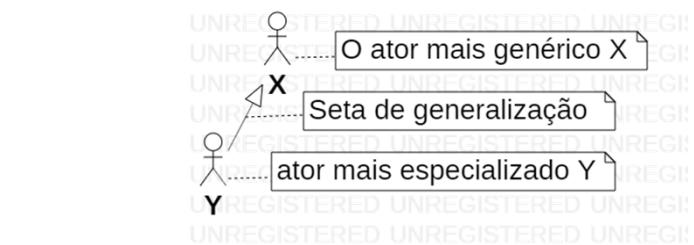
## Relação de generalização entre atores

- Aplica-se quando existem atores com
  - algumas propriedades que os distinguem
  - outras propriedades em comum
- Exemplo:
  - quer o Professor Assistente quer o Professor Responsável podem consultar informação sobre os alunos
  - apenas o Professor responsável pode lançar pautas

03 Diagrama de Casos de Uso

10

## Especialização / generalização / herança entre atores - 1

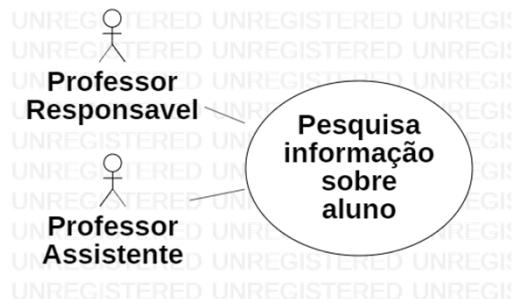


- O ator Y é uma especialização do ator X
- O ator X é uma generalização do ator Y
- O ator Y participa em todos os casos de uso em que X participe
  - e possivelmente em outros mais onde X não participa

03 Diagrama de Casos de Uso

11

## Especialização / generalização / herança entre atores - 2

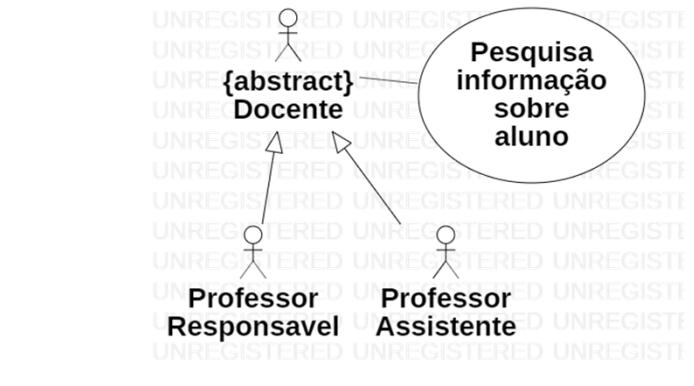


- O caso de uso pesquisa informação requer a participação de um Professor Responsável e de um Professor Assistente

03 Diagrama de Casos de Uso

12

## Especialização / generalização / herança entre atores - 3

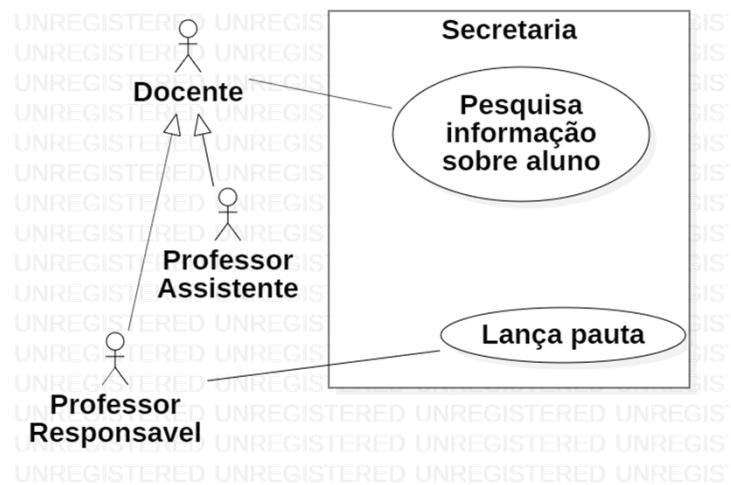


- Quer o Professor Responsável quer o Professor Assistente herdam a associação:
  - Cada um deles participa no caso de uso individualmente

03 Diagrama de Casos de Uso

13

## Exemplo relação de generalização entre atores - UML



03 Diagrama de Casos de Uso

14

## Relação entre casos de uso

- Serve para decompor o comportamento do sistema em "pedaços" mais simples
- Usa-se quando:
  - Existem um conjunto de passos comum a vários casos de uso
    - Ex., um caso de uso que pode ser usado diretamente, mas também faz sentido enquanto parte de outro caso de uso
  - O caso de uso tem diferentes modos de funcionamento de acordo com a situação específica em que decorre

03 Diagrama de Casos de Uso

15

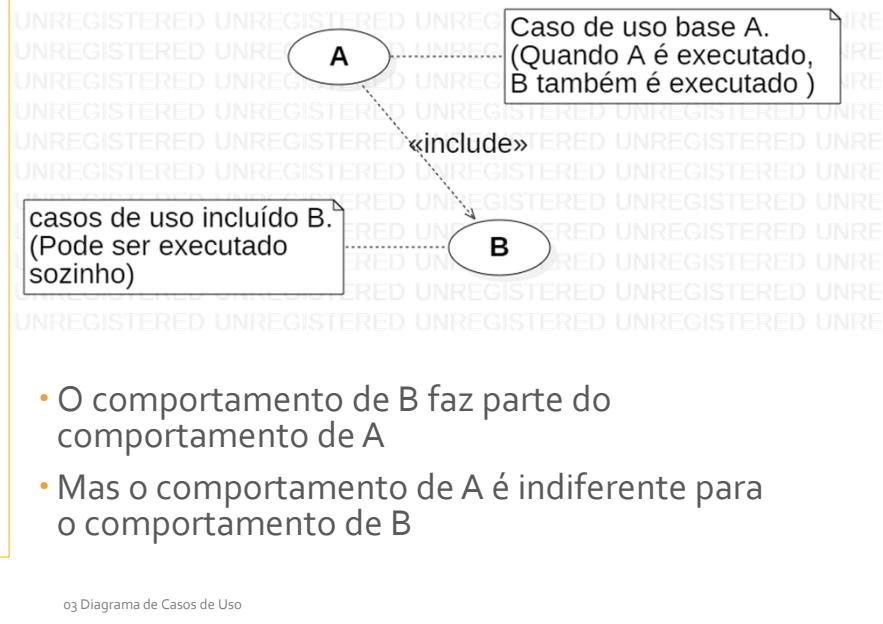
## Relação de «include»: casos de uso com redundâncias 1

- A relação de «include» («incluir») usa-se se:
  - existem comportamentos que obrigatoriamente se repetem em vários casos de uso
- Um caso de uso pode:
  - incluir vários casos de uso
  - ser incluído por vários casos de uso

03 Diagrama de Casos de Uso

16

## Relação de «include»: notação UML



17

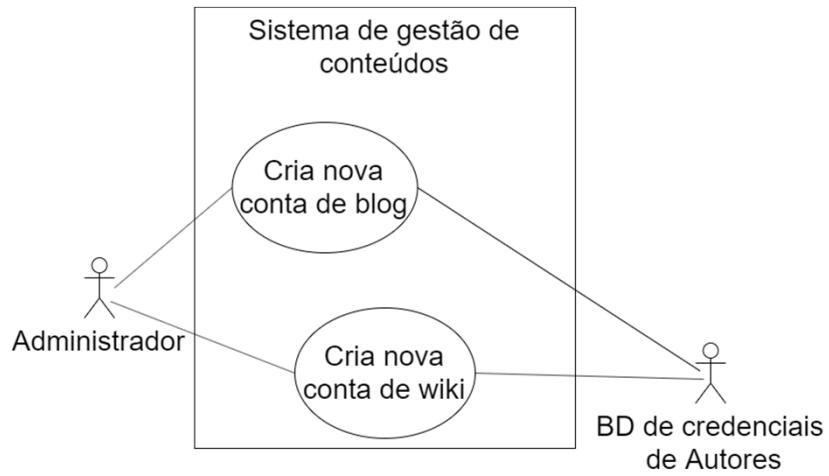
## Relação de «include»: exemplo 1

- Requisito A2:
  - O sistema de gestão de conteúdo irá permitir a um administrador criar uma nova wiki pessoal, desde que os detalhes pessoais do autor que requereu a wiki sejam validados usando a base de dados de credenciais dos autores

03 Diagrama de Casos de Uso

18

## Relação de «include»: exemplo 2



- Qual o comportamento que se repete?

03 Diagrama de Casos de Uso

19

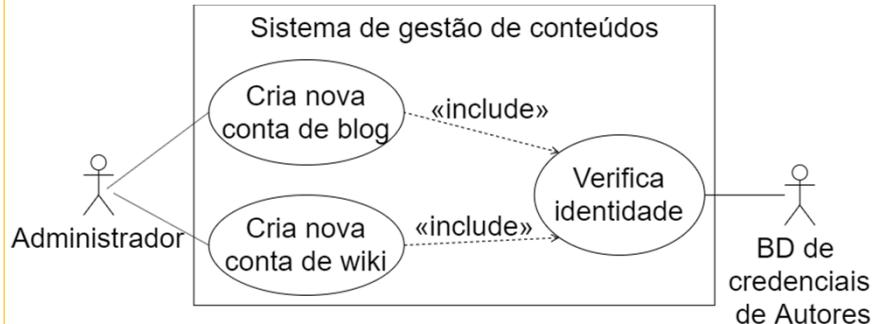
## Relação de «include»: exemplo 3

- *Cria nova conta de blog* e *Cria nova conta de wiki* têm uma série de passos em comum:
  - os que tratam da verificação da identidade do autor
- Melhor criar caso de uso que pode ser reutilizado
  - *Verifica identidade*
  - e usar uma relação de «include»

03 Diagrama de Casos de Uso

20

## Relação de «include»: exemplo 4

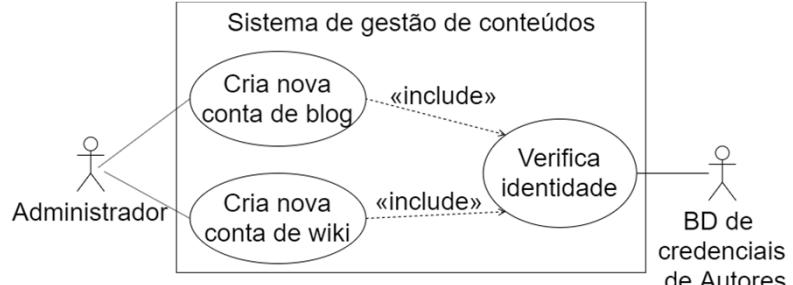


- Cada um dos passos de *Verifica identidade* é também dados por
  - *Cria nova conta de blog*
  - *Cria nova conta de wiki*

03 Diagrama de Casos de Uso

21

## Relação de «include»: exemplo 5



- *Verifica identidade* está associado:
  - diretamente com a *BD de credenciais de Autores*
  - indiretamente com o *Administrador*
- Apenas *Verifica identidade* depende diretamente de uma ligação à *BD de credenciais de Autores*

03 Diagrama de Casos de Uso

22

## Relação de «include»: vantagens

- Benefícios da reutilização usando a relação de «include»:
  - Se houver de alteração/atualização dos requisitos,
    - não temos de repetir as alterações em vários casos de uso
  - Ao fazer o design do sistema
    - sabemos que *Verifica identidade* deverá ser preparada para ser reutilizada

03 Diagrama de Casos de Uso

23

## Relação de generalização: casos de uso com variantes 1

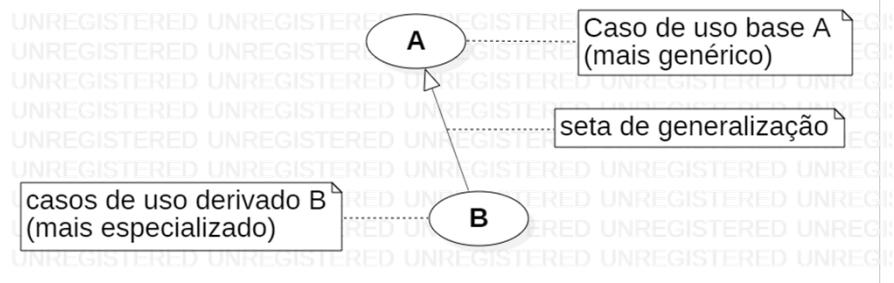
- A relação de generalização entre casos de uso usa-se se:
  - Um caso de uso em diferentes situações
    - mas consoante a situação o seu comportamento é ligeiramente diferente
- A herança/generalização entre casos de uso é útil quando
  - queremos mostrar que um caso de uso é uma variante mais específica de outro

03 Diagrama de Casos de Uso

24

## Relação de generalização: representação UML

- Representa-se usando a seta de generalização
- O caso derivado herda o comportamento e as relações do caso base



03 Diagrama de Casos de Uso

25

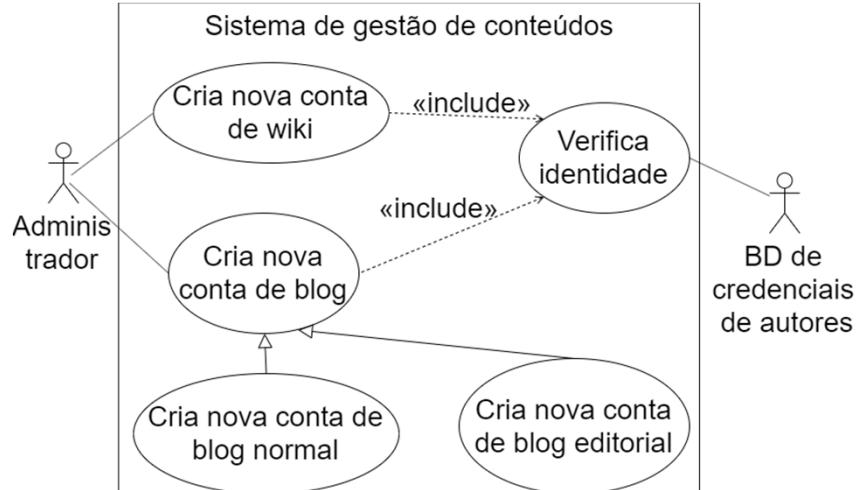
## Relação de generalização: casos de uso com variantes: Exemplo

- Exemplo:
  - O sistema de gestão de conteúdos permite dois tipos de contas:
    - a conta normal, que permite ter um blog;
    - e a conta editorial que permite alterar conjuntos de blogs.
  - Para cada tipo de conta a sequencia de passos a executar
    - é genericamente a mesma
    - mas não exatamente igual

03 Diagrama de Casos de Uso

26

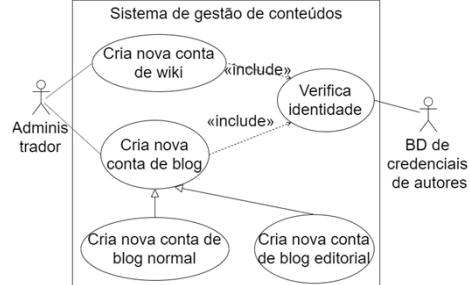
## Relação de generalização: Exemplo



03 Diagrama de Casos de Uso

27

## Relação de generalização: Exemplo



- *Cria nova conta de blog editorial* e *Cria nova conta de blog normal* reutilizam a maior parte do comportamento do *Cria nova conta de blog*
  - alteram apenas detalhes específicos deixados em aberto no caso base

03 Diagrama de Casos de Uso

28

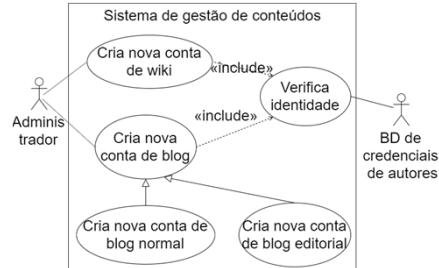
## Relação de generalização: casos de uso com variantes 3

- Usar herança se
  - todos os passos do caso de uso base vão ser executados no caso derivado
- O caso derivado pode
  - tornar mais específicos os passos herdados,
  - ter passos adicionais em relação ao base

03 Diagrama de Casos de Uso

29

## Relação de generalização: casos de uso com variantes 4



- As relações do caso base são herdadas pelo caso derivado
  - ex. a relação de «include» com *Verifica identidade* deve de continuar a fazer sentido em *Cria nova conta de blog normal* e *Cria nova conta de blog editorial*

03 Diagrama de Casos de Uso

30

## Relação de generalização: casos de uso com variantes 5

- Se o potencial caso de uso derivado na realidade só inclui alguns dos passos do caso base
  - então a relação de herança possivelmente não é a mais indicada

o3 Diagrama de Casos de Uso

31

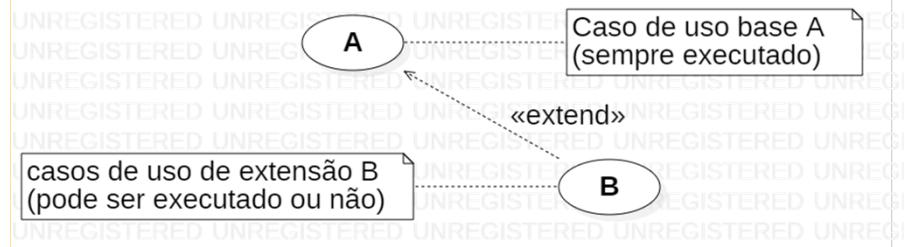
## Relação de «extend»: reutilização opcional de comportamentos os 1

- Nome semelhante mas significado diferente da noção de extend (herança) da orientação a objetos
- Semelhante ao «include» mas no «extend» a reutilização é opcional:
  - depende de decisões de implementação ou mesmo tomadas em *runtime*

o3 Diagrama de Casos de Uso

32

## Relação de «extend»: notação UML



- O comportamento de B pode fazer parte do comportamento de A
  - mas também pode não fazer

03 Diagrama de Casos de Uso

33

## Relação de «extend»: exemplo

- Exemplo:
  - no Sistema de Gestão de Conteúdos:
    - Caso um autor tenha requerido uma conta de blog e esta lhe tenha sido negada queremos poder registar essa informação no histórico de requisições do autor.
    - O mesmo comportamento deve ser possível para as contas de wiki.

03 Diagrama de Casos de Uso

34

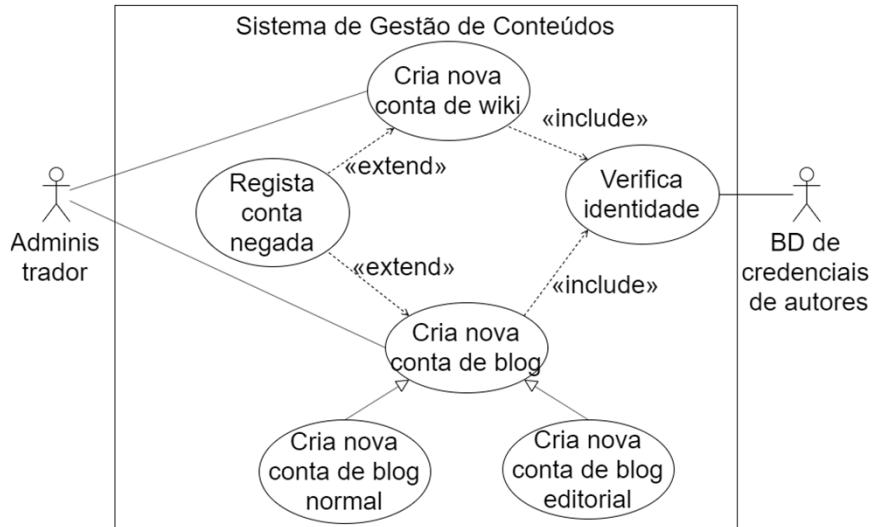
## Relação de «extend»: análise

- Podemos adicionar o caso de uso *Regista conta negada* que será, opcionalmente, executado em função do resultado da execução dos casos de uso *Cria conta de wiki* e *Cria conta de blog*
  - *Regista conta negada* é executado apenas caso o pedido de conta seja rejeitado
  - Se o pedido de conta for aceite o sistema não deverá executar *Regista conta negada*

03 Diagrama de Casos de Uso

35

## Relação de «extend»: exemplo - diagrama UML



03 Diagrama de Casos de Uso

36

# Modelação e Design

## 04: Descrição de Casos de usos

Leonor Melo

leonor@isec.pt

1

### Sumário

- Descrição de casos de uso
  - Objetivos
  - Formato breve
  - Formato casual
  - Formato completo
    - secções a incluir
    - relação de include
    - relação de generalização
    - exemplos de erros a evitar

## Descrição de caso de uso

- Caso de uso é uma descrição textual
- Clarificar e acrescentar detalhe sobre
  - qual o comportamento desejado,
  - situações de erro,
  - comportamentos alternativos,
  - relações entre casos de uso,
  - etc.
- Podem ser escritos com diferentes níveis de detalhe e formalismo

Leonor Melo

04 Descrição de Casos de Uso

3

3

## Descrição de caso de uso: formato breve

- Formato breve:
  - Sumário de um único parágrafo, usualmente com o cenário principal de sucesso:
- **Processa venda:** Um cliente chega à caixa com os itens que quer comprar. O caixa usa o sistema de ponto-de-venda para registar cada item comprado. O sistema apresenta o total e uma linha por cada item com detalhes. O cliente introduz os detalhes de pagamento, que o sistema valida e regista. O sistema atualiza o inventário. O cliente recebe o recibo do sistema e sai com os produtos.

Leonor Melo

04 Descrição de Casos de Uso

4

4

2

## Descrição de caso de uso: formato casual - 1

- **Formato casual:**
  - Alguns parágrafos de texto que cobrem alguns cenários:
- **Processa venda :**
  - *Cenário principal de sucesso:* Um cliente chega à caixa com os itens que quer comprar. O caixa usa o sistema de ponto-de-venda para registar cada item comprado. O sistema apresenta o total e uma linha por cada item com detalhes. O cliente introduz os detalhes de pagamento, que o sistema valida e regista. O sistema ...

Leonor Melo

04 Descrição de Casos de Uso

5

## Descrição de caso de uso: formato casual - 2

- **Processa venda(continuação):**
- **Cenários alternativos:**
  - Se o cliente pagou a crédito e a operação for rejeitada pelo sistema bancário informar o cliente e permitir apenas venda por débito ou a dinheiro.
  - Se o identificador do item não for encontrado no sistema notificar gerente e sugerir introdução manual do código identificador.
  - Se o sistema detetar falha na comunicação com o sistema de contabilidade externo...

Leonor Melo

04 Descrição de Casos de Uso

6

## Descrição de caso de uso: formatos breve e casual

- Os formatos breve e casual usam-se:
  - durante a fase inicial da análise de requisitos
  - para dar uma noção rápida do assunto e alcance de cada caso de uso (e do sistema)
- Podem levar apenas alguns minutos a criar

Leonor Melo

04 Descrição de Casos de Uso

7

7

## Descrição de caso de uso: formato completo - 1

- Formato completo:
  - Todos os passos e variantes são descritos em detalhe,
  - existem secções adicionais como
    - précondições e
    - critérios de sucesso.
  - As secções específicas a escolher neste formato podem variar

Leonor Melo

04 Descrição de Casos de Uso

8

8

## Descrição de caso de uso: formato completo - 2

- Casos de uso no formato completo criam-se:
  - depois de a maior parte dos casos de uso terem sido
    - identificados, e
    - escritos no formato breve.
- Em cada um das iterações iniciais do projeto,
  - uma pequena percentagem dos casos de uso de
    - maior valor, e
    - maior relevância em termos de arquitetura
  - são reescritos no formato completo

Leonor Melo

04 Descrição de Casos de Uso

9

9

## Descrição de caso de uso: formato completo - 3

- *Stakeholders* (grupo de interesse)
  - Pessoas ou instituições com interesse nos resultados produzidos pelo sistema
  - Podem ser
    - Atores
      - primários ou secundários
    - Entidades que não interagem diretamente com o sistema
      - Finanças, accionistas, investidores, proprietários, gestores, sindicatos, Governo, concorrentes, etc.

Leonor Melo

04 Descrição de Casos de Uso

10

10

| Descrição de caso de uso: formato completo - secções - 1 | Secção do caso de uso Comentário |   |
|--|----------------------------------|---|
|  | Nome do caso uso                 | deve começar por um verbo                                   |
|  | Sistema                          | O sistema a ser desenhado                                   |
|  | Nível                            | "objetivo do utilizador" ou "subfunção"                     |
|  | Atores principais                | Que atores recolhem o principal benefício deste caso de uso |
|  | Atores secundários               | Que atores fornecem suporte a este caso de uso              |

Leonor Melo

04 Descrição de Casos de Uso

11

11

| Descrição de caso de uso: formato completo - secções - 2 | Secção do caso de uso Comentário        |  |
|--|---|--|
|  | Stakeholders e seus interesses          | Quem tem interesse neste caso de uso e que benefícios quer ter?  |
|  | Précondições                            | Que situação (não trivial) se deve verificar para o caso de uso poder acontecer?                                       |
|  | Pós condições (ou critérios de sucesso) | Estado em que o sistema deverá ficar em caso de sucesso e no que concerne a satisfação dos requisitos dos stakeholders |

Leonor Melo

04 Descrição de Casos de Uso

12

12

| Secção do caso de uso             | Comentário  |
|-----------------------------------|---|
| Fluxo de eventos                  | Sucessão de passos do cenário de sucesso principal  |
| Situações de erro                 | Que situações podem fazer com que o caso falhe?   |
| Estado do sistema em caso de erro | Estado em que o sistema deverá ficar em caso de erro  |
| Fluxo de eventos alternativo      | Outras sucessões de passos a realizar possíveis, quer em cenários de sucesso quer em cenários de erro |

Leonor Melo

04 Descrição de Casos de Uso

13

13

|   |   |
|---|---|
| <p>Descrição de caso de uso: formato completo - secções - 4</p> | <ul style="list-style-type: none"> <li>• Outras secções possíveis:           <ul style="list-style-type: none"> <li>• requisito relacionado,</li> <li>• objetivo,</li> <li>• requisitos não funcionais,</li> <li>• restrições tecnológicas,</li> <li>• frequência de utilização,</li> <li>• evento causador (<i>trigger</i>),</li> <li>• ...</li> </ul> </li> </ul> |
|---|---|

Leonor Melo

04 Descrição de Casos de Uso

14

14

## Descrição de caso de uso: formato completo - exemplo - 1

| <b>Nome</b>             | <b>Cria nova conta de blog</b>  |
|-------------------------|---|
| Requisito relacionado   | Requisito A.1   |
| Descrição sumária       | Um autor novo ou pré-existente requer uma nova conta de blog ao Administrador                     |
| Précondições            | O autor tem de ser conhecido pelo sistema e tem ter fornecido uma prova de identidade apropriada; |
| Póscondições de sucesso | É criada uma nova conta de blog para o autor.   |

Leonor Melo

04 Descrição de Casos de Uso

15

15

## Descrição de caso de uso: formato completo - exemplo - 2

| <b>Nome</b>           | <b>Cria nova conta de blog</b>   |
|-----------------------|--|
| Póscondições de falha | O pedido de conta de blog é rejeitado, a conta não é construída  |
| Atores principais     | Administrador  |
| Atores secundários    | BD de credenciais de autores   |
| Fluxo de eventos      | <ol style="list-style-type: none"> <li>1. O Administrador pede ao sistema para criar uma nova conta de blog</li> <li>2. O sistema apresenta os tipos de conta possíveis</li> <li>3. O Administrador seleciona o tipo de conta</li> </ol> |

Leonor Melo

04 Descrição de Casos de Uso

16

16

| Descrição de caso de uso: formato completo - exemplo - 3 | <b>Nome</b> <b>Cria nova conta de blog</b><br><b>Fluxo de eventos</b><br>4. Sistema indica lista de detalhes que devem ser fornecidos<br>5. O Administrador introduz os detalhes do autor<br>6. Os detalhes do autor são verificados usando a Base de Dados de credenciais dos autores<br>7. O sistema cria a nova conta de blog e informa o Administrador<br>8. O sistema envia por mail para o autor um sumário dos detalhes da sua nova conta |
|--|--|
| Leonor Melo  | 04 Descrição de Casos de Uso   |

17

17

| Descrição de caso de uso: formato completo - exemplo - 4 | <b>Nome</b> <b>Cria nova conta de blog</b><br><b>Fluxo alternativo</b><br>6.1. A Base de Dados de credenciais de autores não confirma os detalhes do autor<br>6.1.a O pedido de nova conta do autor é rejeitado<br><br>8.1 Ocorreu um erro no envio do mail<br>8.1.a O Sistema avisa o Administrador do erro e pede-lhe que introduza um novo endereço de mail<br>8.1.b O Administrador indica novo endereço de mail<br>8.1.c Volta para o ponto 8. |
|--|---|
| Leonor Melo  | 04 Descrição de Casos de Uso  |

18

18

## Descrição de casos de uso e a relação «include»

- Quando existe uma relação de «include» pode ser acrescentado um campo,
  - Casos de uso incluídos
- Para evitar redundâncias a formulação include::<nome do caso de uso>
- pode ser usado no sítio onde o caso de uso incluído é invocado

Leonor Melo

04 Descrição de Casos de Uso

19

19

## Descrição de caso de uso: exemplo com include - 1

|                         | <b>Nome</b> | <b>Cria nova conta de blog</b>  |
|-------------------------|-------------|---|
| Requisito relacionado   |             | Requisito A.1   |
| Descrição sumária       |             | Um autor novo ou pré-existente requer uma nova conta de blog ao Administrador                     |
| Précondições            |             | O autor tem de ser conhecido pelo sistema e tem ter fornecido uma prova de identidade apropriada; |
| Póscondições de sucesso |             | É criada uma nova conta de blog para o autor.   |
| Póscondições de falha   |             | O pedido de conta de blog é rejeitado, a conta não é construída                                   |

Leonor Melo

04 Descrição de Casos de Uso

20

20

## Descrição de caso de uso: exemplo com include - 2

|  |                        |  |
|--|------------------------|--|
|  | <b>Nome</b>            | <b>Cria nova conta de blog</b>   |
|  | Atores principais      | Administrador  |
|  | Atores secundários     | <b>Não há</b>  |
|  | <b>Casos incluídos</b> | <b>Verifica identidade</b>   |
|  | Fluxo de eventos       | <ol style="list-style-type: none"> <li>1. O Administrador pede ao sistema para criar uma nova conta de blog</li> <li>2. O sistema apresenta os tipos de conta possíveis</li> <li>3. O Administrador seleciona o tipo de conta</li> </ol> |

Leonor Melo

04 Descrição de Casos de Uso

21

21

## Descrição de caso de uso: exemplo com include - 3

|  |                  |   |
|--|------------------|---|
|  | <b>Nome</b>      | <b>Cria nova conta de blog</b>  |
|  | Fluxo de eventos | <ol style="list-style-type: none"> <li>4. Sistema indica lista de detalhes que devem ser fornecidos</li> <li>5. O Administrador introduz os detalhes do autor</li> <li>6. Os detalhes do autor são verificados<br/><b>include::Verifica identidade</b></li> <li>7. O sistema cria a nova conta de blog e informa o Administrador</li> <li>8. O sistema envia por mail para o autor um sumário dos detalhes da sua nova conta</li> </ol> |

Leonor Melo

04 Descrição de Casos de Uso

22

22

|   |  | <b>Nome</b>       | <b>Cria nova conta de blog</b>   |
|---|--|-------------------|--|
| Descrição de caso de uso: exemplo com include - 4 |  | Fluxo alternativo | <p>6.1. A Base de Dados de credenciais de autores não confirma os detalhes do autor</p> <p>6.1.a O pedido de nova conta do autor é rejeitado</p> <p>8.1 Ocorreu um erro no envio do mail</p> <p>8.1.a O Sistema avisa o Administrador do erro e pede-lhe que introduza um novo endereço de mail</p> <p>8.1.b O Administrador indica novo endereço de mail</p> <p>8.1.c Volta para o ponto 8.</p> |

Leonor Melo

04 Descrição de Casos de Uso

23

23

|  |  | <b>Nome</b>           | <b>Verifica identidade</b>                                  |
|--|--|-----------------------|---|
| Descrição de caso de uso: exemplo com include, caso incluído - 1 |  | Requisito relacionado | Requisito A.1, requisito A.2                                |
|  |  | Descrição sumária     | Os detalhes do autor são verificados quanto à autenticidade |
|  |  | Précondições          | Os detalhes do autor tem de ter sido recebido pelo sistema  |
|  |  | Póscondições          | Os detalhes são confirmados de sucesso                      |
|  |  | Póscondições          | Os detalhes não são confirmados de falha                    |

Leonor Melo

04 Descrição de Casos de Uso

24

24

## Descrição de caso de uso: exemplo com include, caso incluído - 2

| <b>Nome</b>        | <b>Verifica identidade</b>   |
|--------------------|--|
| Atores principais  | BD de credenciais dos autores  |
| Atores secundários | Não há   |
| Fluxo de eventos   | <ol style="list-style-type: none"> <li>1. O Sistema fornece à BD os detalhes</li> <li>2. A BD de credenciais de autores confirma se os detalhes são válidos</li> </ol> |

Leonor Melo

04 Descrição de Casos de Uso

25

25

## Descrição de caso de uso: exemplo com include, caso incluído - 3

| <b>Nome</b>       | <b>Verifica identidade</b>   |
|-------------------|--|
| Fluxo de eventos  | <ol style="list-style-type: none"> <li>3. Os detalhes são devolvidos como tendo sido confirmados pela BD de credenciais de autores</li> </ol>  |
| Fluxo alternativo | <ol style="list-style-type: none"> <li>2.1. A BD de credenciais de autores não confirma as detalhes             <ol style="list-style-type: none"> <li>2.1.a Os detalhes são devolvidos como não tendo sido confirmados</li> </ol> </li> </ol> |

Leonor Melo

04 Descrição de Casos de Uso

26

26

## Descrição de casos de uso e a relação de generalização

- Quando existe uma relação de generalização pode ser acrescentado um campo,
  - Caso de uso base
- À partida o caso de uso mais específico irá ser diferente do caso de uso base apenas em detalhes

Leonor Melo

04 Descrição de Casos de Uso

27

27

## Descrição de caso de uso: exemplo com generalização 1

| Nome                    | Cria nova conta de blog editorial  |
|-------------------------|--|
| Requisito relacionado   | Requisito A.1.1  |
| Descrição sumária       | Um autor novo ou pré-existente requer uma <b>nova conta de blog editorial</b> ao Administrador |
| Précondições            | O autor tem de ser conhecido pelo sistema e ter fornecido uma prova de identidade apropriada;  |
| Póscondições se sucesso | É criada uma <b>nova conta de blog editorial</b> para o autor                                  |

Leonor Melo

04 Descrição de Casos de Uso

28

28

Descrição de caso de uso:  
exemplo com generalização  
2

| Nome                    | Cria nova conta de blog editorial                                    |
|-------------------------|--|
| Póscondições se falha   | O pedido de <b>conta de blog editorial</b> é rejeitado               |
| Atores principais       | Administrador  |
| Atores secundários      | Não há   |
| Casos incluídos         | Verifica identidade  |
| <b>Caso de uso base</b> | <b>Cria nova conta de blog</b>                                       |
| Fluxo de eventos        | 1. O Administrador pede ao sistema para criar uma nova conta de blog |

Leonor Melo

04 Descrição de Casos de Uso

29

29

Descrição de caso de uso:  
exemplo com generalização  
3

| Nome             | Cria nova conta de blog  |
|------------------|--|
| Fluxo de eventos | 2. O sistema apresenta os tipos de conta possíveis<br>3. O Administrador seleciona o <b>tipo de conta editorial</b><br>4. O Sistema apresenta os possíveis blogs sobre os quais o autor poderá ter direitos editoriais<br>5. <b>O Administrador seleciona os blogs sobre os quais o autor terá direitos editoriais</b><br>6. O Administrador introduz os detalhes do autor |

Leonor Melo

04 Descrição de Casos de Uso

30

30

## Descrição de caso de uso: exemplo com generalização 4

| Nome              | Cria nova conta de blog  |
|-------------------|--|
| Fluxo de eventos  | <p>7. Os detalhes do autor são verificados include::Verifica identidade</p> <p>8. O sistema cria a nova <b>conta de blog editorial</b> e informa o Administrador</p> <p>9. O sistema envia por mail para o autor um sumário dos detalhes da sua nova conta</p> |
| Fluxo alternativo | <p>7.1. <b>O autor não é autorizado a editar os blogs</b></p> <p>7.1.a <b>O pedido de nova conta de blog editorial é rejeitado</b></p> <p>7.1.b <b>A rejeição do pedido é registada no historial do autor</b></p>  |

Leonor Melo

04 Descrição de Casos de Uso

31

31

# Casos de uso

Erros a evitar

Leonor Melo

04 Descrição de Casos de Uso

32

32

## Erro 1 : modelar processos -1



- O que está errado?

Leonor Melo

04 Descrição de Casos de Uso

33

33

## Erro 1 : modelar processos - 2



- Existe uma relação cronológica entre
  - *Emite certificado*,
  - *Envia notificação* e
  - *Levanta certificado*,
- mas tal não se representa com um diagrama de casos de uso

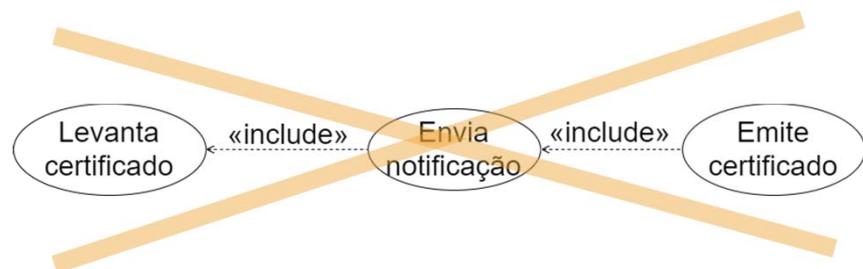
Leonor Melo

04 Descrição de Casos de Uso

34

34

## Erro 1 : modelar processos - 3



- A funcionalidade do caso de uso *Emite certificado* não é parte da funcionalidade do *Envia notificação*,
  - logo não faz sentido usar aqui a relação de «include»
- E entre *Envia notificação* e *Levanta certificado*?

Leonor Melo

04 Descrição de Casos de Uso

35

35

## Erro 2: estabelecer os limites do sistema errados - 1



- O que está errado?

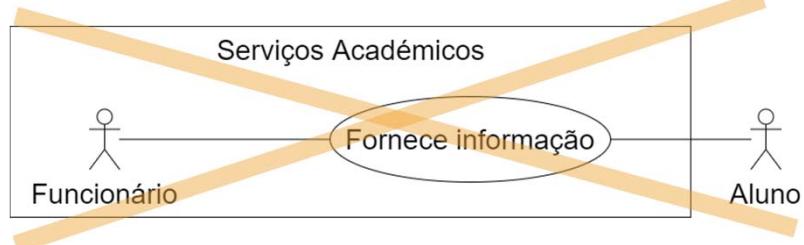
Leonor Melo

04 Descrição de Casos de Uso

36

36

## Erro 2: estabelecer os limites do sistema errados - 2



- Se faz parte do sistema não deve ser modelado como ator
- Se é necessário para o caso de uso mas não faz parte do sistema deve ser colocados fora dos limites

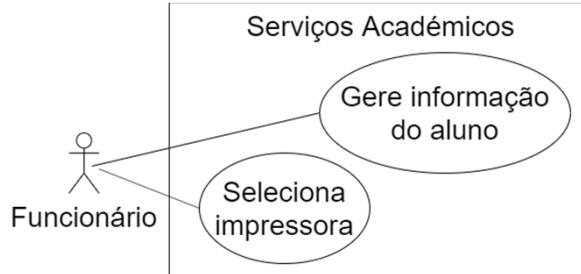
Leonor Melo

04 Descrição de Casos de Uso

37

37

## Erro 3: Misturar níveis de abstração - 1



- O que está errado?

Leonor Melo

04 Descrição de Casos de Uso

38

38

19

### Erro 3: Misturar níveis de abstração - 2



- Primeiro criar o diagrama de casos de uso baseado nos objetivos do negócio (*Gere a informação dos alunos*)
  - Depois refinar esses casos de até chegar aos requisitos técnicos (*Seleciona impressora*)

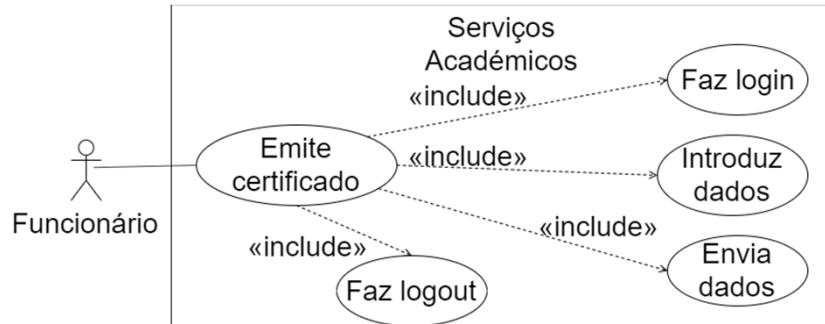
Leonor Melo

04 Descrição de Casos de Uso

39

39

### Erro 4: Decomposição funcional - 1



- O que está errado?

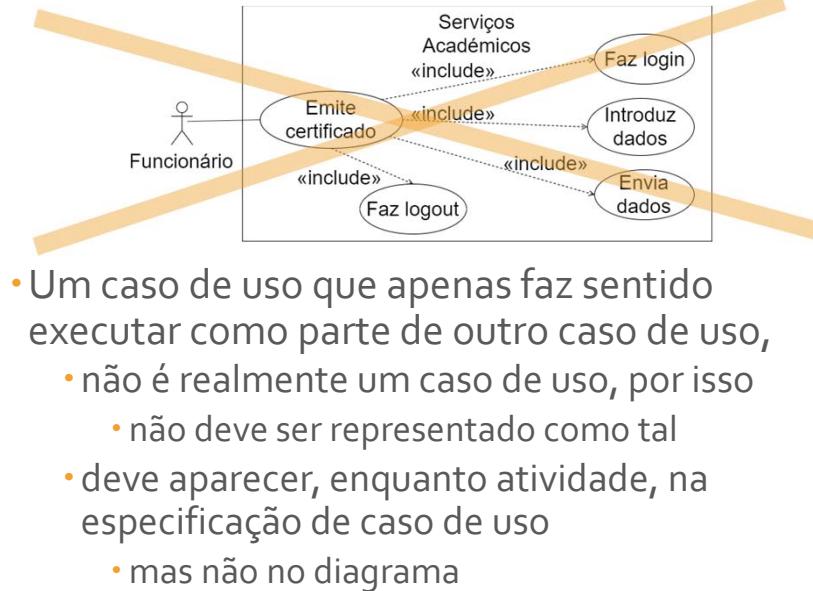
Leonor Melo

04 Descrição de Casos de Uso

40

40

## Erro 4: Decomposição funcional - 2



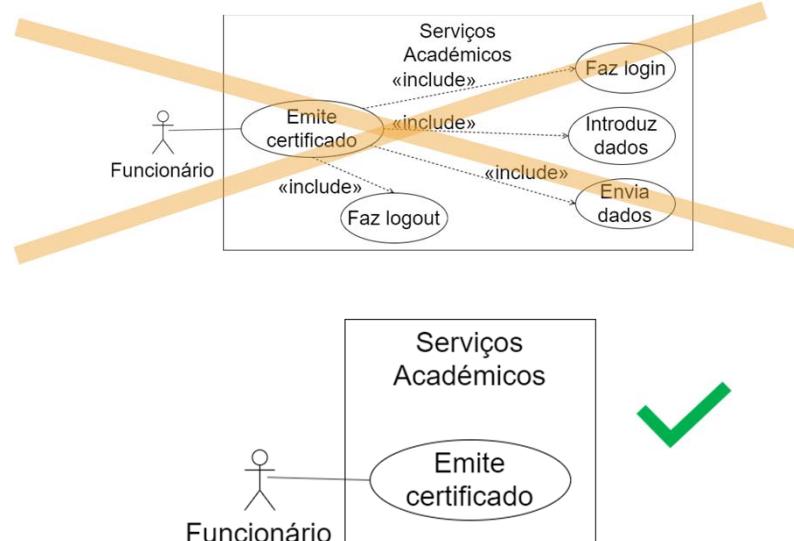
Leonor Melo

04 Descrição de Casos de Uso

4.1

41

## Erro 4: Decomposição funcional - 3



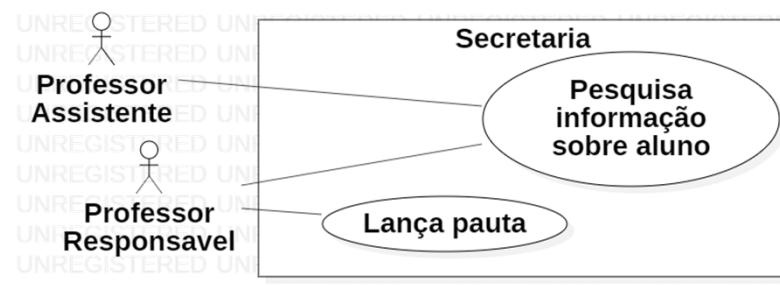
Leonor Melo

04 Descrição de Casos de Uso

4.2

42

## Erro 5: Associações incorretas - 1



- O que está errado?

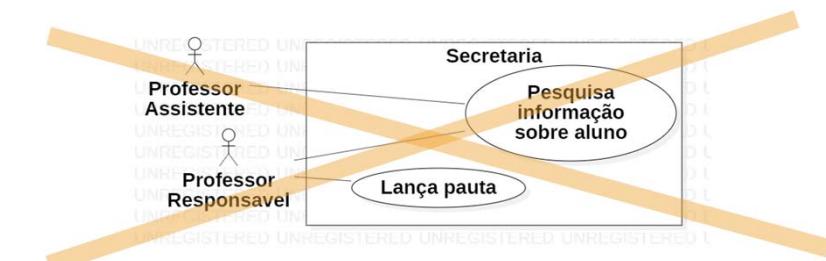
Leonor Melo

04 Descrição de Casos de Uso

43

43

## Erro 5: Associações incorretas - 2



- Caso de uso associado com dois atores
  - ambos são necessários para participar no caso de uso
- Mas quer o Professor Assistente quer o Professor Responsável devem poder Pesquisar a informação sobre um aluno sozinhos

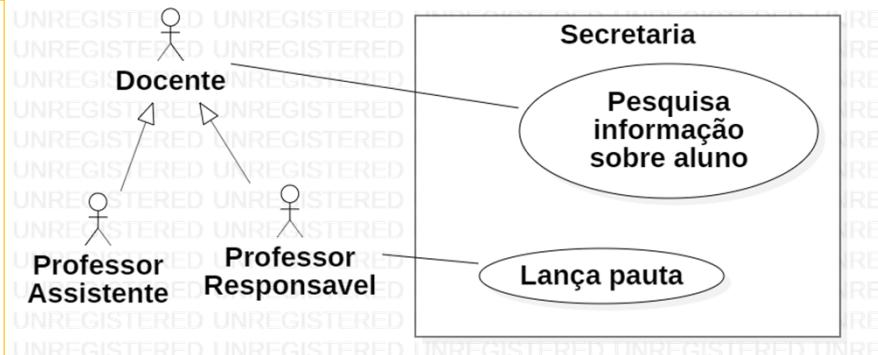
Leonor Melo

04 Descrição de Casos de Uso

44

44

## Erro 5: Associações incorretas - 3



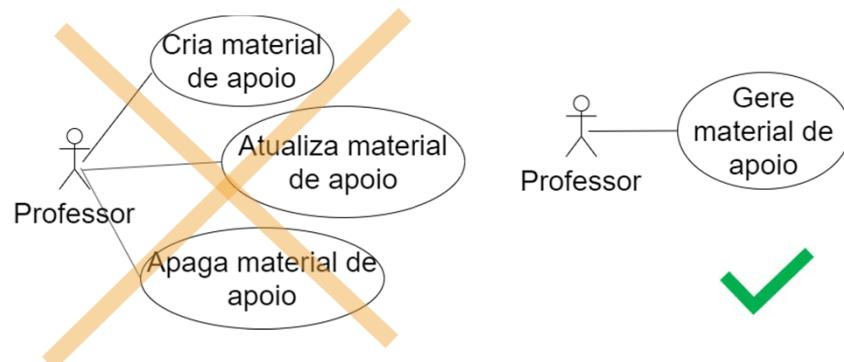
Leonor Melo

04 Descrição de Casos de Uso

45

45

## Erro 6: Casos de uso redundantes



- Pode fazer sentido agrupar casos de uso com o mesmo objetivo
  - para evitar diagramas de caso de uso demasiado complexos

Leonor Melo

04 Descrição de Casos de Uso

46

46

# Modelação e Design

## 05 : Diagrama de Atividade

Leonor Melo

leonor@isec.pt

1

### Sumário

- para que servem os diagrama de atividade
- exemplo inicial
- apresentação dos elementos mais comuns
- nós ação
- nó decisão
- condições de guarda

05 Diagrama de Atividade

2

1

## Diagrama de Atividade: modelar o workflow do sistema

- Diagrama de Atividade permite
  - especificar um comportamento
- Uma atividade pode
  - descrever a implementação de um caso de uso
- Caso de uso
  - O que o sistema deve fazer
- Diagrama de atividade
  - Como irá o sistema alcançar os seus objetivos

05 Diagrama de Atividade

3

## Diagrama de Atividade: modelar o workflow do sistema

- Diagramas de atividade
  - ações de alto-nível
    - encadeadas entre si
  - representar um processo a ocorrer no sistema
- Por exemplo,
  - modelar os passos associados à criação de uma conta de blog

05 Diagrama de Atividade

4

2

## Diagrama de Atividade: e modelos de negócio

- Diagramas de atividade
  - modelar processo de negócio
- Processo de negócio
  - série coordenada de tarefas que servem para alcançar um objetivo de negócio,
  - por exemplo, enviar a encomenda de um cliente

05 Diagrama de Atividade

5

## Diagrama de Atividade: e modelos de negócio

- vantagens de modelar processo de negócio
  - encontrar tarefas redundante
  - detetar "engarrafamentos" no processo
  - integrar novas pessoas na equipa

05 Diagrama de Atividade

6

## Diagrama de Atividade: visão de processo



- Diagrama atividade
- processos de alto-nível

05 Diagrama de Atividade

7

## Diagrama de Atividade: exemplo inicial 1

| Nome                  | Cria nova conta de blog  |
|-----------------------|--|
| Requisito relacionado | Requisito A.1  |
| Descrição sumária     | Um autor novo ou pré-existente requer uma nova conta de blog ao Administrador                  |
| Précondições          | O autor tem de ser conhecido pelo sistema e tem de fornecer uma prova de identidade apropriada |
| Póscondições          | É criada uma nova conta de blog para o autor   |

05 Diagrama de Atividade

8

Diagrama de Atividade:  
exemplo inicial  
2

| Nome                  | Cria nova conta de blog  |
|-----------------------|--|
| Póscondições de falha | O pedido de conta de blog é rejeitado  |
| Atores principais     | Administrador  |
| Atores secundários    | Base de Dados de credenciais de autores  |
| Fluxo de eventos      | <ol style="list-style-type: none"> <li>1. O Administrador pede ao sistema para criar uma nova conta de blog</li> <li>2. O Administrador seleciona o tipo de conta</li> </ol> |

05 Diagrama de Atividade

9

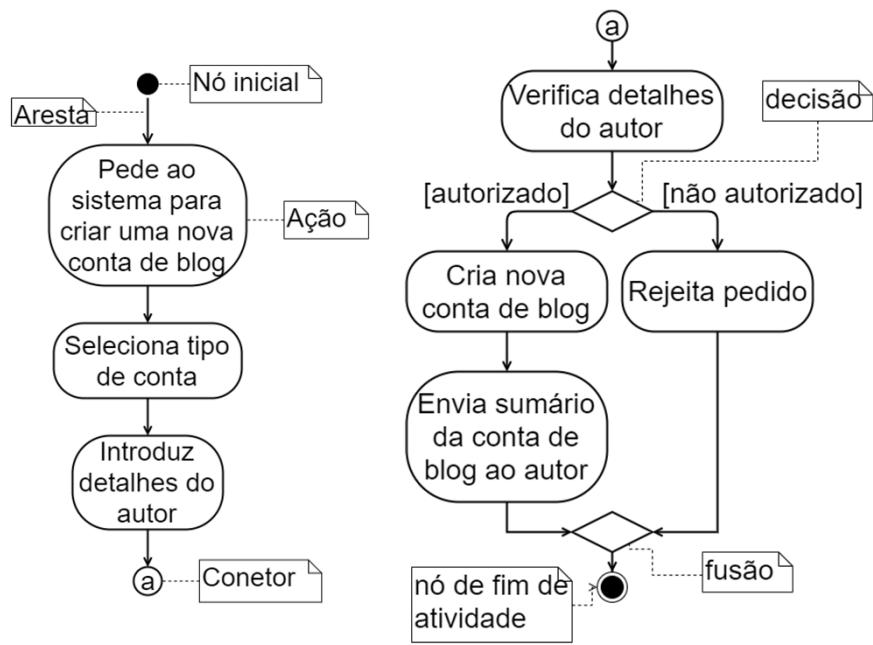
Diagrama de Atividade:  
exemplo inicial  
3

| Nome              | Cria nova conta de blog   |
|-------------------|---|
| Fluxo de eventos  | <ol style="list-style-type: none"> <li>3. O Administrador introduz os detalhes do autor</li> <li>4. Os detalhes do autor são verificados usando a Base de Dados de credenciais dos autores</li> <li>5. A nova conta de blog é criada</li> <li>6. Um sumário dos detalhes da nova conta de blog é enviado por mail para o autor</li> </ol> |
| Fluxo alternativo | <ol style="list-style-type: none"> <li>4.1. A Base de Dados de credenciais de autores não verifica os detalhes do autor             <ol style="list-style-type: none"> <li>4.1.a O pedido de nova conta do autor é rejeitado</li> </ol> </li> </ol>   |

05 Diagrama de Atividade

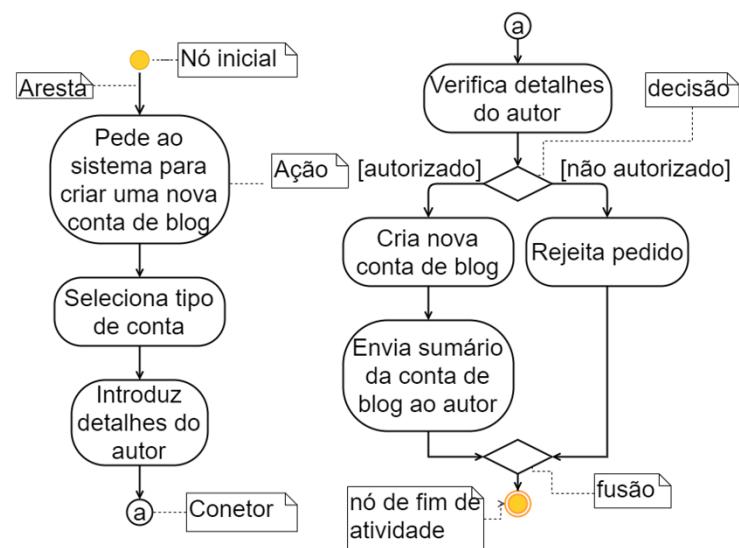
10

Diagrama de Atividade:  
exemplo inicial UML



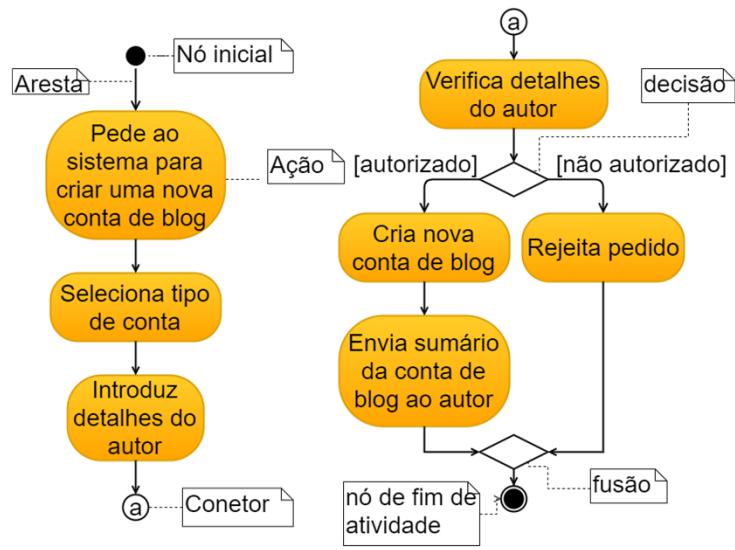
11

Diagrama de Atividade: nós de início e fim de atividade



12

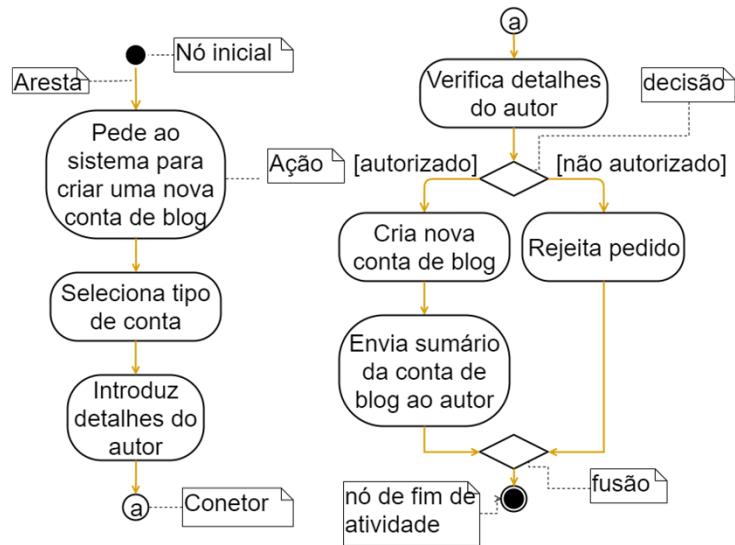
## Diagrama de Atividade: ações



05 Diagrama de Atividade

13

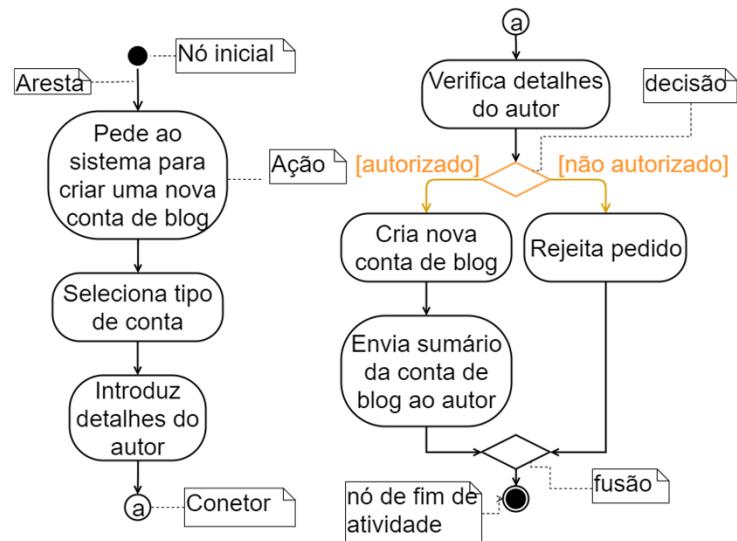
## Diagrama de Atividade: arestas ou caminhos



05 Diagrama de Atividade

14

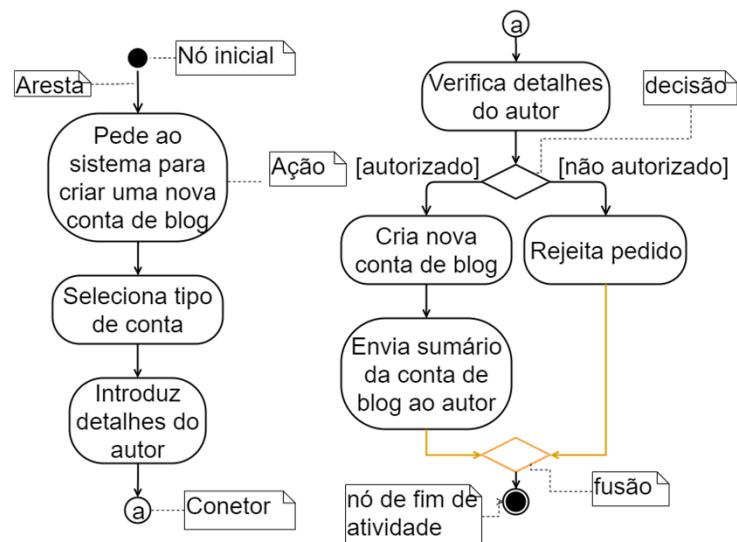
## Diagrama de Atividade: Decisão



05 Diagrama de Atividade

15

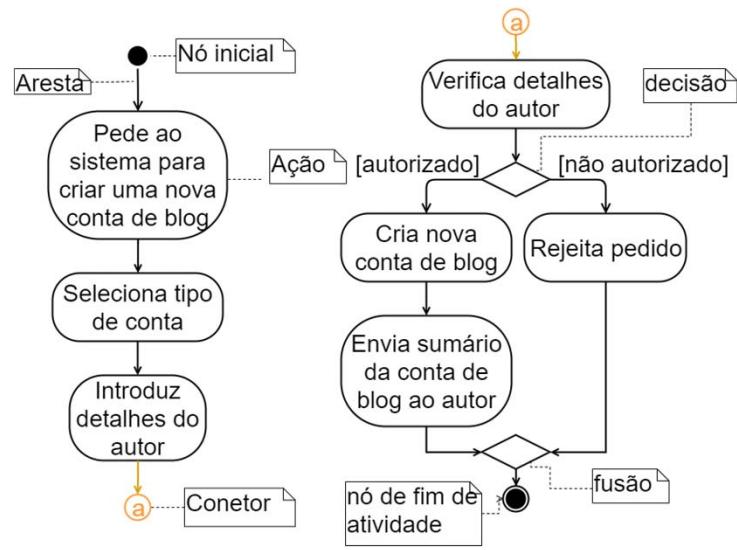
## Diagrama de Atividade: Fusão



05 Diagrama de Atividade

16

## Diagrama de Atividade: conetores



05 Diagrama de Atividade

17

## Diagrama de Atividade: Fluxo

- Os diagramas de atividade
  - representar o fluxo
    - fluxo de execução:
      - modo como se transita de uma ação para outra
    - fluxo de objetos:
      - modo como a informação transita ao longo das ações,
        - como é usada e
        - como é transformada

05 Diagrama de Atividade

18

## Diagrama de Atividade: Atividades vs ações

- Ações
  - passos
    - cálculos ou
    - tarefas
  - constituintes de um processo.
- Por exemplo:
  - CalculaTaxa ou
  - Verificaldentidade

05 Diagrama de Atividade

19

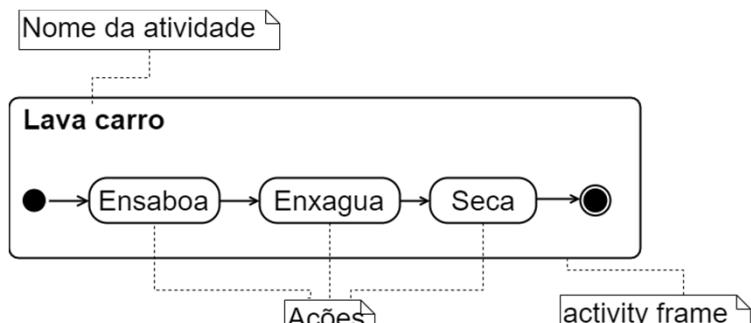
## Diagrama de Atividade: Atividades vs ações

- Atividade
  - processo que estamos a modelar.
- Por exemplo
  - atividade "Lava carro"
  - contém as ações
    - "ensaboa",
    - "enxagua" e
    - "seca"

05 Diagrama de Atividade

20

Diagrama de Atividade:  
"moldura" da atividade  
(activity frame)

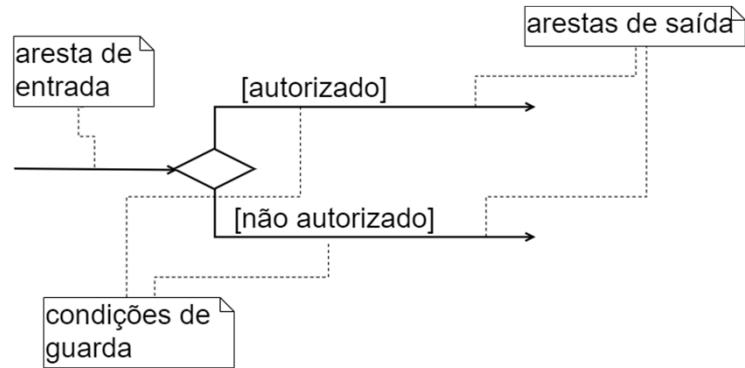


- *activity frames*
  - vários diagramas de atividade na mesma folha.

05 Diagrama de Atividade

21

Diagrama de Atividade:  
decisão



- Uma aresta de entrada,
- várias arestas de saída
  - Cada aresta de saída tem uma condição de guarda

05 Diagrama de Atividade

22

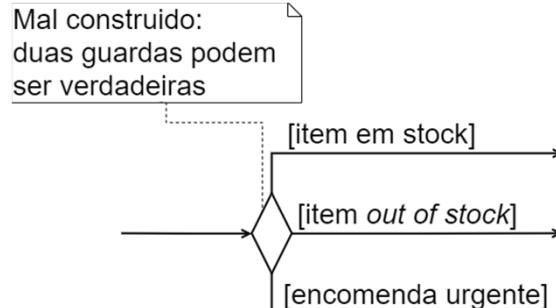
## Diagrama de Atividade: condições de guarda

- Resultado da avaliação de uma condição de guarda
  - verdadeiro ou falso
- Exemplos:
  - [autorizado]
  - [numeroPalavras >= 1000]
  - [item em stock & encomenda urgente]

05 Diagrama de Atividade

23

## Diagrama de Atividade: condições de guarda mal construídas



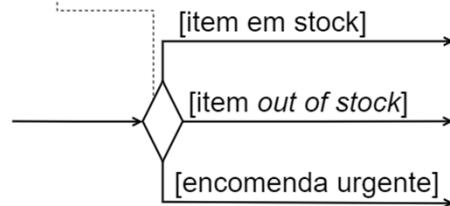
- Item em stock e encomendado de urgência
  - que aresta escolher?

05 Diagrama de Atividade

24

## Diagrama de Atividade: condições de guarda mal construídas

Mal construído:  
duas guardas podem ser verdadeiras



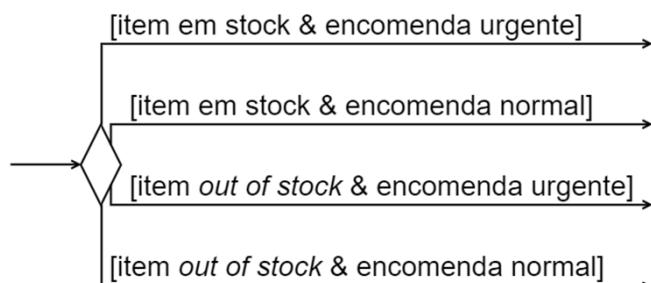
- Guardas devem ser
  - mutuamente exclusivas e
  - conjuntivamente exaustivas:

05 Diagrama de Atividade

25

## Diagrama de Atividade: condições de guarda bem construídas 1

- Solução possível:

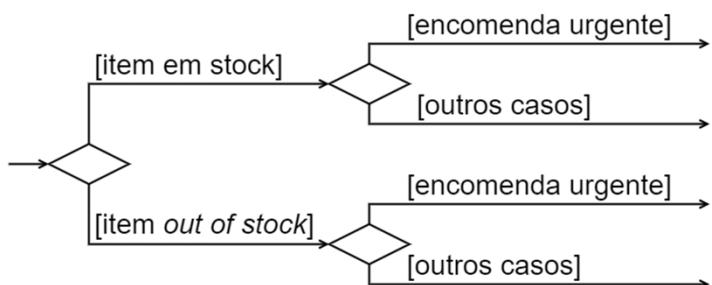


05 Diagrama de Atividade

26

# Diagrama de Atividade: condições de guarda bem construídas

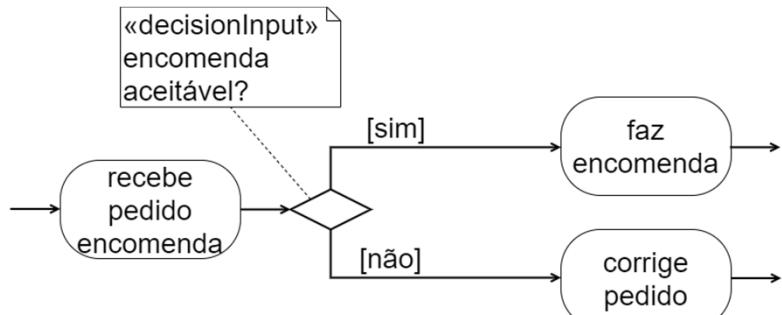
- Ainda melhor:



## 05 Diagrama de Atividade

27

# Diagrama de Atividade: comportamento de decisão



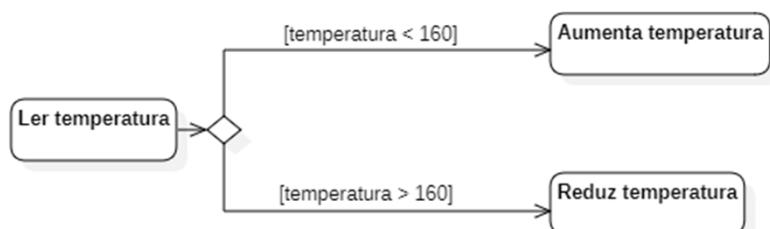
- Resultado do comportamento de decisão
    - fornecido a cada condição de guarda
  - Comportamento de decisão
    - não pode alterar o valor de objetos ou variáveis

## 05 Diagrama de Atividades

28

## Exemplo

- Imagine que está a modelar um controlador dum forno. Qual o problema do seguinte excerto de diagrama?



05 Diagrama de Atividade

# Modelação e Design

## o6 : Diagrama de Atividade

Leonor Melo

leonor@isec.pt

1

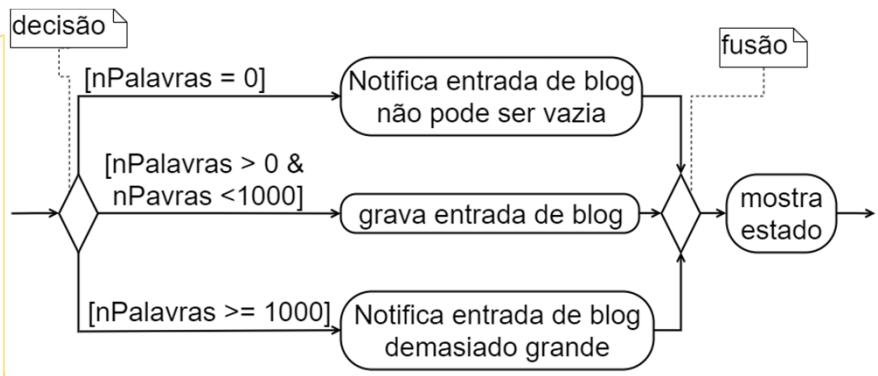
### Sumário

- Decisão (continuação): Nó de fusão
- Ciclos
- Atividades em paralelo
- Eventos de tempo
- Evocar outras atividades
- Nó objeto

o6 Diagrama de Atividade

2

## Diagrama de Atividade: fusão



- **Nó de fusão**

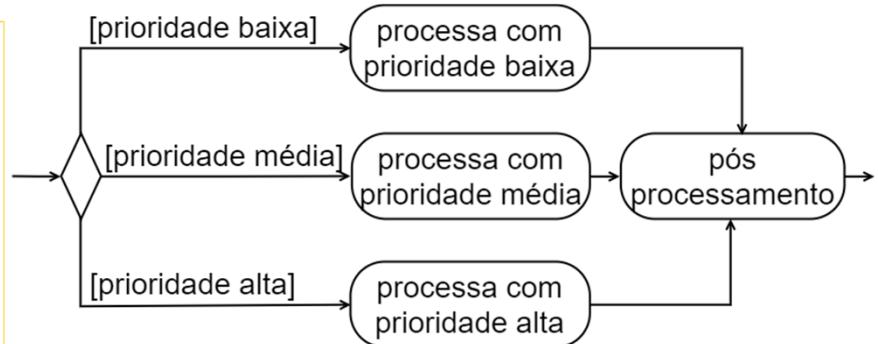
- várias arestas de entrada e
- uma só aresta de saída
- marca o fim do comportamento condicional

o6 Diagrama de Atividade

3

3

## Diagrama de Atividade: fusão UML 1.0



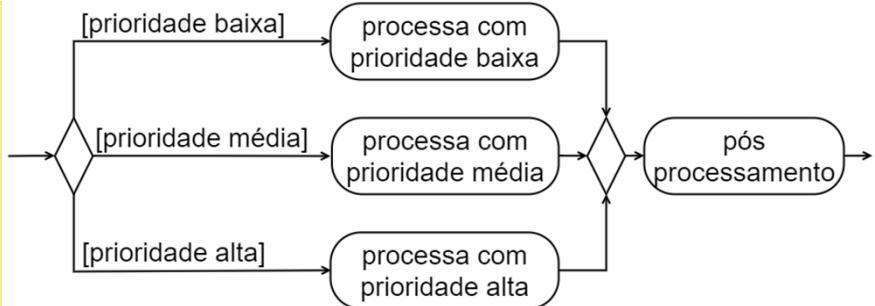
- Notação antiga
- Arestas fundem diretamente numa ação
- Em UML 2.0 significa que o pós-processamento só começa quando os 3 fluxos terminarem
  - Porque é que não faz sentido?

o6 Diagrama de Atividade

4

4

## Diagrama de Atividade: fusão UML 2.0



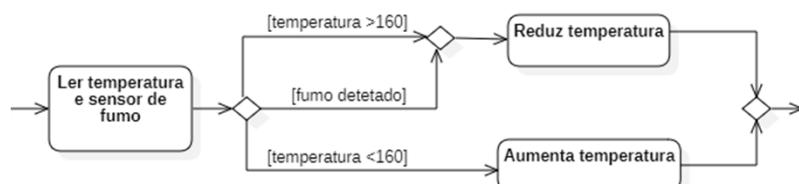
- Notação atual
- Arestas fundem-se antes seguir para a ação: interpretação de acordo com UML 2.0 consistente

o6 Diagrama de Atividade

5

## Questão 1

- O que está errado com o seguinte excerto de diagrama? Como melhorá-lo?



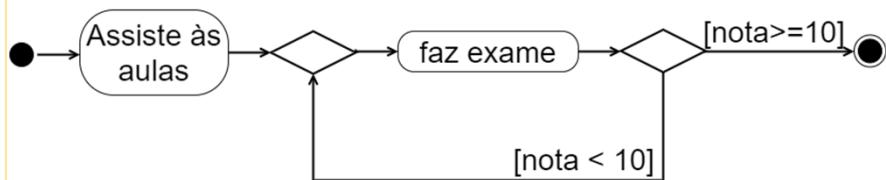
o6 Diagrama de Atividade

6

6

## Diagrama de Atividade: ciclos

- Combinando nós de fusão e decisão também é possível modelar ciclos



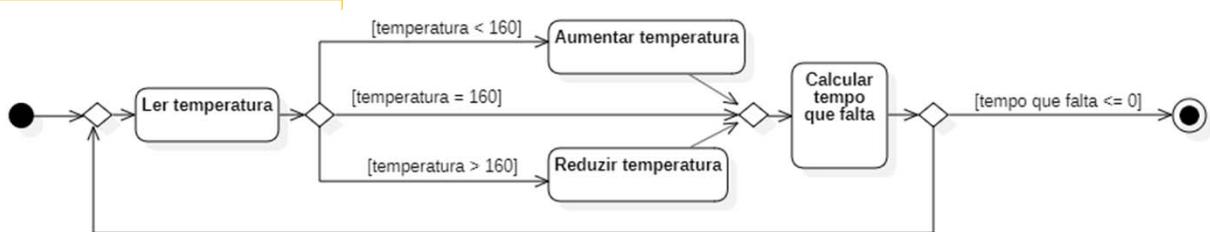
o6 Diagrama de Atividade

7

7

## Diagrama de Atividade: Questão 2

- Como se lê o diagrama acima?



o6 Diagrama de Atividade

8

8

## Diagrama de Atividade: Questão 3

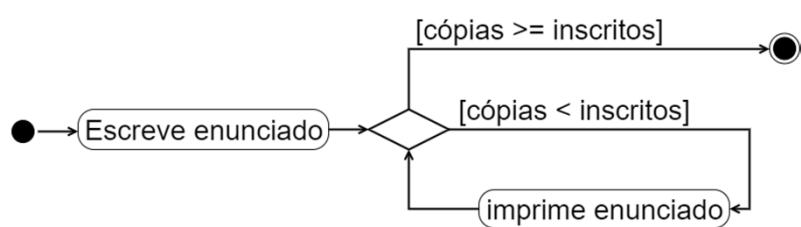
- Como modelar o seguinte processo:
  - "Enquanto não chegar à parede dar passo em frente"

o6 Diagrama de Atividade

9

9

## Diagrama de Atividade: combinar nós de fusão e decisão



- Combinar a fusão e decisão no mesmo nó
  - várias arestas de entrada
  - várias arestas de saída
  - em cada instante
    - apenas uma aresta de entrada ou saída ativa

o6 Diagrama de Atividade

10

10

## Diagramas de atividade: tarefas em paralelo

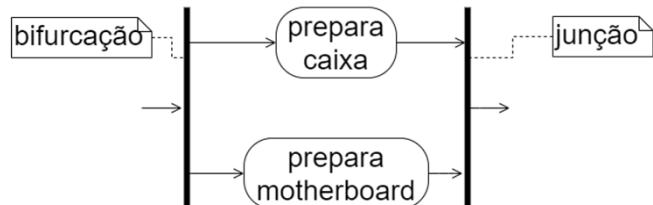
- o workflow da montagem de um computador consiste nos seguintes passos:
  - prepara "caixa"
  - prepara motherboard
  - instala motherboard
  - instala drives
  - instala cartões de vídeo e som
- O preparar da caixa e da motherboard podem ser feitos ao mesmo tempo
  - são independentes

o6 Diagrama de Atividade

11

11

## Diagramas de atividade: representação de tarefas em paralelo



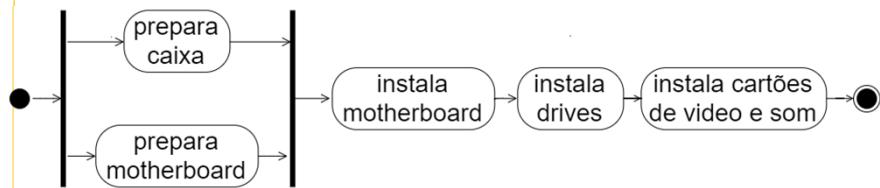
- Bifurcação (*fork*)
  - Ambas as arestas são percorridas
- Num modelo detalhado podem representar
  - múltiplos processos ou *threads*

o6 Diagrama de Atividade

12

12

## Diagramas de atividade: representação de tarefas em paralelo



- Ações em paralelo
  - podem ter durações diferentes
- Junção
  - ambas as tarefas tem de ter terminado antes da execução continuar

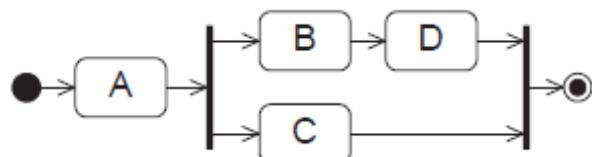
o6 Diagrama de Atividade

13

13

## Questão 4

- Quais as sequências de ações possíveis?



o6 Diagrama de Atividade

14

14

## Diagramas de atividade: Eventos de tempo

- Modelar o tempo:
  - modelar tempo de espera:
    - "a conta deve ser enviada 3 dias depois de enviar a encomenda"
  - modelar ações que são espoletadas a intervalos regulares:
    - "deve ser feito um backup ao sistema uma vez por semana"

o6 Diagrama de Atividade

15

15

## Diagramas de atividade: Eventos de tempo - eventos únicos



- Eventos de tempo
  - tempo de espera entre duas ações
- Com aresta de entrada
  - Evento ocorre apenas uma vez
  - Legenda
    - tempo a esperar

o6 Diagrama de Atividade

16

16

## Diagramas de atividade: Eventos de tempo - eventos recorrentes

- Sem aresta de entrada
  - Evento recorrente
  - Legenda
    - frequência de ativação
- Modo alternativo de começar uma atividade
  - modelar atividades lançadas periodicamente



06 Diagrama de Atividade

17

17

## Diagramas de atividade: evocar outras atividades 1

- 
- Diagrama de Atividade que ilustra a evocação de outras atividades. Ele mostra uma sequência de atividades: "prepara caixa", "instala motherboard", "instala drives" e "instala cartões de vídeo e som". As primeiras duas atividades estão agrupadas por uma barreira (parênteses verticais). A atividade "prepara caixa" tem uma seta apontando para a barreira. A atividade "prepara motherboard" é evocada a partir da barreira, com uma seta saindo da barreira e apontando para ela. As atividades subsequentes ("instala motherboard", "instala drives" e "instala cartões de vídeo e som") são executadas em paralelo, com suas respectivas setas saindo da barreira.
- *call activity node*
    - nó que evoca outra atividade
  - Atividade evocada
    - Modelada num diagrama separado.
    - Nome novo diagrama = nome do nó que o evocou

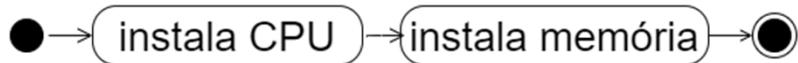
06 Diagrama de Atividade

18

18

## Diagramas de atividade: evocar outras atividades 2

### prepara motherboard



- Ação/atividade evocada
  - tem nó inicial
  - tem nó final
  - contida numa *activity frame*
  - nome
- Depois de alcançado o nó final
  - fluxo de ação volta para a atividade original

o6 Diagrama de Atividade

19

19

## Diagramas de atividade: objetos

- Modelar fluxo de dados
- Exemplo:
  - Definir processo para gerir encomendas
  - Cada passo precisa de informação sobre a encomenda
    - informação de pagamento,
    - custo da transação,...
- Solução
  - nó objeto Encomenda no diagrama de atividade
    - para modelar essa informação

o6 Diagrama de Atividade

20

20

## Diagramas de atividade: Nó objeto - 1

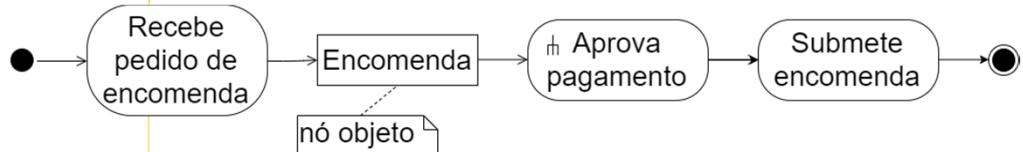
- **Nó objeto (*object node*)**
  - representa o fluir dos dados
  - informação:
    - em formato digital
  - **objeto físico:**
    - exemplo: num sistema não automatizado, pode representar uma ordem de trabalho física que dá origem ao processo

o6 Diagrama de Atividade

21

21

## Diagramas de atividade: Nó objeto - 2

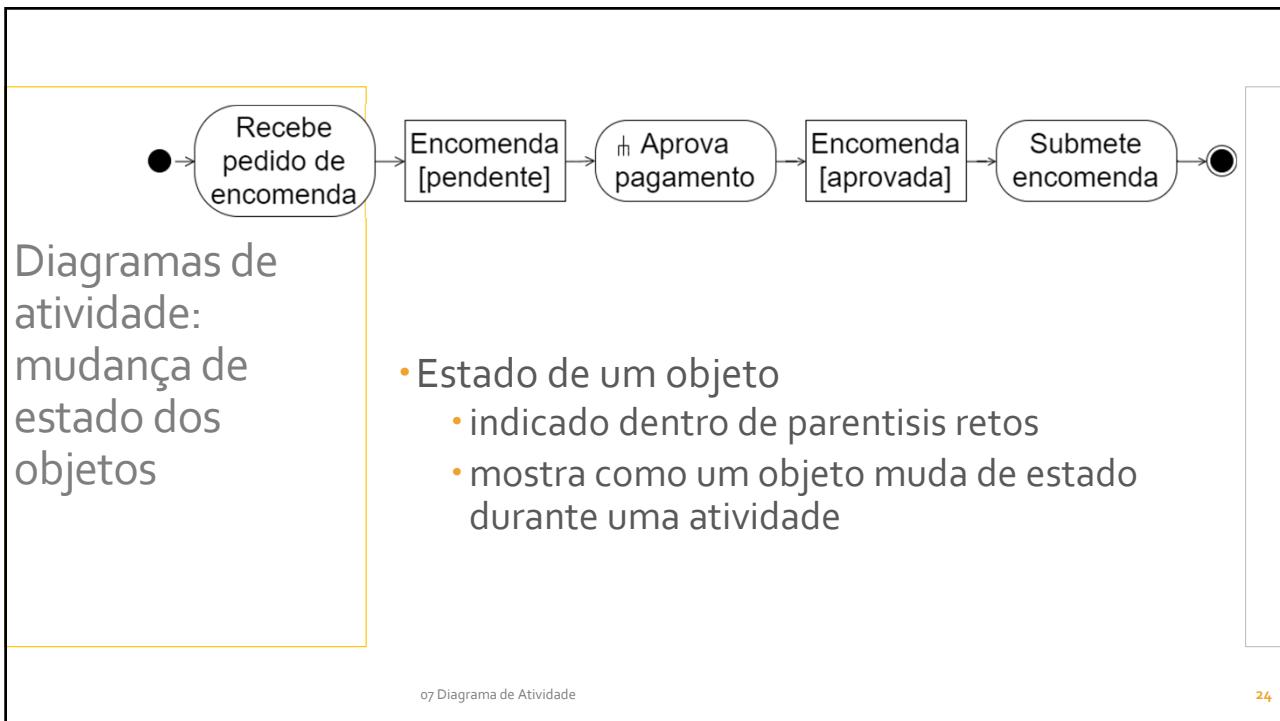
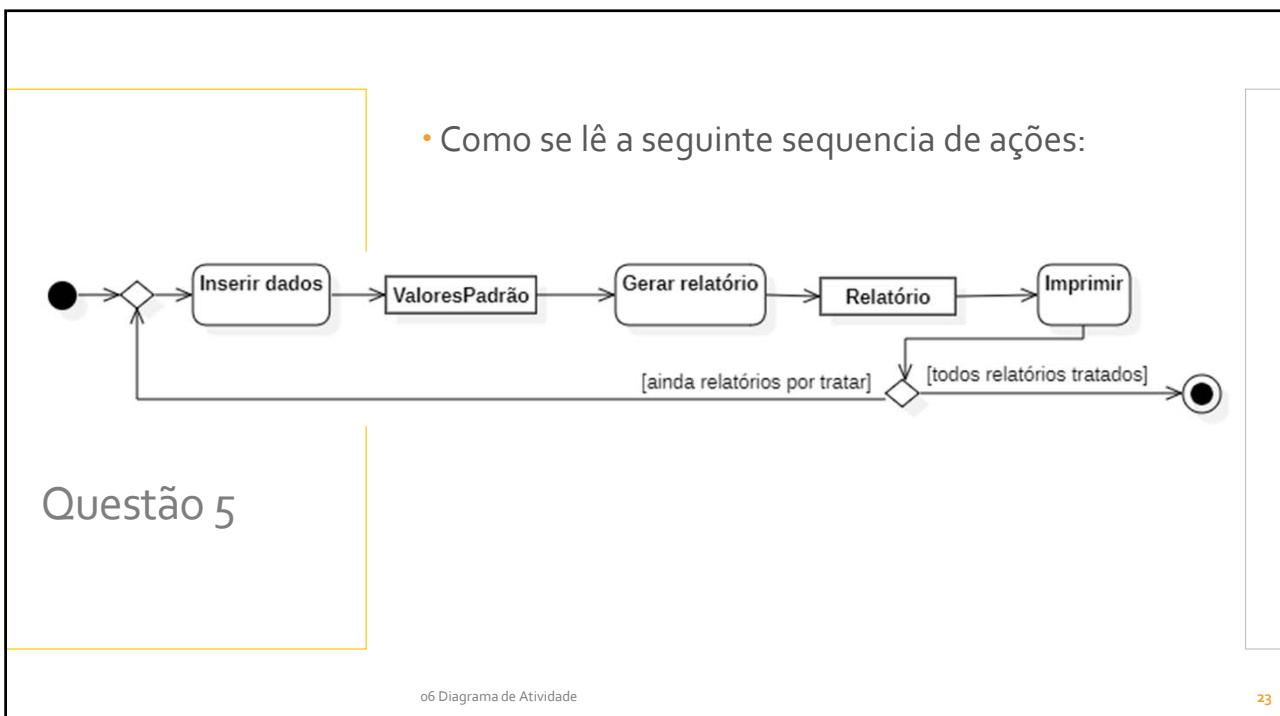


- Objeto que está disponível em determinado instante da atividade
- Mostra como esse objeto é
  - criado,
  - modificado ou
  - usado
- pelas várias ações

o6 Diagrama de Atividade

22

22



# Modelação e Design

## 07 : Diagrama de Atividade

Leonor Melo

leonor@isec.pt

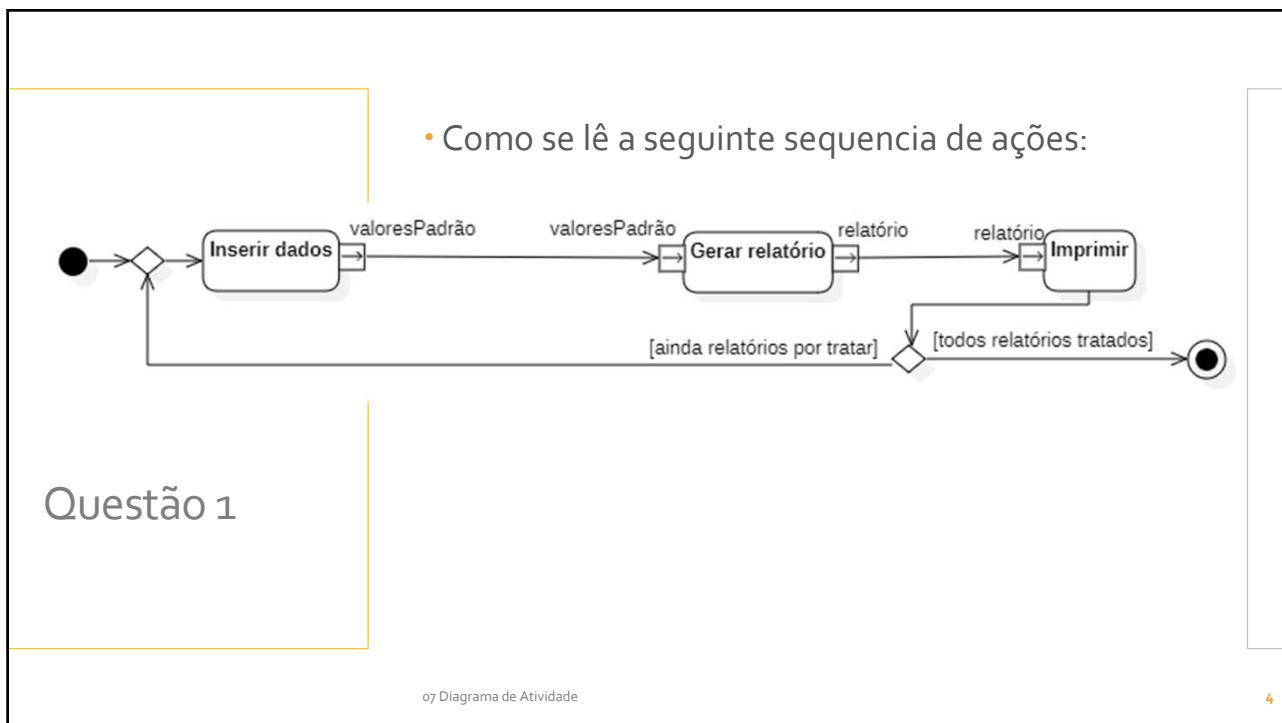
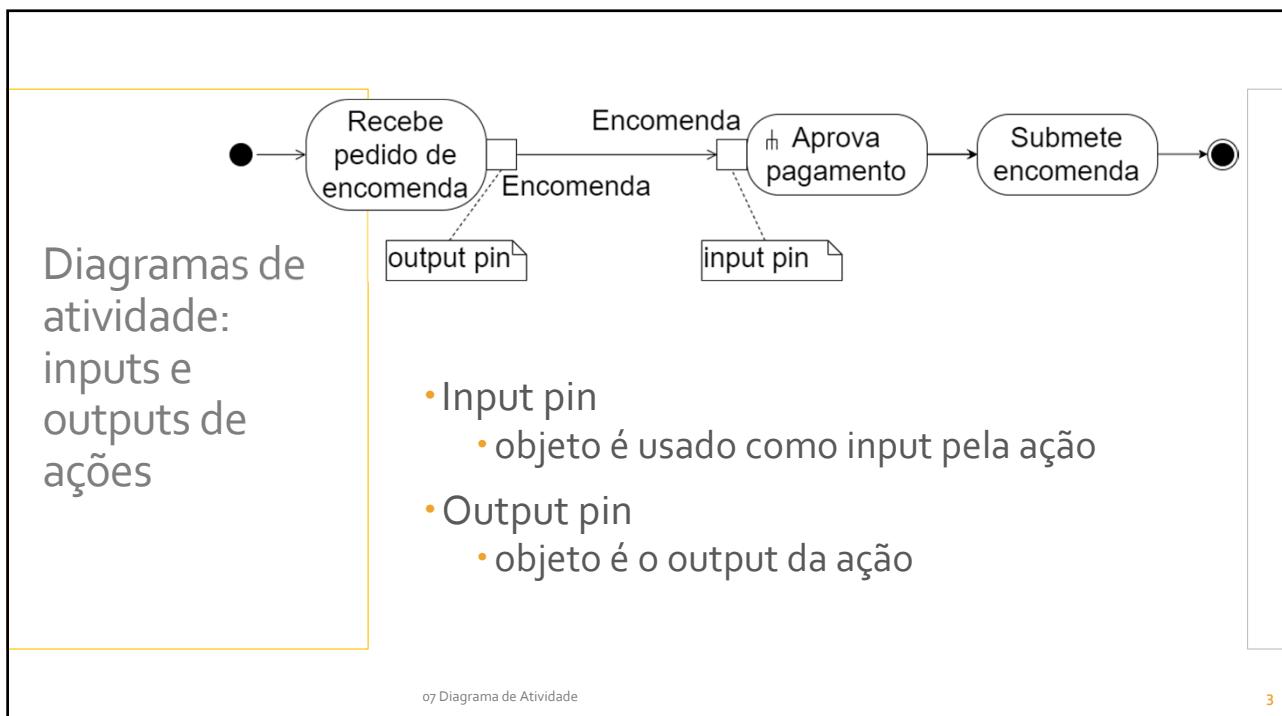
1

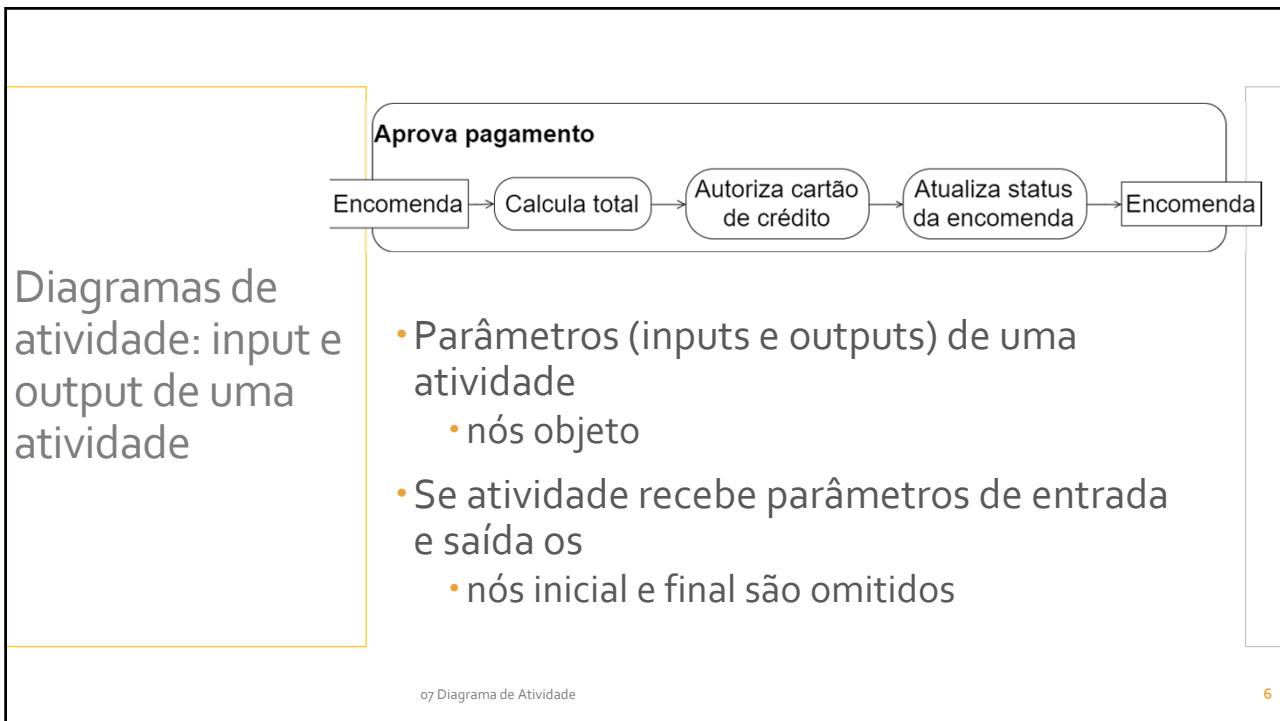
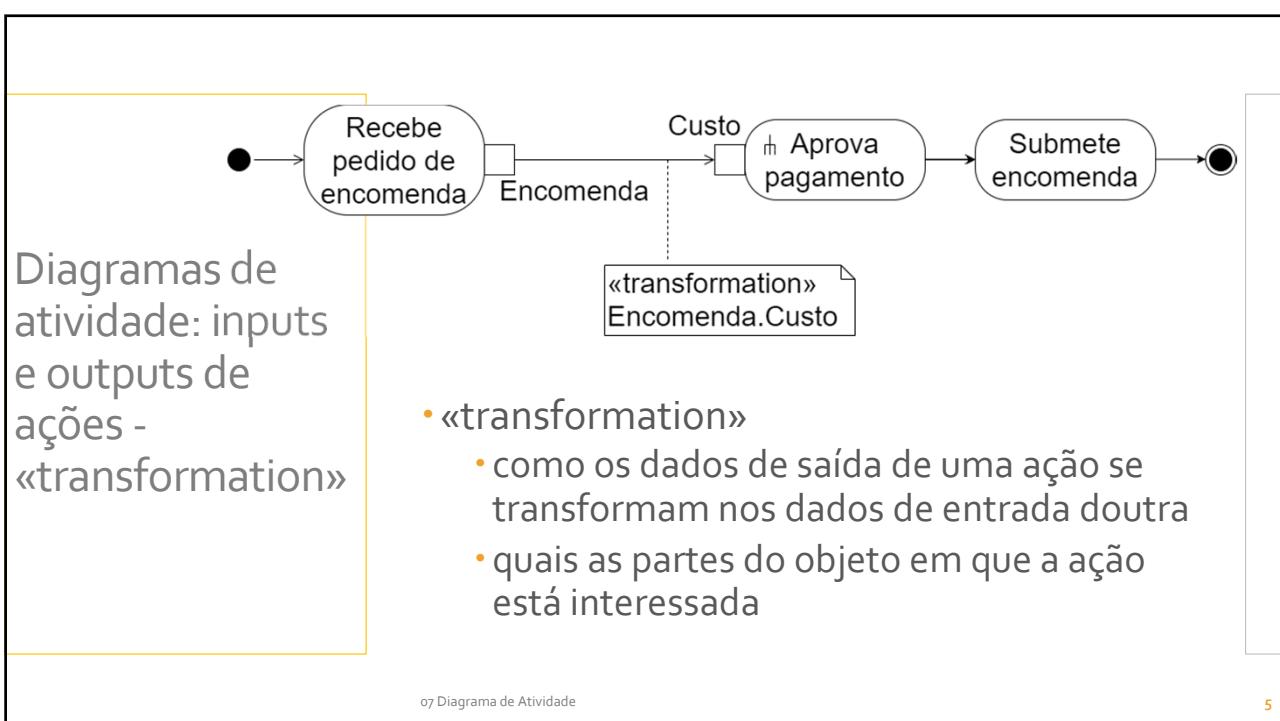
### Sumário:

- Nós objeto
- Sinais
- Exceções
- Fim do fluxo da atividade
- Swimlanes: partições

07 Diagrama de Atividade

2





Diagramas de  
atividade:  
receber e enviar  
sinais

- Sinais

- interações com participantes externos
- enviado (*send*) ou recebido (*receive*)
  - do ponto de vista da atividade
- Exemplo:
  - O click de um botão faz com que o código associado ao botão seja executados
    - sinal *recebido* do ponto de vista da atividade de handling do botão

07 Diagrama de Atividade

7

Diagramas de  
atividade:  
receber e enviar  
sinais

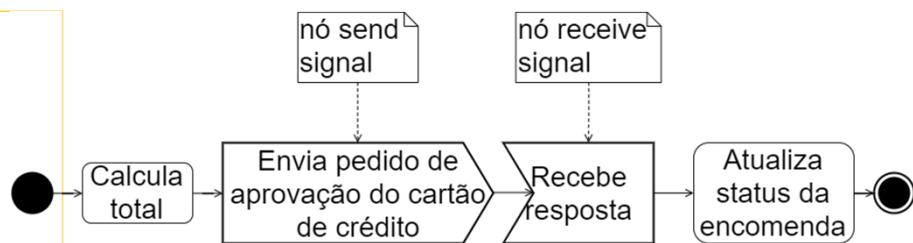
- São sinais enviados ou recebidos?

- O sistema avisa o cliente que a encomenda está atrasada
  - atividade de envio da encomenda
- O sistema envia um pedido à empresa responsável pela gestão das transações com cartões de crédito para aprovar uma transação e recebe depois a resposta da empresa
  - atividade *Aprova cartão de crédito*

07 Diagrama de Atividade

8

## Diagramas de atividade: receive signal



- *receive signal*

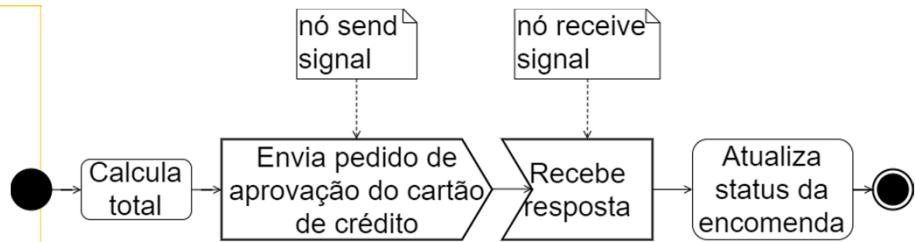
- faz com que a ação se inicie.
- A ação
  - sabe como lidar com o sinal e
  - espera receber o sinal
- mas não sabe quando isso irá acontecer

07 Diagrama de Atividade

9

9

## Diagramas de atividade: send signal



- *send signal*

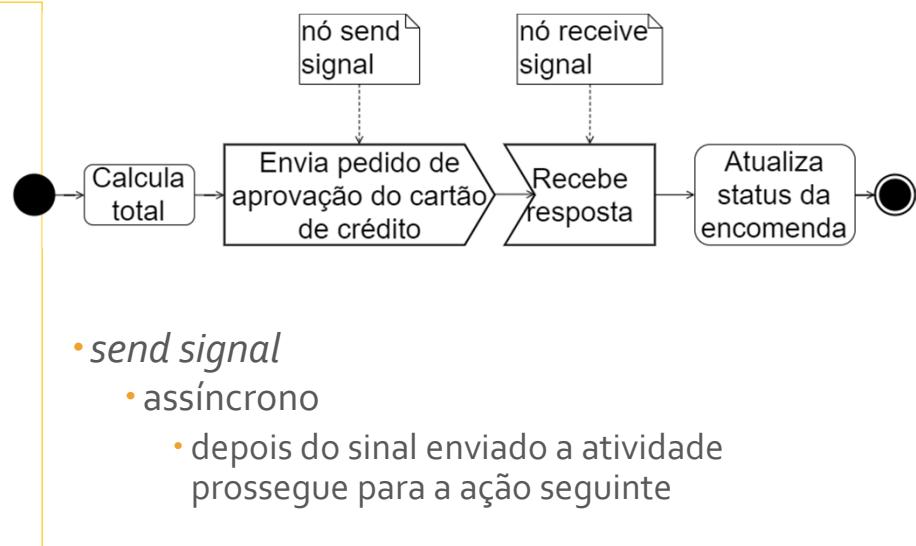
- sinal enviado para um participante externo.
- A reação do participante ao sinal não faz parte do diagrama

07 Diagrama de Atividade

10

10

## Diagramas de atividade: receber e enviar sinais - sincronia

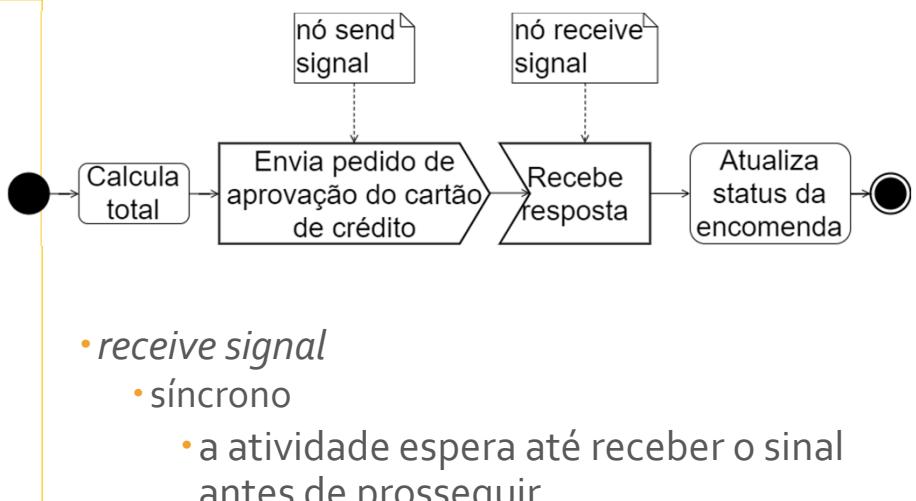


07 Diagrama de Atividade

11

11

## Diagramas de atividade: receber e enviar sinais - sincronia

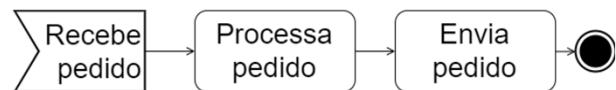


07 Diagrama de Atividade

12

12

Diagramas de atividade:  
receive signal sem aresta de entrada



- receive signal sem aresta de entrada
  - enquanto a atividade de que faz parte estiver ativa
    - está sempre à espera do sinal
- receive signal com aresta de entrada
  - só fica à espera imediatamente após a ação anterior

07 Diagrama de Atividade

13

13

Diagramas de atividade:  
começar uma atividade

- Representar o início de uma atividade
  - com um nó de início de atividade (página 13)
  - com a receção de input (pagina 7)
  - usando um nó de evento de tempo (slides 06 - diagrama atividade)
  - usando um nó de *receive signal* (página 14)

07 Diagrama de Atividade

14

14

## Diagramas de atividade: exceções

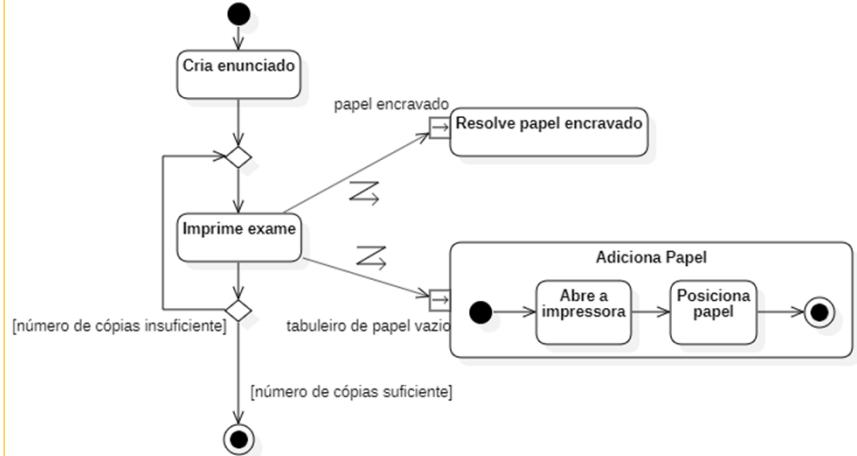
- No funcionamento normal se uma ação falhar a execução termina
- Se uma ação tiver um exception handler para uma situação de erro específica, e essa situação de erro ocorrer
  - O handler é executado em vez da ação que causou o erro
  - A execução continua normalmente

07 Diagrama de Atividade

15

15

## Diagramas de atividade: exceções



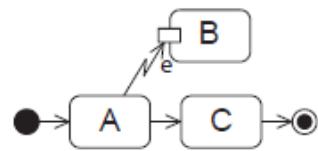
07 Diagrama de Atividade

16

16

## Questão 2:

- Quais as sequências de ação possíveis?

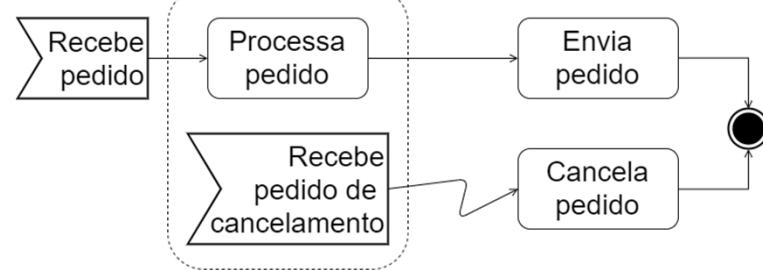


07 Diagrama de Atividade

17

17

## Diagramas de atividade: interromper atividades - 1



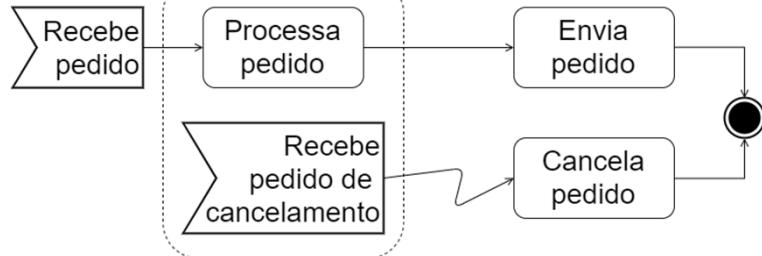
- Região de interrupção
  - parte do processo que pode ser terminado por um evento
    - por exemplo, um processo que pode ser interrompido pelo utilizador

07 Diagrama de Atividade

18

18

## Diagramas de atividade: interromper atividades - 2



- Ação interrompida
  - se sinal for recebido enquanto a ação dentro da região de interrupção estiver ativa

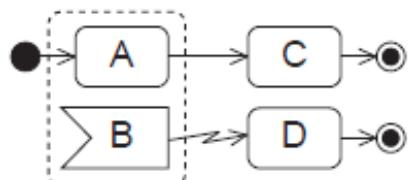
07 Diagrama de Atividade

19

19

## Questão 3:

- Quais as sequências de ação possíveis?

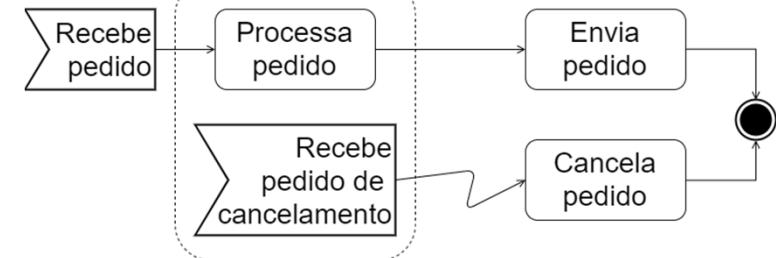


07 Diagrama de Atividade

20

20

## Diagramas de atividade: terminar a atividade



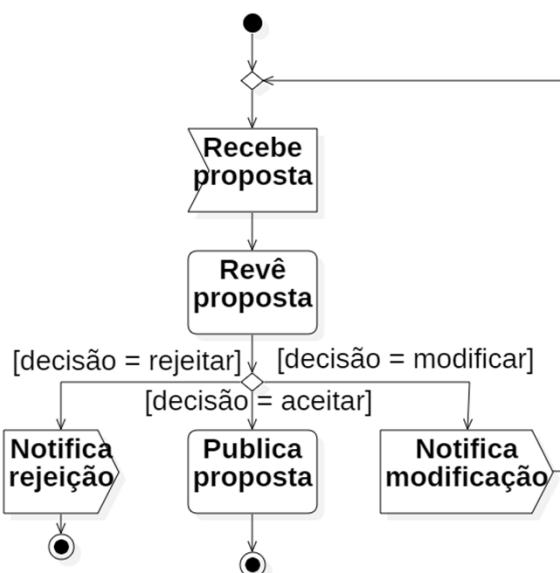
- Nó terminal com várias arestas de entrada
  - atividade termina assim que o nó for alcançado

07 Diagrama de Atividade

21

21

## Diagramas de atividade: terminar a atividade



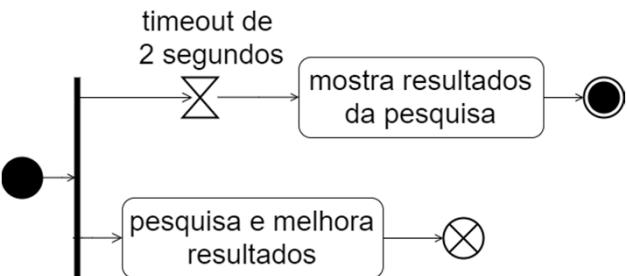
- Vários nós terminais
  - atividade termina assim que o primeiro deles for alcançado

07 Diagrama de Atividade

22

22

## Diagramas de atividade: terminar o fluxo



- nó de fim de fluxo (*final flow*)
  - termina o seu caminho, mas não a atividade
  - atenção aos nós finais depois de uma bifurcação

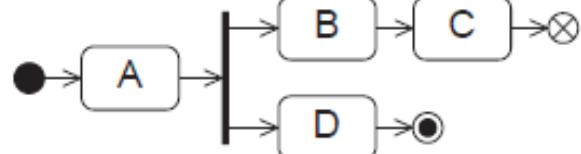
07 Diagrama de Atividade

23

23

## Questão 4:

- Quais as sequências de ação possíveis?



07 Diagrama de Atividade

24

24

## Diagramas de atividade: atribuir responsabilida des: partições - 1

- Partições
  - que intervenientes
  - são responsáveis por que ações
- Exemplo:
  - Atividade de processamento de encomendas necessita de
    - departamento de envios
      - para enviar os produtos
    - departamento de contabilidade
      - para enviar as contas aos clientes

07 Diagrama de Atividade

25

25

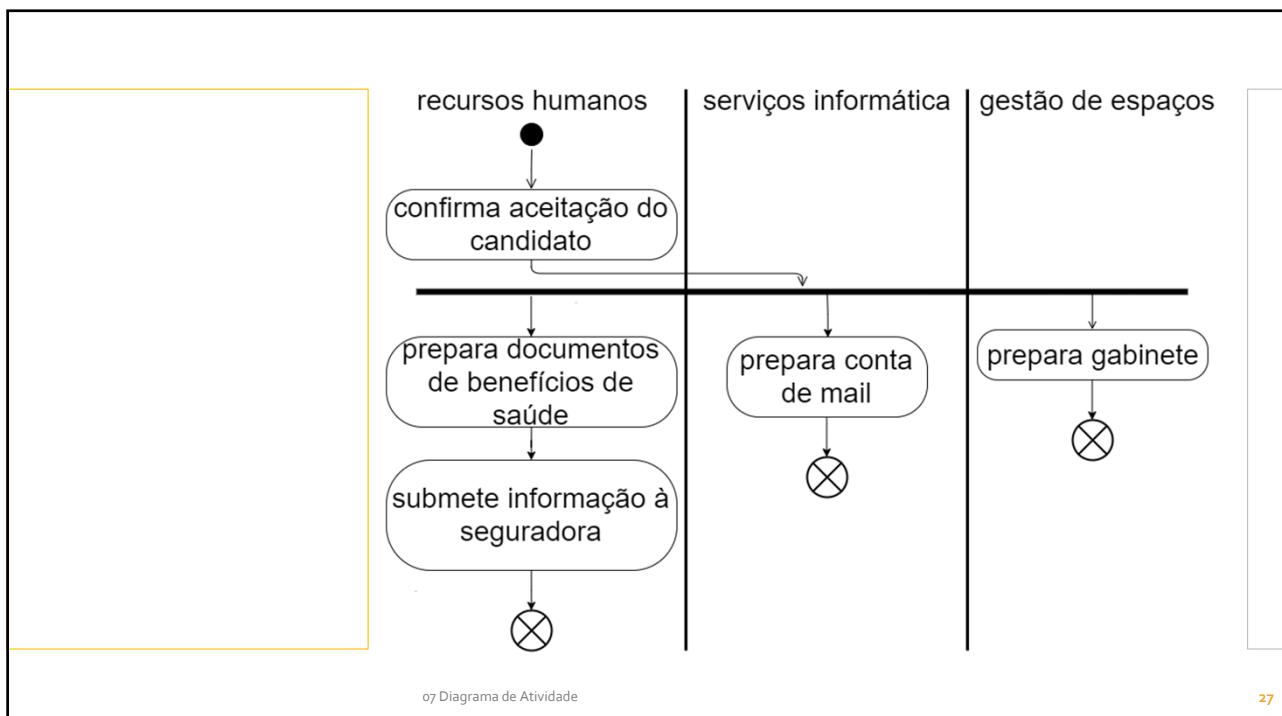
## Diagramas de atividade: atribuir responsabilida des: partições - 2

- Exemplo
  - Para contratar um funcionário estão envolvidos
    - departamentos de recursos humanos,
      - fazer o contrato
      - preparar documentos sistema de saúde
      - preparar documentos seguradora
    - serviços de informática,
      - criar conta mail
    - gabinete de gestão de espaços
      - arranjar gabinete

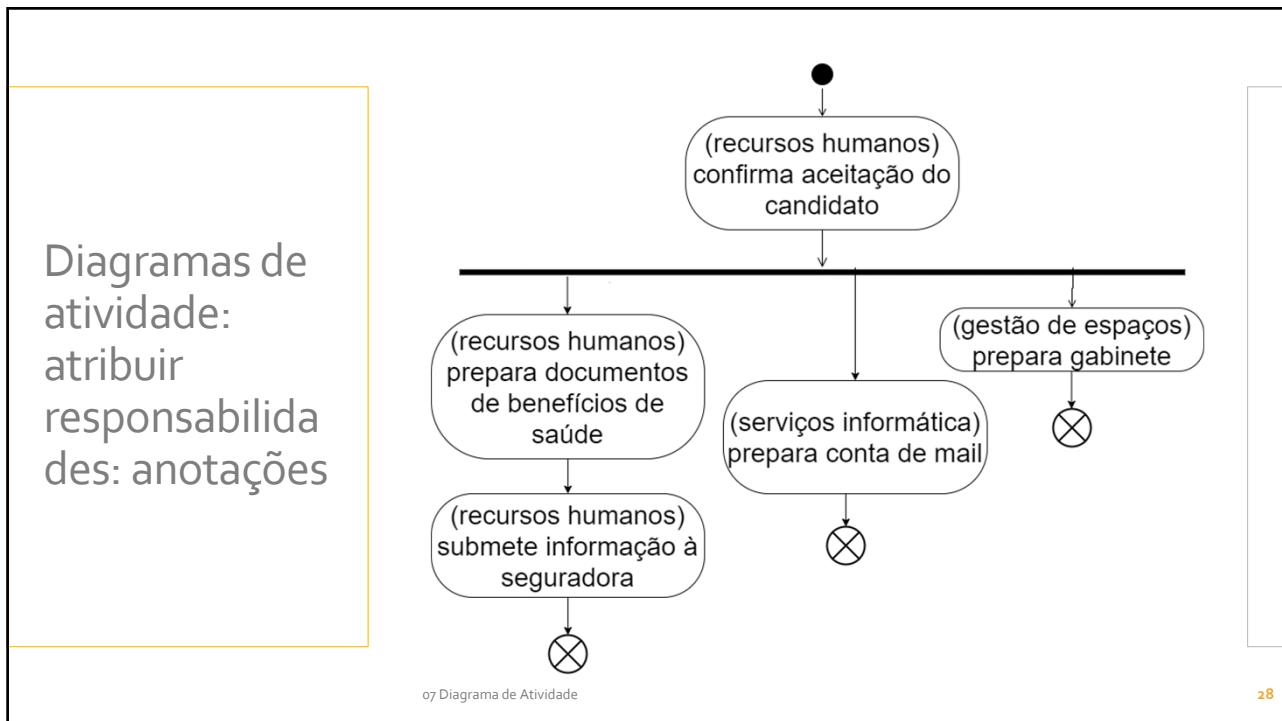
07 Diagrama de Atividade

26

26



27



28

# Modelação e Design

## o8: Modelo do Domínio

Leonor Melo

leonor@isec.pt

1

### Sumário

- O que é o modelo do domínio?
- Para que serve o modelo do domínio?
- Notação
- Classes conceptuais
- Atributos

Leonor Melo

o8 Modelo do Domínio

2

## Modelo do Domínio

- Modelo da Análise Orientada a Objetos
  - mais importante e
  - clássico
- Visualizar os
  - conceitos mais relevante do domínio
  - modo como se relacionam entre si
- Feito nas fases iniciais do projeto

Leonor Melo

08 Modelo do Domínio

3

## Modelo do Domínio

- Baseado em:
  - caso de uso
  - peritos no domínio
  - informações dos outros stakeholders

Leonor Melo

08 Modelo do Domínio

4

## Modelo do Domínio

- Classes conceptuais
  - não são classes de software
  - podem servir de inspiração para objetos de software
- Descrevem o problema
  - e não a solução!

Leonor Melo

08 Modelo do Domínio

5

## Modelo do Domínio

- Visualizar os conceitos
  - Mais fáceis de compreender
  - Mais fácil comunicar sobre eles
- Diagramas de classes UML
  - Subconjunto elementos

Leonor Melo

08 Modelo do Domínio

6

6

## Diagramas de classe

- Diagrama UML mais conhecido
- Modelar a estrutura de um sistema:
  - os elementos
  - as relações entre os elementos

Leonor Melo

08 Modelo do Domínio

7

## Diagramas de classe

- Aplicado em várias fases do processo de desenvolvimento de software
  - **fases iniciais:**
    - modelo conceptual e
    - vocabulário do sistema
  - **fase de design:**
    - modelo refinado e
    - transformado
      - classes de software

Leonor Melo

08 Modelo do Domínio

8

Modelo do Domínio:  
exemplo: caso de uso em causa - 1

| Caso de Uso      | Processa venda   |
|------------------|--|
| Descrição        | Operador da caixa regista cada um dos itens que o cliente quer comprar e recebe o pagamento  |
| Pré-condições    | Operador identificado e autenticado  |
| Pós-condições    | Venda registada. Pagamento recebido  |
| Autor principal  | operador   |
| fluxo de eventos | <ol style="list-style-type: none"> <li>1. Cliente chega à caixa com os produtos ou serviços que quer comprar</li> <li>2. Operador inicia uma nova venda</li> </ol> |

Leonor Melo

08 Modelo do Domínio

9

Modelo do Domínio:  
exemplo: caso de uso em causa - 2

| Caso de Uso      | Processa venda   |
|------------------|--|
| fluxo de eventos | <ol style="list-style-type: none"> <li>3. Operador introduz identificador do item e o número de unidades</li> <li>4. Sistema regista cada por cada item uma linha de venda que apresenta a descrição do item, a quantidade, o preço e o subtotal. O preço é calculado a partir de uma série de regras.</li> <li>Operador repete os passos 2-3 até que assinala que terminou o registo</li> <li>5. Sistema apresenta o total com o imposto calculado</li> </ol> |

Leonor Melo

08 Modelo do Domínio

10

10

Modelo do Domínio:  
exemplo: caso de uso em causa - 3

### Caso de Uso

- fluxo de eventos
6. Operador comunica total ao cliente e pede o pagamento
  7. Cliente paga e sistema lida com o pagamento
  8. O sistema regista a venda e envia a informação sobre a venda e pagamento para a contabilidade externa (para contabilidade e cálculo de comissões) e para o sistema de inventário (para atualização do inventário)

Leonor Melo

08 Modelo do Domínio

11

11

Modelo do Domínio:  
exemplo: caso de uso em causa - 4

### Caso de Uso

- fluxo de eventos
9. Sistema apresenta a fatura
  10. Cliente sai com produtos (se houverem) e fatura
- fluxo alternativo
- 7a. Pagamento com dinheiro
    1. Operador introduz quantia entregue
    2. Sistema apresenta o troco e abre a gaveta do dinheiro
    3. Operador deposita dinheiro entregue e entrega troco ao cliente
    4. Sistema regista pagamento feito

Leonor Melo

08 Modelo do Domínio

12

12

## Modelo de domínio: questão 1

- Quais parecem ser os conceitos mais relevantes?

Leonor Melo

08 Modelo do Domínio

13

13

## Modelo do Domínio: decomposição do domínio

- Processo de desenvolvimento de software *Unified Process (UP)*:
  - Decomposição do domínio:
    - conceitos e
    - objetos relevantes
  - Modelo do domínio / modelo conceptual
    - representação visual
      - de classes conceptuais ou objetos do domínio
      - *da situação real* (não de objetos de software)

Leonor Melo

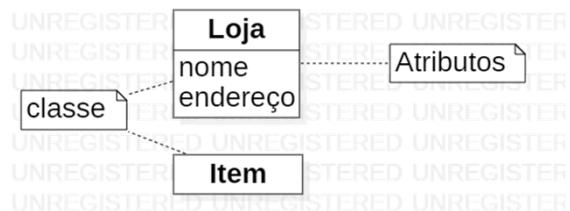
08 Modelo do Domínio

15

15

# Modelo do Domínio: Classes representação UML

- Modelo de domínio
    - classes contêm
      - um nome
      - um nome e atributos



Leonor Melo

## o8 Modelo do Domínio

16

16

- Diagrama de classe de design
    - Classes podem ter
      - Nome
      - Atributos
      - Operações



Leonor Melo

## o8 Modelo do Domínio

17

17

## Modelo do Domínio: abstração

- Um modelo do domínio
  - série de diagramas de classe
  - não são definidas operações (métodos)
- perspetiva conceptual:
  - objetos do domínio
    - ou classes conceptuais
  - associações entre classes conceptuais
  - atributos de classes conceptuais

Leonor Melo

08 Modelo do Domínio

18

18

## Modelo do Domínio: abstração

- Modelo do domínio:
  - abstração das classes conceptuais
    - Muitos detalhes omitidos,
    - Evidencia nos mais relevantes
  - contexto do caso de uso

Leonor Melo

08 Modelo do Domínio

19

19

## Modelo do Domínio: "dicionário visual"

- "dicionário visual"
  - vocabulário do domínio de interesse,
  - conceitos relevantes,
  - relações entre os conceitos
  - informação que lhes está associada
- ajuda a
  - comunicar
  - clarificar dúvidas de interpretação
  - visualiza as relações entre os conceitos

Leonor Melo

08 Modelo do Domínio

20

20

## Modelo do Domínio: objetos conceptuais e não objetos de software - 1

- Modelo do domínio
  - representação esquemática
    - coisas e conceitos
    - relevantes da situação real
- Não representa
  - objetos de software com responsabilidades  
(como classes de C++ ou Java)
- Perceber o problema
  - Antes de desenhar a solução

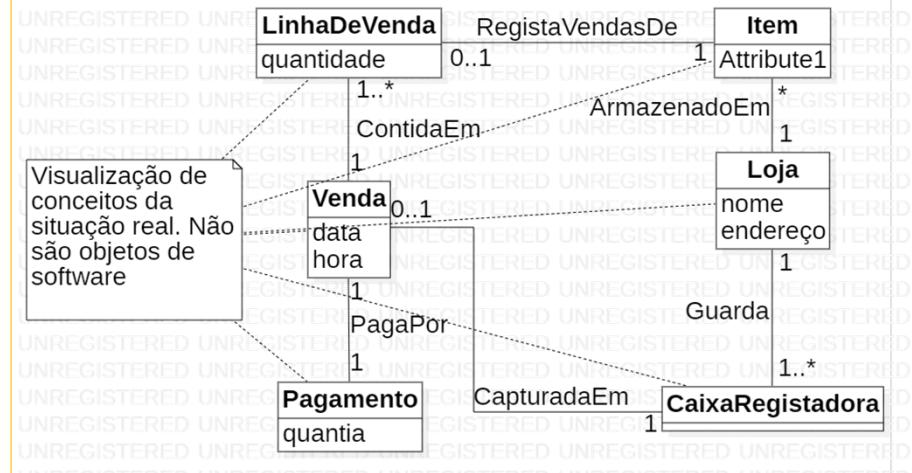
Leonor Melo

08 Modelo do Domínio

21

21

## Modelo do Domínio: objetos conceptuais e não objetos de software - 2

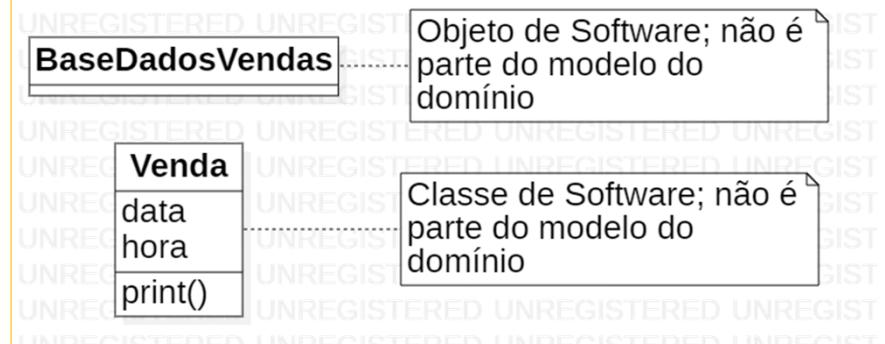


Leonor Melo

o8 Modelo do Domínio

22

## Modelo do Domínio: a evitar



Leonor Melo

o8 Modelo do Domínio

23

## Modelo do Domínio: mais que um significado

- Modelo do Domínio pode ser uma perspetiva conceptual do objetos numa situação real (definição usada na metodologia de desenvolvimento de software Unified Process (UP) )
- modelo do domínio pode ser também: "os objetos de software da camada do domínio" (i.e os objetos usados na camada "de negócio")

Leonor Melo

08 Modelo do Domínio

24

24

## Modelo do Domínio: classes conceptuais

- Uma classe conceptual é uma ideia, coisa ou objetos
- Formalmente, possui:
  - Símbolo - palavra(s) que representam a classe
  - Intenção - aquilo que representa e a caracteriza
  - Extensão - os exemplos a que se pode aplicar
- Exemplo:
  - Símbolo - palavra "Venda"
  - Intenção - Uma venda representa o evento de uma compra. Tem uma data e uma hora
  - Extensão - pode-se aplicar a qualquer venda

Leonor Melo

08 Modelo do Domínio

25

25

## Modelo do Domínio: porquê?

- Modelo do Domínio é criado para compreender o vocabulário e conceitos chave
  - especialmente importante quando se desenvolve um sistema para uma área de negócio que não se conhece bem
- Muitas vezes alguns conceitos acabam por ser usados no software
  - diminuindo a distância entre a nossa representação mental e a representação em software

Leonor Melo

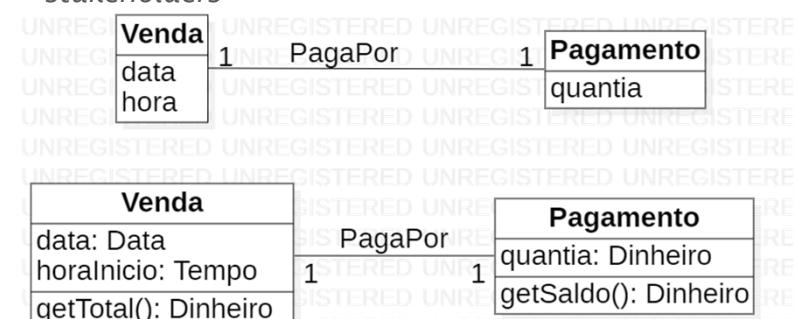
08 Modelo do Domínio

26

26

## Modelo do Domínio: porquê?

- Modelo do Domínio (do UP):
  - visão dos conceitos relevantes do domínio segundo os *stakeholders*
- Modelo de Design (do UP):
  - criar as classes de software inspirado no domínio real



Leonor Melo

08 Modelo do Domínio

27

27

## Modelo do Domínio: como criar?

- Tendo em conta os requisitos que se querem resolver nessa altura:
  - Encontrar as classes conceptuais
  - Desenhá-las como classes UML simples
  - Adicionar associações e atributos

Leonor Melo

08 Modelo do Domínio

28

28

## Modelo do Domínio: encontrar classes conceptuais

- Encontrar classes conceptuais:
  - Reutilizar e modificar modelos existentes
  - Usar uma lista de categorias
  - Identificar substantivos

Leonor Melo

08 Modelo do Domínio

29

29

## Modelo do Domínio: Categorias conceptuais comuns- 1

- Transações comerciais –
  - começar por estas são vitais (envolvem dinheiro).
  - Ex: Venda, Pagamento
- Partes de transações –
  - as transações muitas vezes incluem várias partes.
  - Ex: LinhaDeVenda
- Produtos ou serviços
  - relacionados a transação ou parte da transação.
  - Ex: Item
- Onde fica a transação registada?
  - Ex. RegistoContabilistico, fatura, recibo

Leonor Melo

08 Modelo do Domínio

30

30

## Modelo do Domínio: Categorias conceptuais comuns- 2

- Papéis de pessoas ou organizações
  - associadas com a transação; atores do caso de uso.
  - Ex. Operador, Cliente, Loja
- Local onde ocorre a transação/serviço
  - Ex. Loja, Caixa registadora
- Eventos relevantes
  - muitas vezes associados com uma data ou local que serão necessários.
  - Ex. Venda, Pagamento
- Objetos físicos
  - Ex. Item, CaixaRegistadora

Leonor Melo

08 Modelo do Domínio

31

31

## Modelo do Domínio: Categorias conceptuais comuns- 3

- Descrições de coisas.
  - Ex. DescriçãoProduto
- Catálogos.
  - Ex. CatalogoProdutos
- Contentores (de informação ou físicos).
  - Ex. Loja, Armazém, Fatura
- Coisas dentro de contentores.
  - Ex. Item
- Outros sistemas externos.
  - Ex. SistemaDeAutorizaçãoCredito

Leonor Melo

08 Modelo do Domínio

32

32

## Modelo do Domínio: Categorias conceptuais comuns- 4

- Registos financeiros, de trabalho, contratos, assuntos legais.
  - Ex. Fatura, RegistoContabilidade
- Instrumentos financeiros.
  - Ex. Dinheiro, Cheques, LinhaDeCredito
- Horários, manuais, documentos
  - que são consultados regularmente para executar a tarefa.
  - Ex. ListaDePromoçõesDiária

Leonor Melo

08 Modelo do Domínio

33

33

|  |  |
|--|--|
| <p>Modelo do Domínio:<br/>exemplo:<br/>Identificar os substantivos - 1</p> | <p><b>Caso de Uso</b></p> <p>Processa venda</p> <p>fluxo de eventos</p> <p>1. <b>Cliente</b> chega à <b>caixa</b> com os <b>produtos ou serviços</b> que quer comprar<br/>     2. <b>Operador</b> inicia uma nova <b>venda</b><br/>     3. Operador introduz <b>identificador</b> do item<br/>     4. Sistema regista cada <b>linha de venda</b> do <b>item</b> e apresenta a <b>descrição do item</b> o <b>preço</b> e o <b>total corrente</b>. O preço é calculado a partir de uma série de regras.<br/>     Operador repete os passos 2-3 até que assinala que terminou o registo</p> |
|--|--|

Leonor Melo

08 Modelo do Domínio

34

|  |  |
|--|--|
| <p>Modelo do Domínio:<br/>exemplo:<br/>Identificar os substantivos - 2</p> | <p><b>Caso de Uso</b></p> <p>Processa venda</p> <p>fluxo de eventos</p> <p>5. Sistema apresenta o total com o <b>imposto</b> calculado<br/>     6. Operador comunica total ao cliente e pede o <b>pagamento</b><br/>     7. Cliente paga e o sistema lida com o pagamento<br/>     8. O sistema regista a <b>venda</b> e envia a informação sobre a venda e pagamento para a <b>contabilidade externa</b> (para contabilidade e cálculo de comissões) e para o <b>sistema de inventário</b> (para atualização do inventário)</p> |
|--|--|

Leonor Melo

08 Modelo do Domínio

35

## Modelo do Domínio: exemplo:

| Caso de Uso       | Processa venda  |
|-------------------|---|
| fluxo de eventos  | 9. Sistema apresenta a <b>fatura</b><br>10. Cliente sai com produtos (se houverem) e fatura   |
| fluxo alternativo | 7a. Pagamento com dinheiro<br>1. Operador introduz <b>quantia entregue</b><br>2. Sistema apresenta o <b>troco</b> e abre a <b>gaveta do dinheiro</b><br>3. Operador deposita dinheiro entregue e entrega troco ao cliente<br>4. Sistema regista pagamento feito |

Leonor Melo

08 Modelo do Domínio

36

36

## Regras básicas

- Termos do domínio –
  - os termos usados por quem vai usar o sistema
- Excluir características irrelevantes
  - ou fora do âmbito do caso de uso a tratar nessa altura
- Não adicionar o que não existe!

Leonor Melo

08 Modelo do Domínio

37

37

## Atributo ou classe?

- Se não pensamos em determinada entidade conceptual como sendo um número ou texto,
  - então provavelmente não é um atributo,
  - é uma classe conceptual



Leonor Melo

08 Modelo do Domínio

38

# Modelação e Design

## 09: Modelo do Domínio

Leonor Melo

leonor@isec.pt

1

### Sumário

- Associações
- Classes descritoras
- Tipos de dados

2

## Modelo do domínio: associações

- Associação é uma
  - relação entre instâncias de classes,
  - relação interessante e relevante

Leonor Melo

09 Modelo do Domínio

3

3

## Modelo do domínio: associações

- Informação sobre uma dada relação entre instâncias deve ser conhecida durante algum tempo?
  - provavelmente associação relevante
  - Ex. precisamos saber quais são as LinhasDeVenda de uma Venda para
    - reconstruir a venda,
    - imprimir a fatura ou
    - calcular o total

Leonor Melo

09 Modelo do Domínio

4

4

2

## Modelo do domínio: associações

- **Modelo do Domínio**
  - perspetiva conceptual da realidade,
  - a "necessidade de lembrar a relação" deve ocorrer no caso real
- **Adicionar apenas as associações relevantes:**
  - Muitas associações tornam o modelo do domínio confuso

Leonor Melo

09 Modelo do Domínio

5

## Modelo do domínio: associações

- **Associação no modelo do domínio**
  - relação relevante do ponto de vista conceptual
  - Não indica:
    - fluxo de dados,
    - relação de chave primária numa base de dados
    - o modelo do domínio não é um modelo de dados

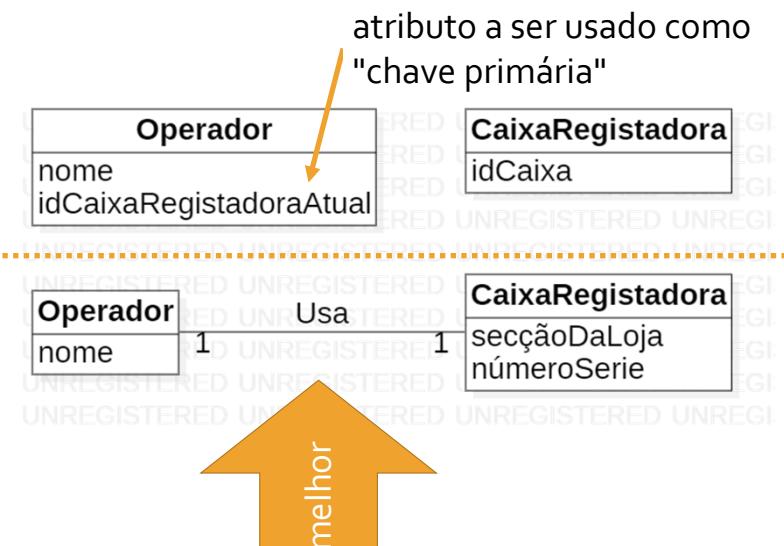
Leonor Melo

09 Modelo do Domínio

6

6

Modelo do domínio:  
atributos  $\neq$   
chave primárias



Leonor Melo

09 Modelo do Domínio

7

Modelo do domínio:  
associações

- Associações
- esquematizar as relações
  - na situação real
  - não na solução de software que iremos implementar

Leonor Melo

09 Modelo do Domínio

8

## Modelo do domínio: associações representação UML



- Representa-se como uma linha com o nome que começa em maiúscula
- Nos extremos da associação pode estar indicada a multiplicidade
  - a relação numérica entre as instâncias da classe
- A relação é bidirecional

Leonor Melo

09 Modelo do Domínio

9

## Modelo do domínio: associações representação UML: nomes

- Normalmente o nome das relações lê-se da esquerda para a direita e de cima para baixo
- O nome deve facilitar a leitura do diagrama:
  - *Venda SaldadaPor PagamentoDinheiro*
  - *Jogador EstaNa PosicaoTabuleiro*
- maus exemplos
  - *Venda Usa PagamentoDinheiro*
  - *Jogador Tem PosicaoTabuleiro*

Leonor Melo

09 Modelo do Domínio

10

Modelo do domínio:  
associações  
representação  
UML:  
multiplicidade



- Um item está armazenado numa loja, uma loja armazena zero ou mais itens

| símbolo | significado           |
|---------|-----------------------|
| *       | zero ou mais          |
| 1..*    | 1 ou mais             |
| 1..40   | 1 a 40                |
| 5       | exatamente 5          |
| 3, 5, 8 | exatamente 3, 5, ou 8 |

Leonor Melo

09 Modelo do Domínio

11

11

Modelo do domínio:  
associações  
representação  
UML:  
multiplicidade

- Como se interpreta a multiplicidade da associação do diagrama de cima?



- E do diagrama de baixo?

Leonor Melo

09 Modelo do Domínio

12

12

## Modelo do domínio: lista de associações comuns - 1

| categoria   | exemplo                   |
|---|---------------------------|
| A é uma transação relacionada com outra transação B             | Venda - PagamentoDinheiro |
| A é uma parte física ou lógica de B                             | LinhaDeVenda - Venda      |
| A é um produto ou serviço de uma transação ou "linha-item" de B | Item - LinhaDeVenda       |
| A é um papel relacionado com uma transação B                    | Cliente - Venda           |

Leonor Melo

09 Modelo do Domínio

13

13

## Modelo do domínio: lista de associações comuns - 2

| categoria                                     | exemplo                 |
|---|-------------------------|
| A está contido de forma física ou lógica em B | Registro - Loja         |
| A é uma descrição de B                        | DescriçãoProduto - Item |
| A é registado / gravado em B                  | Venda - Registro        |
| A trabalha em B                               | Operador - Loja         |
| A usa ou gera B                               | Operador - Registro     |

Leonor Melo

09 Modelo do Domínio

14

14

## classes descritoras (*Item- Descriptor pattern*)

- Uma classe descritora contém informação para descrever as instâncias de outra classe
- Deve ser acrescentada se:
  - A descrição de um item ou serviço é necessária mesmo que em determinado momento não exista nenhum exemplar desse item ou serviço
  - Apagar as instâncias que eles descrevem resulta em perda de informação que deveria ser mantida
  - Reduz informação redundante ou duplicada

Leonor Melo

09 Modelo do Domínio

15

## classes descritoras (*Item- Descriptor pattern*)

- Classe descritora
  - Informação comum a várias outras
  - Reduz informação redundante ou duplicada
- Acrescentar se:
  - Descrição de um item ou serviço
    - necessária ainda que não exista nenhum exemplar desse item ou serviço
  - Apagar a descrição
    - perda de informação que se devia manter

Leonor Melo

09 Modelo do Domínio

16

classes  
descriptoras:  
exemplo

| Item        |
|-------------|
| descrição   |
| preço       |
| numeroSerie |
| idItem      |

- Item:
  - numeroSerie é exclusivo de uma instância
  - outros atributos são comuns a várias instâncias
- Se se deixar de comercializar o Item temporariamente a restante informação perde-se
- Solução?

Leonor Melo

09 Modelo do Domínio

17

17

classes  
descriptoras:  
exemplo

| DescriçãoItem                | descreve | Item          |
|------------------------------|----------|---------------|
| descrição<br>preço<br>idItem | 1        | * numeroSerie |

- Descrição do produto e o produto em si são distintos
  - Por ex. descrição do item pode estar num catálogo
- A mesma descrição aplica-se a muitos itens,
  - determinado item tem só uma descrição

Leonor Melo

09 Modelo do Domínio

18

18

## Modelo do domínio: atributos

- Atributo
  - característica do objeto
  - mensurável (ou classificável) com um valor
- Identificar apenas os atributos que
  - Ajudam a clarificar a situação a modelar

Leonor Melo

09 Modelo do Domínio

19

19

## Modelo do domínio: atributos

- Atributos
  - informação que tem de ser conhecida durante algum tempo
  - procurar nos requisitos
    - por ex. casos de uso

Leonor Melo

09 Modelo do Domínio

20

20

10

## Modelo do domínio: atributos exemplo

- Exemplo POS:
  - uma fatura normalmente inclui
    - data e hora,
    - nome da loja,
    - endereço da loja,
    - identificação do operador, ...
- Que atributos usar?

Leonor Melo

09 Modelo do Domínio

21

21

## Modelo do domínio: atributos exemplo

- Exemplo POS:
  - Venda
    - *data e hora*
  - Loja
    - *nome*
    - *endereço*
  - Operador
    - *identificação*

Leonor Melo

09 Modelo do Domínio

22

22

## Modelo do domínio: atributos: notação UML



- Atributos aparecem no segundo compartimento do diagrama de classes

Leonor Melo

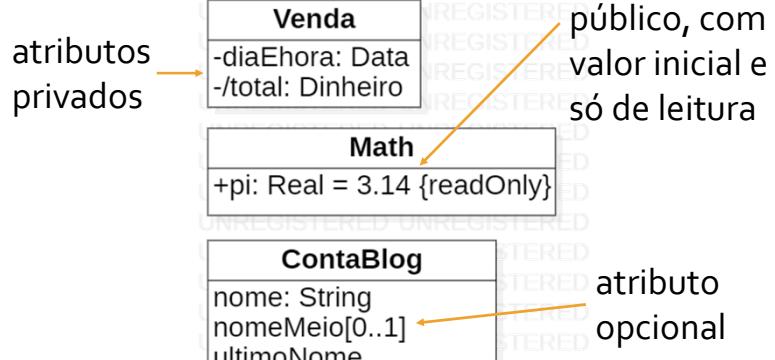
09 Modelo do Domínio

23

23

## Modelo do domínio: atributos: sintaxe UML de um atributo

visibilidade nome : tipo multiplicidade = default {propriedade}



Leonor Melo

09 Modelo do Domínio

24

24

## Modelo do domínio: atributos derivados

- Atributos derivados
  - valor pode ser calculado ou derivado
    - de informação guardada noutra classe
  - Ex: *total* da classe *Venda*
    - pode ser calculado a partir da informação em *LinhaDeVenda*
- notação UML:
  - / nomeAtributo

Leonor Melo

09 Modelo do Domínio

25

25

## Modelo do domínio: atributos derivados exemplo



Leonor Melo

09 Modelo do Domínio

26

26

Modelo do domínio:  
atributos de tipos de dados  
- 1

- Tipo dos atributos no Modelo do domínio
  - Sobretudo tipos de dados "primitivos"
    - números,
    - valores booleanos,
    - carácter,
    - strings,..

Leonor Melo

09 Modelo do Domínio

27

27

Modelo do domínio:  
atributos de tipos de dados  
- 2

- Tipo dos atributos no Modelo do domínio (continuação)
  - Podem ser "tipos de dados" comuns
    - Endereço,
    - Número de Telefone,
    - Código Postal,
    - URL,
    - Cor,
    - Figura Geométrica,...

Leonor Melo

09 Modelo do Domínio

28

28

Modelo do domínio:  
atributos de tipos de dados  
- 3

- Tipo dos atributos no Modelo do domínio (continuação)
  - Não devem ser
    - conceitos do domínio complexos
      - Venda,
      - Aeroporto,...

Leonor Melo

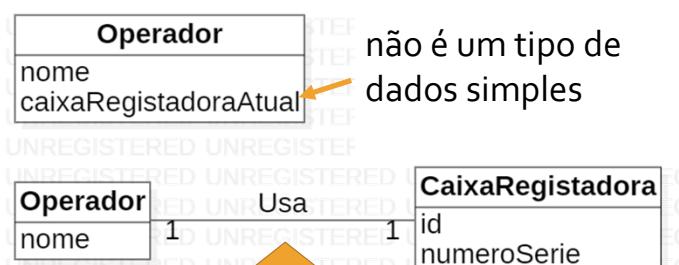
09 Modelo do Domínio

29

29

Modelo do domínio:  
atributos de tipos de dados  
- 4

- As relações entre classes conceptuais
  - representadas usando associações
    - não como atributos



melhor

Leonor Melo

09 Modelo do Domínio

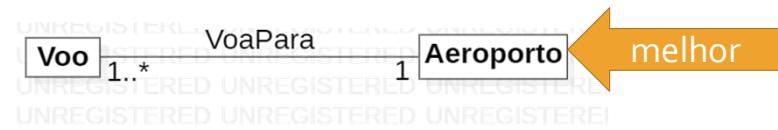
30

30

## Modelo do domínio: atributos de tipos de dados - 5

- Conceitos complexos

- classes conceptuais



Leonor Melo

09 Modelo do Domínio

31

31

## Modelo do domínio: tipos de dados - 1

- “Tipos de dados” comuns

- conjuntos de valores para os quais
  - uma identidade única não é relevante
    - no nosso modelo
- teste de igualdade
  - baseiam-se no valor
  - não na identidade

Leonor Melo

09 Modelo do Domínio

32

32

## Modelo do domínio: tipos de dados - 2

- Exemplo:
- Não faz sentido distinguir:
  - instancias separadas do valor inteiro 5
  - instancias separadas da string "Bom dia"
  - instancias separadas da data "13 de abril 2016"
- Faz sentido distinguir
  - instancias separadas de *Pessoa* mesmo se por acaso tem ambas o nome "João Silva"

Leonor Melo

09 Modelo do Domínio

33

## Modelo do domínio: tipos de dados - 2

- Do ponto de vista do software:
  - comparamos dois inteiros ou duas datas
    - por valor
  - comparamos duas Pessoas
    - usando o endereço de memória:
      - é possível duas pessoas com
        - os mesmos atributos
      - mas identidades diferentes

Leonor Melo

09 Modelo do Domínio

34

## Modelo do domínio: tipos de dados - 3

- Tipos de dados
  - (usualmente) imutáveis ao longo do tempo:
    - a instância do inteiro 5
      - não muda
    - a instância da data "13 de abril 2016"
      - provavelmente não mudará
    - uma Pessoa
      - pode mudar de nome

Leonor Melo

09 Modelo do Domínio

35

35

## Modelo do domínio: Número/string ou classe conceptual?

- Classe conceptual
  - composta por várias secções
  - secções usadas individualmente
    - nome,
    - número de telefone
      - Indicativo de país,
      - Indicativo região / operadora,
      - número
    - número BI
      - 7/8 algarismos
      - 1º dígito de controlo
      - 2 letras (número de emissões)
      - 2º dígito de controlo

Leonor Melo

09 Modelo do Domínio

36

36

## Modelo do domínio: Número / string ou classe conceptual?

- Classe conceptual (continuação):
  - Tem outros atributos
    - preço promocional,
    - valor,
    - data de início,
    - data de fim
  - Quantidade com uma unidade associada
    - pagamento
      - valor numérico,
      - moeda
    - medida
      - valor,
      - unidade

Leonor Melo

09 Modelo do Domínio

37

37

## Modelo do domínio: representar classes tipos de dados? - notação UML - 1

- Classes tipo de dados
  - Representadas explicitamente no modelo do domínio
    - relevância de conhecer os atributos dessa classe para
      - representar,
      - compreender, e
      - comunicar
    - sobre o domínio
  - Não representadas explicitamente
    - tipo de determinado atributo

Leonor Melo

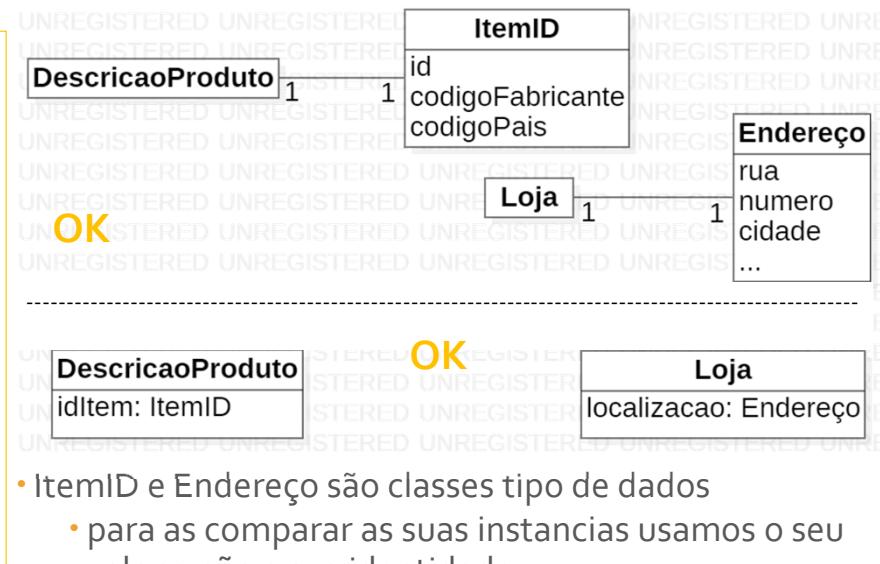
09 Modelo do Domínio

38

38

19

Modelo do domínio:  
representar classes tipos de dados? -  
notação UML -  
2



Leonor Melo

09 Modelo do Domínio

39

Modelo do domínio:  
conclusão

- O modelo do domínio é uma ferramenta para:
  - compreender o domínio (conceitos, terminologia e relações)
  - comunicar entre um grupo de pessoas
- Um modelo do domínio útil captura as abstrações essenciais necessárias para compreender o domínio no contexto do requisito a ser tratado nesse momento

Leonor Melo

09 Modelo do Domínio

40

# Modelação e Design

## 10: Diagrama de Classes: Classes de Design

Leonor Melo

leonor@isec.pt

1

### Classes de design

- Noção de classe e de objeto
- Diagrama de classes aplicado a classes de design
- Principais conceitos de Orientação a Objetos
- Nomenclatura de classes e atributos

Leonor Melo

10 Diagramas de classe

2

2

1

## Sistema Orientado a Objetos

- Num sistema Orientado a Objetos
  - os objetos são os conceitos centrais
  - os objetos podem saber como:
    - guardar informação
    - receber informação
    - criar nova informação
    - fornecer informação

Leonor Melo

10 Diagramas de classe

3

## Sistema Orientado a Objetos

- Dentro do mesmo sistema é frequente termos vários objetos do mesmo "tipo"
  - com características e comportamentos idênticos entre si
- Os "tipos" correspondem a classes
- Os objetos são instâncias das classes
- Normalmente um sistema tem mais de um tipo de objeto:
  - tem várias classes

Leonor Melo

10 Diagramas de classe

4

## O que é uma classe - 1

- Exemplo de objeto:
  - meu carro favorito!



- Este carro tem uma identidade:
  - é aquele que tive entre 2003 (?) e 2018

Leonor Melo

10 Diagramas de classe

5

## O que é uma classe - 2

- Mas a Opel não fabricou apenas esse Agila
  - fabricou muitos carros deste tipo
  - fabricou muitos carros desta *classe*
- Classe =
  - tipo de
  - "planta de construção" de

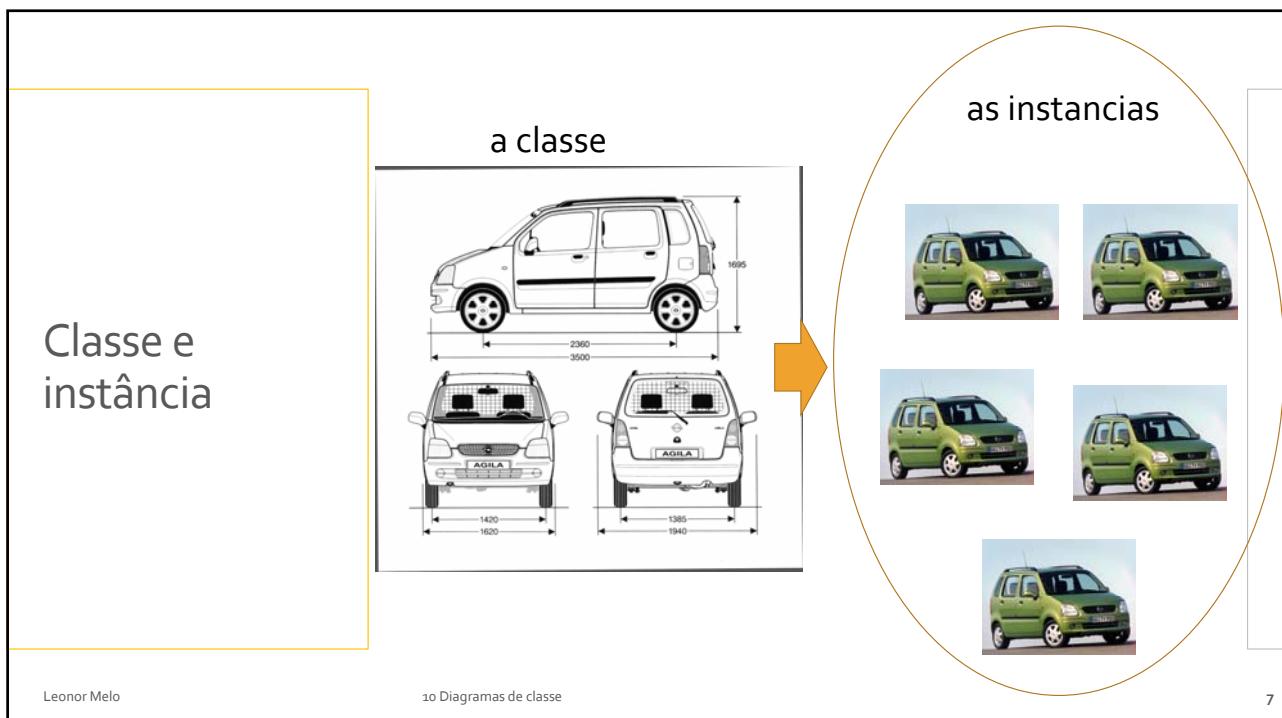
Leonor Melo

10 Diagramas de classe

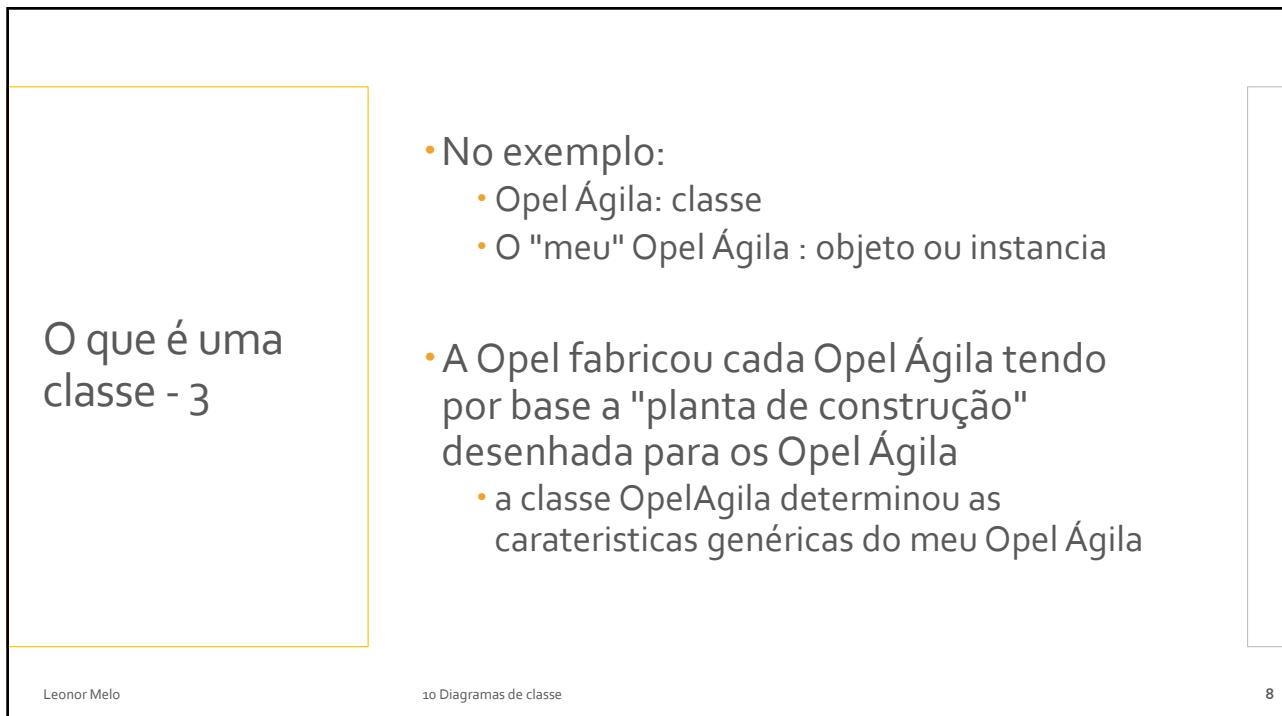
6

6

3



7



8

## O que é uma classe - 4

- Uma classe contém:
  - estado:
    - informação que os objetos da classe irão conter
  - comportamento:
    - ações que os objetos serão capazes de desempenhar
- No exemplo:
  - estado:
    - cor exterior, estofos, ...
  - comportamento
    - aumentar a velocidade, travar,...

Leonor Melo

10 Diagramas de classe

9

## O que é uma classe - 5

- Estado:
  - atributos da classe e/ou das instâncias da classe
- Comportamento:
  - Operações associadas à classe e/ou às instâncias da classe
- **Atributos e operações da classe**
  - descrevem o que caracteriza os elementos dessa classe

Leonor Melo

10 Diagramas de classe

10

## Diagramas de classe - 1

- Aplicado em várias fases do processo de desenvolvimento de software
  - fases iniciais: modelo conceptual e vocabulário do sistema
  - **fase de design:** modelo refinado e transformado de forma a traduzir as classes do sistema de software

Leonor Melo

10 Diagramas de classe

11

11

## Diagramas de classe - 2

- Classes
  - conceito chave da orientação a objetos
- Diagrama UML mais conhecido
  - diagrama de classes
- Estrutura de um sistema
  - coleção de "peças"
  - peça = objeto
- Classes descrevem os diferentes tipos de objetos

Leonor Melo

10 Diagramas de classe

12

12

## Diagramas de classe - 3

- Diagramas de classes
  - diferentes tipos de objetos que existem no sistema
  - relações entre eles
- Casos de uso
  - comportamento que sistema deve ter
- Classes do modelo de domínio (problema)
  - entidades relevantes para o caso de uso que estamos a resolver
- Classes de design (solução)
  - tipo de objetos que devem existir no sistema para que este consiga cumprir o indicado pelo caso de uso

Leonor Melo

10 Diagramas de classe

13

13

## Principais Conceitos de Orientação a objetos

- Abstração
- Herança
- Polimorfismo
- Encapsulamento
- Envio de mensagens
- Relações entre objetos

Leonor Melo

10 Diagramas de classe

14

14

## Abstração - 1

- Abstração
  - Ignorar detalhes irrelevantes para um determinado contexto
    - Demasiadas informações
      - modelo confuso
      - Informações de menos
        - modelo pouco rigoroso
  - Focar na informação que o sistema necessita conhecer

Leonor Melo

10 Diagramas de classe

15

15

## Abstração - 2

- Os atributos e operações de uma classe são uma abstração
  - apenas registamos os relevantes para o problema/solução
- Numa oficina de reparações:
  - atributos:
    - riscos e marcas na pintura
- Na fábrica
  - atributos:
    - peso da carroçaria

Leonor Melo

10 Diagramas de classe

16

16

## Herança - 1

- As classes podem ser organizadas em hierarquias:
- Classe base
  - mais genérica
- Classe derivada
  - representa um caso especial da classe base
  - mais específica
  - "herda" atributos e comportamentos da classe base

Leonor Melo

10 Diagramas de classe

17

17

## Herança - 2

- Exemplo: Aplicação de aluguer de viaturas:
- Classe base
  - Viatura
- Classe derivada
  - Carro
  - Carrinha
  - Motociclo

Leonor Melo

10 Diagramas de classe

18

18

## Polimorfismo

- Capacidade de um objeto de um tipo derivado poder substituir um objeto do tipo base:
  - tipo específico do objeto necessário deixa de ser relevante
  - código mais fácil de manter a longo prazo: novas classes podem ser adicionadas mais tarde sem comprometer a integridade da arquitetura

Leonor Melo

10 Diagramas de classe

19

19

## Encapsulamento - 1

- Uma das características mais importantes da orientação a objetos
  - Cada classe expõe apenas aquilo de que as outras classes necessitam para comunicar com ela
    - a classe esconde todos ou maior parte dos seus atributos
    - algumas operações permitem aceder aos atributos mas de forma controlada
  - classe pode mudar a sua arquitetura interna sem afetar o resto do projeto
- Exemplo carro:
  - atributo: parte elétrica: escondido
  - operação: ligar faróis: acessível através de um botão

Leonor Melo

10 Diagramas de classe

20

20

## Encapsulamento - 2

- Permite esconder os detalhes internos de funcionamento da classe
  - Mesmo que a classe altere o modo como funciona internamente o sistema não sofre:
    - a interação com a classe continua igual
- Numa abordagem orientada a objetos
  - pequenas alterações no modo como a classe funciona internamente
    - não devem causar falhas no sistema

Leonor Melo

10 Diagramas de classe

21

21

## Envio de mensagens - 1

- Num sistema Orientado a Objetos
  - os objetos trabalham juntos enviando mensagens uns aos outros
  - mensagem = pedido para que o receptor da mensagem execute uma dada operação
  - Operações que uma classe exibe = mensagens que está disponível para receber

Leonor Melo

10 Diagramas de classe

22

22

11

## Envio de mensagens - 2

- exemplo:
  - Controlo remoto: envia uma mensagem para a TV a pedir-lhe para se desligar
  - TV: quando recebe uma mensagem para se desligar, desliga-se



imagem: vecteezy.com

Leonor Melo

10 Diagramas de classe

23

23

## Relações

- Os objetos que estão relacionados entre si conseguem comunicar uns com os outros
- As relações podem ser de diferentes tipos
  - apenas numa direção ou bidirecionais
  - tem uma multiplicidade associada
  - exprimir diferentes graus de dependência entre as classes

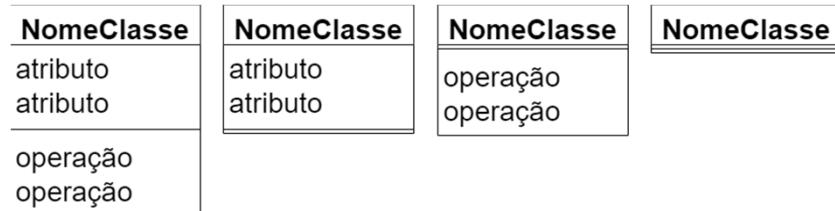
Leonor Melo

10 Diagramas de classe

24

24

## Classes em UML - 1



- Se as secções de atributos ou operações não forem mostradas não quer dizer que estejam vazias
  - apenas que o diagrama é mais fácil de ler com alguma informação escondida

Leonor Melo

10 Diagramas de classe

25

25

## Classes em UML - 2



- Nome da classe = tipo de dados das instâncias dessa classe

Leonor Melo

10 Diagramas de classe

26

26

## Classes, atributos e operações - 1

- Classes:
  - Nomes "classificadores"
    - pessoa, empregado, curso, ...
  - Nomes de valores raramente são classes
    - João, programação orientada a objetos, ...
- Valores de atributos
  - Adjetivos
  - Nomes
  - Se forem derivados devem ser denotados como tal
    - idade derivado de data de nascimento

Leonor Melo

10 Diagramas de classe

27

27

## Classes, atributos e operações - 2

- Operações
  - verbos
  - que operações pode um objeto de uma classe executar? (que mensagens pode receber)
  - que eventos, aos quais um objetos de uma classe terá de reagir, podem ocorrer?
    - que outros eventos podem ocorrer como resultado?
- Considerar extensibilidade do sistema

Leonor Melo

10 Diagramas de classe

28

28

## Classes, atributos e operações - 3

- Que atributos e operações deve ter a TV? e o telecomando?



imagem: vecteezy.com

Leonor Melo

10 Diagramas de classe

29

# Modelação e Design

## 11: Diagrama de Classes: conceitos básicos

Leonor Melo

leonor@isec.pt

1

### Conceitos básicos

- Visibilidade
- Sintaxe dos Atributos

2

## Visibilidade

- O encapsulamento é uma das vantagens mais unâimes da orientação a objetos:
  - Oferece robustez ao código
- O encapsulamento é conseguido controlando a visibilidade dos atributos e operações das classes

11 - Diagrama de classe

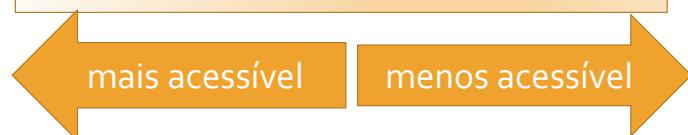
3

3

## Visibilidade

- Indica aquilo que a classe esconde
  - e aquilo que disponibiliza
- UML permite 4 níveis de visibilidade:
  - visibilidade UML pode ser diferente da visibilidade de algumas linguagens de programação!

| Public | Protected | Package | Private |
|--------|-----------|---------|---------|
| (+)    | (#)       | (~)     | (-)     |



11 - Diagrama de classe

4

4

## Visibilidade Public - 1

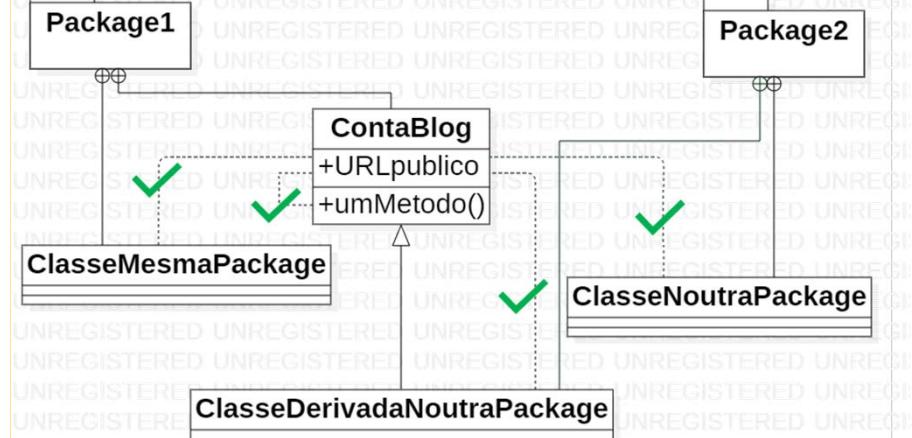
- Visibilidade public
  - a mais acessível
  - + antes do atributo ou operação
  - acessível para métodos de qualquer outra classe

11 - Diagrama de classe

5

5

## Visibilidade Public - 2



11 - Diagrama de classe

6

6

## Visibilidade Public - 3

- interface público de uma classe
  - coleção de atributos e operações públicos dessa classe
  - deve mudar o mínimo ao logo do tempo para não afetar o funcionamento das outras classes
- Normalmente evita-se usar atributos públicos
  - se um atributo for visível também é modificável
    - exceção: se atributo for uma constante usada por outras classes (ex. Pi)

11 - Diagrama de classe

7

7

## Visibilidade Protected - 1

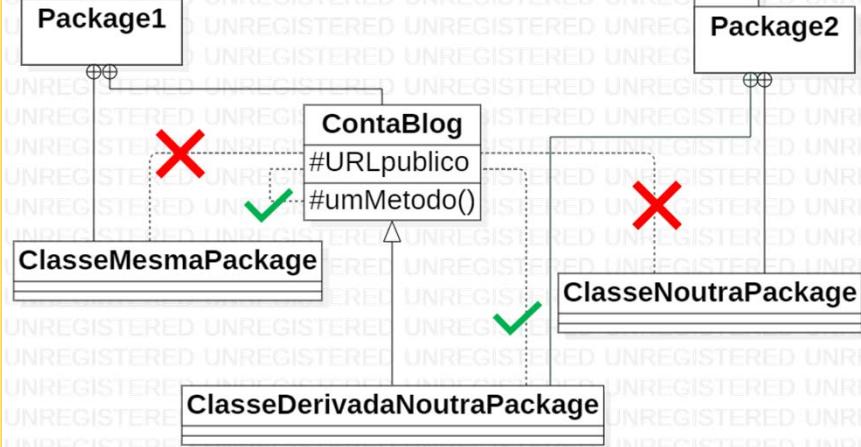
- Visibilidade protected
  - menos acessível que a public
  - # antes do atributo ou operação
  - acessível para métodos da própria classe e para métodos das classes descendentes dela
    - inacessível para os métodos das restantes classes, estejam elas na mesma package ou não

11 - Diagrama de classe

8

8

## Visibilidade Protected - 2



11 - Diagrama de classe

9

9

## Visibilidade Protected - 3

- Crucial para deixar que as classes especializadas aceder aos atributos e operações da classe mais genérica
  - mas sem disponibilizar o atributos e operações a todo o sistema
  - operações e atributos úteis para o funcionamento interno da classe (e suas derivadas) mas que mais ninguém deve usar

11 - Diagrama de classe

10

10

## Visibilidade Package - 1

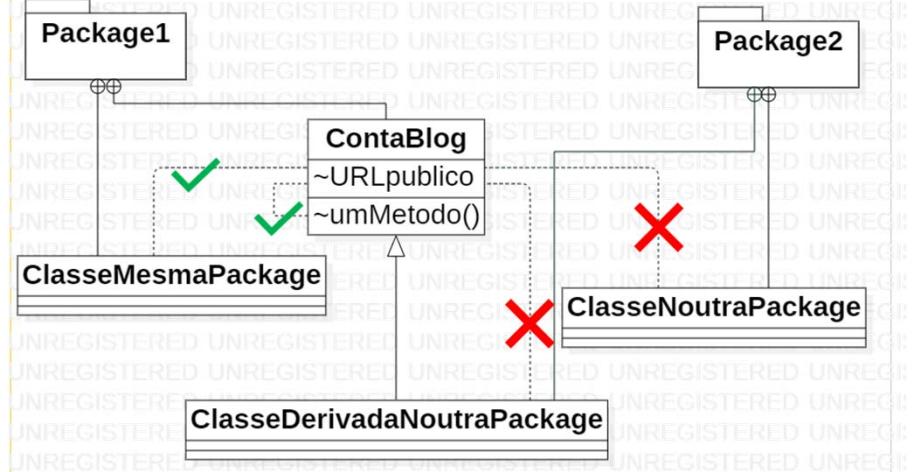
- Visibilidade package
  - mais acessível que a private
  - ~ antes do atributo ou operação
  - acessível para os métodos da própria classe e para os métodos das classes da mesma package
  - inacessível para os métodos das restantes classes, mesmo que sejam classes derivadas dessa

11 - Diagrama de classe

11

11

## Visibilidade Package - 2



11 - Diagrama de classe

12

12

## Visibilidade Package - 3

- Usada sobretudo quando temos uma coleção de método que queremos reutilizar dentro da mesma package
  - Ex. Numa package de "utility classes", queremos reutilizar os métodos dentro das classes dessa package, mas não queremos que fiquem expostos ao resto do sistema.
  - métodos para partilhar dentro da package: package
  - métodos para partilhar com o sistema: public

11 - Diagrama de classe

13

13

## Visibilidade Private - 1

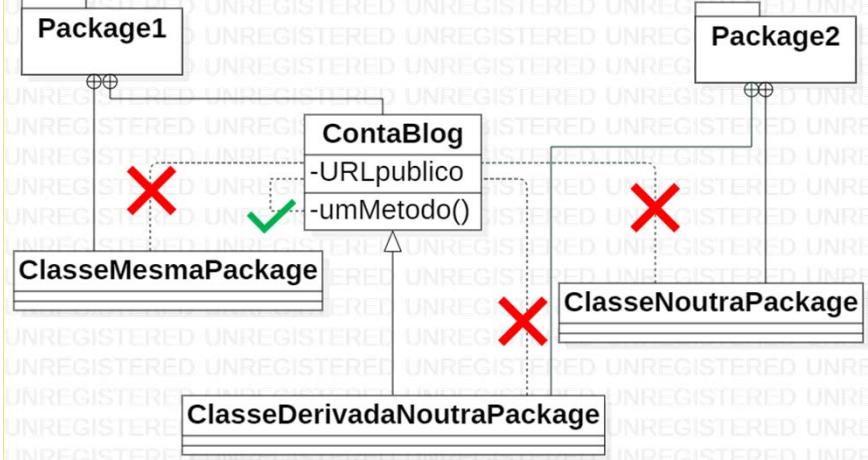
- Visibilidade private
  - a menos acessível
  - - antes do atributo ou operação
  - acessível para métodos da própria classe

11 - Diagrama de classe

14

14

## Visibilidade Private - 2



11 - Diagrama de classe

15

15

## Visibilidade Private - 3

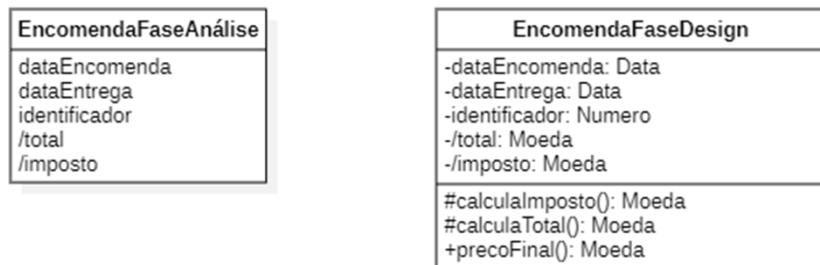
- Por regra os atributos são private
  - eventualmente protected se forem necessários nas classes descendentes
- Operações private são aquelas que dizem respeito ao funcionamento interno da classe
  - ou que queremos poder alterar mais tarde

11 - Diagrama de classe

16

16

Exemplo visibilidade:  
(exerto de diagrama de classes sobre o processamento de encomendas)



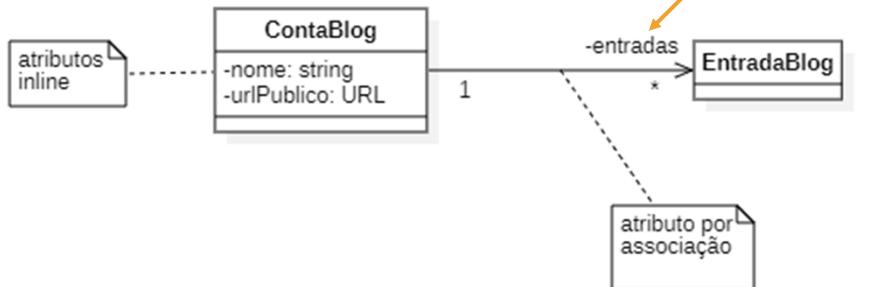
11 - Diagrama de classe

17

17

Atributos - 1

- Atributos (propriedades) da classe
  - estado do objetos
- Podem ser representados
  - dentro da classe (*inline*)
  - por associação com outra classe



11 - Diagrama de classe

18

18

## Atributos - 2

- Formato completo:
  - visibilidade nome : tipo multiplicidade = default {modificador-de-propriedade}
- No mínimo, a assinatura do atributo tem:
  - visibilidade
  - nome
  - tipo
- mas só nome é realmente obrigatório

11 - Diagrama de classe

19

19

## Atributos - Nome e tipo

- Nome:
  - conjunto de caracteres
  - único dentro da classe
  - deve descrever a informação que o atributo representa
- Tipo:
  - tipo de dados primitivo
    - inteiro, booleano, ...
  - outra classe
- Podem adequar-se às convenções da linguagem usada na implementação

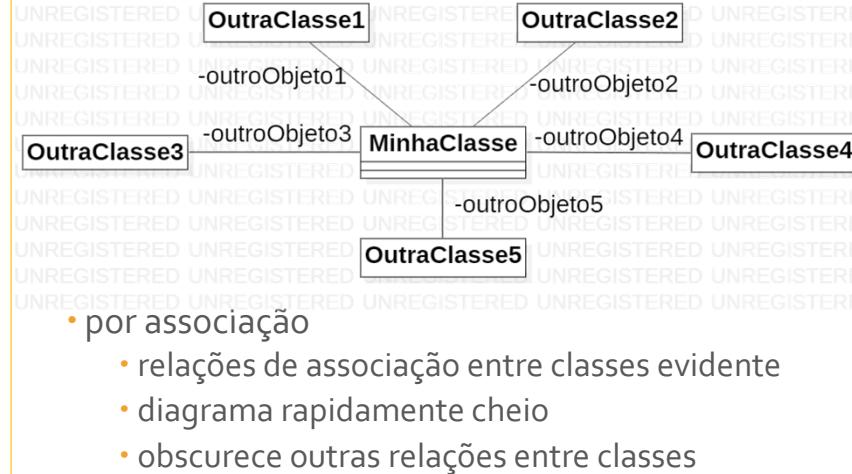
11 - Diagrama de classe

20

20

10

## Atributos inline vs Atributos por associação - 1

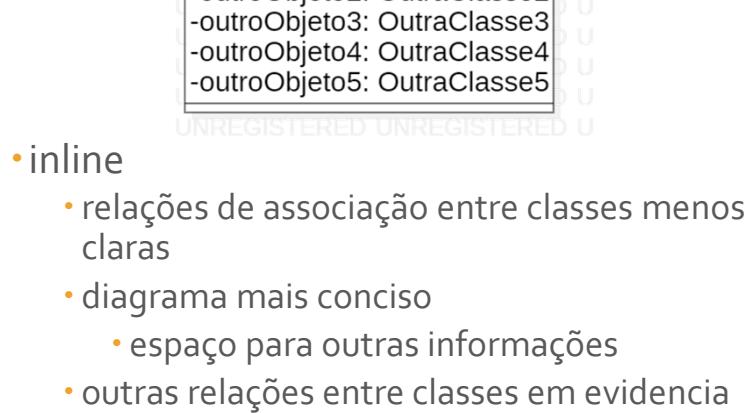


11 - Diagrama de classe

21

21

## Atributos inline vs Atributos por associação - 2



11 - Diagrama de classe

22

22

## Atributos - Multiplicidade - 1

- Um atributo pode representar mais do que um objeto
  - pode representar todo um conjunto de objetos desse tipo
- Multiplicidade
  - indica que um atributo é na realidade uma coleção
  - aplica-se a atributos
    - inline
    - por associação

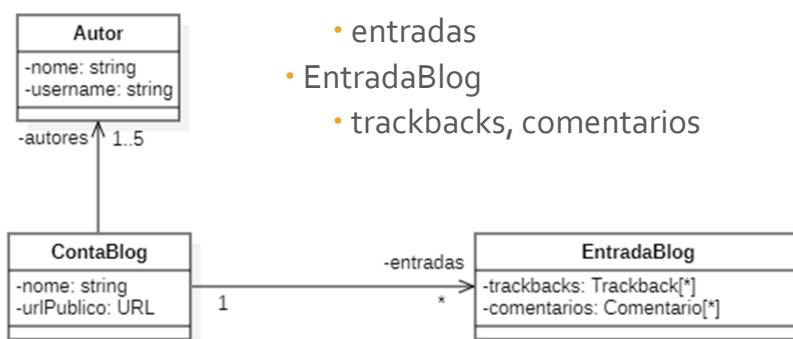
11 - Diagrama de classe

23

23

## Atributos - Multiplicidade - 2

- Quantos objetos podem ser guardados?
  - ContaBlog:
    - nome, urlPublico
    - autores
    - entradas
  - EntradaBlog
    - trackbacks, comentarios



11 - Diagrama de classe

24

24

## Modificadores de Propriedade quando multiplicidade > 1

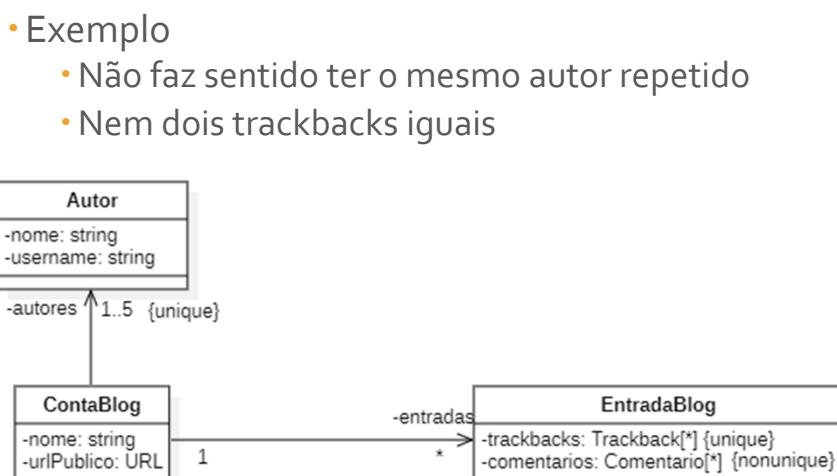
- Podemos usar "modificadores de propriedade" para especificar ainda com mais detalhe a multiplicidade:
  - modificador de propriedade *unique*
    - não existem elementos repetidos no atributo múltiplo
  - modificador de propriedade *nonunique*
    - podem existir elementos repetidos no atributo múltiplo
  - por omissão os atributos múltiplos são unique

11 - Diagrama de classe

25

25

## Modificadores de Propriedade quando multiplicidade > 1

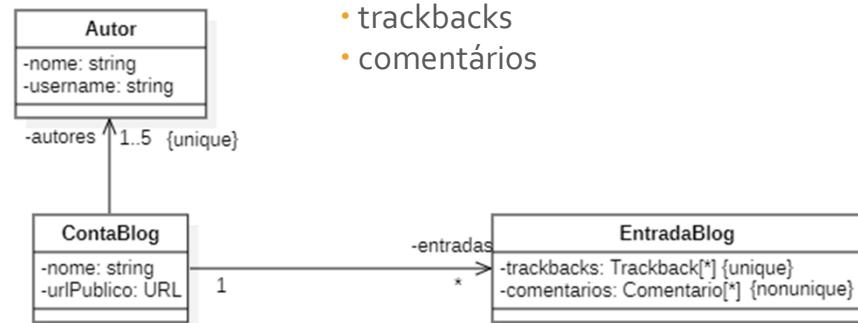


11 - Diagrama de classe

26

26

## Modificadores de Propriedade quando multiplicidade > 1



- Em quais dos seguintes atributos podem existir elementos repetidos?

- autores
- entradas
- trackbacks
- comentários

11 - Diagrama de classe

27

27

## Modificadores de Propriedade quando multiplicidade > 1

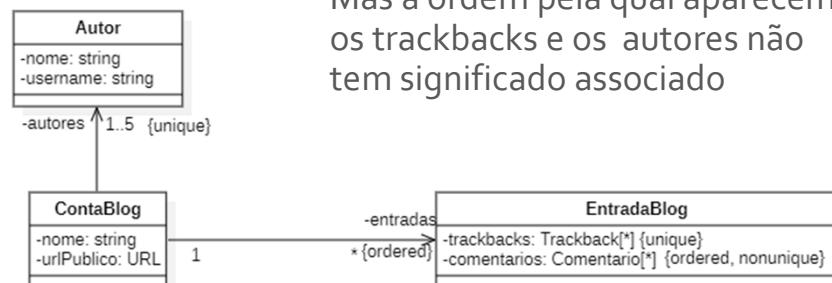
- Podemos indicar se os elementos de determinado atributo múltiplo estão ordenados
  - modificador de propriedade *unique*
    - os elementos do atributo estão ordenados de acordo com algum critério
- por omissão os atributos múltiplos não têm nenhuma ordem associada

11 - Diagrama de classe

28

28

## Modificadores de Propriedade quando multiplicidade > 1



- Faz sentido associar uma ordem às entradas do blog e aos comentários
  - por exemplo, a ordem pela qual foram adicionadas
- Mas a ordem pela qual aparecem os trackbacks e os autores não tem significado associado

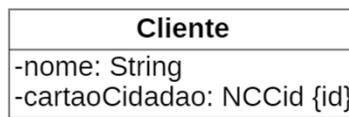
11 - Diagrama de classe

29

29

## Outros modificadores de propriedade comuns

- SistemaGestaoConteudos**
- +criadoPor: String = "Adelina Silva, SA." {readOnly}
- Modificador de propriedade *readOnly*
    - valor não pode mudar depois da atribuição inicial
  - Modificador de propriedade *id*
    - valor é/faz parte do identificador único desse objeto



11 - Diagrama de classe

30

30

# Modelação e Design

## 12: Diagrama de Classes: Design

Leonor Melo

leonor@isec.pt

1

Diagrama de classes

- Operações num diagrama de classes
- Classes com membros estático

12 Diagramas de classe

2

## Comportamento da classe: operações

- A secção das operações no diagrama de classes descreve
  - o que a classe consegue fazer
  - mas não descreve como o irá fazer
- Apresenta apenas a "assinatura":
  - visibilidade,
  - nome da operação,
  - tipos de dados que recebe,
  - tipos de dados que devolve

12 Diagramas de classe

3

## Comportamento da classe: operações

- Operação:
  - "contrato" que indica que a classe irá conter um comportamento que fará o que a assinatura da operação sugere
- Conjunto das operações
  - serviços que a classe disponibiliza devido à sua "natureza"
  - mensagens que podem ser enviadas a essa classe e/ou suas instâncias

12 Diagramas de classe

4

## Comportamento da classe: operações UML - 1

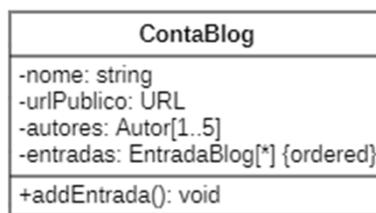
- Formato completo:
  - [visibilidade] nome ( [parâmetro [, parâmetro]] ) [: tipo] [multiplicidade] [{ propriedade [, propriedade]}]
- Também pode ser usando a sintaxe de uma linguagem de programação
  - +getJogador( nome : String ) : Jogador {exception IOException}
  - public Jogador getJogador(String nome ) throws IOException

12 Diagramas de classe

5

## Comportamento da classe: operações UML - 2

- Assinatura da operação deve ter pelo menos:
  - visibilidade
  - nome
  - par de parêntesis com os parâmetros lá dentro
  - tipo de dados devolvido

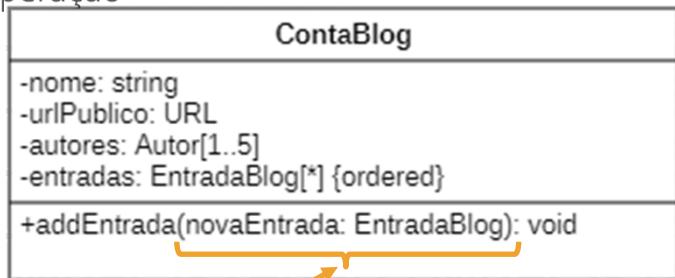


12 Diagramas de classe

6

## Comportamento da classe: operações: parâmetros -1

- Parâmetros
  - especificar a informação passada à operação



- **addEntrada** mais realista

12 Diagramas de classe

7

## Comportamento da classe: operações: parâmetros -2

- Para cada parâmetro é preciso indicar
  - pelo menos o nome
- Podemos passar mais que um parâmetro
  - separando por vírgulas



12 Diagramas de classe

8

## Comportamento da classe: operações: parâmetros -3

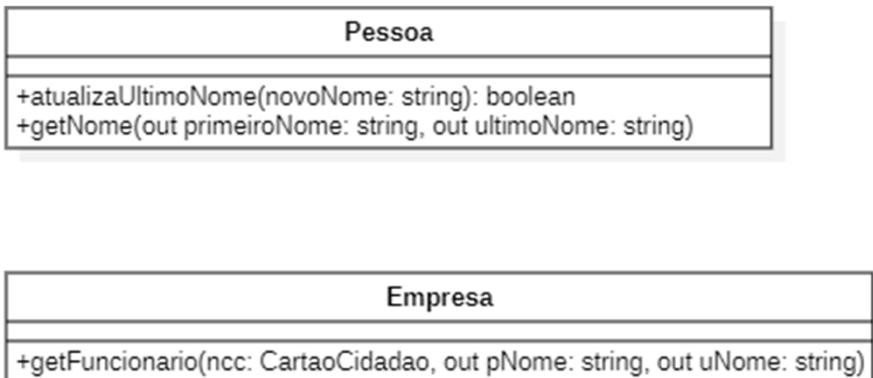
- Sintaxe completa dos parâmetros:
- [in / out / inout] nome [: tipo]  
[multiplicidade] [= default] [{propriedade  
, propriedade}])

12 Diagramas de classe

9

## Comportamento da classe: operações: parâmetros -4

- Exemplos:



12 Diagramas de classe

10

## Comportamento da classe: operações: tipo do retorno

- Tipo de retorno
  - tipo do objeto que será devolvido pela operação
  - *verdadeiro* se conseguiu adicionar, *falso* se não conseguiu



12 Diagramas de classe

11

## Comportamento da classe: construtores

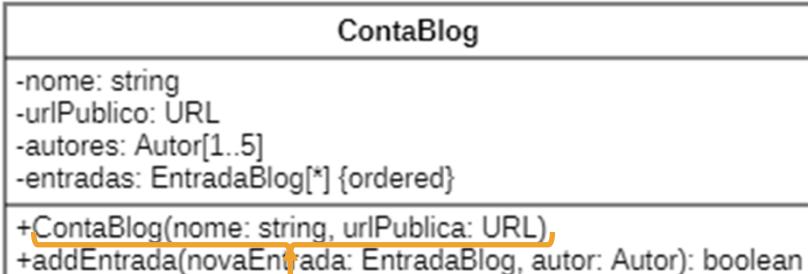
- Construtores
  - funções especiais
  - criam o objeto e inicializam corretamente seus os atributos
  - podem receber parâmetros
  - Devolvem o objeto construído
  - Denotam-se como operações associadas a objetos, mas na realidade são executadas pelo compilador
  - Só se representam em diagramas de classe de design de baixo nível

12 Diagramas de classe

12

## Comportamento da classe: construtores

- Construtores
  - nome operação = nome da classe
  - não precisam de indicar tipo de retorno



- devolve uma ContaBlog

12 Diagramas de classe

13

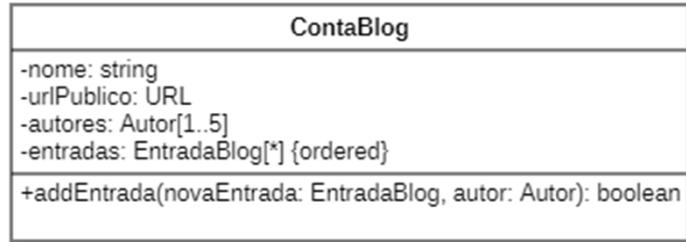
## Partes estáticas e não estáticas da classe

- Atributos e operações podem ser estáticos
- Estático
  - associado com a classe
- não estático
  - associado com a instância

12 Diagramas de classe

14

Partes não -  
estáticas da  
classe - 1

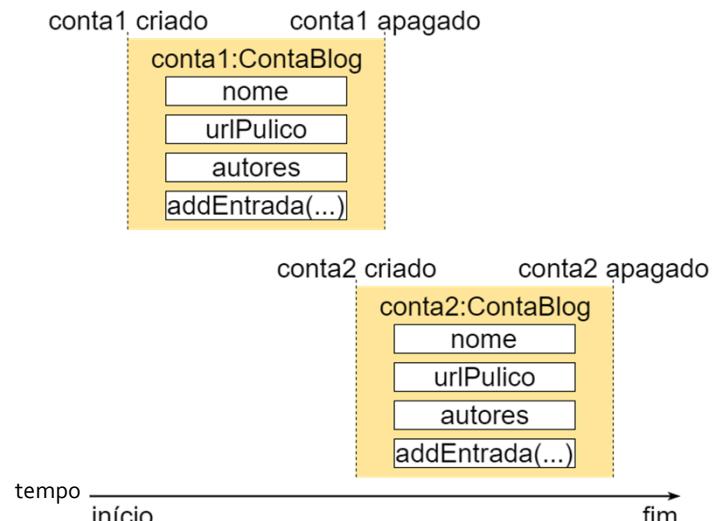


- Cada objeto da classe ContaBlog tem
  - 4 atributos
  - 1 operação
- Partes não estáticas da classe têm a duração das instâncias da qual fazem parte

12 Diagramas de classe

15

Partes não -  
estáticas da  
classe - 2



12 Diagramas de classe

16

## Partes estáticas da classe

- As classes "existem" assim que o programa se inicia e duram até que o programa termine
  - não dependem da existência de nenhum objeto dessa classe
- Membros estático
  - só têm uma "cópia" (associada à classe)
  - estão disponíveis durante toda a execução

12 Diagramas de classe

17

## Partes estáticas da classe - 1

- Problema:
  - queremos criar um atributo privado, contaContas, que registe o número de objetos ContaBlog existentes em cada momento
- Como fazer?

12 Diagramas de classe

18

## Partes estáticas da classe - 2

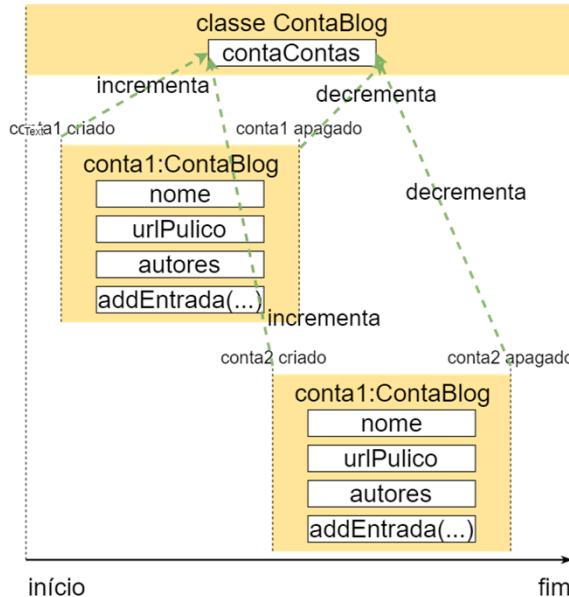
- contaContas
  - regista o número de objetos ContaBlog existentes em determinado momento
  - faz sentido que exista apenas um para toda a classe



12 Diagramas de classe

19

## Partes estáticas da classe – 3



12 Diagramas de classe

20

## padrão de design

- Design pattern
  - Solução para um problema que existe em determinado contexto
- Contexto:
  - situação à qual o problema se aplica
- Problema:
  - objetivo que se está a tentar alcançar (restrições encontradas nesse contexto)
- Solução:
  - design genérico que pode ser aplicado nessa situação e que resolve o problema

12 Diagramas de classe

21

## padrão de design

- Uma solução frequentemente usada para um problema comum
- Problema: odor no lava-louça
- Padrão: sifão em S
- Problema: cruzamento na autoestrada
- Padrão: trevo de quatro-folhas



12 Diagramas de classe

22

## padrão de design de software singleton -1

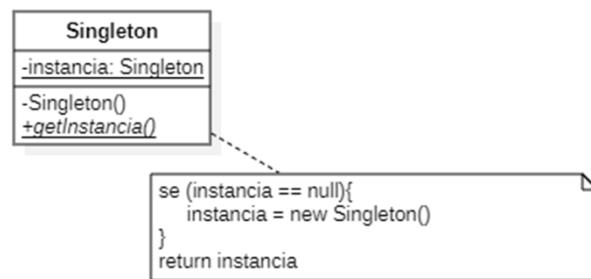
- Padrão de software singleton
  - A aplicação necessita de uma e só uma instância do objeto.
  - Essa instancia tem de ser globalmente acessivel
- Exemplo:
  - classe que regista as configurações do sistema
    - criar mais do que um objeto deste tipo durante a execução do programa pode originar problemas

12 Diagramas de classe

23

## padrão de design de software singleton -2

- Padrão de software singleton
  - A aplicação necessita de uma e só uma instância do objeto.
  - Essa instancia tem de ser globalmente acessivel



12 Diagramas de classe

24

padrão de  
design de  
software  
singleton -3

### Exemplo:

classe que regista as configurações do sistema

criar mais do que um objeto deste tipo durante a execução do programa pode originar problemas

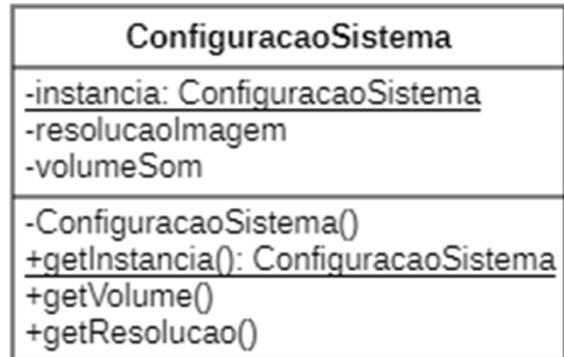
12 Diagramas de classe

25

padrão de  
design de  
software  
singleton -4

### Exemplo:

classe que regista as configurações do sistema



12 Diagramas de classe

26

# Modelação e Design

## 13: Diagrama de Classes: Design

Leonor Melo

leonor@isec.pt

1

Diagrama de classes

- Relações entre classes
  - dependencia
  - associação
    - associação n-ária
    - classes de associação

Leonor Melo

13 Diagramas de classe

2

2

## Relações entre classes - 1

- Classes trabalham em conjunto
  - estabelecem relações com outras classes
- As relações podem ter diferentes graus de força:
  - quanto mais uma classe depender de outra
    - mais forte é a relação
    - mais forte é o acoplamento

Leonor Melo

13 Diagramas de classe

3

3

## Relações entre classes - 2

- Fortemente ligadas = fortemente acopladas
  - alterações numa classe
    - vão afetar a outra classe
- Acoplamento forte é normalmente mau
- Quanto mais forte a relação mais cuidado se deve ter

Leonor Melo

13 Diagramas de classe

4

4

## Relações entre classes - 3

| acoplamento crescente |  |                         |  |                       |
|-----------------------|--|-------------------------|--|-----------------------|
| Dependência           | Associação   | Agregação partilhada    | Agregação composta   | Herança               |
| interação breve       | interação mais longa que dependência, pode, ou não, ser todo-parté | todo-parté mas partilha | o todo contém a parte; a parte não pode existir sem o todo | um é um tipo do outro |

Leonor Melo

13 Diagramas de classe

5

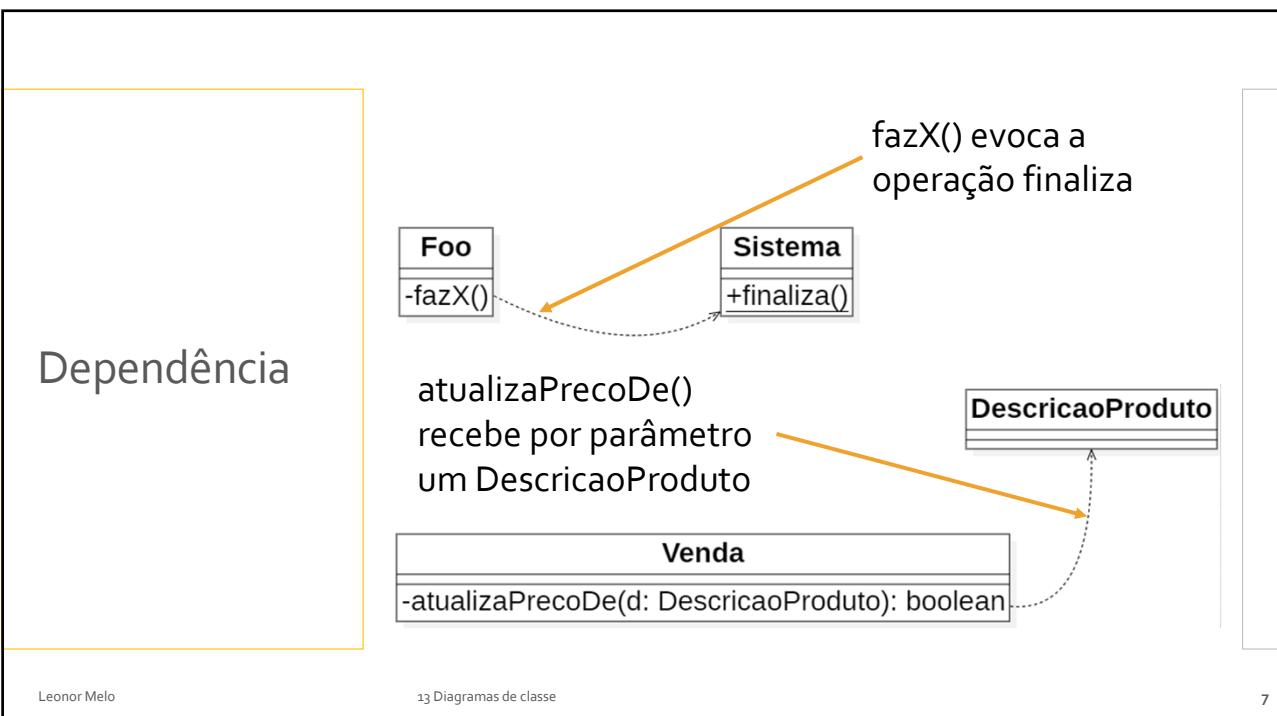
## Dependência

- Um objeto cliente usa um objeto fornecedor
  - uma alteração no fornecedor pode afetar o cliente
- Dependência de objetos feita por
  - atributos ou métodos estáticos
  - parâmetros ou valores devolvidos
  - variáveis locais
- Representa-se como dependência
  - relação não é suficientemente forte para ser uma das outras

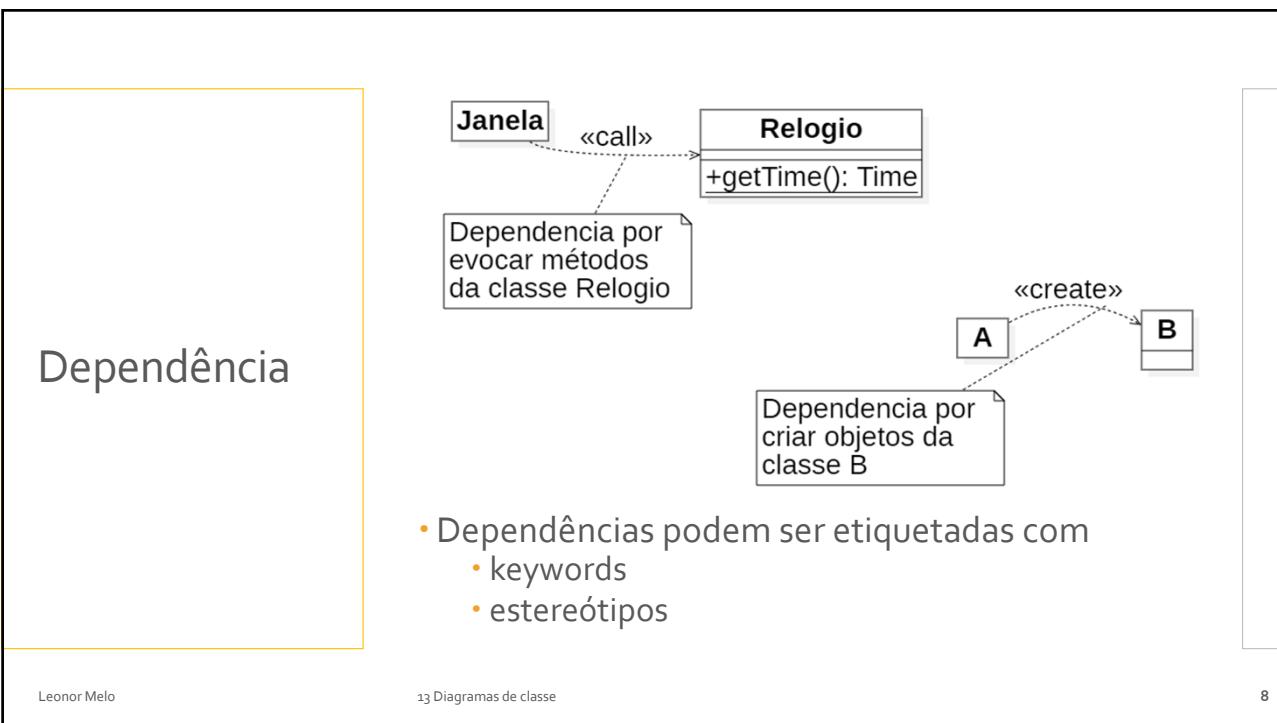
Leonor Melo

13 Diagramas de classe

6



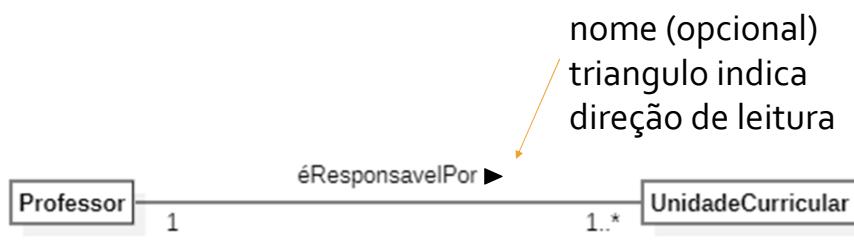
7



8

## Associação: modelo do domínio

- Relação de associação entre duas classes
  - uma classe tem uma relação lógica com outra



- exemplo: associação usada para representar uma ligação lógica entre as classes

Leonor Melo

13 Diagramas de classe

9

9

## Associação: classes de design

- Relação de associação entre duas classes
  - uma classe tem uma relação lógica com outra

Papel do objeto  
na relação



- exemplo: associação usada para representar referências para a outra classe

Leonor Melo

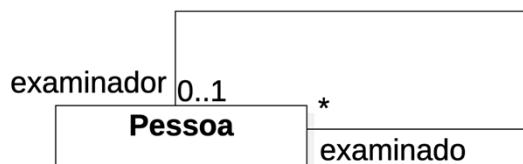
13 Diagramas de classe

10

10

## Associação reflexiva

- Associação entre instâncias da mesma classe
  - Papel de cada objeto explícito



- Uma pessoa
  - Pode ter, ou não, um examinador
  - Pode examinar, ou não, pessoas

Leonor Melo

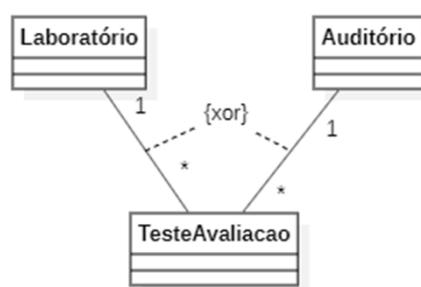
13 Diagramas de classe

11

11

## Associação com restrição xor

- Para exprimir que um objeto da classe A pode estar associado a um objeto da classe B ou C, mas não com ambos
  - usa-se a restrição xor



Leonor Melo

13 Diagramas de classe

12

12

## Associação Navegabilidade e

- Navegabilidade
  - qual das classes "conhece" a outra e pode aceder aos seus atributos e operações visíveis
- Exemplo: Professor contém um atributo que é um conjunto de referências para objetos UnidadeCurricular



Leonor Melo

13 Diagramas de classe

13

13

## Associação Navegabilidade e

- Pode-se omitir
  - o X (se for essa a convenção) e
  - a multiplicidade da origem



Leonor Melo

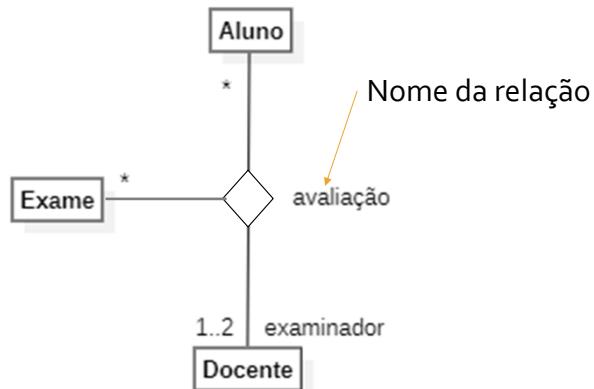
13 Diagramas de classe

14

14

## Associação n-ária

- Associação entre  $n$  instâncias
  - não tem navegabilidade associada



Leonor Melo

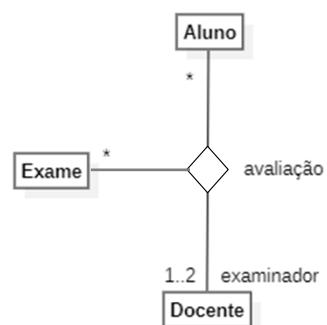
13 Diagramas de classe

15

15

## Associação n-ária

- Multiplicidade na relação n-ária
  - Quantos objetos dessa classe se relaciona com um tuplo de  $(n-1)$  objetos (1 de cada uma das outras classes)
  - um aluno, num dado exame, é avaliado por 1 ou 2 docentes
  - Um docente pode avaliar zero ou mais alunos de um determinado exame
  - Um docente pode avaliar zero ou mais exames de um dado aluno



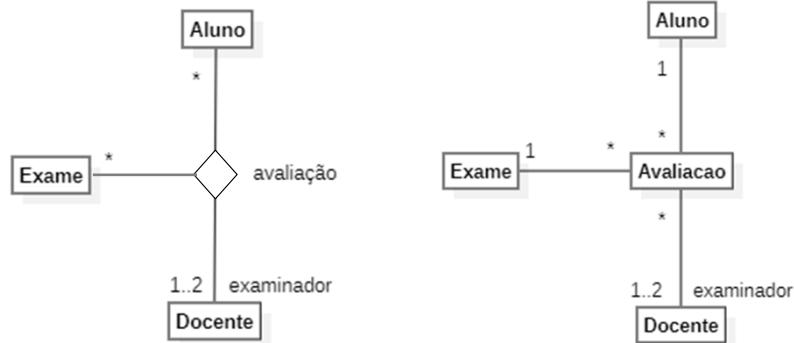
Leonor Melo

13 Diagramas de classe

16

16

## Associação n-ária



- No diagrama da direita

- É possível um aluno ter várias avaliações para o mesmo exame (cada uma corrigida por 1 ou 2 docentes, eventualmente diferentes)

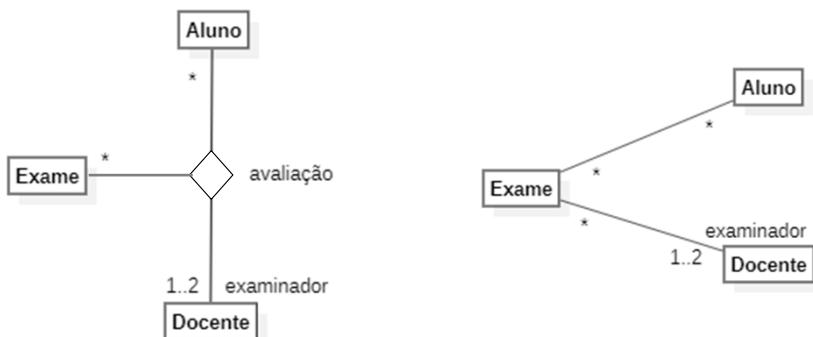
Leonor Melo

13 Diagramas de classe

17

17

## Associação n-ária



- No diagrama da direita

- o mesmo professor ou par de professores tem de corrigir todas as provas (alunos) de um dado exame

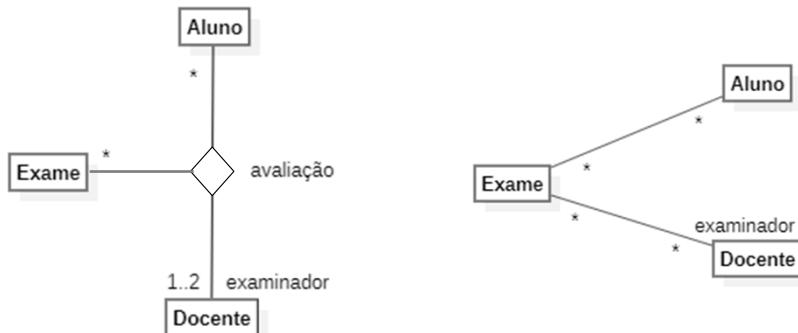
Leonor Melo

13 Diagramas de classe

18

18

## Associação n-ária



- No diagrama da direita
  - um exame pode ser corrigido por mais de 2 docentes

Leonor Melo

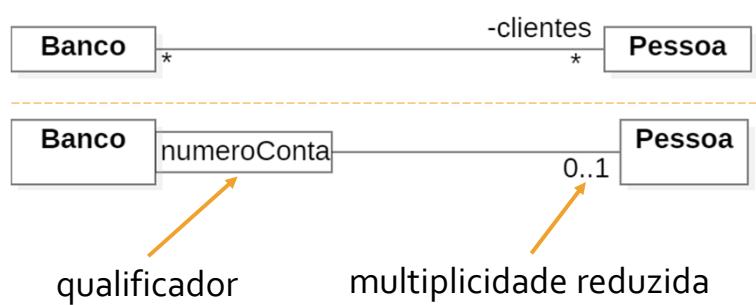
13 Diagramas de classe

19

19

## Associação qualificada

- tem um qualificador usado para selecionar um objeto (ou vários) de acordo com o valor da chave do qualificador



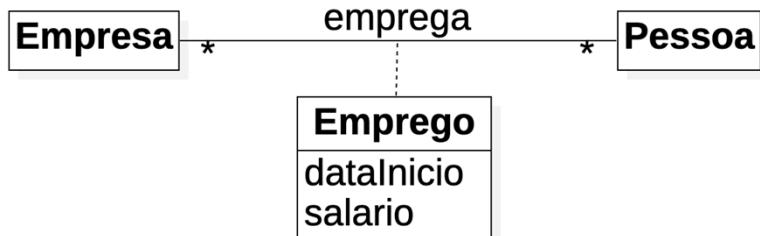
Leonor Melo

13 Diagramas de classe

20

20

# Classes de associação



- Classe de associação:
    - Relações mais complexas, com características (data de início, salário) próprias
    - Classe relacionada com outras duas devido à associação entre elas

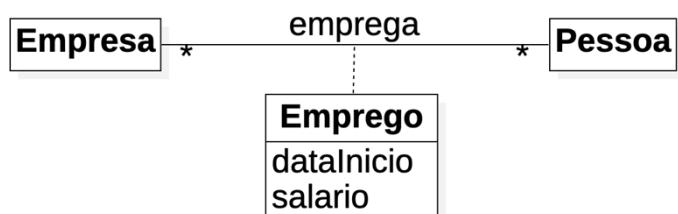
Leonor Melo

13 Diagramas de classe

21

21

# Classes de associação



- Uma Empresa emprega várias Pessoas
  - Uma Pessoa pode trabalhar para várias Empresas
  - Mas para cada par (Empresa, Pessoa) existe apenas um Emprego

Leonor Melo

13 Diagramas de classe

22

22

# Modelação e Design

## 14: Diagrama de Classes

Leonor Melo

leonor@isec.pt

1

Diagrama de classes

- Relações entre classes:
  - agregação partilhada
  - agregação composta
  - generalização
- Constraints

Leonor Melo

14 Diagramas de classe

2

## Agregação

- Relação do tipo "todo-parte"
- Binária
- Assimétrica
- Transitiva
- Pode ser
  - Partilhada
  - Composta

Leonor Melo

14 Diagramas de classe

3

3

## Agregação partilhada

- Agregação partilhada
  - Também conhecida por agregação
- Semanticamente não é clara a distinção entre
  - agregação partilhada
  - associação
- Considerada mais forte que a associação  
denota uma relação de "posse"

Leonor Melo

14 Diagramas de classe

4

4

## Agregação partilhada

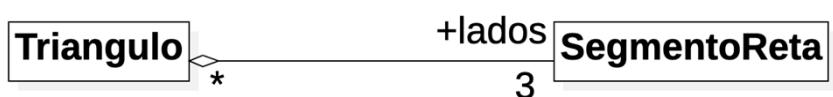
- O objeto de uma das classes (o todo) "é dono" do objeto da outra classe (a parte)
  - Uma das componentes do todo é a parte
  - A parte pode ser partilhada por vários todos
  - A parte pode existir sem o todo
- em código muitas vezes traduz-se por um atributo que é uma referência para um objeto

Leonor Melo

14 Diagramas de classe

5

5



## Agregação partilhada

- Um triângulo é composto por três lados
- Um segmento de reta pode pertencer a vários triângulos
  - Ou a nenhum

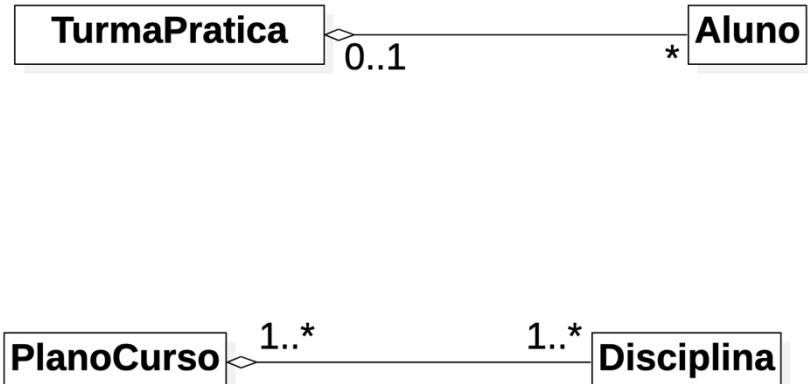
Leonor Melo

14 Diagramas de classe

6

6

Agregação  
partilhada:  
outros  
exemplos



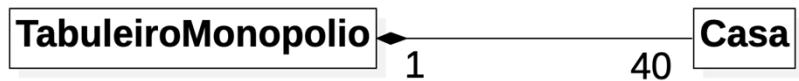
Leonor Melo

14 Diagramas de classe

7

Agregação  
composta

- Agregação Composta:
  - também conhecida por composição
- Mais forte das agregações
  - Uma das componentes do todo é a parte
  - A parte pertence apenas a um todo
  - A parte não existe sem o todo



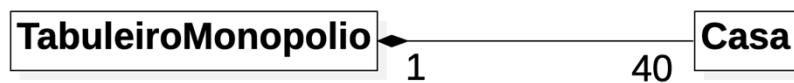
Leonor Melo

14 Diagramas de classe

8

## Agregação composta

- Em código muitas vezes traduz-se por um atributo que é um objeto
- Normalmente o todo é quem tem a responsabilidade de criar, destruir, e "propagar" as mensagens à parte



Leonor Melo

14 Diagramas de classe

9

9

## Agregação composta outros exemplo



Leonor Melo

14 Diagramas de classe

10

10

## Generalização

- Usada para descrever uma classe que é *um tipo* de outra classe
- Uma classe relaciona-se com outra
  - Tendo uma
    - associação,
    - agregação,
    - composição
  - Sendo uma
    - generalização
    - especialização

Leonor Melo

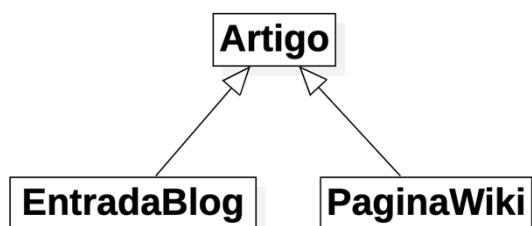
14 Diagramas de classe

11

11

## Generalização UML

- Classe mais geral
  - Classe mãe
  - Classe base
  - Superclasse
- Classe mais especializada
  - Classe filha
  - Classe derivada



Leonor Melo

14 Diagramas de classe

12

12

## Generalização

- Classe derivada
  - herda
    - Atributos
    - Operações
    - (que sejam visíveis)
  - Pode acrescentar
    - Atributos
    - Operações
  - que só façam sentido na classe derivada
- Generalização
  - Só faz sentido num sentido

Leonor Melo

14 Diagramas de classe

13

13

## Generalização e reutilização

- Generalização
  - Permite reutilização
  - Mas é o acoplamento mais forte
- Design deve favorecer acoplamento fraco
  - Usar generalização
    - se uma classe for uma especialização de outra
  - Não usar generalização
    - apenas para reutilizar algumas partes

Leonor Melo

14 Diagramas de classe

14

14

## Generalização múltipla

- Possível de representar em UML
  - Existe casos em que é útil
  - Algumas linguagens permitem
- Potencialmente problemática

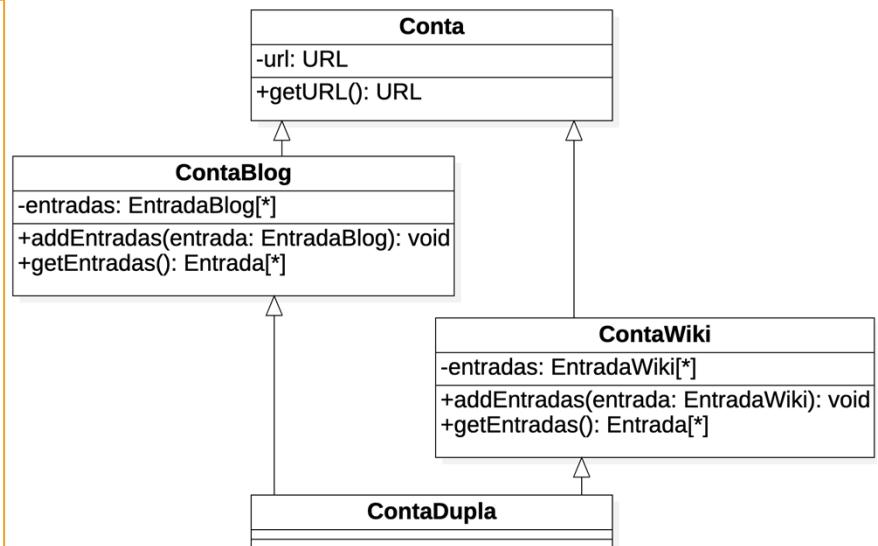
Leonor Melo

14 Diagramas de classe

15

15

## Problemas devido a Generalização múltipla?



Leonor Melo

14 Diagramas de classe

16

16

## Agregação, Composição ou Generalização ?

- Uma conta do banco tem um histórico de transações
- O animal de estimação tem um dono
- O animal de estimação tem uma cauda
- O animal de estimação é um cão
- O veículo tem 4 rodas (numa oficina de reparações); (num sistema aluguer carros)

Leonor Melo

14 Diagramas de classe

17

17

## *Constraints* (restrições)

- Condição ou restrição num elemento (classe ou associação) UML
- Usado sobretudo em diagramas de classes
- Indicam:
  - Uma condição específica que nunca se pode verificar
  - Que o valor de um atributo se baseia noutro
  - Que uma operação nunca pode deixar a classe num estado irregular
- Podem aplicar-se
  - a apenas um elemento
  - a mais de um elemento

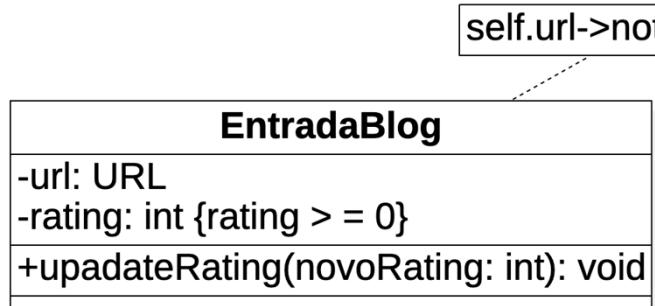
Leonor Melo

14 Diagramas de classe

18

18

## *Constraints* invariantes



- Invariante
  - Tem de ser sempre verdadeira
  - Aplica-se aos atributos da classe

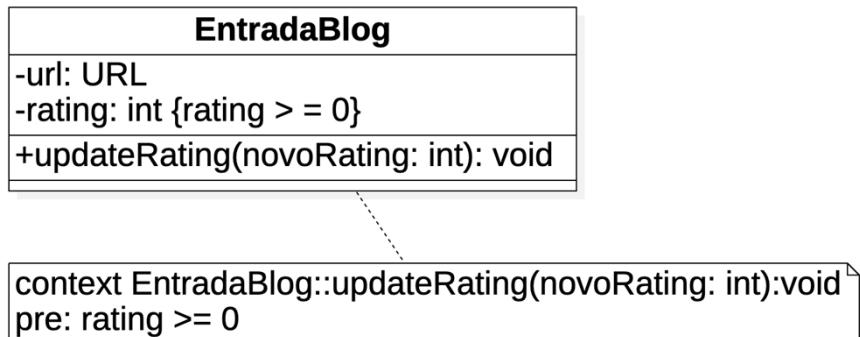
Leonor Melo

14 Diagramas de classe

19

19

## *Constraints* como pré-condições



- Pré-condição
  - Verificada antes do método ser executado
  - Aplica-se aos métodos da classe

Leonor Melo

14 Diagramas de classe

20

20

## *Constraints* como pós-condições

push(){post condition: novo tamanho = velho tamanho + 1 }

```
PilhaDe5
+tamanho: int {tamanho >= 0}
+push(elemento: Object)
+pop(): Objet
```

```
pop()
post condition: novo tamanho = velho tamanho -1
```

- Pós-condição
  - Verificada depois do método ser executado
  - Aplica-se aos métodos da classe

Leonor Melo

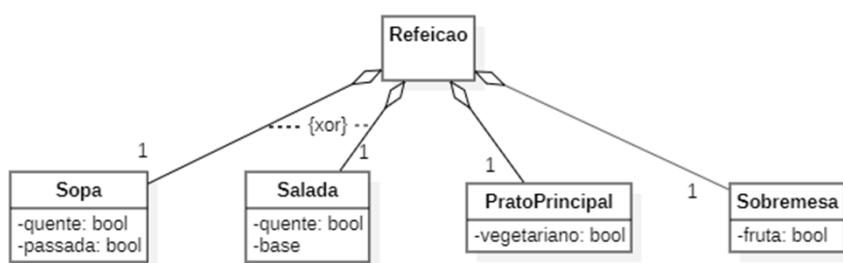
14 Diagramas de classe

21

21

## *Constraints* sobre agregações

- Uma refeição é composta por
  - sopa ou salada
  - prato principal
  - sobremesa



Leonor Melo

14 Diagramas de classe

22

22

# Modelação e Design

## 15: Diagrama de Classes

Leonor Melo

leonor@isec.pt

1

### Diagrama de classes

- Classes abstratas
- Interfaces
- Templates

Leonor Melo

15 - Diagramas de classe

2

## Classes abstratas

- Aplica-se a
  - classes mais genéricas
  - desenhadas para serem reutilizadas
  - agregam atributos e comportamentos comuns às subclasses
- Algum do comportamento da classe mais genérica é desconhecido
  - Sei que o comportamento deve existir
  - Mas implementação depende exclusivamente da classe derivada

Leonor Melo

15 - Diagramas de classe

3

## Classes abstratas

|   |
|---|
| <b>Repositorio</b>                        |
| +guarda( <i>artigos: Artigo[]</i> ): void |
| +recupera(): Artigo[]                     |

- Métodos abstratos
  - assinatura escrita em itálico
  - não contém implementação
  - "espaço reservado"
    - a implementação será feita pelas subclasses

Leonor Melo

15 - Diagramas de classe

4

## Classes abstratas

### **Repositorio**

```
+guarda(artigos: Artigo[*]): void  
+recupera(): Artigo[*]
```

- Se algum dos métodos for abstrato
  - a classe é abstrata
- Classe abstrata
  - Não pode ser instanciada

Leonor Melo

15 - Diagramas de classe

5

## Classes abstratas

### **Repositorio**

```
+guarda(artigos: Artigo[*]): void  
+recupera(): Artigo[*]
```

### **RepositorioBlog**

```
+guarda(artigos: Artigo[*]): void  
+recupera(): Artigo[*]
```

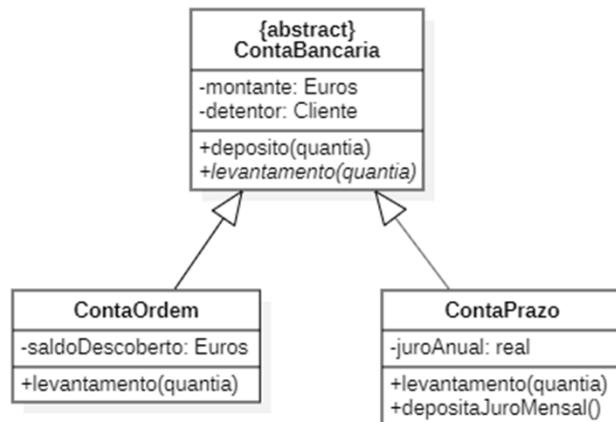
- Subclasse de classe abstrata
  - Ou implementa os todos os métodos abstratos
  - Ou é também abstrata

Leonor Melo

15 - Diagramas de classe

6

## Classes abstratas



- normalmente
  - tem atributos
  - apenas parte das operações são abstratas

Leonor Melo

15 - Diagramas de classe

7

## Classes abstratas

- Classes abstratas
  - Mecanismo que promove a reutilização
  - Define-se atributos e operações que deverão existir / ser garantidos
    - Ainda que os detalhes venham a ser definidos pelas sub- classes
  - Usadas frequentemente em design patterns
  - Usa relação de herança
    - Acoplamento forte
      - Usar com cuidado

Leonor Melo

15 - Diagramas de classe

8

## Interfaces

- Representa um tipo de comportamento que determinada classe pederá, ou não, ser capaz de fornecer
- Conjunto de operações
  - sem implementação dos métodos
- "Contrato" que indica
  - a assinatura das operações que devem existir
- Raramente contém atributos
  - e apenas estáticos ou constantes
- Não pode ser instanciada

Leonor Melo

15 - Diagramas de classe

9

9

## Notação UML



Notação com esteriótipo



Notação com "bola"

Leonor Melo

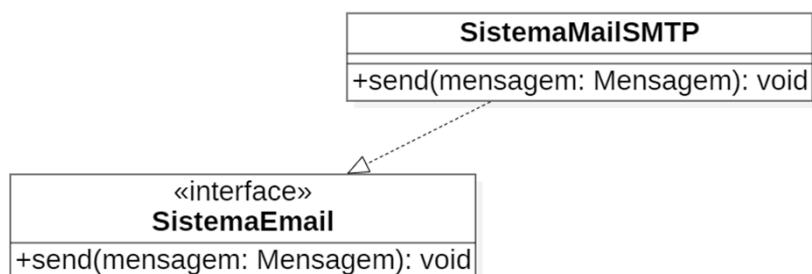
15 - Diagramas de classe

10

10

## Relação de realização

- “relação de realização”
  - Estabelecida entre uma classe e um interface
  - A classe esté em conformidade com o contrato especificado pela interface



Leonor Melo

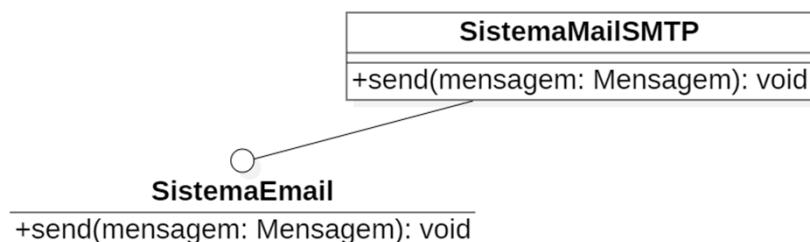
15 - Diagramas de classe

11

11

## Relação de realização

- representação UML
  - na notação de bola
  - linha de associação
- Notação com esteriótipos
  - seta de realização



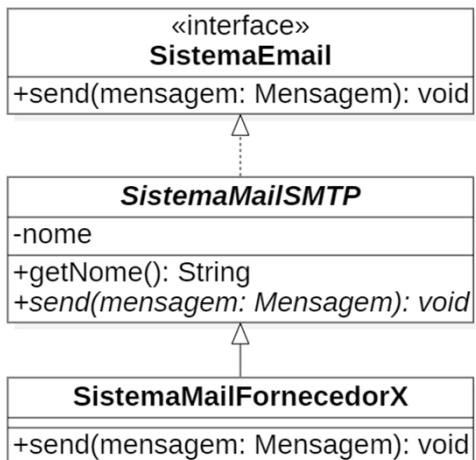
Leonor Melo

15 - Diagramas de classe

12

12

## Relação de realização



- Uma classe que realize um interface mas não implemente todos os métodos do interface tem de ser declarada abstrata

Leonor Melo

15 - Diagramas de classe

13

13

## Interface: relação de dependencia

- Os interfaces existem para que as classes dependam dos interfaces e não de classes específicas (abstração, redução de acoplamento)
- Uma classe A depende de um interface se
  - é indiferente os detalhes da classe que na realizade vai executar os métodos
  - a classe A sabe que a classe que eventualmente implementar o interface está preparada para receber mensagens com determinada assinatura

Leonor Melo

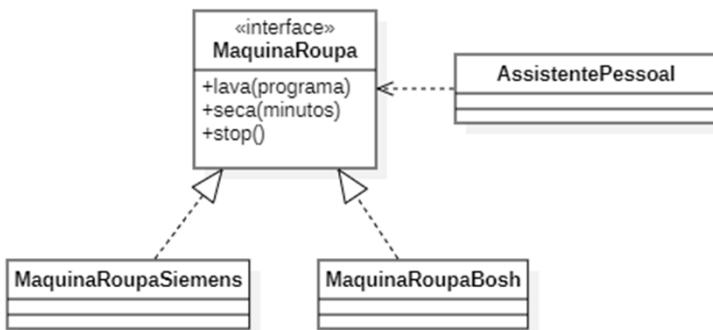
15 - Diagramas de classe

14

14

## Interface: relação de dependencia

- O assistente sabe como comunicar com uma máquina da roupa, desde que essa máquina realize o interface MaquinaRoupa



Leonor Melo

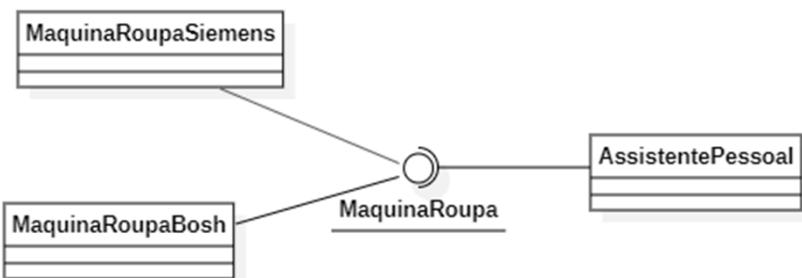
15 - Diagramas de classe

15

15

## Interface: relação de dependencia

- Notação de bola:
- Tanto pode comunicar com uma máquina Bosch como com uma Siemens (ou outras que entretanto realizem o interface)



Leonor Melo

15 - Diagramas de classe

16

16

## Interfaces

- Uma classe pode realizar vários interfaces
- Evita problemas da generalização múltipla
  - porquê?
- Semelhantes a classes abstratas
  - mas com acoplamento mais fraco
  - usadas para "desacoplar" classes
- Usadas frequentemente em design patterns

Leonor Melo

15 - Diagramas de classe

17

17

## Interfaces

- Separar o comportamento que a classe deve ter
  - da implementação desse comportamento
- Se uma classe implementar um interface
  - objetos dessa classe podem ser referidos pelo nome do interface
- As outras classes podem passar a depender do interface
  - e não da classe que o implementa

Leonor Melo

15 - Diagramas de classe

18

18

## Templates

- classe parameterizada
  - útil quando se quer adiar a decisão sobre com que classe(s) a nossa classe irá trabalhar
- sei que a minha classe irá trabalhar com outras
  - mas não quero ter de especificar exatamente com qual

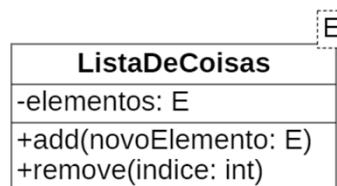
Leonor Melo

15 - Diagramas de classe

19

19

## Templates



- E não é nenhuma classe existente no modelo
  - está apenas no lugar onde aparecerá a classe cujos objetos serão guardados na lista

Leonor Melo

15 - Diagramas de classe

20

20

## Usar Templates

- usar uma classe template
  - fazer o *bind* (ligar) dos parâmetros
- Fazer o *bind*:
  - *binding* por subclasse
  - *binding* em *runtime*

Leonor Melo

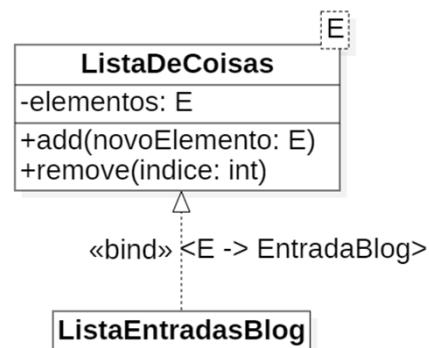
15 - Diagramas de classe

21

21

## Usar Templates: *binding* por subclasse

- **ListaEntradasBlog:**
  - comportamento genérico de **ListaDeCoisas**
  - apenas para instâncias de **EntradaBlog**



Leonor Melo

15 - Diagramas de classe

22

22

## Usar Templates: *binding* em *runtime*

- O *template* só conhece o tipo de dados com que vai trabalhar quando é criado
  - em *runtime*
- *Binding* que diz respeito aos objetos
  - representado usando diagramas de objeto

Leonor Melo

15 - Diagramas de classe

23

23

# Modelação e Design 16: Diagrama de Objeto

Leonor Melo

leonor@isec.pt

1

## Diagrama de objetos

- Notação UML geral
- Objetos e relações reflexivas
- Objetos e relações ternárias
- runtime bind de classes template

Leonor Melo

16 Diagramas de Objeto

2

## Diagramas de objetos

- Sistema orientado a objetos em execução
  - objetos
- Diagrama de objeto
  - descrever objetos existentes
    - num cenário/momento específico
  - destacar
    - objetos existentes e
    - as suas interações
  - Visão lógica do modelo

Leonor Melo

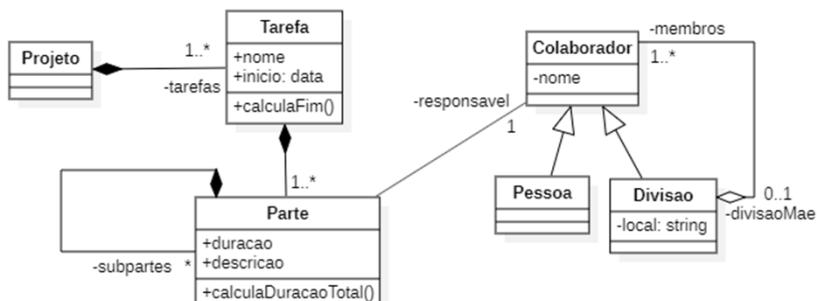
16 Diagramas de Objeto

3

3

## Diagramas de objetos

- Ajudar a visualizar diagramas de classes mais complexos



Leonor Melo

16 Diagramas de Objeto

4

4

## Objeto: representação UML

**entrada**

**entrada: EntradaBlog**

- objeto cuja identidade é entrada

- objeto da classe EntradaBlog, cuja identidade é entrada

- Identidade de um objeto é única:

- estes dois objetos não poderiam pertencer ao mesmo diagrama

Leonor Melo

16 Diagramas de Objeto

5

## Objeto: representação UML

**entrada**

**entrada: EntradaBlog**

- Objeto UML = Especificação de objeto
  - podem estar apenas parcialmente definidos
    - certos atributos podem ficar por preencher
  - é possível mostrar especificações de objetos mesmo se a classe for abstrata

Leonor Melo

16 Diagramas de Objeto

6

## Objeto anónimo: representação UML

- Quando a identidade do objeto não é relevante:
- ActionListener**  
+actionPerformed(e: ActionEvent): void

: ActionListener

• Classe ActionListener

• objeto anónimo da classe ActionListener

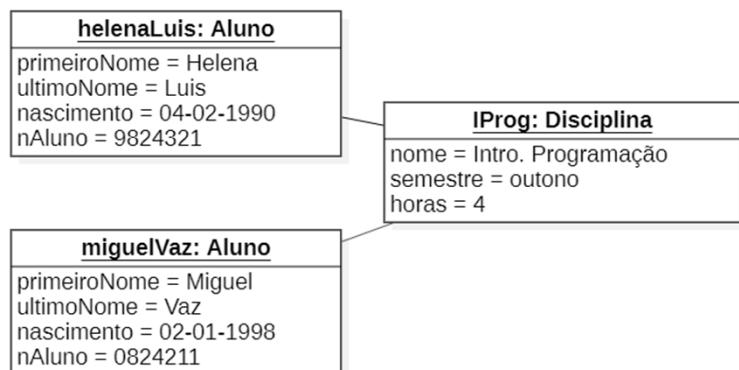
Leonor Melo

16 Diagramas de Objeto

7

## Objetos com atributos

- Podemos incluir valor que os atributos do objeto têm nesse momento



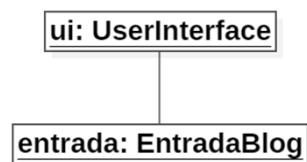
Leonor Melo

16 Diagramas de Objeto

8

## Link

- Indica que dois objetos
  - podem comunicar entre si
  - implica relação entre as classes correspondentes
- Tem de coincidir com o que foi representado no diagrama de classes



Leonor Melo

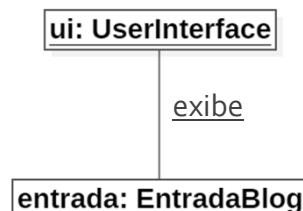
16 Diagramas de Objeto

9

9

## Link com label

- informação adicional (opcional)
  - propósito do link



Leonor Melo

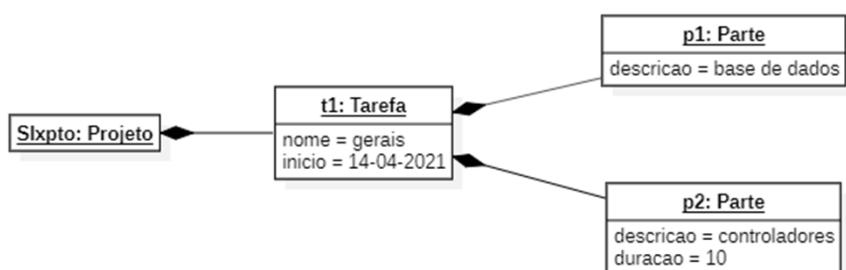
16 Diagramas de Objeto

10

10

## Extremidades do link

- também pode ter:
  - nome do papel desempenhado
  - naveabilidade
  - agregação / composição



Leonor Melo

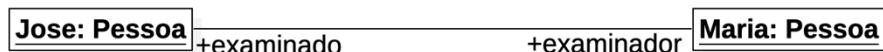
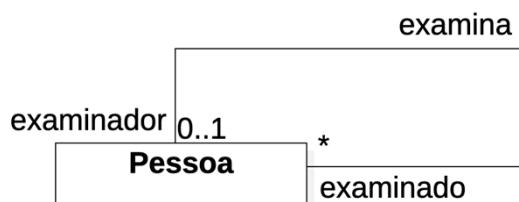
16 Diagramas de Objeto

11

11

## Objetos e relações reflexivas

- Dado o diagrama de classes,
  - O diagrama de objeto está correto?



Leonor Melo

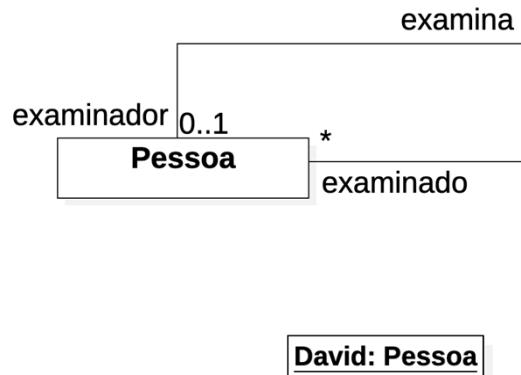
16 Diagramas de Objeto

12

12

## Objetos e relações reflexivas

- Dado o diagrama de classes,
  - O diagrama de objeto está correto?



Leonor Melo

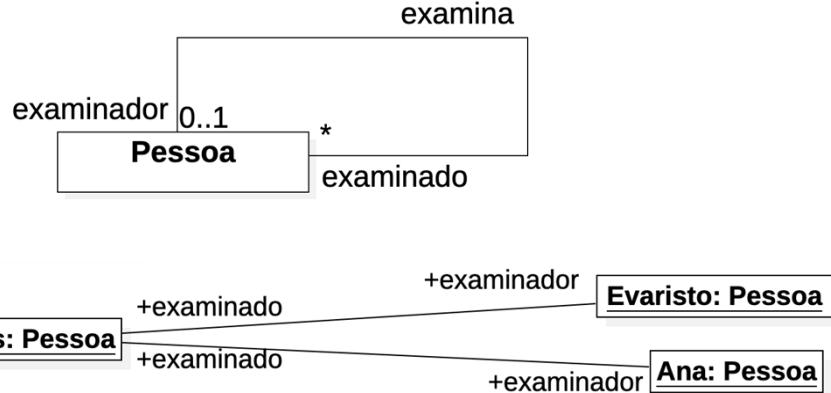
16 Diagramas de Objeto

13

13

## Objetos e relações reflexivas

- Dado o diagrama de classes,
  - O diagrama de objeto está correto?



Leonor Melo

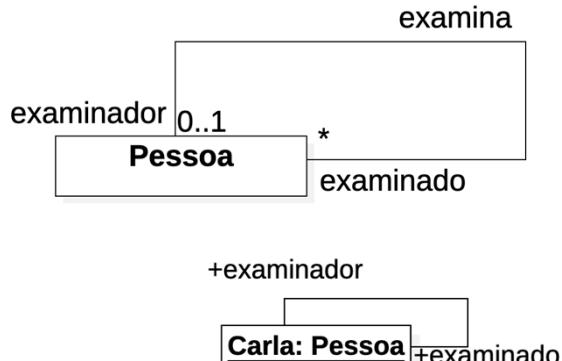
16 Diagramas de Objeto

14

14

## Objetos e relações reflexivas

- Dado o diagrama de classes,
  - O diagrama de objeto está correto?



Leonor Melo

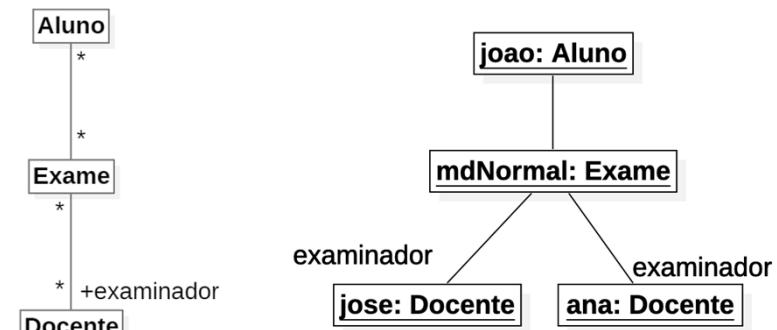
16 Diagramas de Objeto

15

15

## Objetos e relações ternárias

- Dado o diagrama de classes,
  - O diagrama de objeto está correto?



Leonor Melo

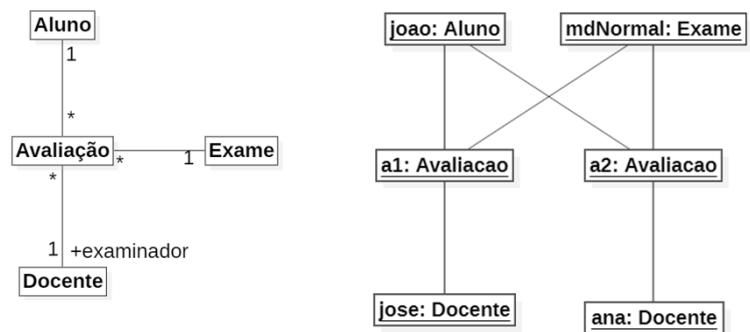
16 Diagramas de Objeto

16

16

## Objetos

- Dado o diagrama de classes,
  - O diagrama de objeto está correto?



Leonor Melo

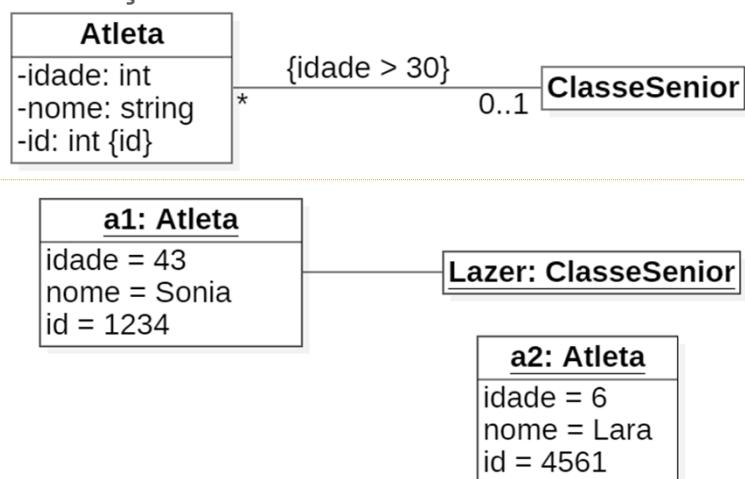
16 Diagramas de Objeto

17

17

## Links e constraints

- Links têm de respeitar as constraints nas associações:



Leonor Melo

16 Diagramas de Objeto

18

18

## Runtime bind de classes Template

- *Runtime bind* é a maior vantagem das classes template
- Só no momento de construir o objeto
  - é que se indica o tipo que os parâmetros devem ser

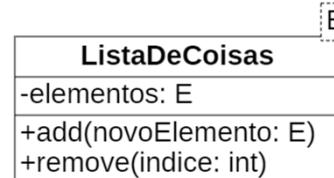
Leonor Melo

16 Diagramas de Objeto

19

19

## Runtime bind de classes Template



- classe Template

**Object1: ListaDeCoisas <E -> EntradaBlog>**

- runtime bind

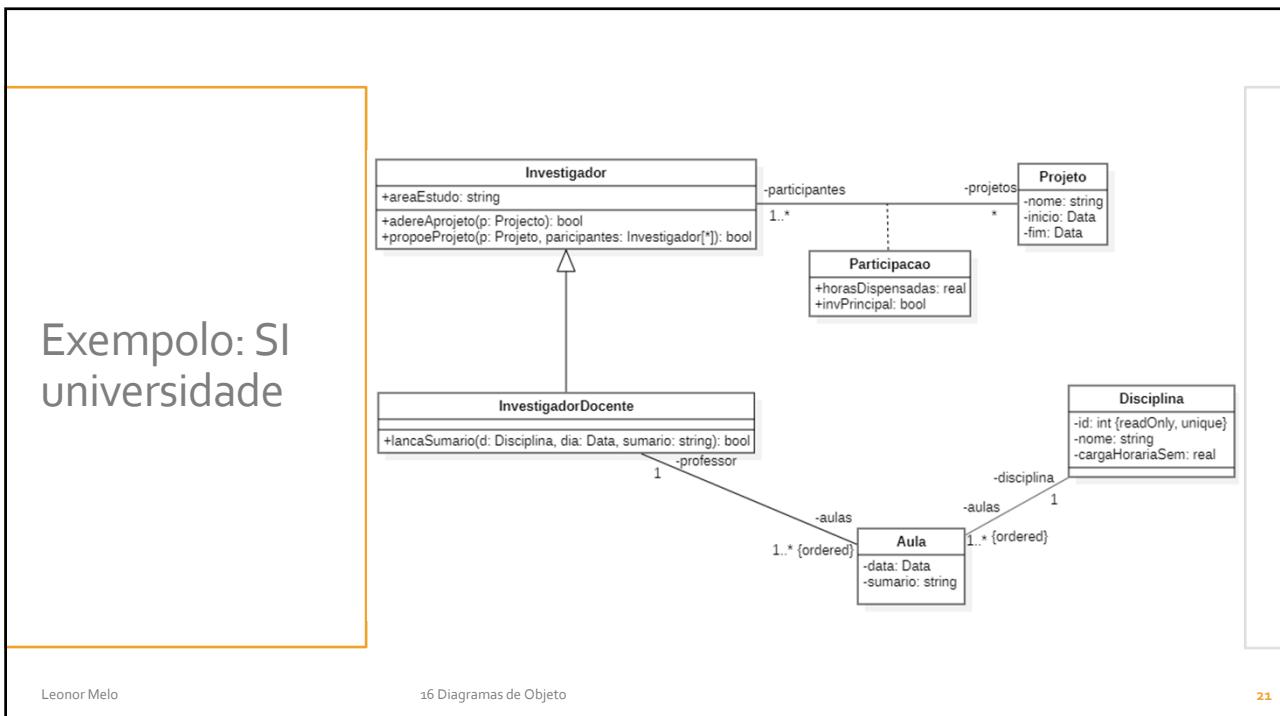
Leonor Melo

16 Diagramas de Objeto

20

20

## Exempolo: SI universidade



21

## Diagrama de objetos - atributos

- Dia 15 de abril o docente António
  - Lecionou 1 aula de Programação – que tem uma carga letiva semanal de 4 horas
  - E uma aula de Sistemas Inteligentes – que tem uma carga letiva semanal de 2 horas

Leonor Melo

16 Diagramas de Objeto

22

## Diagrama de objetos - classes de associação

- A investigadora Ivone
  - é o investigador principal do projeto FCT-Mito-Fit, onde colabora a 30% do seu tempo;
  - Participa igualmente no projeto CAPES-Energy-landscapes a 15% do seu tempo

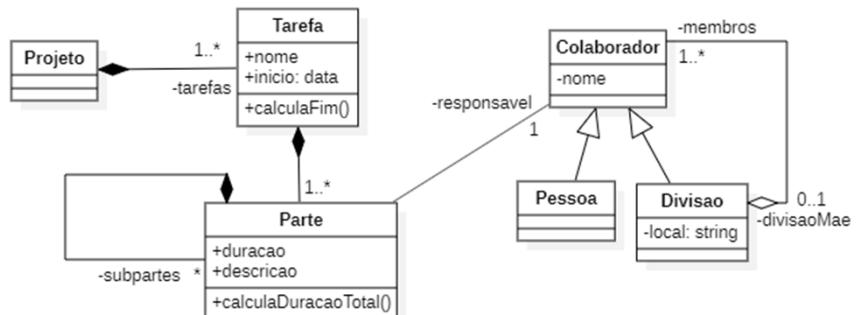
Leonor Melo

16 Diagramas de Objeto

23

23

## Exemplo: Projeto



Leonor Melo

16 Diagramas de Objeto

24

24

## Diagrama de objetos - generalização + associação reflexiva

- A Divisão XXYY localizada em Lisboa
  - Tem um Colaborador que é uma pessoa de nome Carlos Gomes
  - E uma subdivisão XXYYexpert localizada em Évora, que tem como membros a Joana Silva e o António Cunha
- O Carlos Gomes deve ficar responsável pela Parte da base de dados, e a subdivisão XXYYexpert deve ficar responsável pela parte dos controladores, com duração estimada de 10 dias

Leonor Melo

16 Diagramas de Objeto

25

25

## Diagrama de objetos - generalização + associação reflexiva

- É possível uma parte ser atribuída a uma Divisão, e a Divisão não ter Pessoas nem subdivisões?
- É possível uma Pessoa não pertencer a nenhuma divisão?

Leonor Melo

16 Diagramas de Objeto

26

26

# Modelação e Design

## 17: Diagrama de Sequencia

Leonor Melo

leonor@isec.pt

1

### Diagrama de Sequencia

- Diagramas de interação
- Noção de parceiro de interação
- Troca de mensagens enquanto especificação de eventos
- Sintaxe das mensagens

Leonor Melo

17 Diagrama de Sequencia

2

## Diagramas de Interação

- **Interação**
  - Ordem pela qual mensagens e dados são trocados entre determinados parceiros de interação (ou participantes)
- **Exemplo de parceiros de interação**
  - Humanos
    - Professores,
    - Estudantes, ...
  - Não humanos
    - servidor,
    - impressora,
    - aplicação,
    - objetos de software,...

Leonor Melo

17 Diagrama de Sequencia

3

## Diagramas de Interação

- **Exemplos de interação**
  - Conversa entre várias pessoas
  - Exame oral
  - Resposta a um sinal
    - Soar de um alarme de incendio e correspondente procedimentos a adotar
  - Troca de mensagens entre humanos e sistema de software
    - Utilização, por parte do professor e aluno do software de gestão académica
  - Sequencia de chamadas de métodos
    - Execução de um programa

Leonor Melo

17 Diagrama de Sequencia

4

## Diagramas de interação

- Usados para
  - um designer ou grupo de designers compreenderem melhor os detalhes de determinada interação
  - na fase de design detalhado assegurar que os protocolos de comunicação inter-processos são cumpridos; desenhar esses protocolos
  - na fase de teste comparar o trace do sistema com o que tinha sido estipulado
  - explicar aos utilizadores potenciais / stakeholders o modo como o sistema pode ser usado para obter determinada funcionalidade
- trace = sequência de eventos que surgiram ou devem / podem surgir

Leonor Melo

17 Diagrama de Sequencia

5

## Diagramas de Interação

- Representam o comportamento inter-objeto do sistema
- Incluem, entre outros
  - Diagrama de sequencia
  - Diagrama de comunicação

Leonor Melo

17 Diagrama de Sequencia

6

6

## Diagramas de Interação

- **Interação** descreve
  - Sequencia de troca de mensagens entre vários parceiros de interação
- **Envio/recepção de mensagem**
  - originado pela ocorrência de certos eventos
    - p.ex. a receção de outra mensagem
- **Interações** podem ter associadas pré-condições
  - Condições que têm de se verificar para a interação decorrer com sucesso
    - p.ex. o professor tem de estar registado no sistema antes de introduzir as notas dos alunos

Leonor Melo

17 Diagrama de Sequencia

7

## Diagramas de Interação UML

- **Diagramas de interação UML**
  - Modelam um cenário concreto
    - A troca de mensagem
      - ocorre dentro de um contexto específico
      - serve para realizar determinada tarefa
  - Cobrem uma parte específica da situação
  - Pode representar dados trocados / processados / guardados entre os parceiros de interação
    - Mas o enfase é em como determinada sequência de mensagens leva à obtenção de dada funcionalidade

Leonor Melo

17 Diagrama de Sequencia

8

## Diagramas de Interação UML

- Diagramas de interação UML
  - podem ser usados a diferentes níveis de abstração:
    - Interacção de um sistema com o seu meio ambiente
    - Interacção entre as partes que constituem o sistema
    - Protocolos de comunicação inter-processo
    - Interacção entre as classes que constituem o sistema

Leonor Melo

17 Diagrama de Sequencia

9

9

## Parceiros de interação

- Parceiros de interação (ou participantes)
  - Representados por *lifelines*
  - Cada *lifeline* representa um único objeto
  - Identificação:
    - nome\_objeto [seletor] : nome\_classe
    - Seletor só quando queremos 1 elemento de um multivalue
    - nome\_objeto ou nome\_classe pode ser omitido

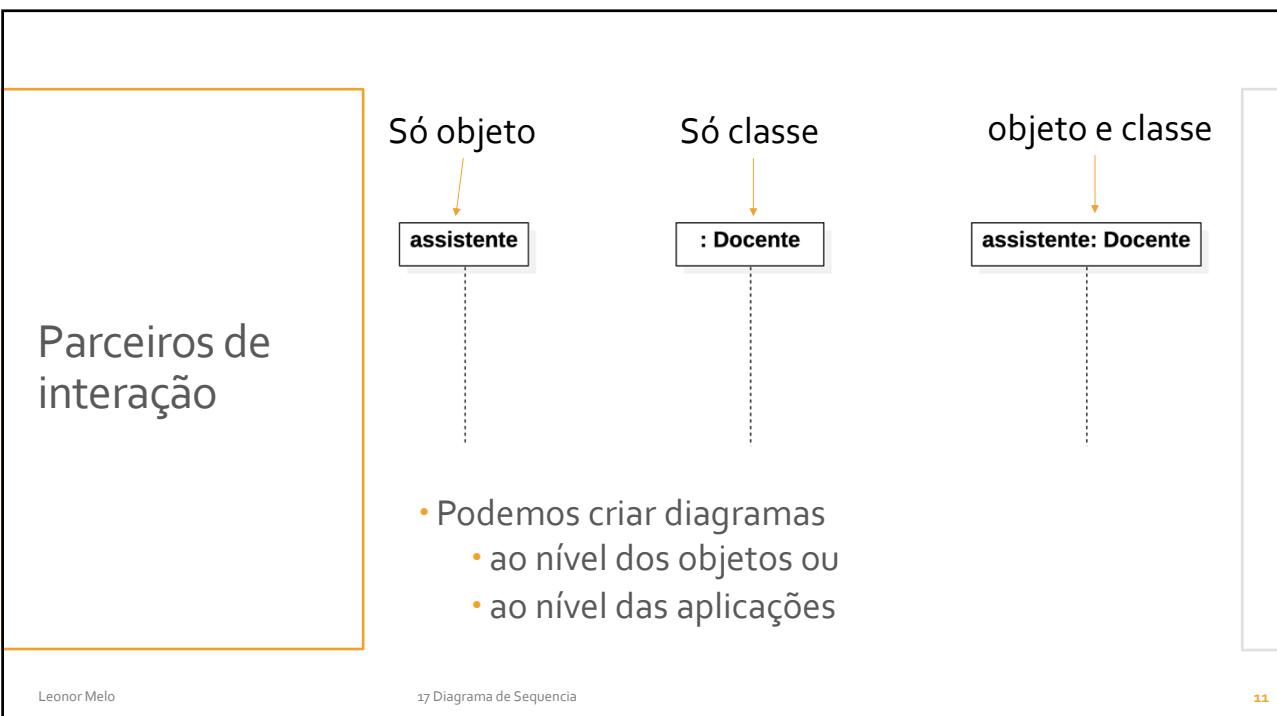
assistente: Docente

Leonor Melo

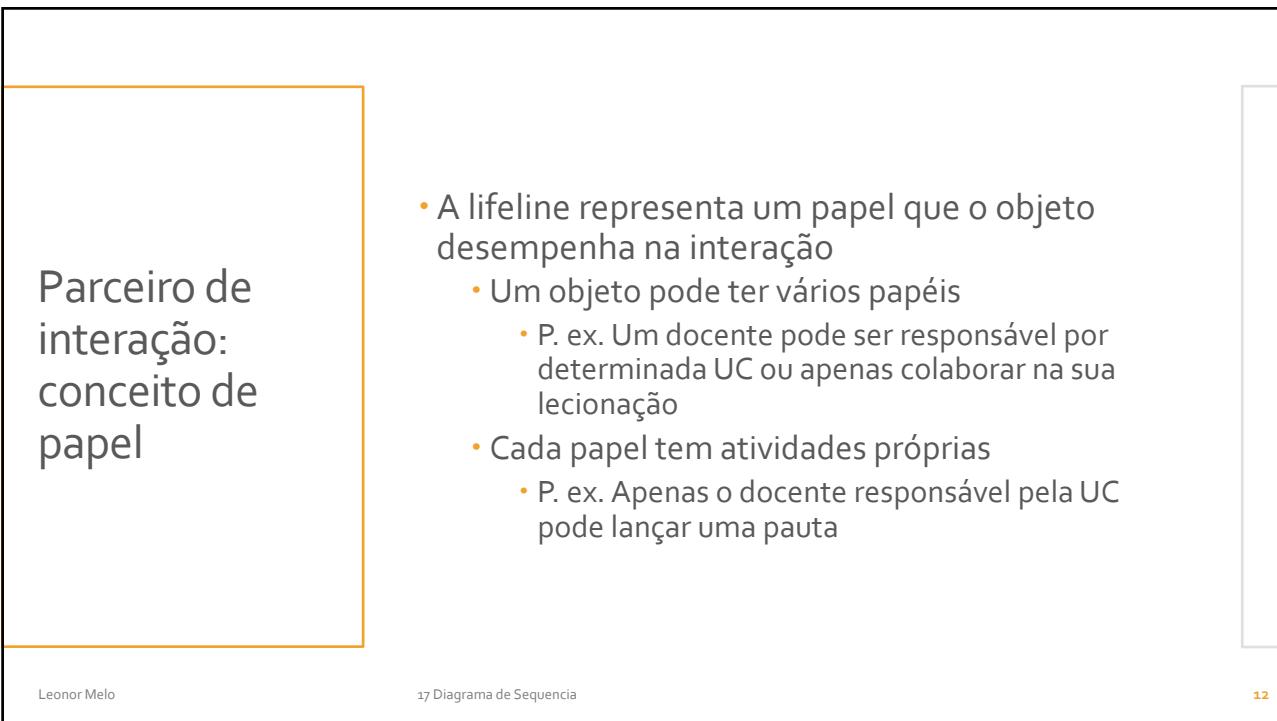
17 Diagrama de Sequencia

10

10



11



12

## Parceiro de interação: Seletor do papel

- assistente[i]: Docente
- o objeto é um atributo múltiplo
    - nesse caso usa-se um seletor para representar o (sub)objeto específico
  - o seletor é indicado entre parêntesis retos
  - podemos ter vários participantes selecionados na mesma interação
    - cada um com a sua lifeline

Leonor Melo

17 Diagrama de Sequencia

13

13

## Troca de mensagens: Tempo

- assistente : Docente
- 
- Diagrama de sequencia
    - ordem pela qual as interações ocorrem
  - Enfase na ordem pela qual as interações ocorrem
    - não na duração
    - espaço vertical ocupado pela interação não tem significado

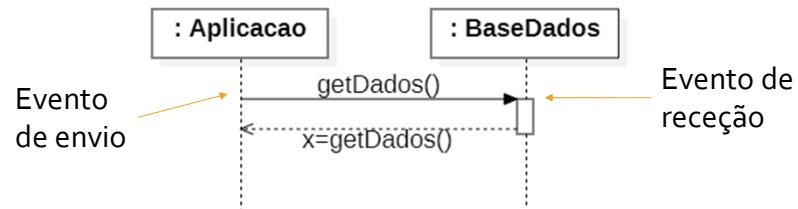
Leonor Melo

17 Diagrama de Sequencia

14

14

## Troca de mensagens: especificação de eventos



- As interações são consideradas especificações de eventos
- Eventos
  - enviar mensagem
  - receber mensagem
  - eventos baseados em tempo
    - ter sido atingido determinado ponto no tempo

Leonor Melo

17 Diagrama de Sequencia

15

## Troca de mensagens: especificação de eventos

- A linha vertical determina a sequencia de ocorrência de eventos numa *lifeline*
  - Independente das outras *lifelines*
- Só existe alguma relação de ordem entre várias *lifelines*
  - se existirem mensagens trocadas entre elas
- Normalmente assume-se que a transmissão é instantânea

Leonor Melo

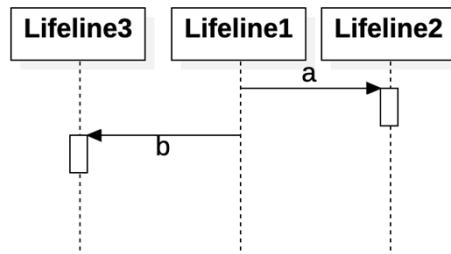
17 Diagrama de Sequencia

16

16

## Troca de mensagens: especificação de eventos

- A ligação cronológica (*trace*) entre a mensagem **a** e a mensagem **b** representa-se usando o símbolo →
  - $a \rightarrow b$
  - a mensagem **a** é enviada antes da mensagem **b**



Leonor Melo

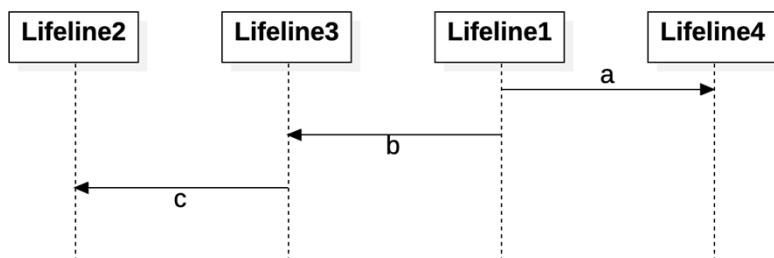
17 Diagrama de Sequencia

17

17

## Troca de mensagens: especificação de eventos

- Qual o trace da seguinte interação?



Leonor Melo

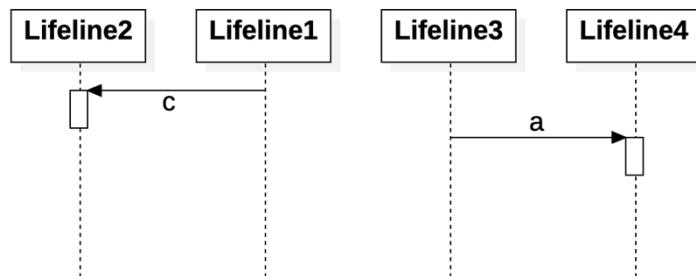
17 Diagrama de Sequencia

18

18

## Troca de mensagens: especificação de eventos

- Qual o trace da seguinte interação?



Leonor Melo

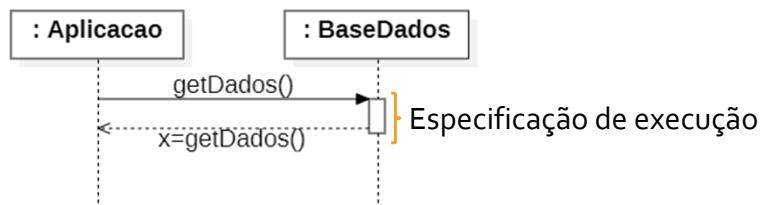
17 Diagrama de Sequencia

19

19

## Troca de mensagens: comportamento de execução

- Especificação de execução (barra de ativação)
- Representa o comportamento adotado face à receção da mensagem
- Pode ser omitido



Leonor Melo

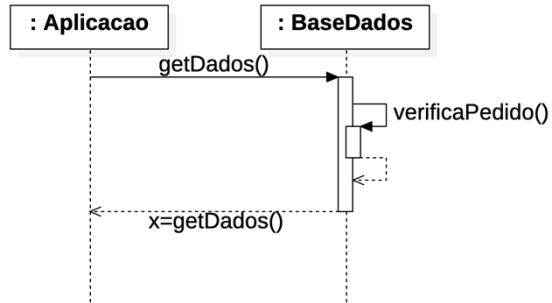
17 Diagrama de Sequencia

20

20

## Troca de mensagens: mensagens para si próprio

- Especificação de execução sobrepostas
  - significa que os comportamentos também se sobrepõem
  - Exemplo, se um participante enviar uma mensagem a si próprio



Leonor Melo

17 Diagrama de Sequencia

21

21

## Mensagens

- Mensagem é enviada pelo emissor com o objetivo provocar determinado comportamento no receptor
  - chamar um método
  - devolver um valor (em consequência de uma mensagem de chamada)
  - criar um objeto
  - destruir um objeto
- Representada por uma seta
- O tipo de seta indica o tipo de mensagem

Leonor Melo

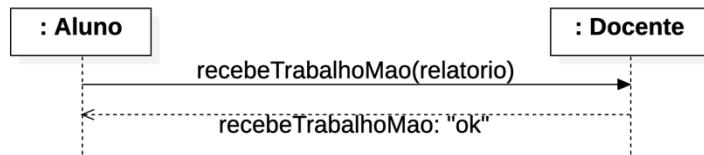
17 Diagrama de Sequencia

22

22

## Mensagens: mensagens síncronas

- Mensagem síncrona (seta a cheio)
  - O emissor fica à espera até receber a mensagem de resposta enviada pelo recetor
    - Transferência do fluxo de execução
- Mensagem de resposta pode ser omitida
  - se conteúdo e momento de envio/recepção evidentes



Leonor Melo

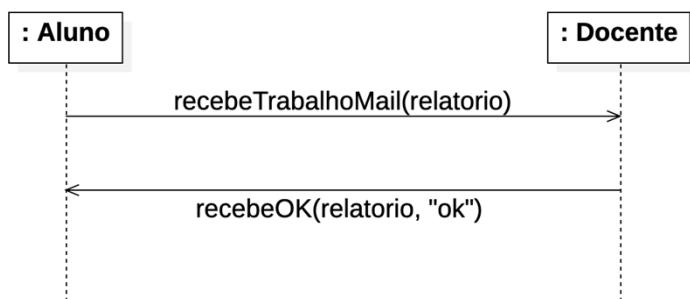
17 Diagrama de Sequencia

23

23

## Mensagens: mensagens assíncronas

- Mensagem assíncrona (seta aberta)
  - O emissor envia a mensagem e continua a sua própria execução
    - Emissor e recetor têm execuções em paralelo



Leonor Melo

17 Diagrama de Sequencia

24

24

## Mensagens de envio: sintaxe

*nomeMensagem([(argumento\_in] [, argumento\_in]\*])*

*argumento\_in ::= nome\_argumento [= valor]*

- Apenas o nomeMensagem é obrigatório
- argumentos pode ser uma lista de argumentos de entrada separada por vírgulas
- A receção da mensagem normalmente resulta na evocação do método com esse nome por parte do recetor da mensagem
  - Os argumentos da mensagem devem ser compatíveis com os parâmetros de entrada do método
  - Se o nome de todos parâmetros for usado a ordem não interessa

Leonor Melo

17 Diagrama de Sequencia

25

25

## Mensagens de resposta: sintaxe

*[atributo =] nomeMensagem([(argumento\_out] [, argumento\_out]\*]) [: valor]*

*argumento\_out ::= nome\_argumento [: valor] | atributo = nome\_argumento [: valor]*

- Apenas o nomeMensagem é obrigatório
- Só se indicam os argumentos de saída ou inout
- A legenda pode ser inteiramente omitida se for evidente qual a mensagem que lhe deu origem e o valor devolvido não for relevante para a interação

Leonor Melo

17 Diagrama de Sequencia

26

26

## Mensagens: exemplo

- O que podemos ficar a saber das pelas mensagens?
1. fazQualquerCoisa()
  2. fazQualquerCoisa( 1, 20 )
  3. fazQualquerCoisa( numero2 = 50 )

Leonor Melo

17 Diagrama de Sequencia

27

27

## Mensagens: exemplo

- O que podemos ficar a saber das pelas mensagens de resposta?
1. fazQualquerCoisa() : "ok"
  2. resposta = fazQualquerCoisa: "ok"
  3. v=mymsg(w=myout:16):96

Leonor Melo

17 Diagrama de Sequencia

28

28

# Modelação e Design

## 18: Diagrama de Sequencia

Leonor Melo

leonor@isec.pt

1

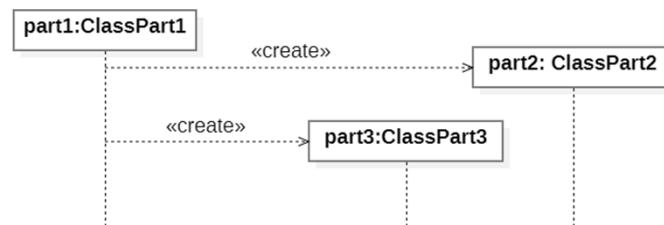
Diagrama de  
sequencia

- Mensagens especiais
- Fragmentos combinados
  - alt
  - opt
  - loop
  - break
  - seq

2

## Mensagens especiais: criar objetos

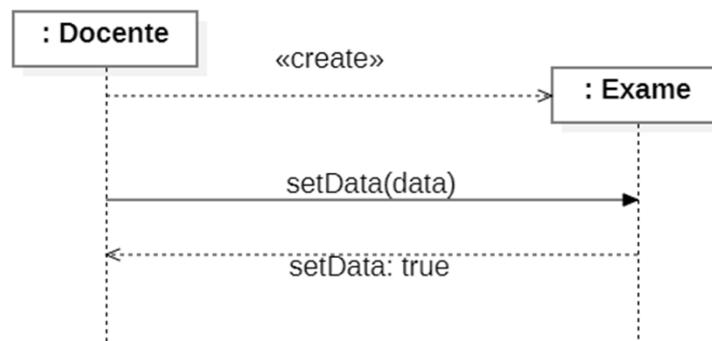
- Um objeto
  - pode existir desde o início da interação
  - pode ser criado durante a interação
- Mensagem de *create*
  - Corresponde à evocação do construtor do objeto



3

## Mensagens especiais: criar objetos

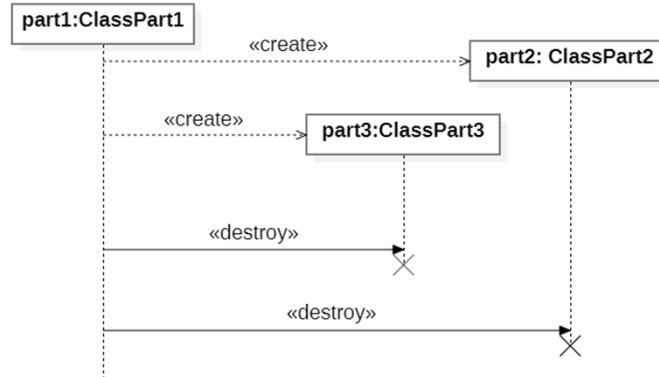
- O que representa a seguinte interação:



4

## Mensagens especiais: destruir objetos

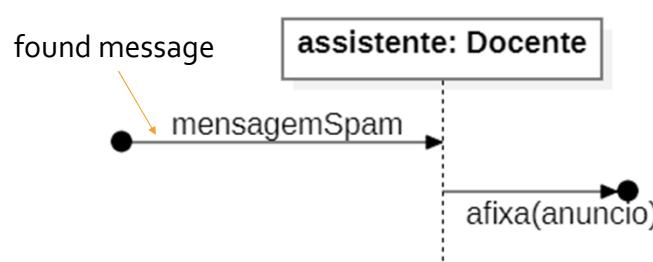
- Se um objeto for destruído durante uma interação pode-se denotar o facto com um evento de destruição



5

## Mensagens especiais: found message

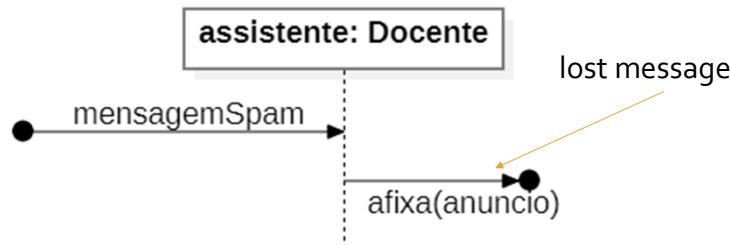
- found message
- mensagem com receptor conhecido e remetente desconhecido
- origem fora do âmbito do diagrama



6

## Mensagens especiais: lost message

- lost message
  - mensagem remetente conhecido e receptor desconhecido



7

## Fragments combinados

- Fragmentos combinados
  - modelar o controlo de fluxo
- Dependendo do operador tem um ou mais operandos
- Cada operando pode conter
  - interações
  - fragmentos combinados
  - referencia para outros diagramas de sequencia



8

## Fragmentos combinados

|                       | operador | propósito               |
|-----------------------|----------|-------------------------|
| ramificações e ciclos | alt      | interações alternativas |
|                       | opt      | interações opcionais    |
|                       | loop     | interações iterativas   |
|                       | break    | interações de exceção   |
| ordem e concurrencia  | seq      | ordem fraca             |
|                       | strict   | ordem estrita           |

9

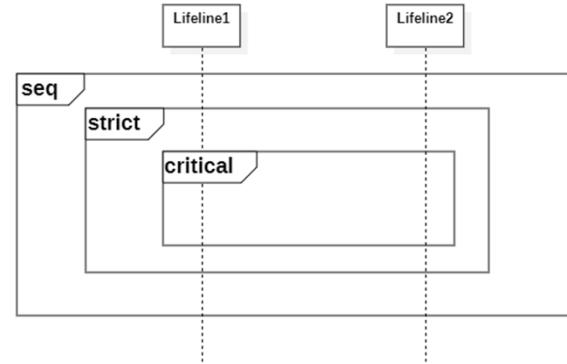
## Fragmentos combinados

|                      | operador | propósito                      |
|----------------------|----------|--------------------------------|
| ordem e concurrencia | par      | interação concurrente          |
|                      | critical | interação atómica              |
| filtros e afirmações | ignore   | parte da interação irrelevante |
|                      | consider | parte da interação relevante   |
|                      | assert   | parte obrigatória              |
|                      | neg      | interação inválida             |

10

## Fragmentos Combinados

- Os fragmentos combinados podem ser colocados uns dentro dos outros
  - um em cada retângulo
  - todos no mesmo retângulo (os vários operadores no mesmo por ordem no pentágono)



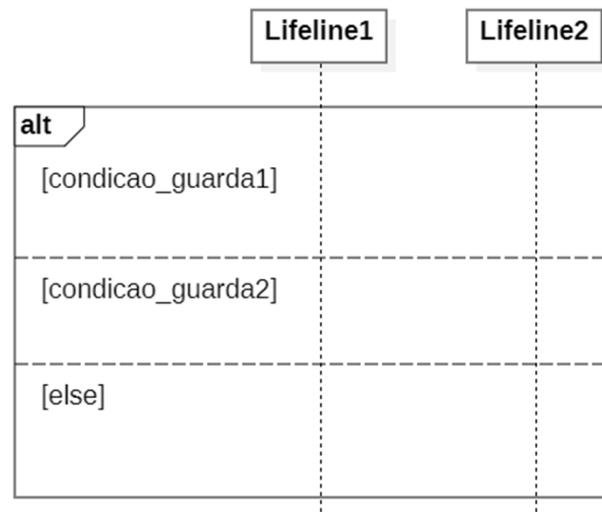
11

## Fragmentos combinados: alt

- alt:
  - especificar interações em alternativa
  - pelo menos dois operandos
  - cada operando tem uma guarda (expressão booleana)
    - se não houver guarda assume-se o valor true
  - apenas uma guarda pode originar o valor true
  - [else] é verdadeira se nenhuma das outras guardas for

12

Fragmentos combinados:  
alt



13

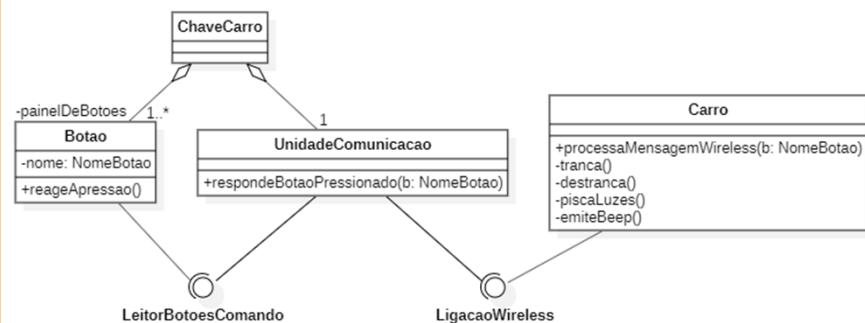
Fragmentos combinados:  
alt



14

## Fragmentos combinados: alt

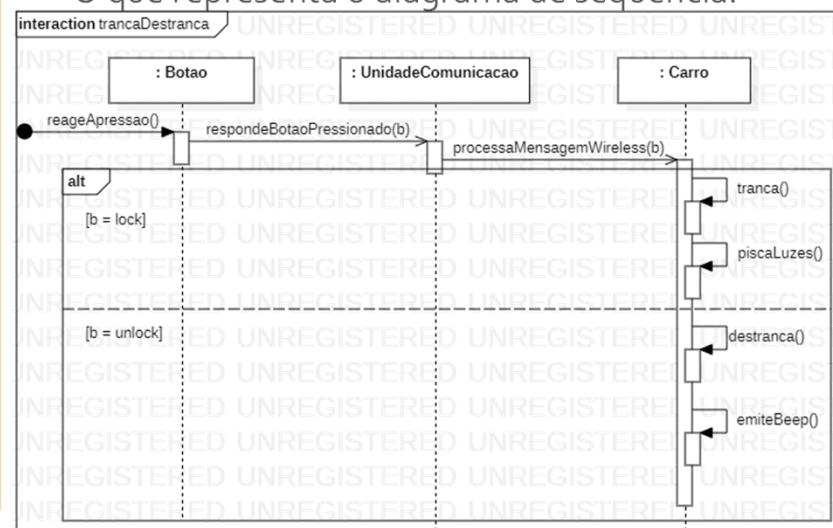
- Dado o seguinte diagrama de classes:



15

## Fragmentos combinados: alt

- O que representa o diagrama de sequência:



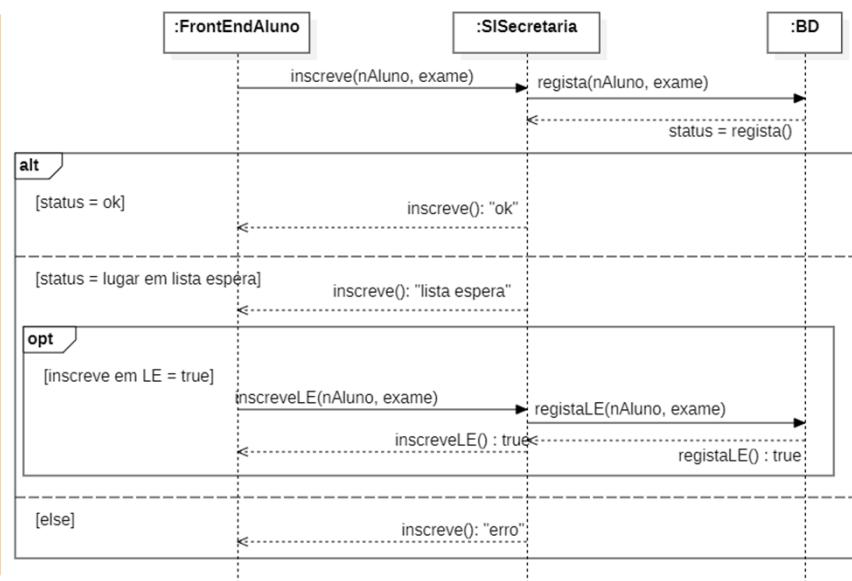
16

## Fragmentos combinados: opt

- fragmento opcional
- a interação só ocorre se a guarda for verdadeira

17

## Fragmentos combinados: opt



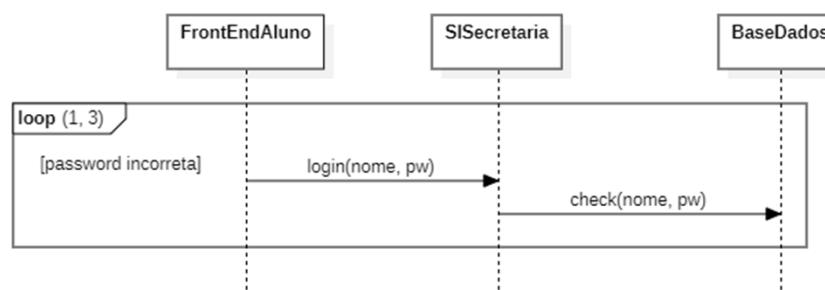
18

## Fragmentos combinados: loop

- loop:
  - o fragmento deve ser executado várias vezes
  - *loop(n)*: n = número de vezes que deve ser repetido
  - *loop(min, max)*: min, max = número mínimo e máximo de vezes que deve ser repetido
  - *loop(\*)* ou *loop*: repetir nº indeterminado de vezes
  - guarda: testada a quando se chega ao número mínimo de vezes
    - se der false sai-se do loop

19

## Fragmentos combinados: loop



20

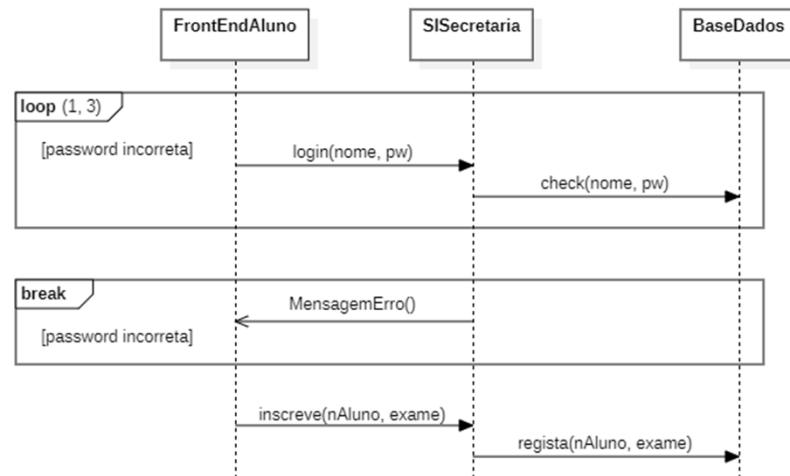
## Fragmentos combinados: break

- break:

- modo de lidar com exceções
- se guarda for verdadeira o fragmento é executado
- restantes operações que rodeiam o fragmento são omitidas
- continua-se no fragmento de nível imediatamente superior

21

## Fragmentos combinados: break



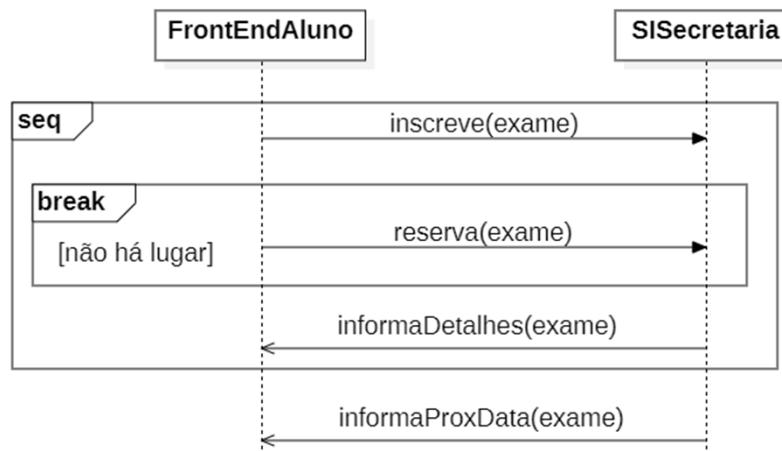
22

## Fragmentos combinados: seq

- seq:
  - representa a ordem por omissão
  - pode-se usar para rodear um break
  - se break ocorrer resto do fragmento seq é ignorado e continua-se logo a seguir

23

## Fragmentos combinados: seq



24

# Modelação e Design 20: Diagrama de Sequencia do Sistema

Leonor Melo

[leonor@isec.pt](mailto:leonor@isec.pt)

# Diagrama de Sequencia do Sistema

- O que é o DSS
- Para que serve o DSS
- Sintaxe e semântica
- Exemplos

# Diagrama de Sequencia do Sistema

- Artefacto que faz parte do Unified Process (UP)
- Ilustra interações entre
  - Sistema
  - Atores
- Usa notação dos Diagramas de Sequencia UML
  - mas exprime visão de alto-nível
  - Sistema visto como uma “caixa preta”

## Diagrama de Sequencia do Sistema

- Ajuda a analisar o comportamento que o sistema deverá exibir em resposta às interações com os atores
- Desenha-se no âmbito de um cenário de um caso de uso
- Reflete a ordem temporal dos eventos nesse cenário

# Diagrama de Sequencia de Sistema

- Mostra, para um cenário de um caso de uso:
  - Eventos gerados pelos atores
  - A ordem pela qual são gerados
  - Eventos inter-sistema
- Desenha-se para
  - o principal cenário de sucesso e
  - cenários alternativos
    - Mais frequentes
    - Mais complexos

# Porquê desenhar Diagrama de Sequencia de Sistema?

- Necessitamos de saber que eventos podem atingir o nosso sistema
  - Para decidir como o sistema deverá lidar com eles
- Eventos podem ser criados por
  - Rato,
  - Teclado,
  - Outros sistemas,...
- o sistema terá de ser capaz de receber esses eventos
  - e executar a resposta apropriada

# Eventos do sistema

- Um sistema de software reage a:
  1. Eventos externos criados por atores (humanos ou outros sistemas)
  2. Eventos temporais
  3. Falhas ou exceções (que podem vir de sistemas externos)

# Eventos do Sistema e Operações do Sistema

- Evento do sistema
  - Criados por um ator ao interagir com o sistema
- Operação do Sistema
  - Resposta do sistema a determinado evento
- Exemplo: Sistema POS
  - Operador introduz o código de um item
    - evento do sistema
  - Em resposta, o sistema POS regista a venda desse item
    - operação do sistema

## Sistema como uma caixa-preta

- Diagrama de sequencia de sistema
  - comportamento do sistema enquanto “caixa-preta”
- Comportamento do Sistema
  - O que é que o sistema faz
    - Sem explicar como o faz
  - Expresso por:
    - casos de uso
    - diagrama de sequencia do sistema

# Diagrama de Sequencia

- O UML não define o que é um Diagrama de Sequencia do Sistema
- Apenas define Diagrama de Sequencia que pode ser visto:
  - ao nível do sistema
    - Diagrama de sequencia do sistema
  - ou ao nível dos objetos
    - Diagrama de sequencia
  - ou ainda níveis intermédios

# Exemplo de Diagrama de Sequencia

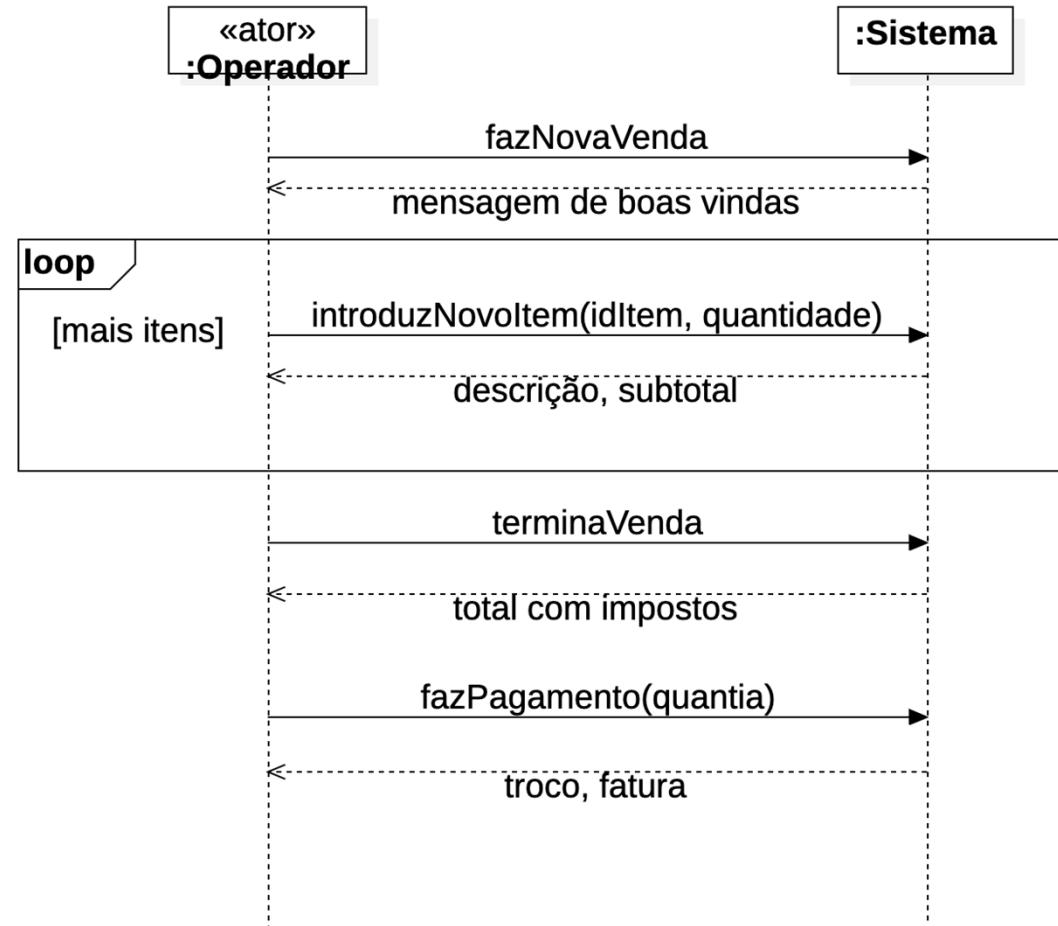
Cenário de sucesso principal do caso de uso de uma venda a dinheiro do caso de uso Processa venda:

1. Cliente aproxima-se da caixa com os produtos e/ou serviços que quer comprar
2. Operador começa uma nova venda
3. Sistema mostra mensagem de boas vindas
4. Operador introduz identificador do item
5. O sistema regista a linha de venda e apresenta a descrição do item, preço e subtotal.  
O operador repete os passos 3-4 até indicar que terminou

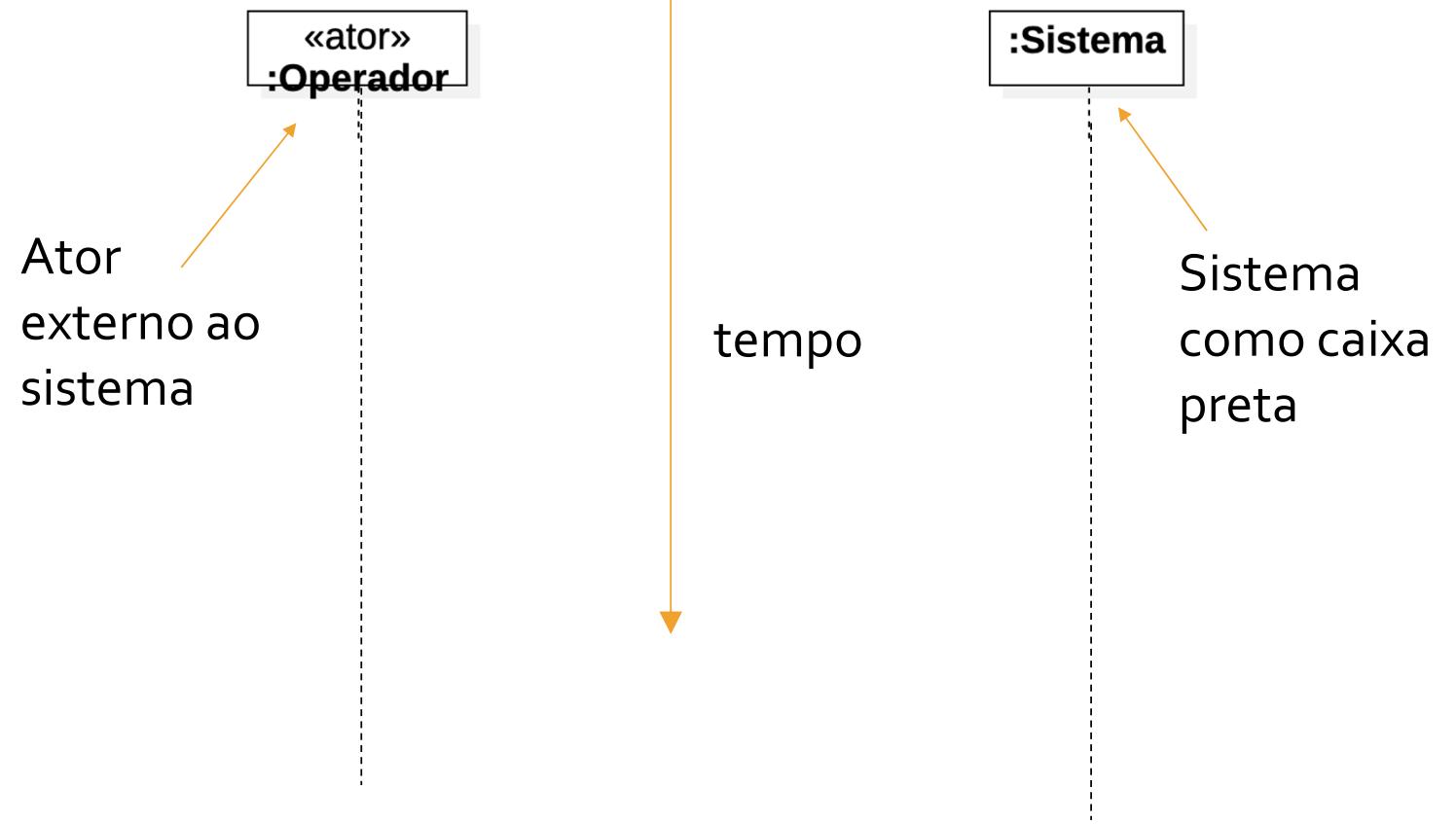
## Exemplo de Diagrama de Sequencia

5. Sistema apresenta o total com os impostos calculados
6. Operador comunica o total ao cliente e solicita o pagamento
7. O cliente paga, o operador indica que o pagamento foi feito e deposita o dinheiro na gaveta
8. O sistema indica o troco a haver e imprime a fatura
9. O operador entrega o troco e a fatura ao cliente

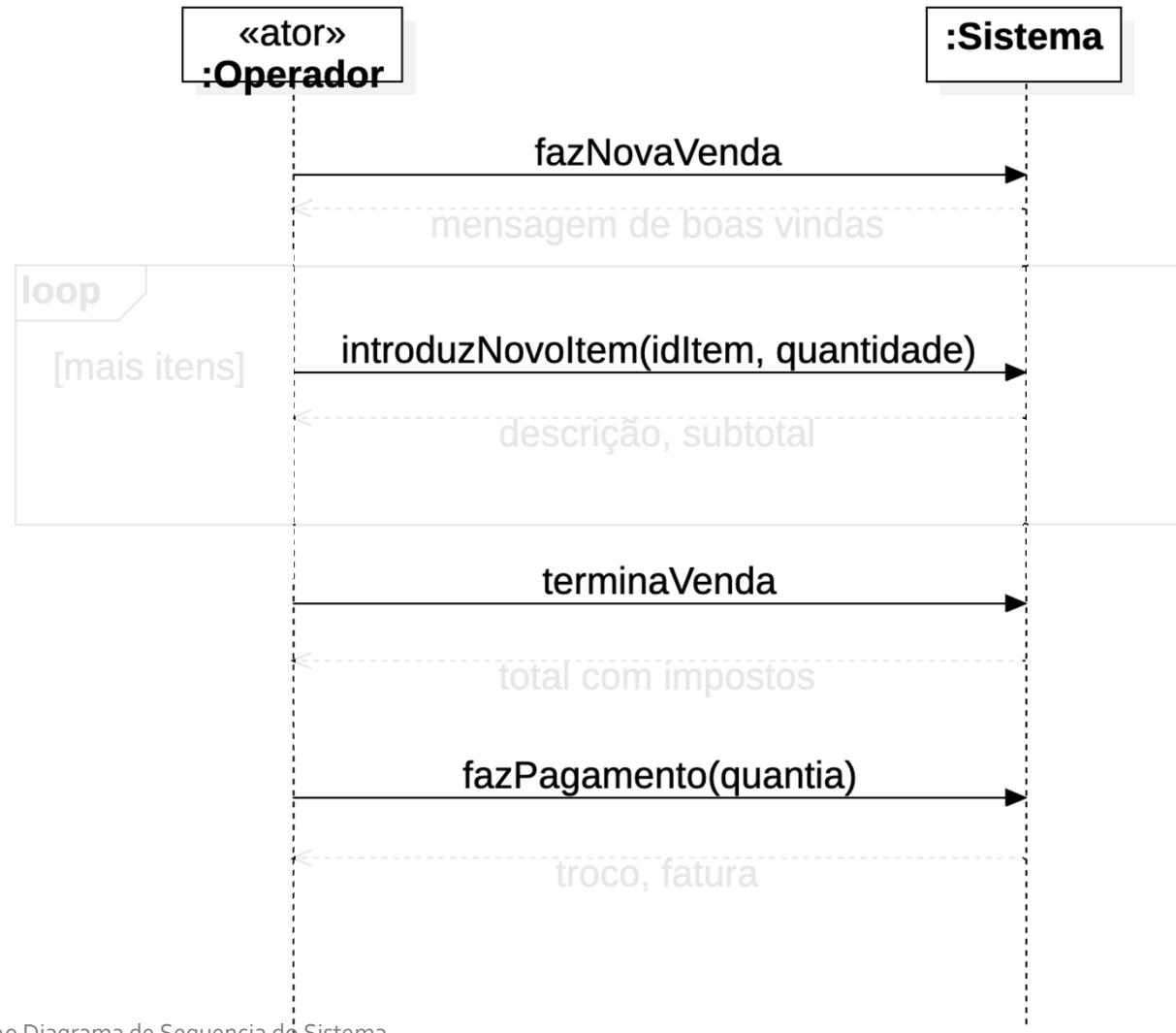
# Exemplo de Diagrama de Sequencia



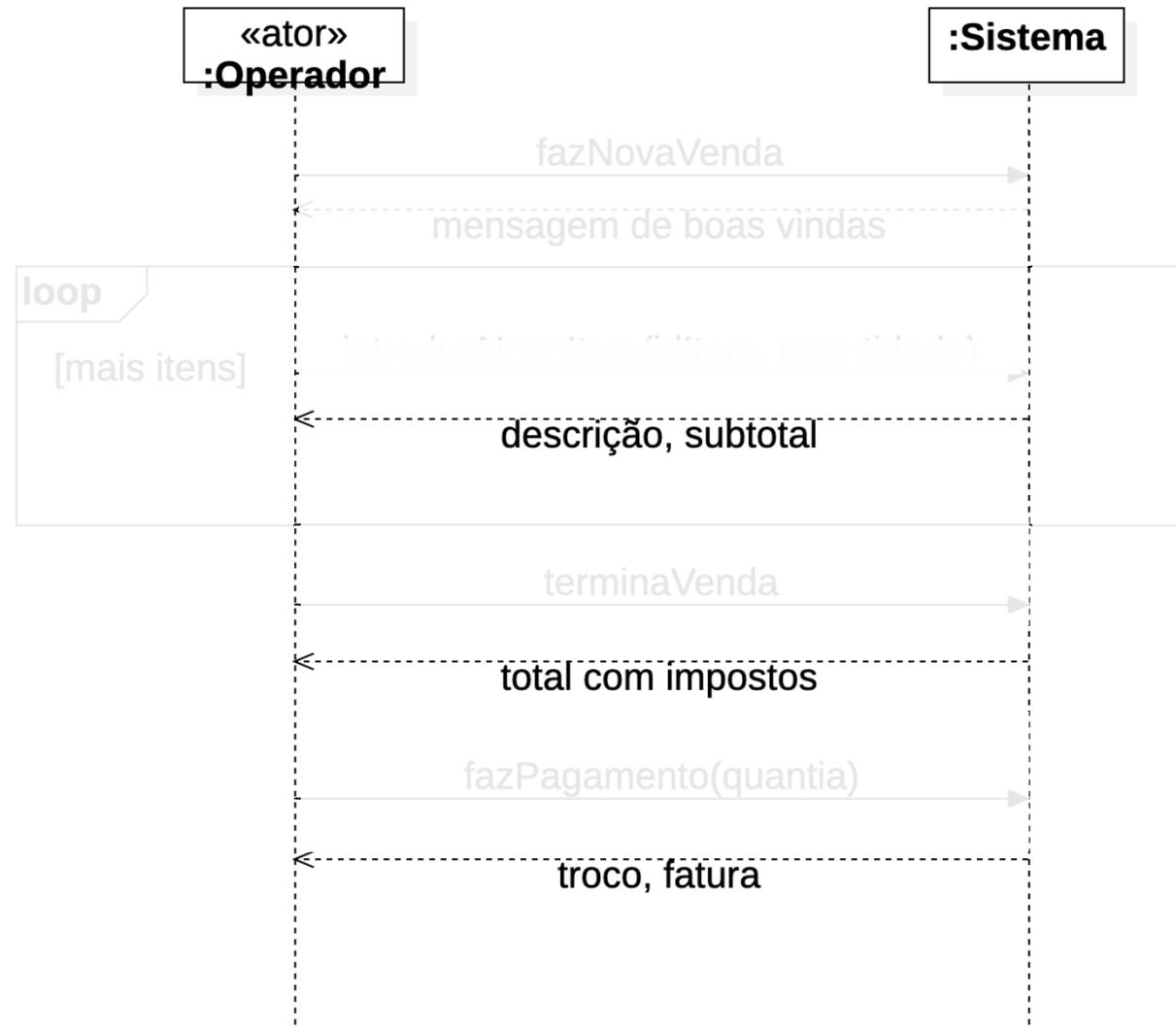
# Exemplo de Diagrama de Sequencia



# Eventos gerados pelo operador



Valores devolvidos associados com as mensagens anteriores



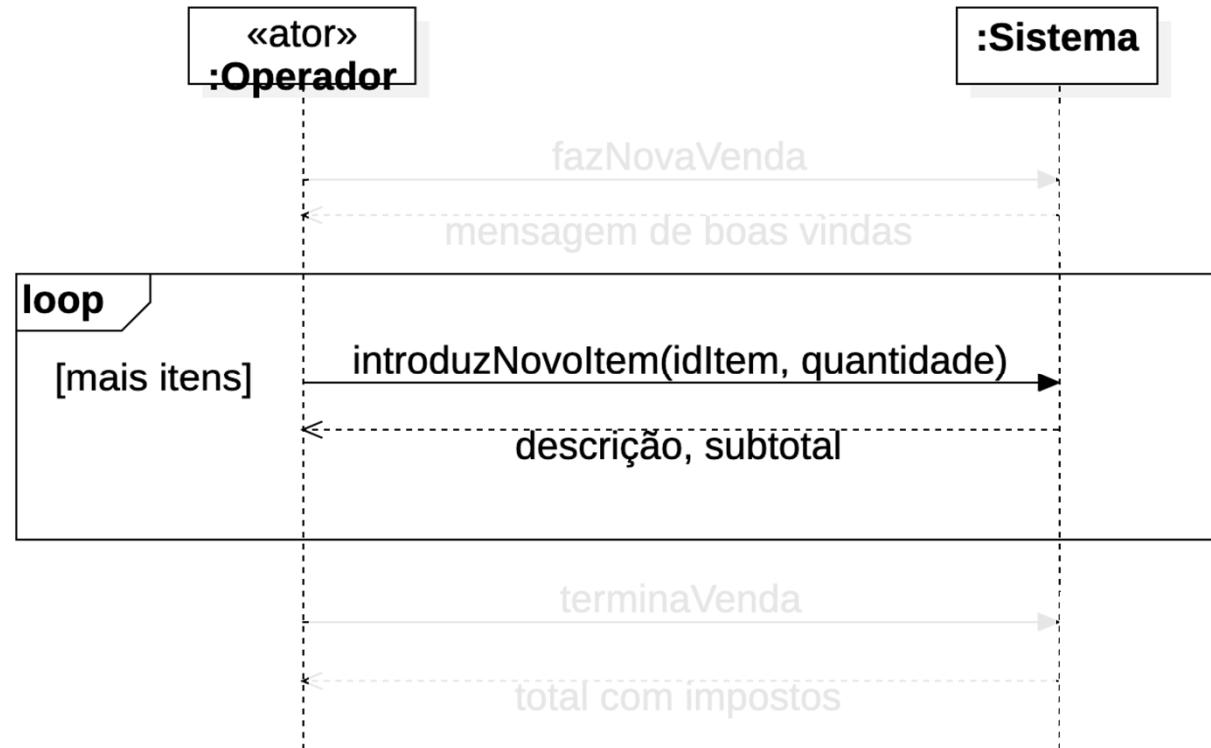
# Mensagens

- Os eventos gerados pelo utilizador são representados como mensagens dirigidas ao sistema
- As mensagens podem ter parâmetros
- Também são usadas mensagens de resposta para indicar os "dados devolvidos"
- Se não existirem valores devolvidos
  - a linha de resposta pode ser omitida

# Mensagens

- As mensagens são abstrações
  - ignoram a apresentação
  - e o mecanismo pelo qual são transmitidas
- fazPagamento(quantia)
  - Representa o evento do sistema em que os dados relativos ao pagamento foram introduzidos no sistema por um qualquer mecanismo

# Elementos adicionais



- Todos os elementos dos DS podem ser usados nos DSS

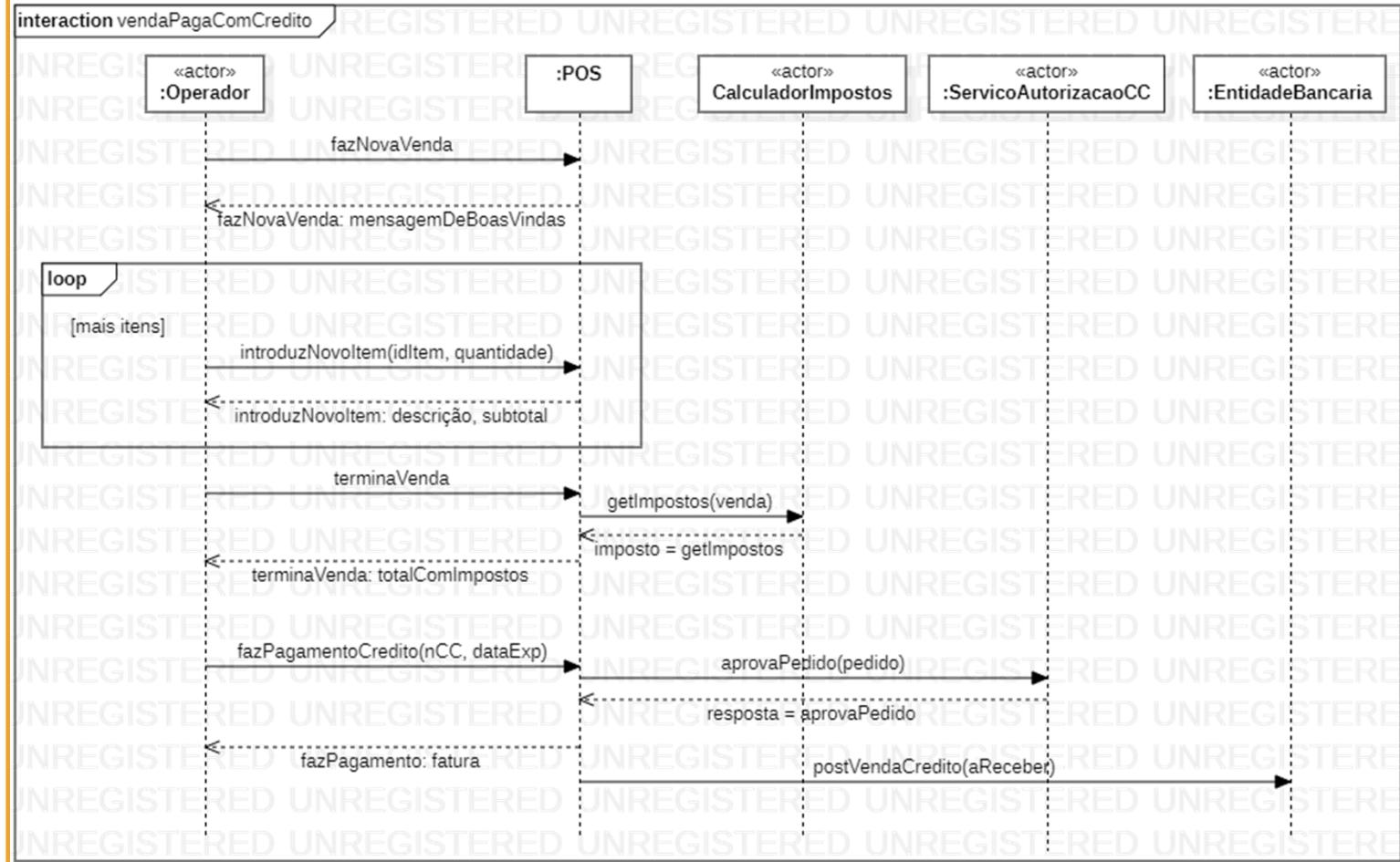
## Boas prática

- Nomes dos eventos do sistema
  - Devem começar por um verbo
    - Enfatizar que são ordens ou pedidos que se estão a fazer ao sistema
  - Devem capturar a intenção da operação mas preservar o nível de abstração
    - Melhor:
      - introduzNovoItem(idItem, quantidade)
    - Pior:
      - fazScan (idItem, quantidade)

# Interação com outros sistemas

- Vamos refazer o Caso de uso do Point of Sale mas no cenário de
  - compra feita usando cartão de crédito e
  - recorrendo a um sistema externo de cálculo de impostos

# Interação com outros sistemas



# Glossário

- Os elementos do DSS (nome operações, parâmetros, valores devolvidos) são concisos
  - podem ser necessárias explicações adicionais
    - necessárias para a fase de design da solução
  - Esses detalhes devem aparecer no glossário

## Glossário: exemplo

**Fatura:** impressa em papel, usada para IRS e devoluções.  
inclui informação sobre :  
    nome, morada e NIF da empresa;  
    nº fatura, data de emissão;  
    nome e morada da loja;  
    nº transação, data limite de devolução, nome do  
operador;  
    Para cada item: quantidade, nome, valor;  
    Total a pagar, total entregue, troco;  
    Modo de pagamento, cartão  
    Para cada escalão de IVA: Taxa de IVA, valor de  
incidência, total de IVA pago  
    Nome e NIF do comprador

## Exemplo 1: Cenário principal de sucesso de cria conta de blog

1. O Administrador pede ao sistema para criar uma nova conta de blog
2. O Administrador seleciona o tipo de conta de blog normal
3. O Administrador introduz os detalhes do autor (nome e endereço de mail)
4. Os detalhes do autor são verificados usando a Base de Dados de credenciais dos autores
5. A nova conta de blog é criada
6. Um sumário dos detalhes da nova conta de blog (username e password de acesso) é enviado por mail para o autor

## Exemplo 2: Iniciar chamada telefónica entre telefones fixos

- Cenário principal de sucesso de uma chamada estabelecida entre dois telefones fixos
  - Emissor = pessoa que telefona
  - Recetor = pessoa que recebe o telefonema
1. O Emissor levanta o auscultador e o sistema emite um sinal de marcar
  2. O Emissor introduz o primeiro dígito e o sistema termina cessa o sinal de marcar
  3. O Emissor introduz os restantes dígitos

## Exemplo 2: Iniciar chamada telefónica entre telefones fixos (cont.)

5. O sistema emite sinal de chamada para o emissor e faz tocar o telefone no recetor
4. O Recetor levanta o auscultador e o sistema cessa de fazer tocar o telefone do recetor e deixa de emitir sinal de marcar para o Emissor e as pessoas passam a poder falar uma com a outra

## Exemplo 3: Terminar chamada telefónica entre telefones fixos

- Terminador = pessoa que termina telefone
- Pessoa = pessoa que estava ao telefone com o terminador
  - 1. O Terminador baixa o auscultador
  - 2. Sistema emite sinal de marcar para o telefone da outra pessoa. Pessoas deixam de poder falar entre si.

# Modelação e Design

## 21: Princípios de Design Orientado a Objetos

Leonor Melo

leonor@isec.pt

1

## Design Orientado a Objetos

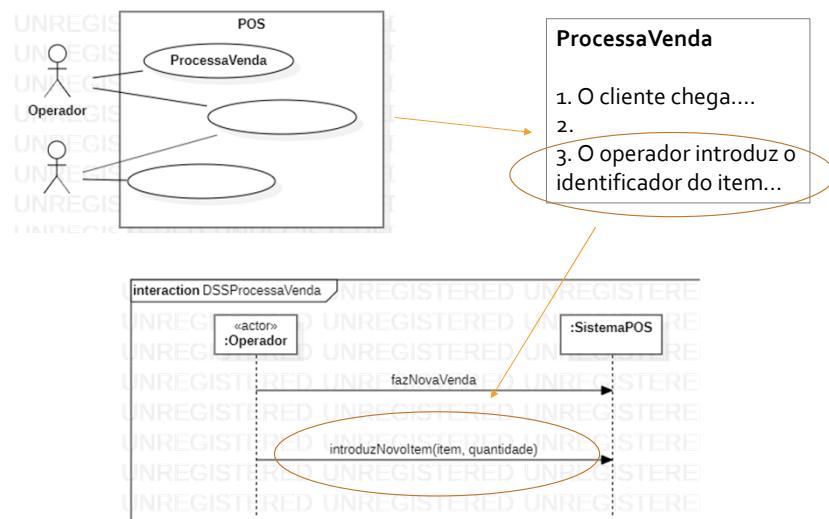
- Responsabilidades
- Princípios GRASP
  - Criador
  - Perito em informação
  - Acoplamento baixo
  - Controlador
  - Coesão elevada
  - Polimorfismo
  - Invenção
  - Variações protegidas

Leonor Melo

21 Design Orientado a Objetos

2

## Onde estamos



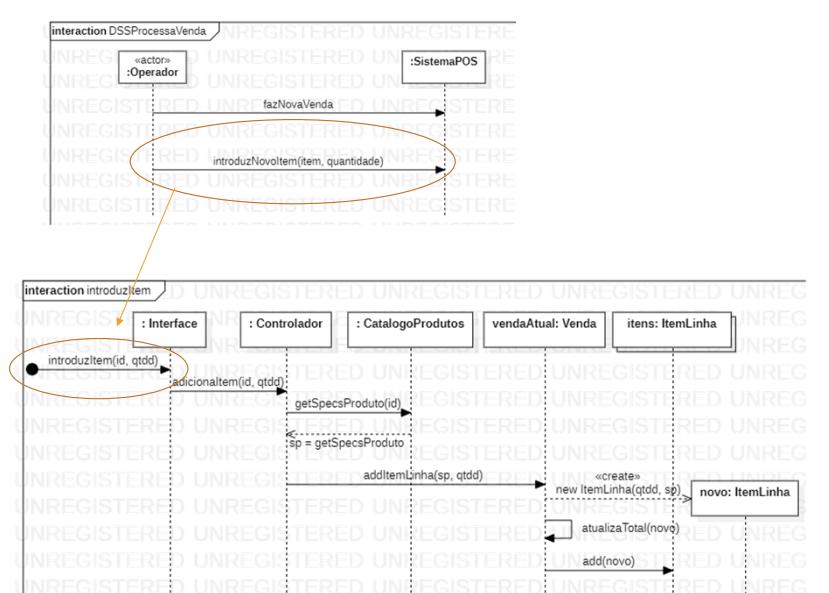
Leonor Melo

21 Design Orientado a Objetos

3

3

## Onde estamos



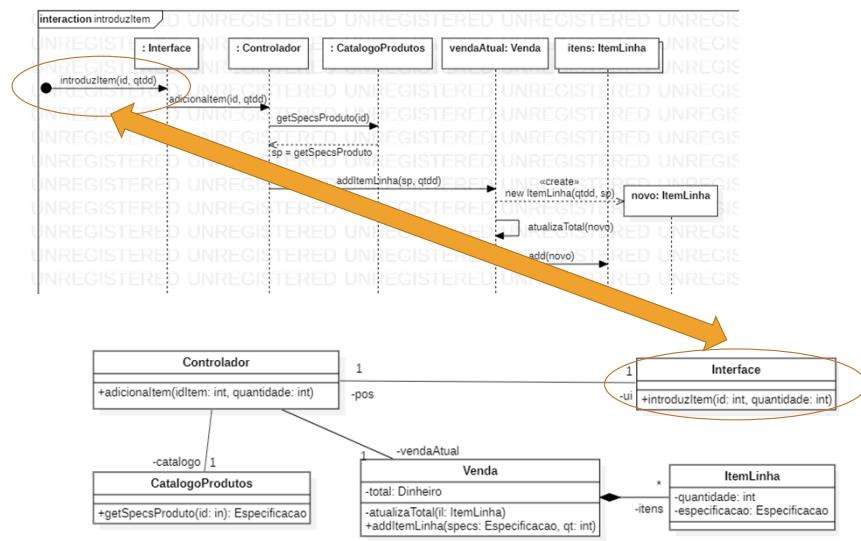
Leonor Melo

21 Design Orientado a Objetos

4

4

## Onde estamos



Leonor Melo

21 Design Orientado a Objetos

5

## Princípios de design

- Estratégias que ajudam a criar um design
  - Limpo
  - Modular
  - Fácil de testar
  - Fácil de corrigir
  - Fácil de manter

Leonor Melo

21 Design Orientado a Objetos

6

## Design guiado por responsabilidades

- Objetos / classe / componentes são vistos como entidades com
  - Responsabilidades
  - Papel a desempenhar
  - Colaborações
- Responsabilidade
  - “contrato ou obrigação de um classificador”
- GRASP – princípios que auxiliam na atribuição das responsabilidades

Leonor Melo

21 Design Orientado a Objetos

7

## Responsabilidades

- Um objeto tem 2 tipos de responsabilidades
  - Responsabilidades de fazer
    - Fazer alguma coisa ele próprio como criar um objeto ou fazer um cálculo
    - Iniciar uma ação em outro objeto
    - Controlar e coordenar atividades com outros objetos
  - Responsabilidades de conhecer
    - Conhecer dados privados e encapsulados
    - Conhecer objetos com os quais tem uma relação
    - Conhecer dados que pode derivar ou calcular

Leonor Melo

21 Design Orientado a Objetos

8

## Responsabilida des

- Objeto Venda
  - Responsabilidades de fazer
    - Uma Venda é responsável por criar um ItemDeLinha
  - Responsabilidades de conhecer
    - Uma Venda é responsável por conhecer o total
- Modelo do domínio muitas vezes inspira as “responsabilidades de conhecer” dos objetos que também pertençam ao modelo de design

Leonor Melo

21 Design Orientado a Objetos

9

9

## Colaborações

- Responsabilidades são implementadas através de métodos que
  - agem sozinhos, ou
  - **colaboram** com outros métodos e objetos
- Responsibility Driven Design
  - Metáfora
  - Sistema é visto como uma comunidade de objetos com responsabilidades que colaboram entre si

Leonor Melo

21 Design Orientado a Objetos

10

10

## Princípios do GRASP

- Criador (Creator)
- Perito (Information Expert)
- Acoplamento Baixo (Low Coupling)
- Controlador (Controller)
- Coesão Elevada (High Cohesion)
- Polimorfismo (Polymorphism)
- Invenção (Pure Fabrication)
- Indireção (Indirection)
- Variações protegidas (Protected Variations)

Leonor Melo

21 Design Orientado a Objetos

11

11

## Criador

- Perguntas a que quer responder:
  - Quem cria um objeto?
  - Quem deveria criar instâncias de determinada classe?
- Resposta:
  - Atribuir à classe B a responsabilidade de criar instâncias da classe A se
    - B “contém” ou “é composta por” A
    - B regista A
    - B usa A com frequência / proximidade
    - B tem a informação necessária para inicializar A

Leonor Melo

21 Design Orientado a Objetos

12

12

## Criador

- Exemplo:
  - Tabuleiro e Casa
  - Tabuleiro tem uma relação de composição com a Casa
  - Tabuleiro é o “todo” e Casa é a “parte”
    - As instâncias de Casa devem ser criadas pela classe Tabuleiro



Leonor Melo

21 Design Orientado a Objetos

13

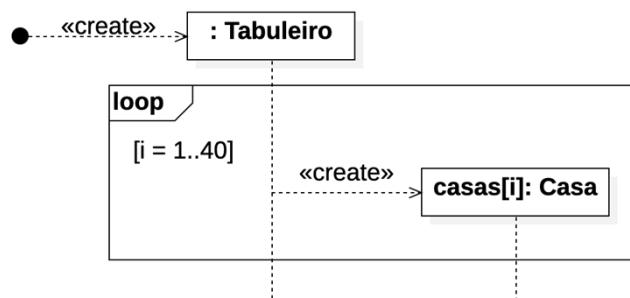
13

## Criador

- Diagrama de Classes:



- Diagrama de sequencia:



Leonor Melo

21 Design Orientado a Objetos

14

14

## Perito

- Perguntas a que quer responder:
  - Qual o princípio básico para atribuir uma responsabilidade a um objeto?
- Resposta:
  - Atribuir a responsabilidade à classe que tem a informação necessária para executar a operação (sozinha ou em colaboração com outras classes)

Leonor Melo

21 Design Orientado a Objetos

15

15

## Perito

- Exemplo:
  - Qual a classe que deve ser responsável por, dado o nome de uma casa, devolver a referência para essa casa?
    - A classe Tabuleiro, que conhece todas as casas



Leonor Melo

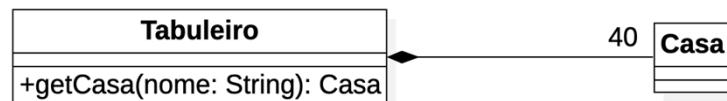
21 Design Orientado a Objetos

16

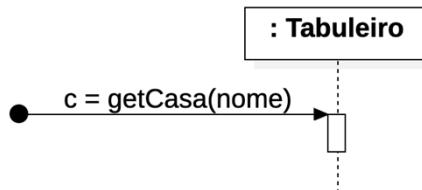
16

## Perito

- Diagrama de Classe:



- Diagrama de Sequencia:



Leonor Melo

21 Design Orientado a Objetos

17

17

## Acoplamento Baixo

- Perguntas a que quer responder:

- Como reduzir o impacto de alterações?

- Resposta:

- Atribuir a responsabilidade de forma a que acoplamentos desnecessários permaneçam baixos.
  - Usar esse princípio para avaliar alternativas

Leonor Melo

21 Design Orientado a Objetos

18

18

## Acoplamento Baixo

- Exemplo de acoplamento desnecessariamente elevado:

- Tornar a classe Peão (ou outra qualquer) responsável por, dado o nome de uma casa, devolver a referência para essa casa



Leonor Melo

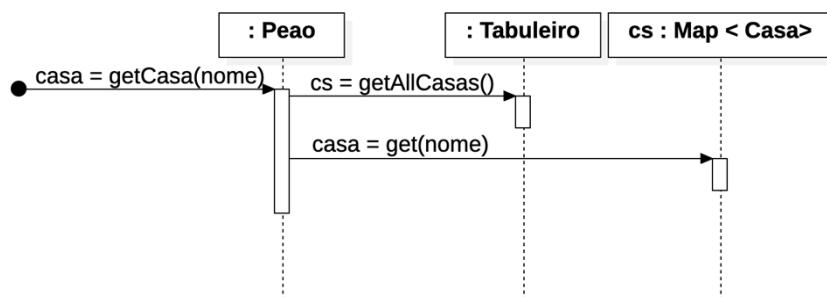
21 Design Orientado a Objetos

19

19

## Acoplamento Baixo

- Diagrama de Sequencia do exemplo de acoplamento desnecessariamente elevado:



Leonor Melo

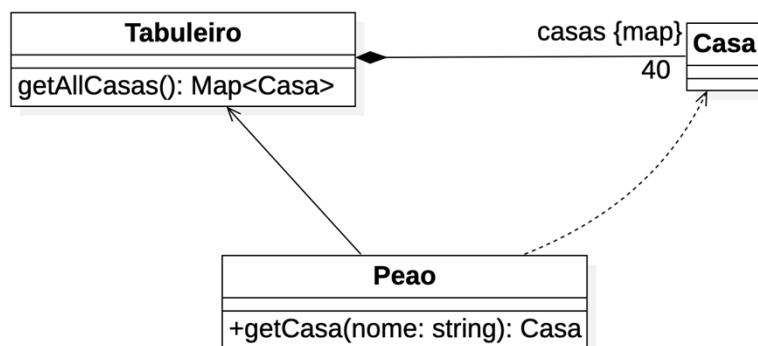
21 Design Orientado a Objetos

20

20

## Acoplamento Baixo

- Diagrama de Classe do exemplo de acoplamento desnecessariamente elevado:



Leonor Melo

21 Design Orientado a Objetos

21

21

## Acoplamento Baixo

- Quando é necessário alterar o software, o baixo acoplamento tende a reduzir
  - Tempo necessário
  - Esforço despendido
  - Defeitos introduzidos

Leonor Melo

21 Design Orientado a Objetos

22

22

## Controlador

- Uma arquitetura simples tem pelo menos uma camada de interface com o utilizador (UI layer) e uma camada de domínio/aplicação/negócio
- Os objetos da camada de UI (botões, caixas de texto,...) recebem eventos do utilizador, mas não devem ser responsáveis pela lógica do domínio
  - Devem delegar o pedido para os objetos do domínio

Leonor Melo

21 Design Orientado a Objetos

23

23

## Controlador

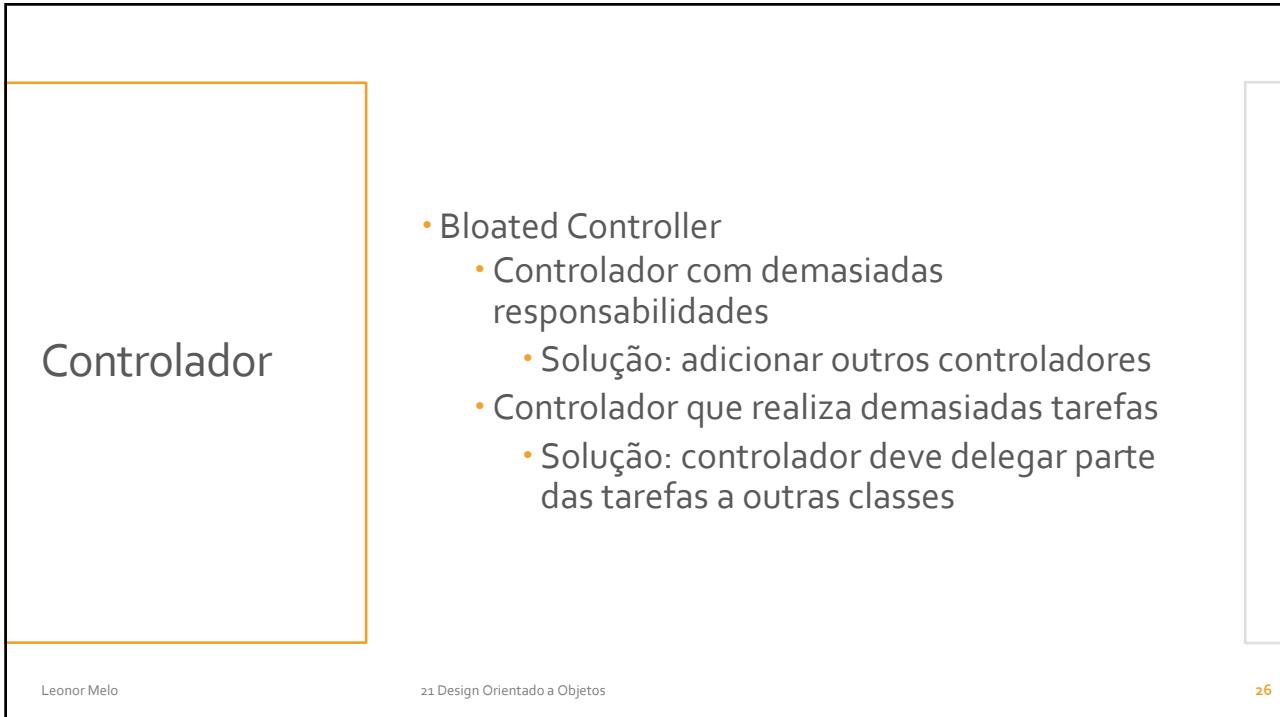
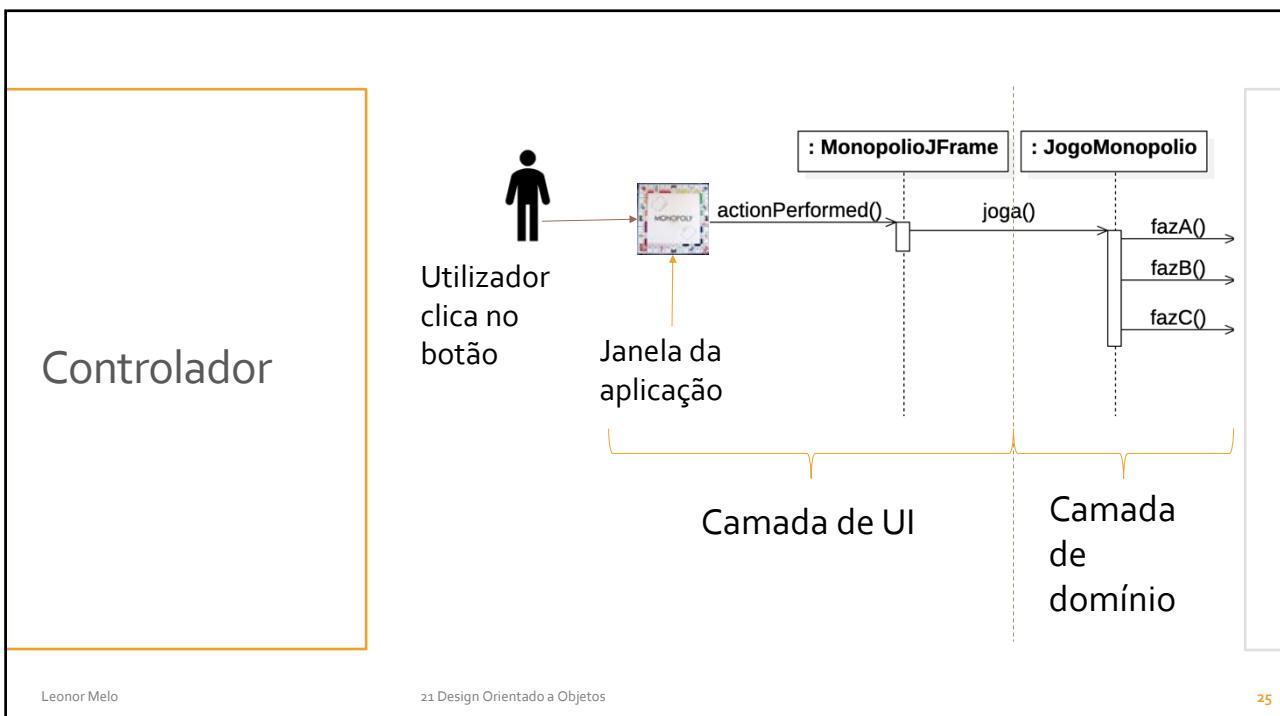
- Pergunta:
  - Qual é o primeiro objeto, depois da camada de interface com o utilizador (UI), que deve receber a mensagem da camada de UI
- Resposta:
  - Atribuir responsabilidade a um objeto que represente uma das seguintes escolhas:
    - Representa o “sistema” completo,
    - Representa o dispositivo onde o sistema se está a executar ou um subsistema
    - Representa um caso de uso dentro do qual a operação ocorre

Leonor Melo

21 Design Orientado a Objetos

24

24



## Coesão Elevada

- Pergunta:
  - Como manter os objetos focados, compreensíveis, manejáveis e como consequência com acoplamento baixo?
- Resposta:
  - Atribuir responsabilidades de forma que a coesão permaneça elevada.
  - Usar a coesão como critério entre soluções alternativas

Leonor Melo

21 Design Orientado a Objetos

27

27

## Coesão Elevada

- Coesão:
  - Mede quão funcionalmente relacionadas estão as operações de um elemento de software; quanto trabalho está a ser realizado por um dado elemento
- Baixa coesão (mau):
  - Um elemento faz muitas atividades não relacionadas entre si
  - Um elemento é responsável por uma parte grande do trabalho
- Baixa coesão tende a originar um acoplamento elevado

Leonor Melo

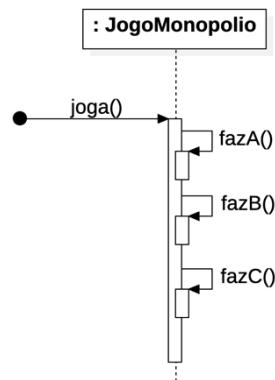
21 Design Orientado a Objetos

28

28

## Coesão Elevada

- Diagrama de Sequencia de um exemplo de coesão baixa (pior):



Leonor Melo

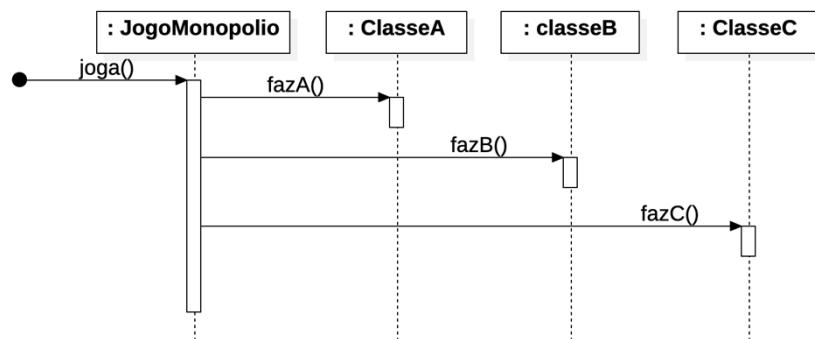
21 Design Orientado a Objetos

29

29

## Coesão Elevada

- Diagrama de Sequencia de um exemplo de coesão alta (melhor):



Leonor Melo

21 Design Orientado a Objetos

30

30

## Polimorfismo

- Problema:
  - Como lidar com alternativas baseadas no tipo dos objetos? Como criar componentes que possam ser facilmente substituídos sem danos para o sistema?
- Solução:
  - Usar operações polimórficas (i.e. com a mesma assinatura, mas com implementações distintas consoante o tipo de objeto) – os diferentes objetos devem estar relacionados entre si pela implementação do mesmo interface ou por relações de generalização

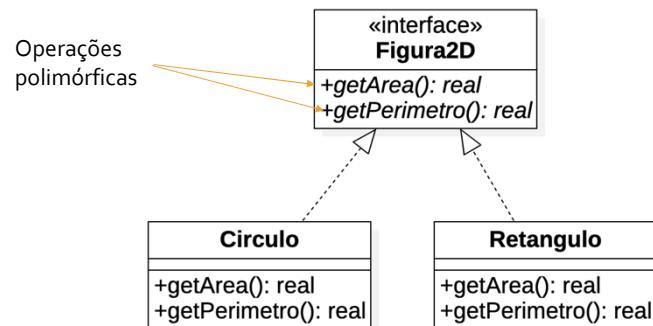
Leonor Melo

21 Design Orientado a Objetos

31

31

## Polimorfismo



- Se enviarmos uma mensagem `getArea()` a um `objeto2D`, a implementação correta vai ser automaticamente usada de acordo com o tipo do objeto.

Leonor Melo

21 Design Orientado a Objetos

32

32

## Invenção

- Problema:
  - Que objeto deve ter determinada responsabilidade quando não queremos violar os princípios de coesão elevada e acoplamento fraco, mas a solução sugerida pelo “perito em informação” não é boa
- Solução:
  - atribuir um conjunto muito coeso de responsabilidades a uma classe de conveniência artificial, mesmo que não represente nenhuma entidade do domínio, de forma a suportar coesão elevada, acoplamento fraco e reutilização

Leonor Melo

21 Design Orientado a Objetos

33

33

## Invenção

- Exemplo:
  - Que objeto deve ser responsável por guarda a informação de uma Venda na base de dados?
  - A informação pertence à Venda pelo que o perito em informação é a Venda
  - Mas guardar informação na base de dados implica uma série de operações relacionadas apenas com a base de dados
  - Implica também relacionamento com determinado interface de base de dados
    - Acrescentar estas operações diminuir a coesão e aumentar o acoplamento da Venda

Leonor Melo

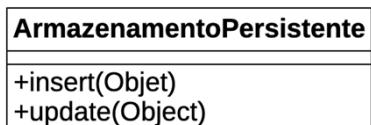
21 Design Orientado a Objetos

34

34

## Invenção

- Exemplo:
  - É melhor criar uma classe unicamente responsável por gravar a informação na base de dados, que eventualmente poderá ser reutilizada por várias outras classes



Leonor Melo

21 Design Orientado a Objetos

35

35

## Indireção

- Problema
  - Como atribuir responsabilidades de forma a evitar acoplamento direto entre duas (ou mais) classes? Como desacoplar objetos de forma que o acoplamento permaneça baixo e o potencial de reutilização continue elevado?
- Solução:
  - Atribuir a responsabilidade a um objeto intermédio que irá mediar as relações entre os outros componentes

Leonor Melo

21 Design Orientado a Objetos

36

36

## Invenção

- Exemplo:
  - O ArmazenamentoPersistente é também um exemplo de indireção, ao servir de intermediário entre a Venda e a Base de Dados



Leonor Melo

21 Design Orientado a Objetos

37

37

## Variações Protegidas

- Problema
  - Como desenhar objetos, subsistemas e sistemas de forma que as variações ou instabilidade nestes elementos não tenham um impacto negativo nos outros elementos?
- Solução:
  - Identificar os pontos de previsível instabilidade e atribuir responsabilidades de forma a criar um interface estável

Leonor Melo

21 Design Orientado a Objetos

38

38

## Variações Protegidas

- Exemplo:

- A criação do interface para aplicação do polimorfismo no exemplo das figuras geométricas:
  - Mais tarde podem ser acrescentadas outras figuras (Triangulo, Pentágono, ...): se estas novas classes também implementarem o interface Figura2D, o restante software, que já depende do Figura2D, não tem de ser alterado.

Leonor Melo

21 Design Orientado a Objetos

39

# Modelação e Design 22: Diagrama de Comunicação

Leonor Melo

leonor@isec.pt

1

## Diagrama de Comunicação

- O que é e para que serve um diagrama de comunicação
- mensagens aninhadas
- mensagens simultâneas
- mensagens para o próprio
- exemplos de conversão dos elementos de um diagrama de sequência em elementos de diagrama de comunicação

Leonor Melo

22 Diagrama de Comunicação

2

## Diagrama de Comunicação

- Diagrama de comunicação
  - É um diagrama de interação
  - Semanticamente equivalente ao diagrama de sequencia
- Diagrama de sequencia
  - Enfase na ordem das mensagens
- Diagrama de comunicação
  - Enfase na ligação entre os participantes

Leonor Melo

22 Diagrama de Comunicação

3

## Diagrama de Comunicação

- Diagrama de objetos
  - Objetos
  - Relações entre objetos
- Diagrama de Comunicação
  - Extensão do diagrama de objetos
    - Ligações entre os objetos
    - Mensagens trocadas entre os objetos
- Diagrama de objetos
  - Fotografia (instancias num instante)
- Diagrama de comunicação
  - Filme (instancias e interações que ocorrem ao longo do tempo)

Leonor Melo

22 Diagrama de Comunicação

4

## Diagrama de comunicação

- Diagrama de comunicação é composto por
  - Participantes
  - Links de comunicação
    - Um par de participantes que não esteja ligado por um link não pode comunicar
  - Mensagens

participante1: ClasseParticipanteA

participante2: ClasseParticipanteB

Leonor Melo

22 Diagrama de Comunicação

5

## Diagrama de comunicação

- Mensagens
  - Ordem de invocação
  - Assinatura da mensagem (mesma sintaxe que diagrama de sequencia)
  - Seta de sentido (direção do link)

participante1: ClasseParticipanteA

1 : mensagemA()

2 : mensagemB()

participante2: ClasseParticipanteB

Leonor Melo

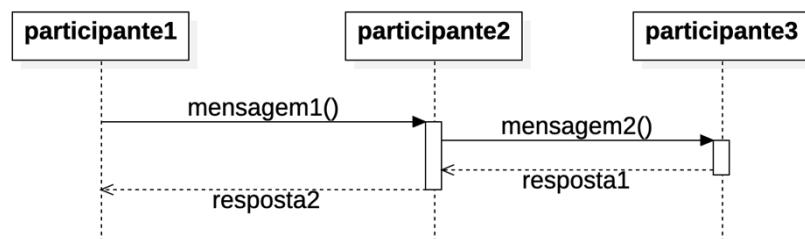
22 Diagrama de Comunicação

6

6

## Mensagens aninhadas (*nested*)

- Mensagens aninhadas
  - Quando a receção de uma mensagem provoca o envio de outra mensagem



Leonor Melo

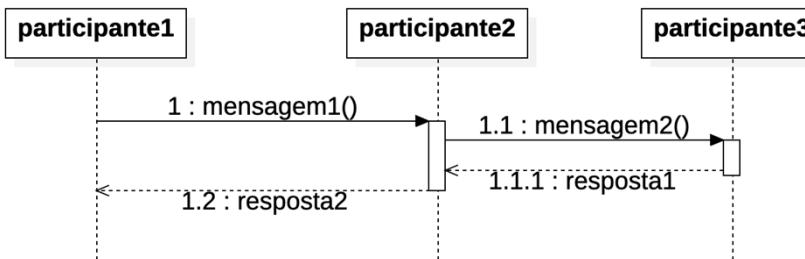
22 Diagrama de Comunicação

7

7

## Mensagens aninhadas: numeração

- Se mensagem original for 1 as mensagens originadas em consequência dessa serão, 1.1, 1.2, ...



Leonor Melo

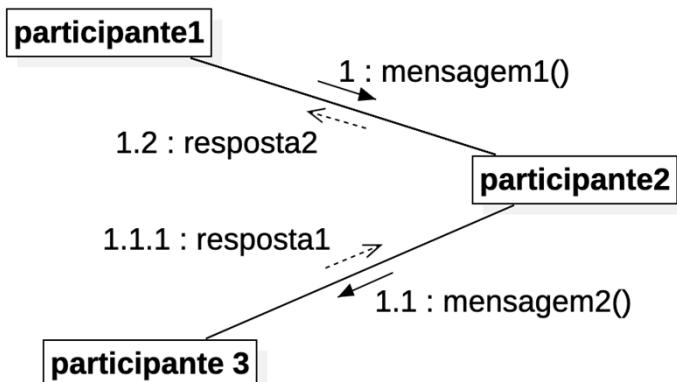
22 Diagrama de Comunicação

8

8

## Mensagens aninhadas: diagrama de comunicação

- Diagrama de comunicação equivalente:



Leonor Melo

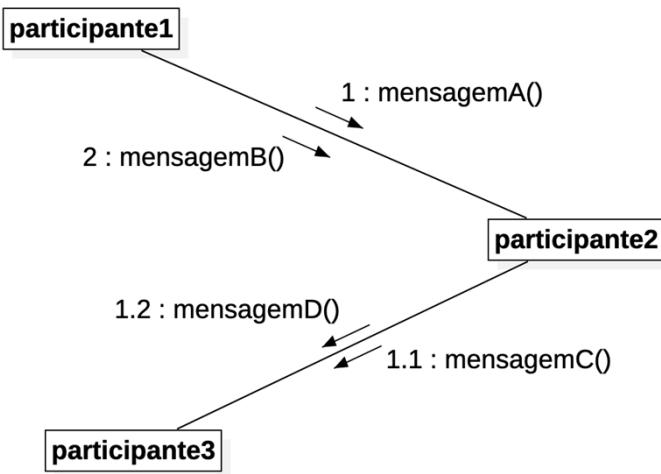
22 Diagrama de Comunicação

9

9

## Mensagens aninhadas: diagrama de comunicação

- Como se lê o seguinte diagrama:



Leonor Melo

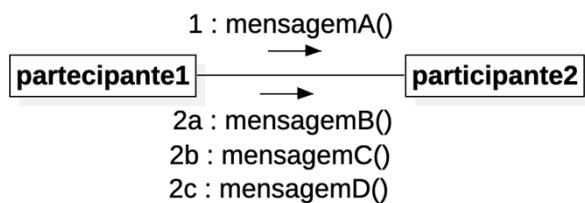
22 Diagrama de Comunicação

10

10

## Mensagens que ocorrem em simultâneo

- Diagramas de sequencia
  - Fragmento combinado par
- Diagrama de comunicação
  - Numeração com notação número-e-letra



Leonor Melo

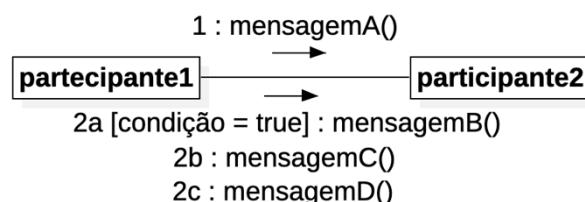
22 Diagrama de Comunicação

11

11

## Envio de mensagens baseado em condições

- Diagramas de sequencia
  - Fragmento opt
- Diagramas de comunicação
  - condição de guarda



Leonor Melo

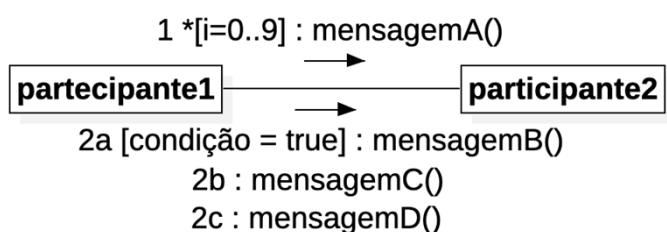
22 Diagrama de Comunicação

12

12

## Envio múltiplo de mensagens

- Diagramas de sequencia
  - Fragmento loop
- Diagramas de comunicação
  - \*condição de guarda



Leonor Melo

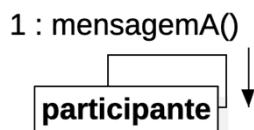
22 Diagrama de Comunicação

13

13

## Mensagem para o próprio participante

- Mensagem para o próprio
  - Representa a evocação de um método pelo próprio participante
  - Necessário um link do participante para ele próprio



Leonor Melo

22 Diagrama de Comunicação

14

14

## Exemplo: criação de conta de blog

1. Administrador pede ao sistema para criar nova conta de blog
2. Administrador escolhe conta de blog normal
3. Administrador introduz detalhes do autor
4. Detalhes do autor são verificados usando a base de dados de credenciais de autores
5. A nova conta de blog normal é criada
6. Um sumário dos detalhes da nova conta de blog são enviados por mail ao autor

Leonor Melo

22 Diagrama de Comunicação

15

15

## Exemplo: criação de conta de blog

- Classes
  - <<ator>> Administrador
  - UlcriaConta
    - Classe responsável pelo Interface com o utilizador para a funcionalidade de criação de uma conta
  - ControladorCriaNovaConta
    - Classe responsável pela criação da conta propriamente dita
  - <<ator>> Bdcredenciais
  - <<ator>> SistemaEmail

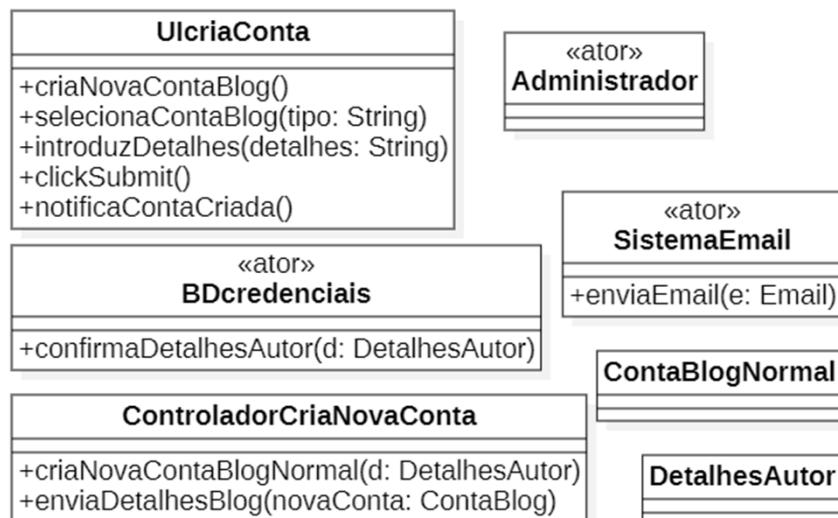
Leonor Melo

22 Diagrama de Comunicação

16

16

Exemplo:  
criação de  
conta de blog



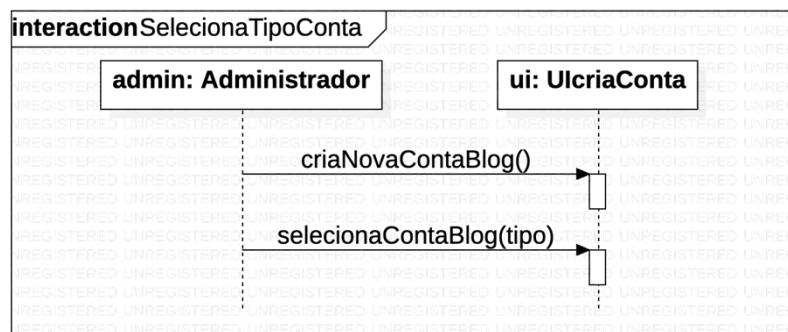
Leonor Melo

22 Diagrama de Comunicação

17

17

Exemplo:  
criação de  
conta de blog:  
seleciona tipo  
de conta,  
diagrama de  
sequencia



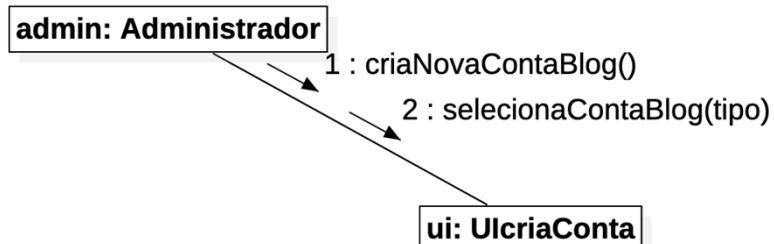
Leonor Melo

22 Diagrama de Comunicação

18

18

Exemplo:  
criação de  
conta de blog:  
seleciona tipo  
de conta  
diagrama de  
comunicação



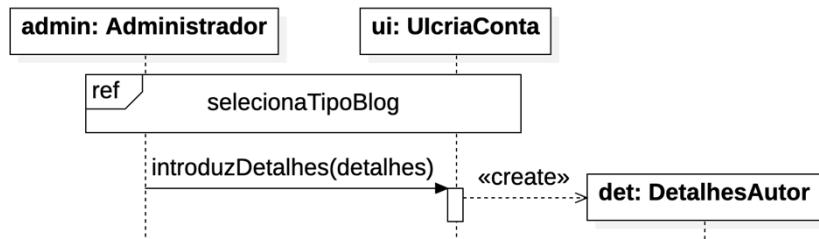
Leonor Melo

22 Diagrama de Comunicação

19

19

Exemplo:  
criação de  
conta de blog:  
cria detalhes  
de autor,  
diagrama de  
sequencia



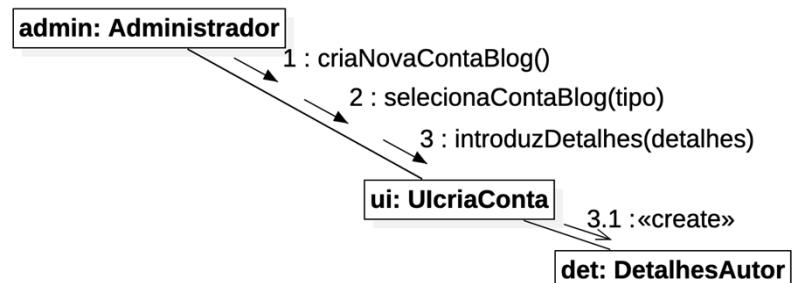
Leonor Melo

22 Diagrama de Comunicação

20

20

Exemplo:  
criação de  
conta de blog:  
cria detalhes  
de autor,  
diagrama de  
comunicação



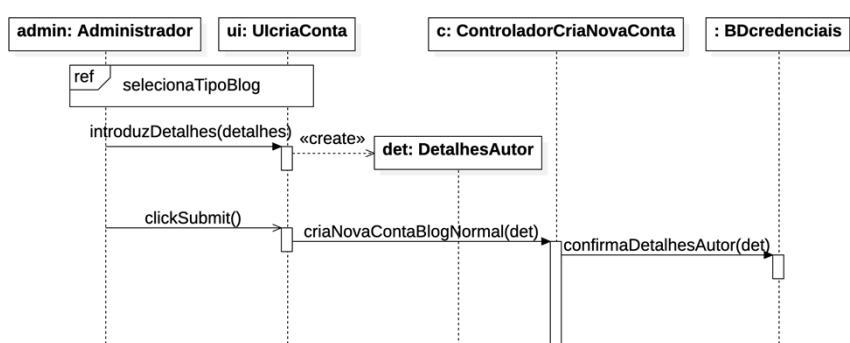
Leonor Melo

22 Diagrama de Comunicação

21

21

Exemplo:  
criação de  
conta de blog:  
verificar  
credenciais do  
autor,  
diagrama de  
sequencia



Leonor Melo

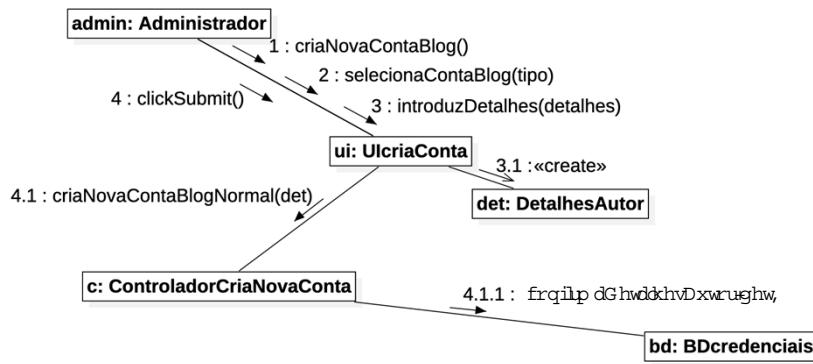
22 Diagrama de Comunicação

22

22

Exemplo:  
criação de  
conta de blog:  
verificar  
credenciais do  
autor,  
diagrama de  
comunicação

Leonor Melo



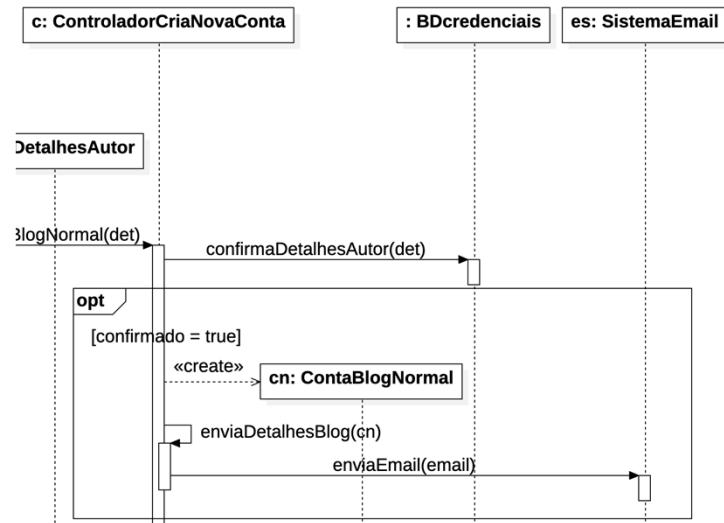
22 Diagrama de Comunicação

23

23

Exemplo:  
criação de  
conta de blog:  
fragmento  
opcional,  
diagrama de  
sequencia

Leonor Melo

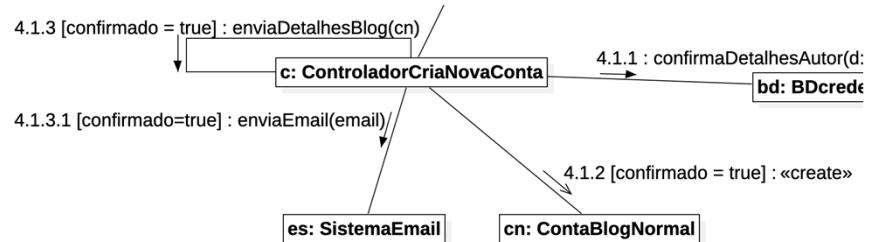


22 Diagrama de Comunicação

24

24

Exemplo:  
criação de  
conta de blog:  
fragmento  
opcional,  
diagrama de  
comunicação



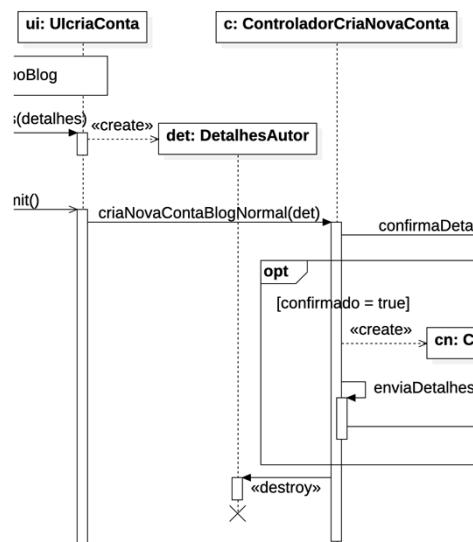
Leonor Melo

22 Diagrama de Comunicação

25

25

Exemplo:  
criação de  
conta de blog:  
destruição de  
um objeto,  
diagrama de  
sequencia



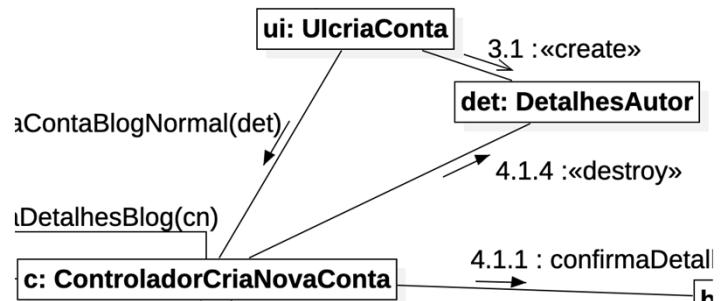
Leonor Melo

22 Diagrama de Comunicação

26

26

Exemplo:  
criação de  
conta de blog:  
destruição de  
um objeto,  
diagrama de  
comunicação



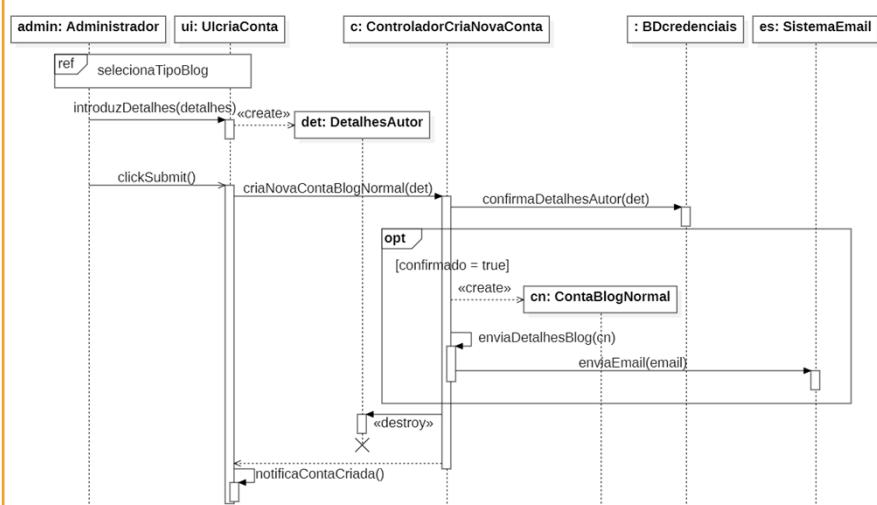
Leonor Melo

22 Diagrama de Comunicação

27

27

Exemplo:  
criação de  
conta de blog:  
interação  
completa,  
diagrama de  
sequencia



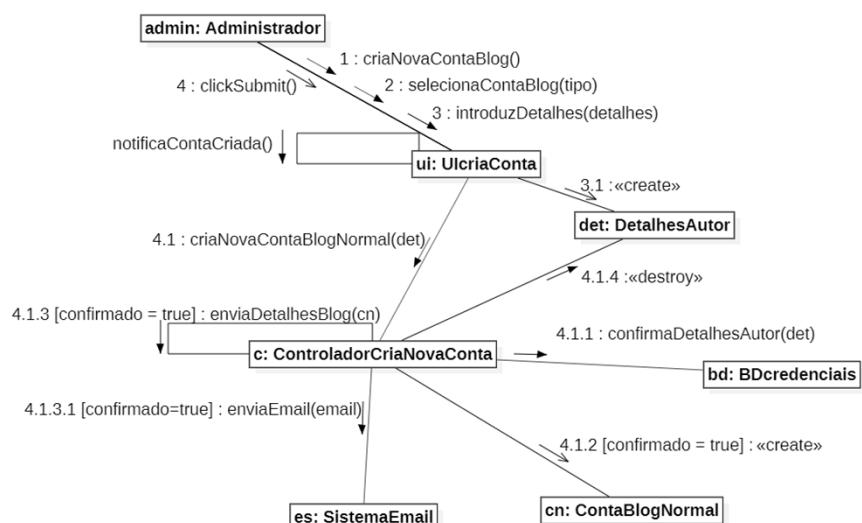
Leonor Melo

22 Diagrama de Comunicação

28

28

## Exemplo: criação de conta de blog: interação completa, diagrama de comunicação



Leonor Melo

22 Diagrama de Comunicação

29

## Comparação entre diagrama de sequencia e diagrama de comunicação

- Diagramas sequencia
  - Mais difícil identificar participantes
  - Links entre participantes subentendidos
  - Fragmentos que permitem decompor o diagrama
  - Fragmento de fluxo de execução que aumentam legibilidade do diagrama
  - Sequencia de mensagens clara
  - Diagramas mais rígidos e tendencialmente largos

Leonor Melo

22 Diagrama de Comunicação

30

## Comparação entre diagrama de sequencia e diagrama de comunicação

- Diagramas comunicação
  - Mais fácil identificar participantes
  - Links entre participantes claros
  - Não prevê decomposição
  - Fluxo de execução representado através da numeração das mensagens
  - Sequencia de mensagens mais difícil de ler
  - Diagramas de formato mais flexível, tendencialmente mais fáceis e manter

Leonor Melo

22 Diagrama de Comunicação

31

31

# Modelação e Design

## 23: Design patterns

Leonor Melo

leonor@isec.pt

1

Design  
patterns

- Princípios de Orientação a Objetos
- Definição e uso de design patterns
- Design patterns
  - creacionais
  - estruturais
  - comportamentais

Leonor Melo

23 Design Patterns

2

## Princípios de Orientação a Objetos

- Princípios OO:
  - Encapsular o que varia
  - Preferir composição a herança
  - Depender de interfaces e não de implementações
  - Design que favoreça o baixo acoplamento entre objetos que interajam
  - "only talk to your friends" (o objeto *a* pode pedir um serviço a um objeto *b*, mas não deve "atravessar" o objeto *b* para chegar ao objeto *c* e pedir-lhe um serviço)

Leonor Melo

23 Design Patterns

3

## Princípios de Orientação a Objetos

- Princípios OO (continuação):
  - Classes devem ser abertas para extensão mas fechadas para modificação
  - Depender de abstrações. Não depender de classes concretas
  - "dont call us, we'll call you": uma classe cliente recebe um serviço de que depende sem ter de o saber construir. O serviço injetado faz parte do estado do cliente.
  - A classe deve ter uma razão para mudar (a classe deve apenas ter um propósito)

Leonor Melo

23 Design Patterns

4

## Definição e uso

- Um design pattern (padrão de desenho) é uma solução
  - Genérica e
  - Reproduzível
  - Para um problema de design comum
- Não é uma solução de design completa que possa ser transformada em código
  - É uma descrição ou template do modo de resolver o problema
    - Que pode ser aplicado a diferentes situações

Leonor Melo

23 Design Patterns

5

## Utilização

- Design Patterns:
  - Fornecem soluções genéricas, documentadas num formato que não requer que fique associado apenas a um caso ou problema específico
  - Podem acelerar o processo de desenvolvimento ao fornecer paradigmas de desenvolvimento testados com sucesso
  - Melhoram a comunicação e legibilidade do código (para quem esteja familiarizado com os patterns)

Leonor Melo

23 Design Patterns

6

## Elementos essenciais

- Elementos essenciais:
  - Nome
    - O nome do design pattern. Fornece um vocabulário comum para os designers de software
  - Problema
    - Descrição da situação onde o pattern deve ser aplicado
  - Solução
    - Elementos que compõe o design: estrutura, participantes, colaborações
  - Consequências
    - Resultados, vantagens e desvantagens de aplicar o pattern

Leonor Melo

23 Design Patterns

7

## Design Patterns do GoF

- Gang of Four (GoF)
  - Autores do livro “Design Patterns Elements of reusable Object-Oriented Software”
  - documentaram 23 problemas comuns e as respetivas soluções mais bem aceites
- Classificados por
  - Propósito
  - Âmbito
    - Classes: relações entre classes e subclasses, estático
    - Objeto: relações entre objetos, dinâmico

Leonor Melo

23 Design Patterns

8

## Tipos de design patterns

- Creational
  - Como podem os objetos ser criados
    - Facilidade de manutenção
    - Controlo
    - Extensibilidade
- Structural
  - Como organizar em estruturas
    - Gestão da complexidade
    - Eficiência
- Behavioural
  - Como atribuir responsabilidades aos objetos
    - Desacoplar os objetos
    - flexibilidade
    - Melhor comunicação

Leonor Melo

23 Design Patterns

9

## Creational patterns

- Tornam o processo de instanciação mais abstrato
- Exemplos:
  - Factory
  - Singleton
  - Builder
  - Prototype

Leonor Melo

23 Design Patterns

10

10

## Padrão Singleton

- Garante que apenas uma instância é criada e fornecem acesso global a esse objeto:
- Construtor é privado
- Um atributo privado para um objeto da própria classe
- Método getInstance()
  - Cria a instancia se esta ainda não existir
  - Devolve a referencia para a instancia

Leonor Melo

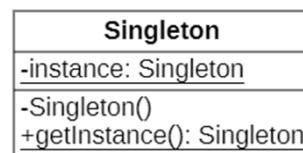
23 Design Patterns

11

11

## Padrão Singleton

- Diagrama de classes:



Leonor Melo

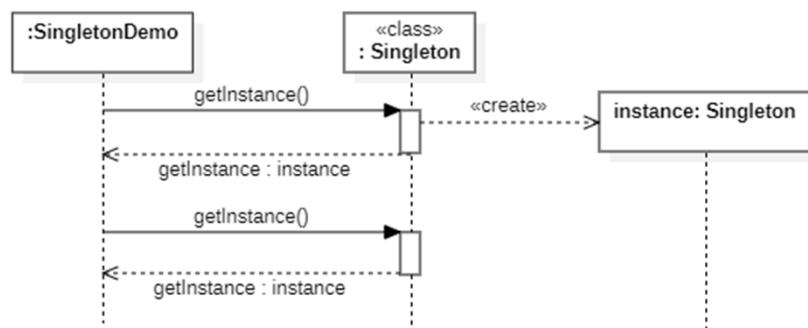
23 Design Patterns

12

12

## Padrão Singleton

- Exemplo de utilização:



Leonor Melo

23 Design Patterns

13

13

## Padrão Singleton

- Quando usar
  - Quando queremos garantir que apenas uma instância da classe é criada
  - Quando essa instância tem de ser acessível por todo o código
  - Em ambientes multi-thread quando vários threads têm de aceder ao mesmo recurso (objeto singleton)
- Usos comuns
  - Classes de configuração

Leonor Melo

23 Design Patterns

14

14

## Padrão Factory

- Cria um objeto omitindo os detalhes da instanciação:
  - Permite que a identidade do objeto seja escolhida apenas em run-time
- Refere-se ao objeto acabado de criar através de um interface comum
- O método que cria o objeto em particular recebe dados genéricos
  - O objeto criado tem de fazer parte de uma hierarquia de classes

Leonor Melo

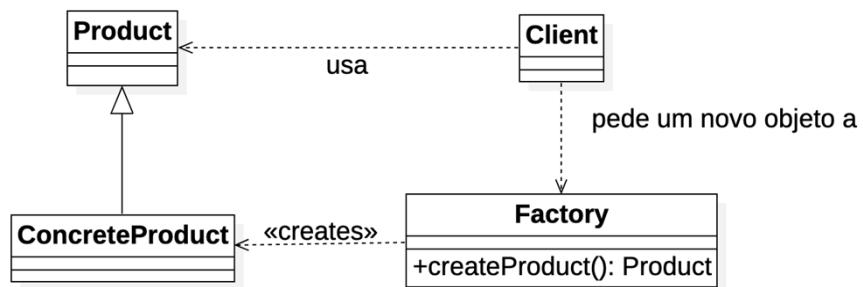
23 Design Patterns

15

15

## Padrão Factory

- Diagrama de classes:



Leonor Melo

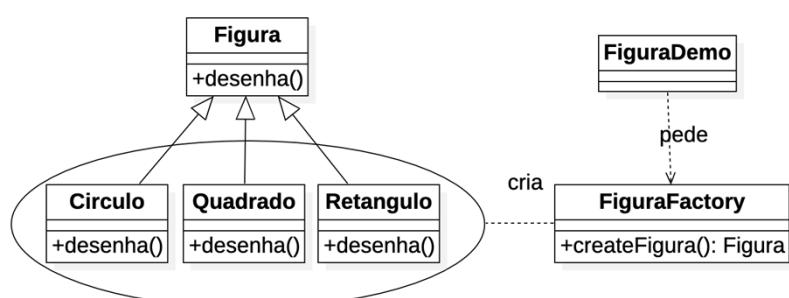
23 Design Patterns

16

16

## Padrão Factory

- Exemplo:



Leonor Melo

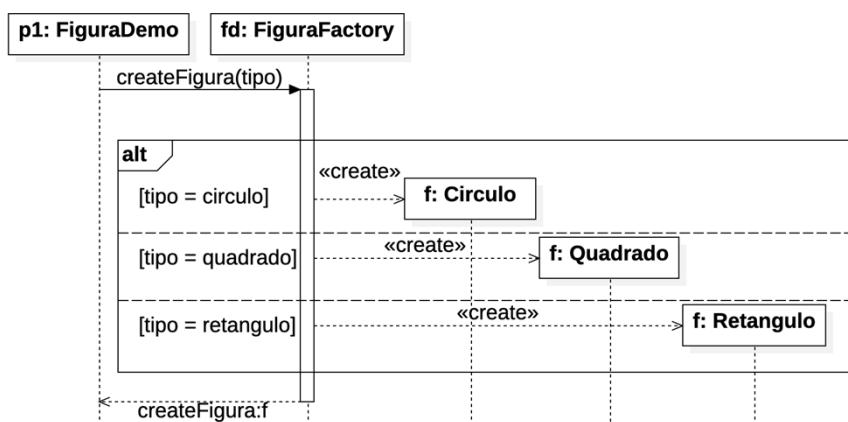
23 Design Patterns

17

17

## Padrão Factory

- Exemplo utilização:



Leonor Melo

23 Design Patterns

18

18

## Structural patterns

- Como devem se devem compor as classes e objetos de maneira a formar estruturas maiores
- Exemplos:
  - Adapter
  - Decorator
  - Proxy
  - Bridge
  - Composite

Leonor Melo

23 Design Patterns

19

19

## Adapter pattern

- Converte o interface de uma classe em outro interface que o cliente está à espera de usar
- Permite que classe que normalmente não trabalhariam juntas devido a interfaces incompatíveis o façam

Leonor Melo

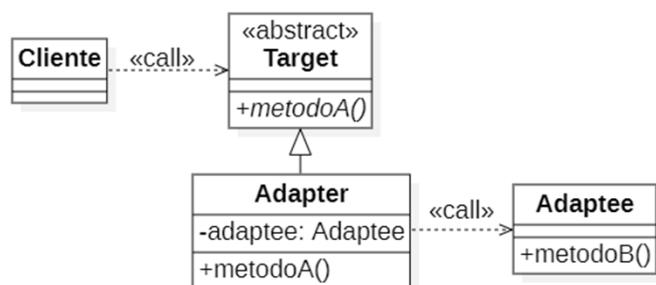
23 Design Patterns

20

20

## Adapter pattern

- Diagrama de classes:



Leonor Melo

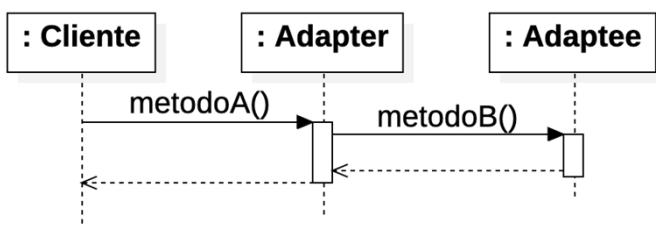
23 Design Patterns

21

21

## Adapter pattern

- Diagrama de sequencia:

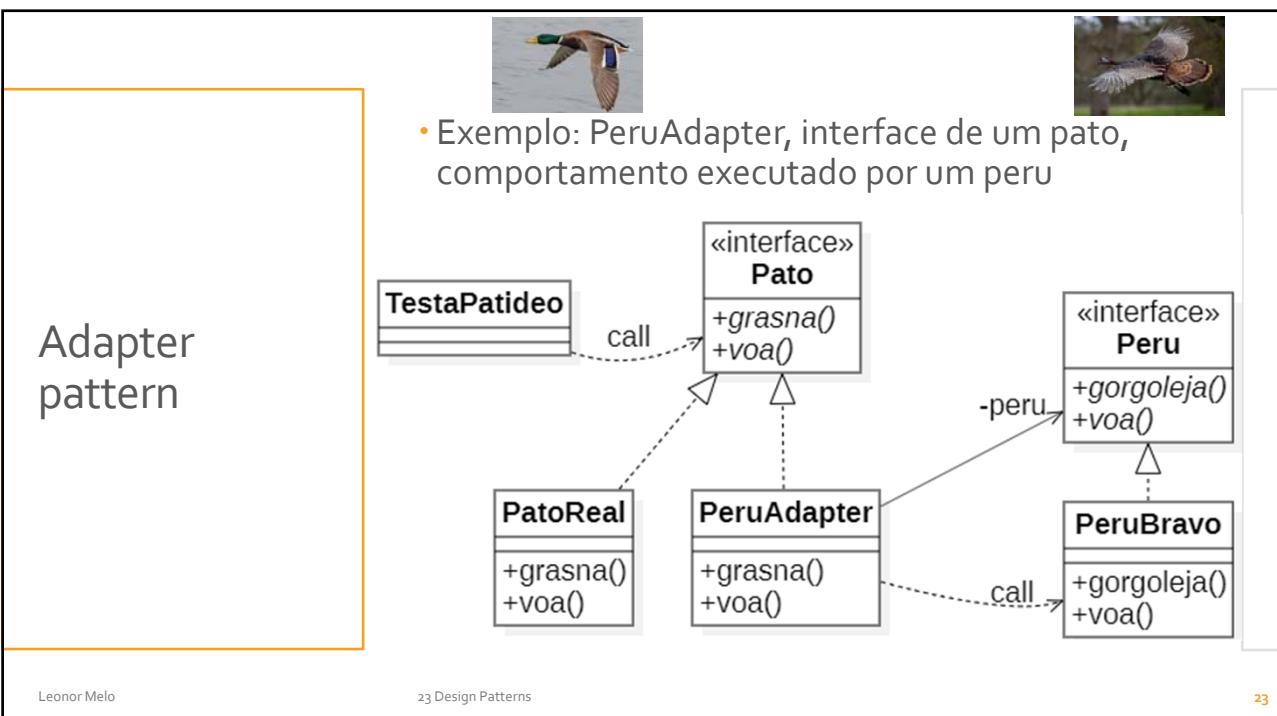


Leonor Melo

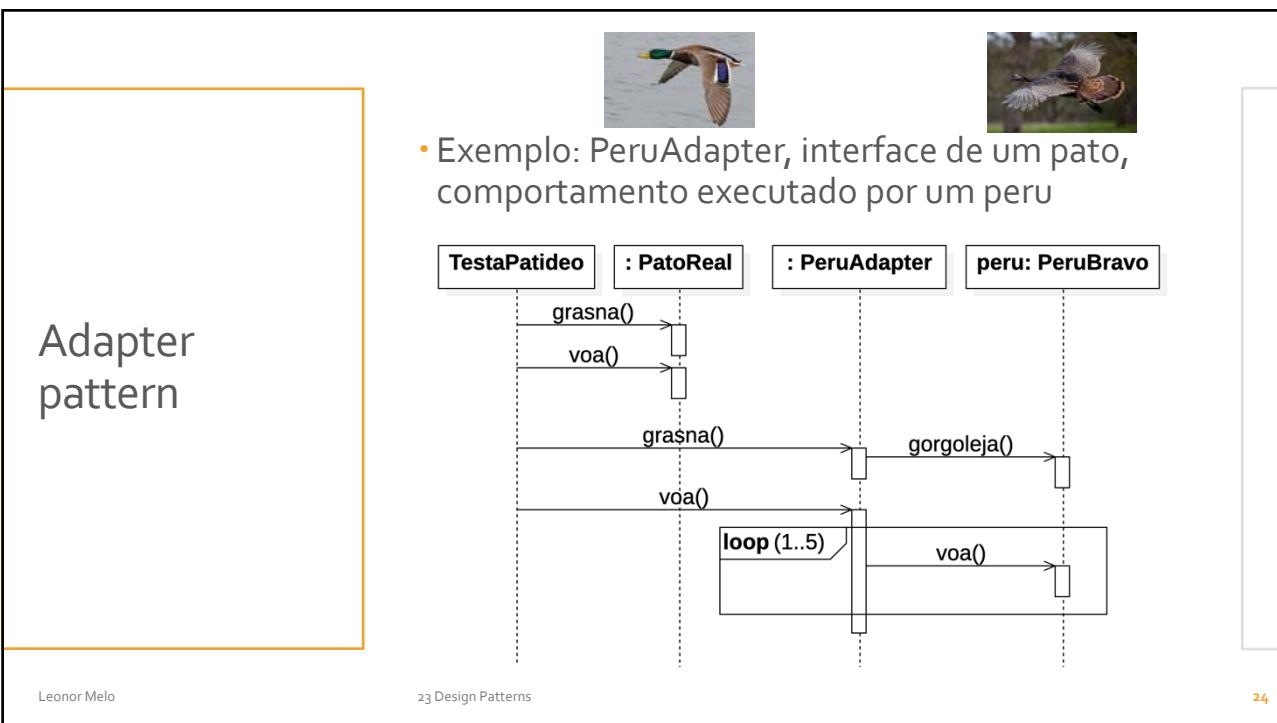
23 Design Patterns

22

22



23



24

## Behavioral patterns

- Dizem respeito a algoritmos e a atribuição de responsabilidades aos objetos
  - Usam herança (classes) ou composição (objetos) para distribuir as responsabilidades
- Exemplo:
  - Strategy
  - Command
  - Iterator
  - Observer
  - Chain of responsibility

Leonor Melo

23 Design Patterns

25

25

## Observer

- Permite que vários objetos sejam notificados da alteração de estado de outros objetos existentes no sistema
- Quando um objeto muda de estado todos os seus dependentes são notificados e atualizados de forma automática
  - A informação pertence apenas ao objeto concreto que está a ser observado. Os observadores concretos têm apenas uma relação de dependência com o observado

Leonor Melo

23 Design Patterns

26

26

## Observer

- Os observadores podem subscrever (e deixar de subscrever) as atualizações do observado
- Quando existe uma mudança de estado do observado este notifica todos os observadores
  - A informação pertence apenas ao objeto concreto que está a ser observado.
  - Os observadores concretos têm apenas uma cópia da informação disponibilizada pelo observado

Leonor Melo

23 Design Patterns

27

27

## Observer

- Responsabilidade do observado:
  - notificar os observadores de que ocorreu uma mudança de estado
- Responsabilidade dos observadores:
  - subscreverem (e deixarem de subscrever) o observado
  - atualizar a sua representação do estado do observado quando são notificados

Leonor Melo

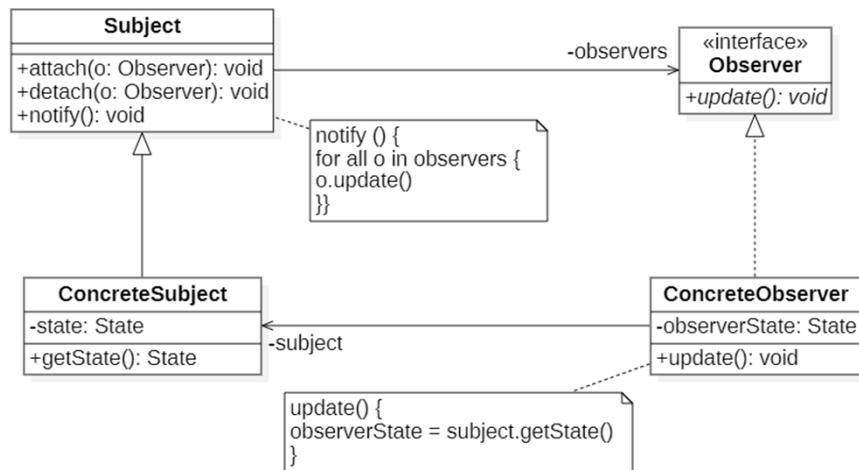
23 Design Patterns

28

28

## Observer

- Diagrama de classes:



Leonor Melo

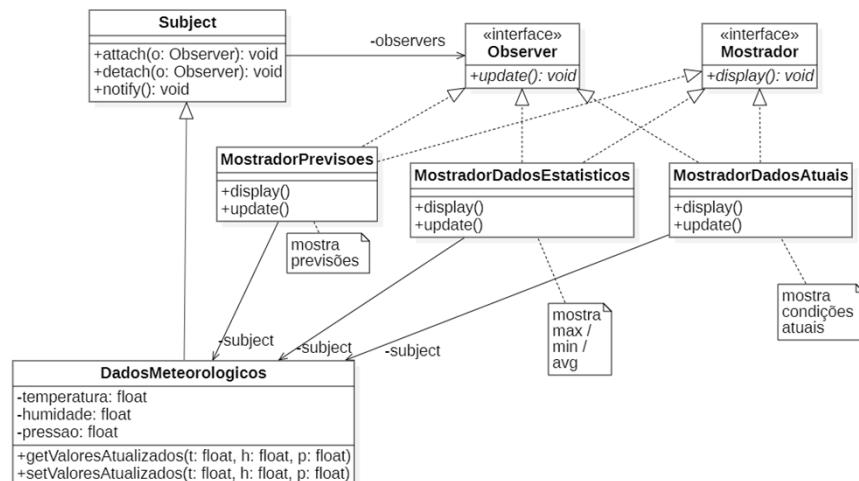
23 Design Patterns

29

29

## Observer

- Exemplo:



Leonor Melo

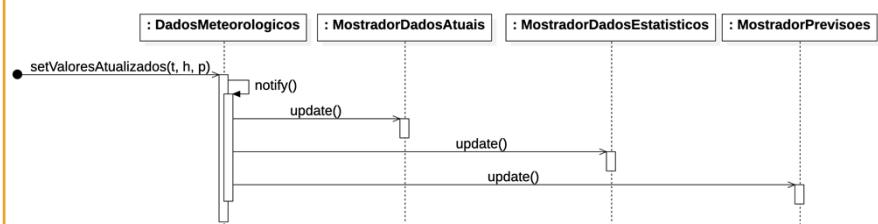
23 Design Patterns

30

30

## Observer

- Exemplo:



Leonor Melo

23 Design Patterns

31

31

# Modelação e Design

## 24: Diagrama de componentes

Leonor Melo

leonor@isec.pt

1

### Diagramas de componentes

- O que é um componente
- Interfaces requeridos e interfaces fornecidos
- Representação em UML
- Portos e estrutura interna
- Whitebox view e Blackbox view

Leonor Melo

24 Diagrama de Componentes

2

## Aplicação dos Inquéritos Pedagógicos, 2.º semestre - 20/21

- Aplicação dos Inquéritos Pedagógicos, 2.º semestre - 20/21
- <https://sigg.ipc.pt/inqueritos>
- Inquérito Unidades Curriculares
- Inquérito avaliação do desempenho pedagógico dos docentes

Leonor Melo

24 Diagrama de Componentes

3

## Diagrama de componentes

- Diagrama de componentes
  - Mostra os blocos principais (ou partes reutilizáveis) do sistema
  - Serve para:
    - Planear a divisão do software em partes numa visão de alto-nível
    - Estabelecer a arquitetura
    - Gerir a complexidade e as dependências entre as partes

Leonor Melo

24 Diagrama de Componentes

4

## Diagrama de componentes



- A perspetiva de desenvolvimento descreve como as partes do sistema estão organizadas em módulos e componentes

Leonor Melo

24 Diagrama de Componentes

5

## Componente

- Componente
  - parte modular do sistema
  - encapsula o seu conteúdo
  - pode ser substituída por outra sem causar impacto negativo no sistema
  - o seu comportamento é definido em termos de
    - interface fornecido e
    - interface requerido
- É um artefacto de design, não existe equivalente concreto em software

Leonor Melo

24 Diagrama de Componentes

6

6

## Componente

- Componentes
  - Podem variar em tamanho
    - de uma (ou poucas) classes a subsistemas grandes
  - Bons candidatos:
    - Item que execute uma funcionalidade chave
    - que seja usado em vários locais do sistema
  - Ex: No sistema de Gestão de Conteúdos, podemos ter um componente que seja um Gestor de Conversão

Leonor Melo

24 Diagrama de Componentes

7

## Componente

- Componente
  - Tal como uma classe pode:
    - ter relações de generalização e associação com outras classes e componentes,
    - implementar interfaces,
    - definir operações,
    - exibir uma estrutura interna,
    - ter portos associados,

Leonor Melo

24 Diagrama de Componentes

8

## Componente

- Componente
  - Tem normalmente um maior número de responsabilidades que uma classe
    - Por exemplo, podemos definir uma classe InfoUtilizador que contenha o contacto do utilizador (nome e mail) e um componente de gestão de utilizadores que permita criar novas contas e verificar a sua autenticidade.
  - É comum um componente conter e usar outras classes e componentes para cumprir os seus objetivos

Leonor Melo

24 Diagrama de Componentes

9

## Componente

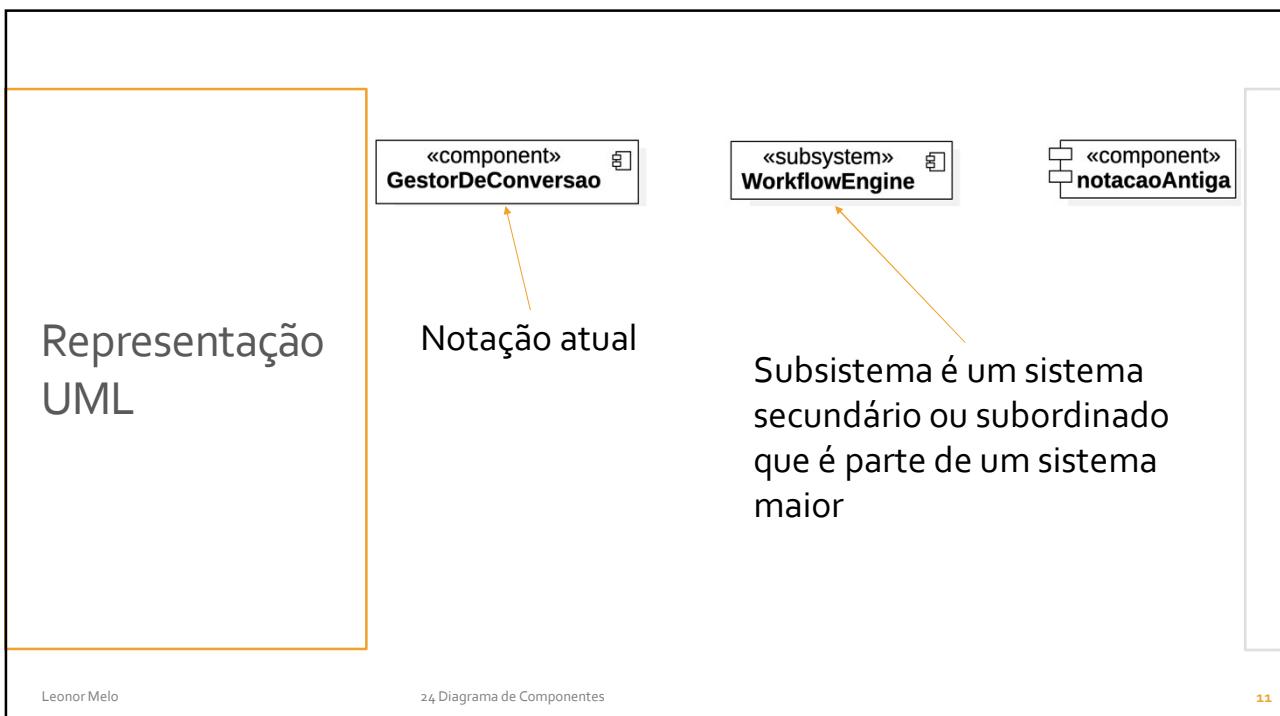
- Componente e interfaces
  - Componente deve ter acoplamento muito fraco com o resto do sistema
  - Comunicação feita através de interfaces
    - (comportamento separado da implementação)
  - Reduz a probabilidade de a substituição de um componente por outro análogo provocar danos graves no sistema

Leonor Melo

24 Diagrama de Componentes

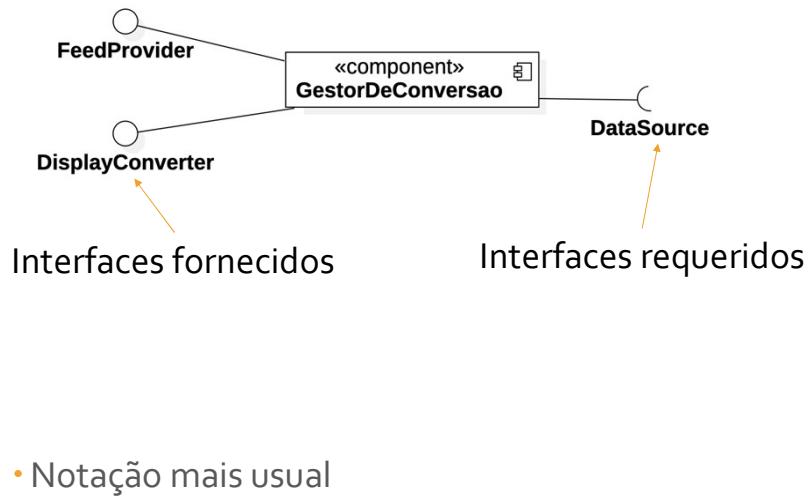
10

10



- Interfaces fornecidos e interfaces requeridos
- Os componentes interagem através dos interfaces fornecido e interfaces requeridos
  - Interfaces fornecidos
    - Interfaces que o componente realiza.
    - Serviços fornecidos pelo componente.
  - Interfaces requeridos
    - Serviços de que o componente necessita para poder funcionar.
  - Acoplamento baixo
    - Comunicação feita através dos interfaces (e não das classes/componentes que os realizam)
- Leonor Melo
- 24 Diagrama de Componentes
- 12

UML:  
Interfaces  
fornecidos e  
interfaces  
requeridos

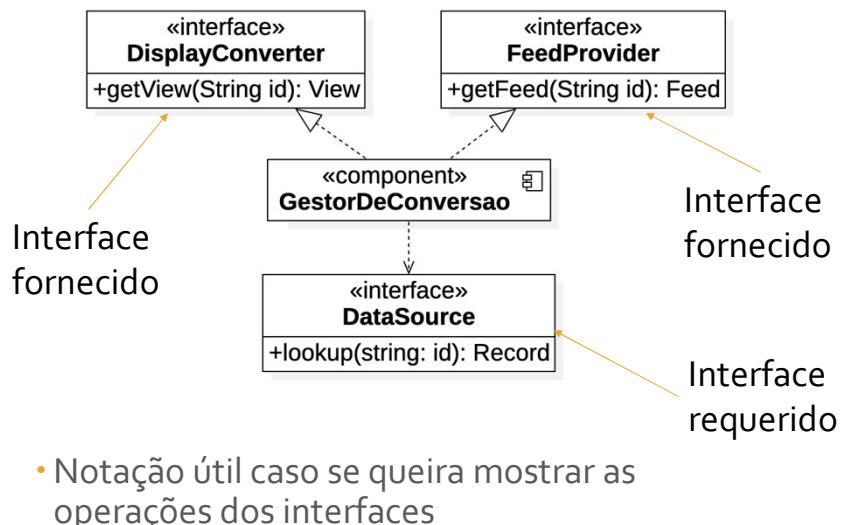


Leonor Melo

24 Diagrama de Componentes

13

UML:  
Interfaces  
fornecidos e  
interfaces  
requeridos



Leonor Melo

24 Diagrama de Componentes

14

UML:  
Interfaces  
fornecidos e  
interfaces  
requeridos

Interfaces  
fornecidos



Interface  
fornecido

- Notação mais compacta, mas mais difícil de ler
- Permite indicar os *artifacts* (ficheiros físicos)

Leonor Melo

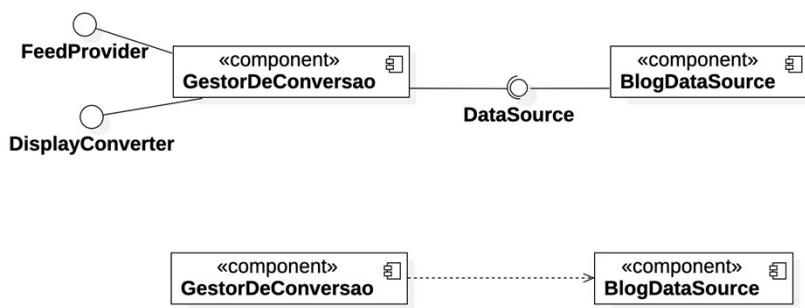
24 Diagrama de Componentes

15

15

UML:  
Componentes  
a trabalhar em  
conjunto

- Duas notações possíveis:
  - Junta esférica (*ball and socket*)
  - Seta de dependência entre os componentes



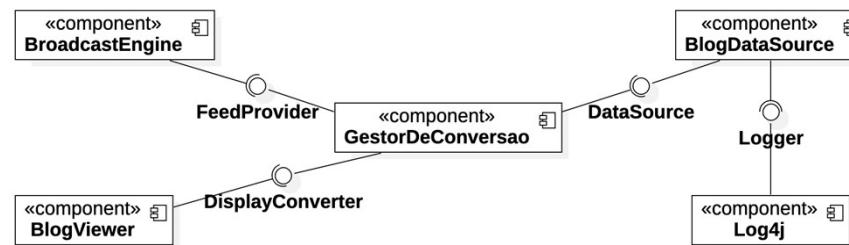
Leonor Melo

24 Diagrama de Componentes

16

16

## UML: Componentes a trabalhar em conjunto



- Realce dado aos elementos chave
  - Componentes e seus interfaces
  - Arquitetura facilmente legível

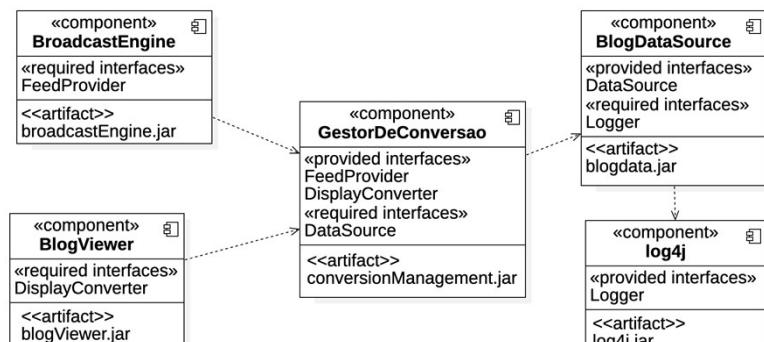
Leonor Melo

24 Diagrama de Componentes

17

17

## UML: Componentes a trabalhar em conjunto



- Visão de alto-nível das dependências entre componentes
  - Pode ajudar na fase de instalação ao mostrar os ficheiros associados aos componentes

Leonor Melo

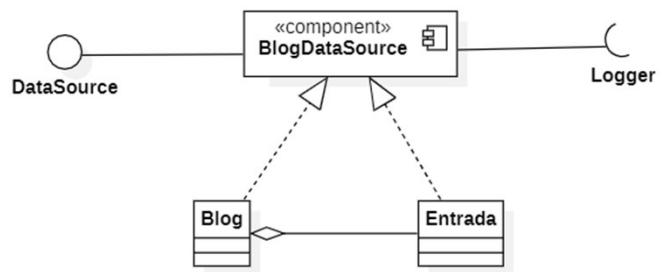
24 Diagrama de Componentes

18

18

## Classes que realizam um componente

- Um componente por vezes contém ou usa classes para implementar as suas funcionalidades
  - Diz-se que essas classes realizam o componente



Leonor Melo

24 Diagrama de Componentes

19

19

## Porto

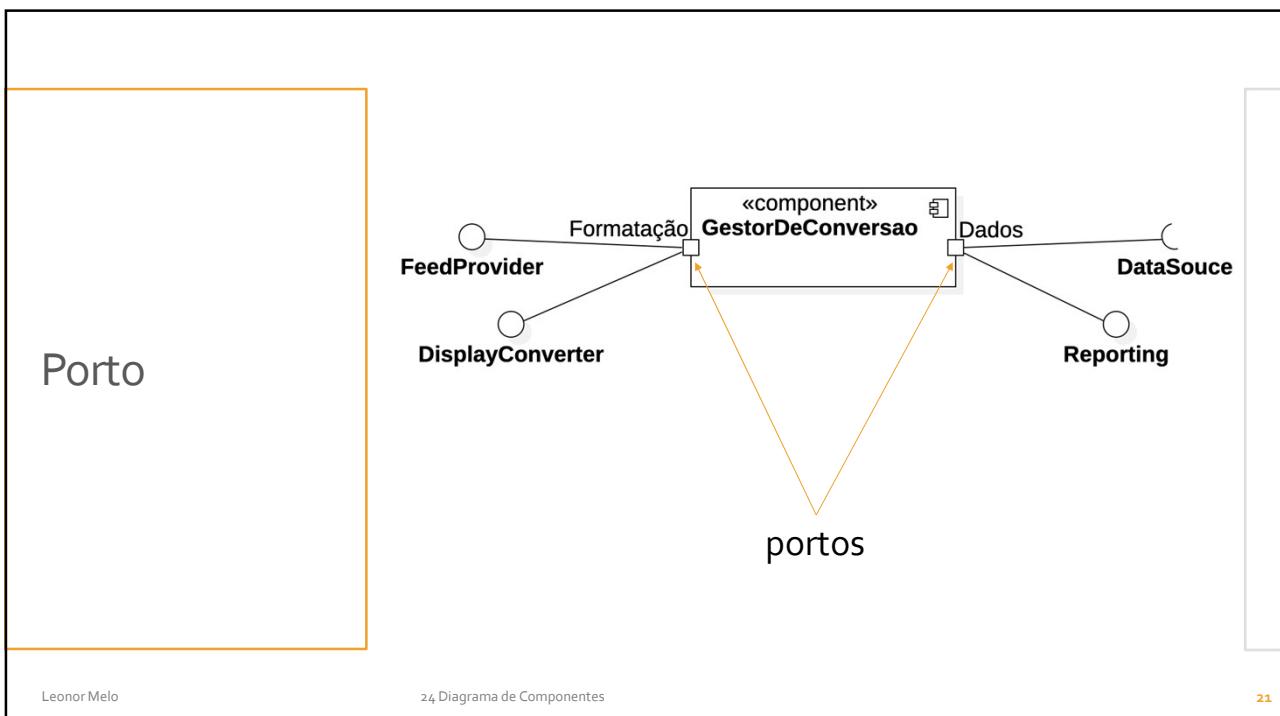
- Porto
  - Ponto de interação
  - Modos específicos como um componente pode ser usado
  - Normalmente tipos de cliente diferentes irão necessitar de portos distintos
  - Pode ter um nome a indicar o propósito do porto
  - Pode agregar mais do que um interface

Leonor Melo

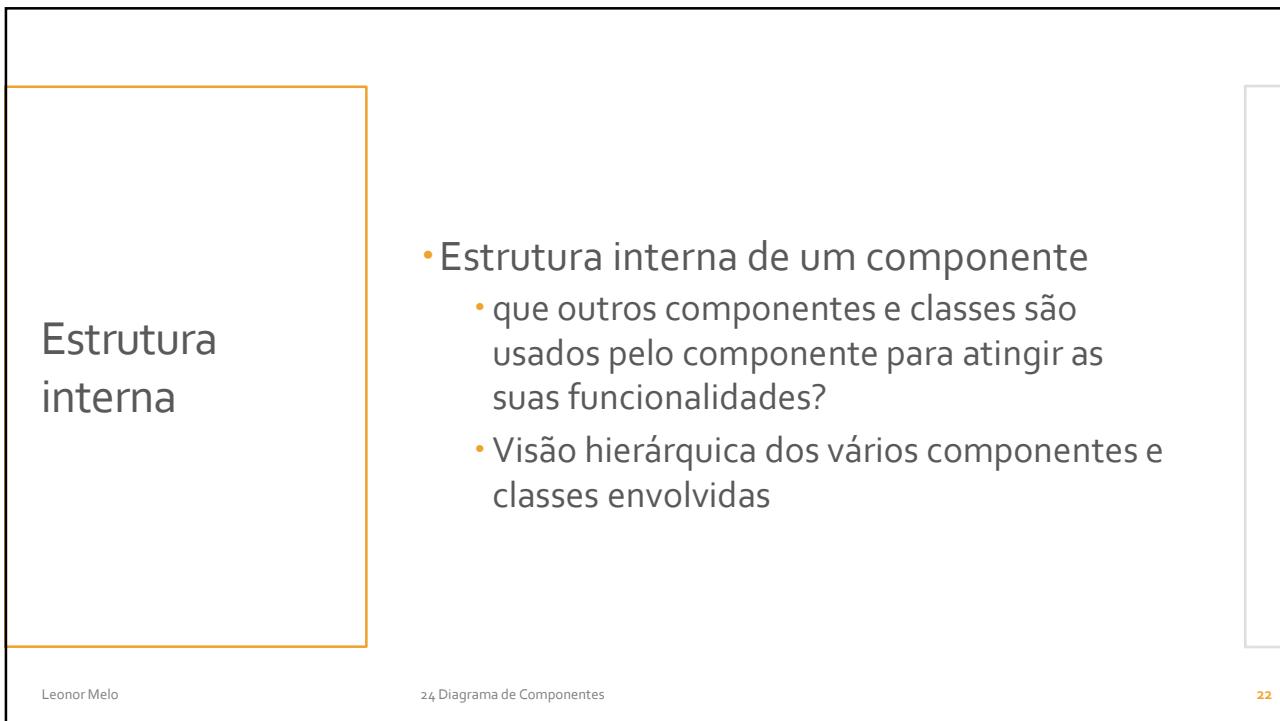
24 Diagrama de Componentes

20

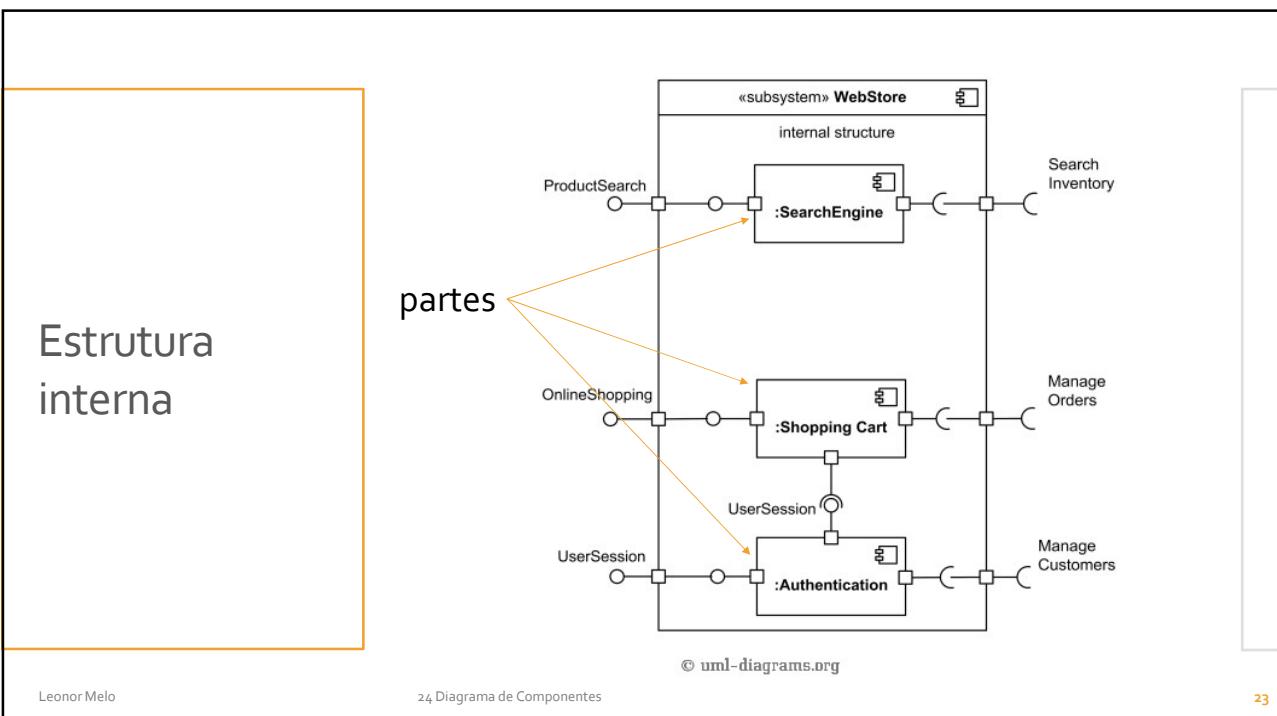
20



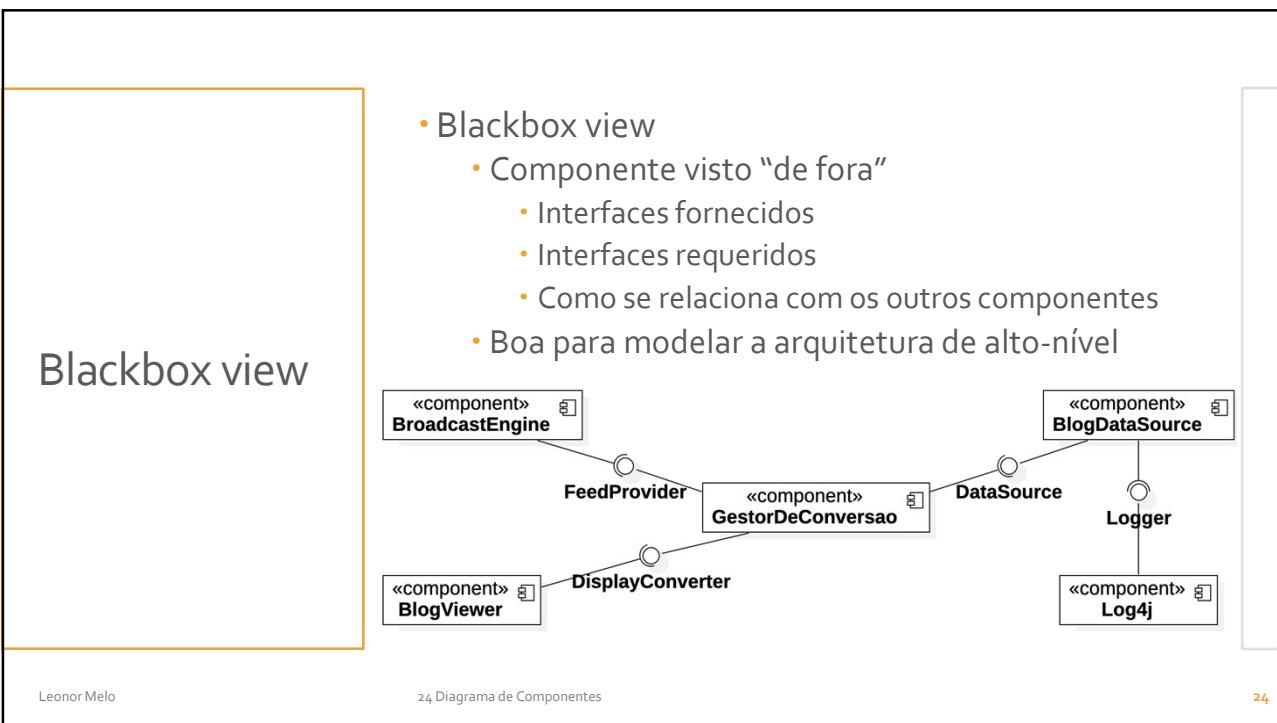
21



22



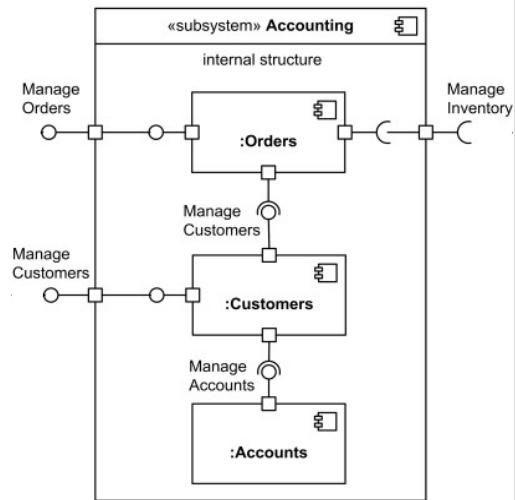
23



24

## Whitebox view

- Whitebox view
  - Estrutura interna do componente
  - Quais os componentes / classes responsáveis pelas várias partes do comportamento do componente



Leonor Melo

24 Diagrama de Componentes

25

25

# Modelação e Design

## 25: Diagrama de instalação

Leonor Melo

leonor@isec.pt

1

### Diagrama Instalação

- O que é e para que serve
- Artefacto
- Nós
- Comunicação entre os nós
- Especificação de instalação

Leonor Melo

25 Diagrama de Instalação

2

## Diagrama de instalação (*Deployment diagram*)



- A perspetiva física diz respeito aos aspectos físicos do sistema, por ex. ficheiros executáveis e hardware onde são executados

Leonor Melo

25 Diagrama de Instalação

3

3

## Diagrama de instalação (*Deployment diagram*)

- Diagrama de instalação
  - Mostra a perspetiva física do sistema
    - Como é que o software é atribuído ao hardware
    - Como é que as diferentes partes do sistema comunicam
  - Sistema no contexto dos diagramas de instalação
    - Software criado
    - Hardware + software onde é executado

Leonor Melo

25 Diagrama de Instalação

4

4

## Instalar um sistema muito simples

- Caso mais simples:
  - Software está contido num único ficheiro executável
  - Irá residir num computador
- Para representar o hardware usa-se um nó
  - etiquetado com o estereótipo <<device>>
  - e com o nome do nó



Leonor Melo

25 Diagrama de Instalação

5

## Instalar um sistema muito simples

- Para representar o software que vai ser executado no hardware usa-se um artefacto
  - Neste caso um ficheiro executável



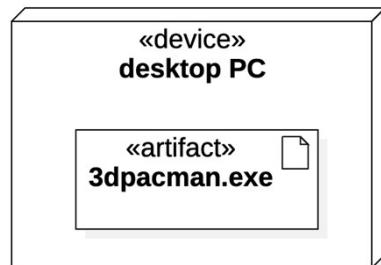
Leonor Melo

25 Diagrama de Instalação

6

## Instalar um sistema muito simples

- Por fim o diagrama completo:
  - Indicar que o executável se deve executar no hardware



Leonor Melo

25 Diagrama de Instalação

7

## Instalar um sistema muito simples

- O diagrama de instalação deve conter os detalhes que sejam importantes para a instalação:
  - Hardware,
  - Firmware,
  - Sistemas operativo,
  - Runtime environment,
  - Device drivers,...
- Se certo aspeto não for relevante, não deve ser adicionado

Leonor Melo

25 Diagrama de Instalação

8

## Artefactos

- Artefactos

- Ficheiros físicos que são executados ou usados pelo software
  - Ficheiros executáveis: .jar, .exe, ...
  - Ficheiros biblioteca: .dll, ...
  - Ficheiros de código: .cpp, .java, ...
  - Ficheiros de configuração usados pelo software em runtime: .txt, .xml, .properties, ...



Leonor Melo

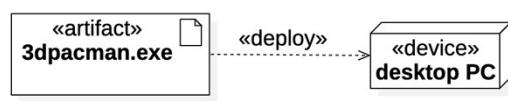
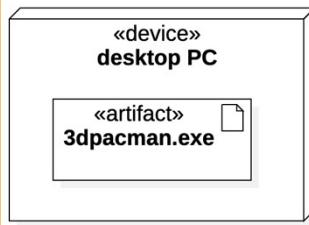
25 Diagrama de Instalação

9

9

## Artefactos instalados num nó

- Mostrar que um artefacto reside (ou foi instalado) num nó
  - Notações possíveis:



Leonor Melo

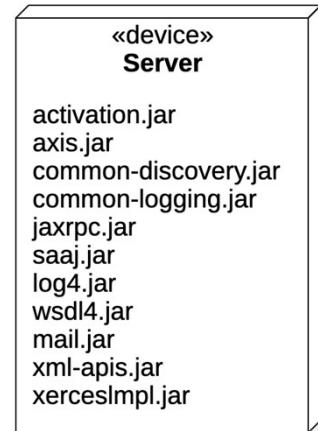
25 Diagrama de Instalação

10

10

## Artefactos instalados num nó

- Notação compacta
  - Poupa espaço no caso de haverem muitos artefactos



Leonor Melo

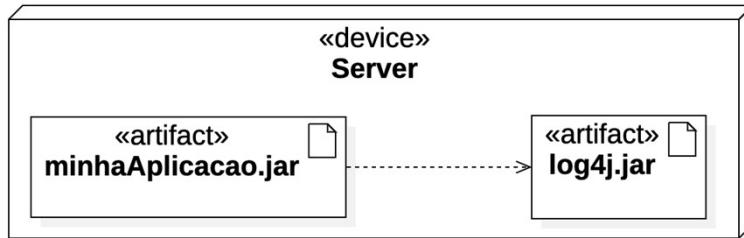
25 Diagrama de Instalação

11

11

## Artefactos instalados num nó

- Notação normal
  - Permite desenhar relações entre os artefactos



Leonor Melo

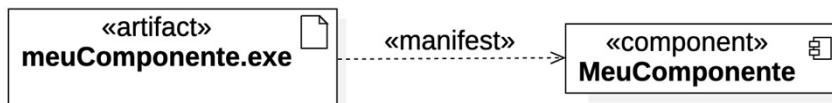
25 Diagrama de Instalação

12

12

## Artefactos a manifestar componentes (ou classes)

- Durante a construção do sistema o código correspondente a um componente é compilado resultando em um ou mais ficheiros – ou artefactos
  - Esse ficheiro *manifesta* o componente
- Relação de manifestação
  - Representada como uma dependência estereotipada entre o artefacto e o componente



Leonor Melo

25 Diagrama de Instalação

13

13

## Artefactos a manifestar componentes (ou classes)

- Relação de manifestação
  - Também pode ser representada dentro do componente



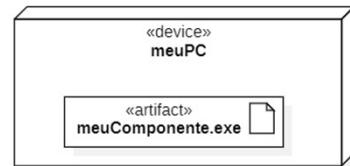
Leonor Melo

25 Diagrama de Instalação

14

14

## Artefactos a manifestar componentes (ou classes)



- Normalmente as relações de manifestação são mostradas à parte das relações de instalação
- Mostra como os componentes ficarão distribuídos pelo hardware

Leonor Melo

25 Diagrama de Instalação

15

15

## Nó

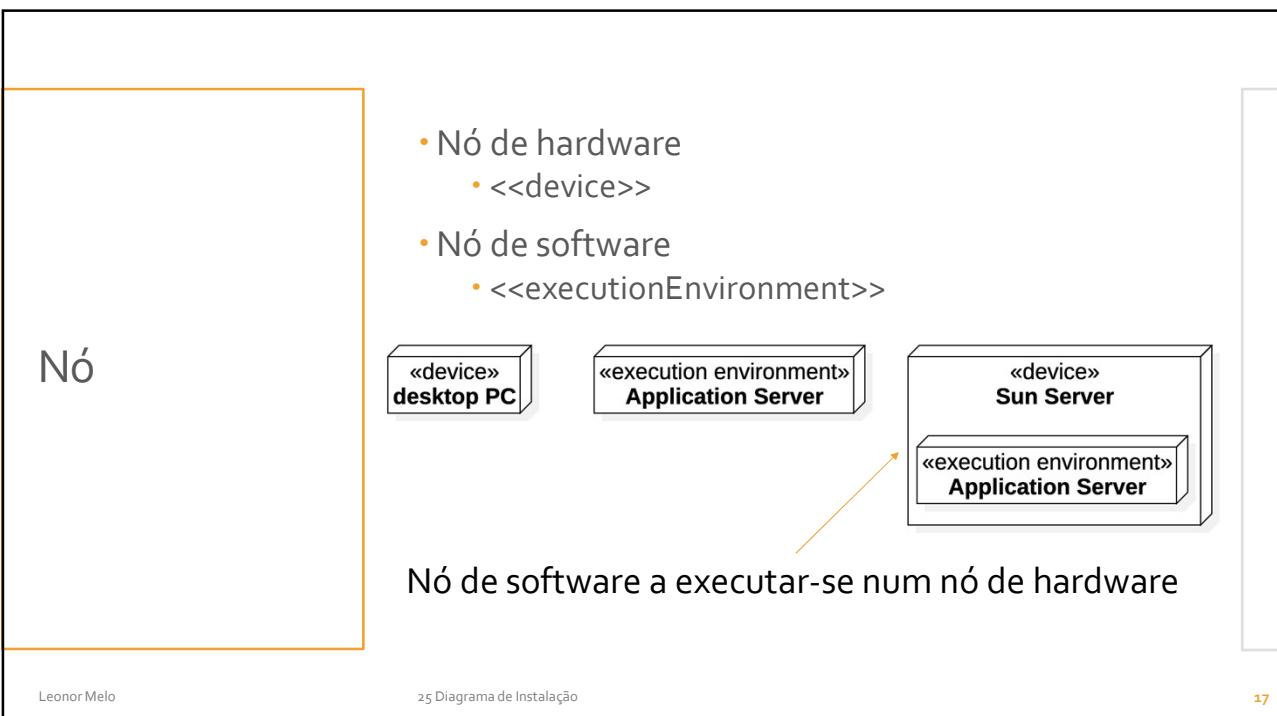
- Nó
  - Pode ser hardware
    - Servidor,
    - Desktop,
    - Dispositivo móvel,...
  - Mas também pode ser software
    - Software que fornece o ambiente onde o sistema ou componente se aloja e executa
      - Sistema operativo,
      - Máquina virtual,
      - Motor de base de dados,
      - Web browser,...

Leonor Melo

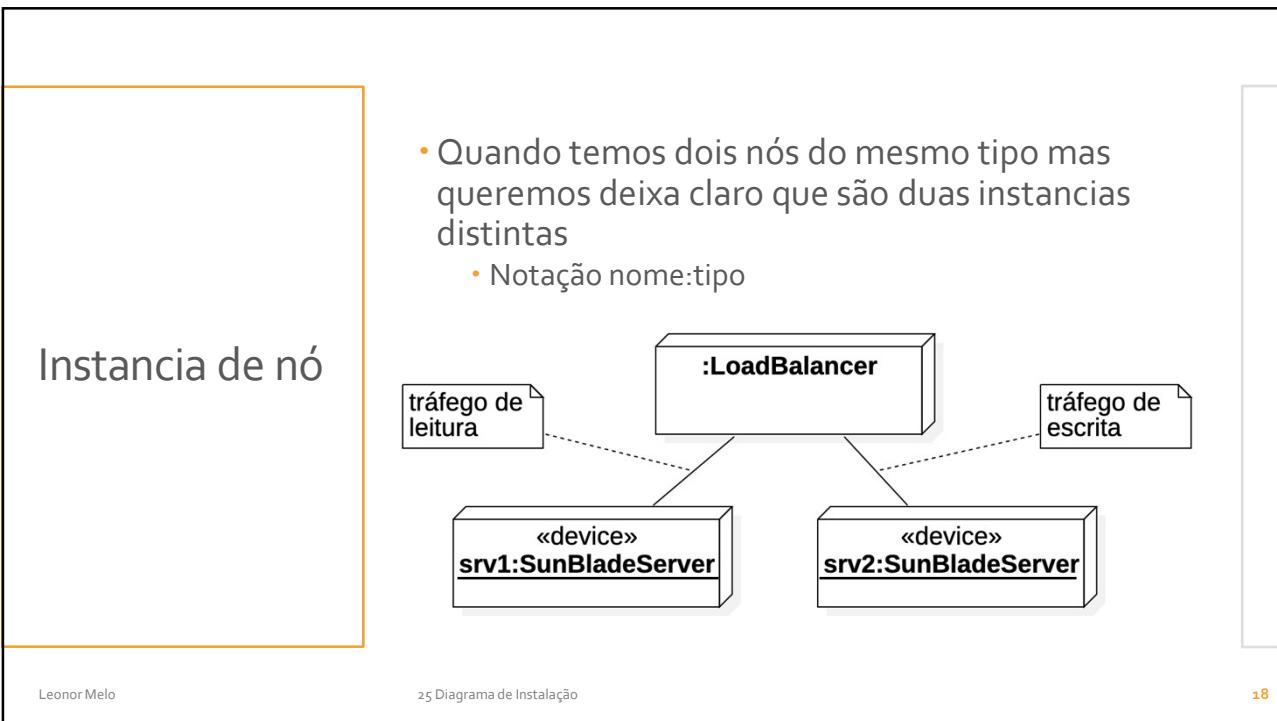
25 Diagrama de Instalação

16

16



17

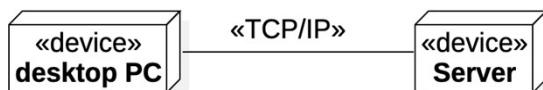


18

## Comunicação entre os nós

- É comum os nós precisarem de comunicar:

- Aplicação cliente a correr num desktop que obtém dados de um server através de uma ligação TCP/IP



- *Communication paths*

- Mostram que os nós são capazes de comunicar
- Não devem mostrar mensagens individuais

Leonor Melo

25 Diagrama de Instalação

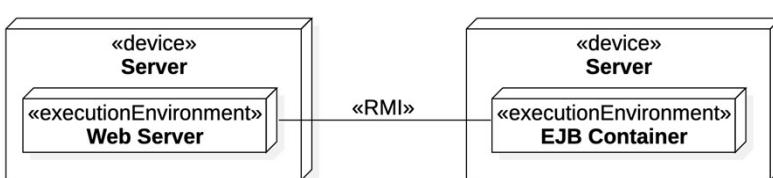
19

19

## Comunicação entre os nós

- Também se pode mostrar comunicação entre nós ambiente de execução

- Um web server que comunica com um EJB container através de RMI



- A “camada” de transporte é TCP/IP, o RMI está “sobre” uma camada de TCP/IP

- É usual usar-se o estereótipo correspondente à “camada” mais alta, mas depende da audiência

Leonor Melo

25 Diagrama de Instalação

20

20

## Especificação de instalação

- Especificação de instalação:
  - Artefacto que fornece informação necessária à instalação de outro artefacto no nó
  - Conjunto de propriedades que determinam os parâmetros de execução de um artefacto
  - A mesma especificação pode ser associada a vários artefactos

Leonor Melo

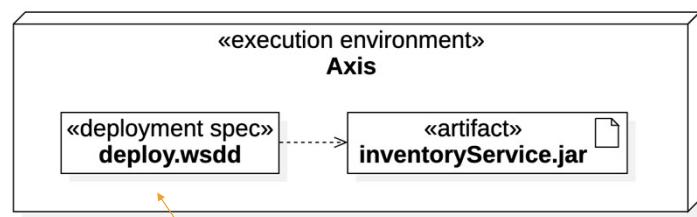
25 Diagrama de Instalação

21

21

## Especificação de instalação

- Especificação de instalação: quando artefacto desenhado dentro do nó



Especificação de instalação

Leonor Melo

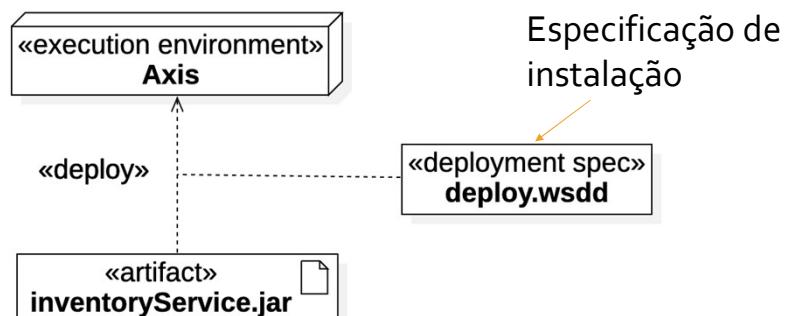
25 Diagrama de Instalação

22

22

## Especificação de instalação

- Especificação de instalação: quando o artefacto a instalar usa uma relação de <>deploy>> com o nó



Leonor Melo

25 Diagrama de Instalação

23

23

## Mostra as propriedades da especificação de instalação

- É possível mostrar as propriedades individuais que constam da especificação de instalação:
  - Quer o formato que essas propriedades devem ter (nome e tipos)
  - Quer a instanciação das propriedades para um caso específico (nomes e valores)

«deployment spec»  
deploy.wsdd  
className: string  
allowedMethods: string[\*]

«deployment specs»  
deploy.wsdd  
className: inventory.InventoryService  
allowedMethods: \*

Formato geral

Instanciação

Leonor Melo

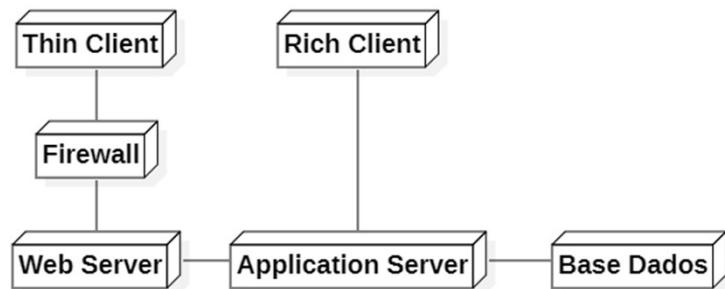
25 Diagrama de Instalação

24

24

## Diagrama inicial

- Possivelmente apenas informação básica sobre a disposição física
- O que podemos ficar a saber com este diagrama?



Leonor Melo

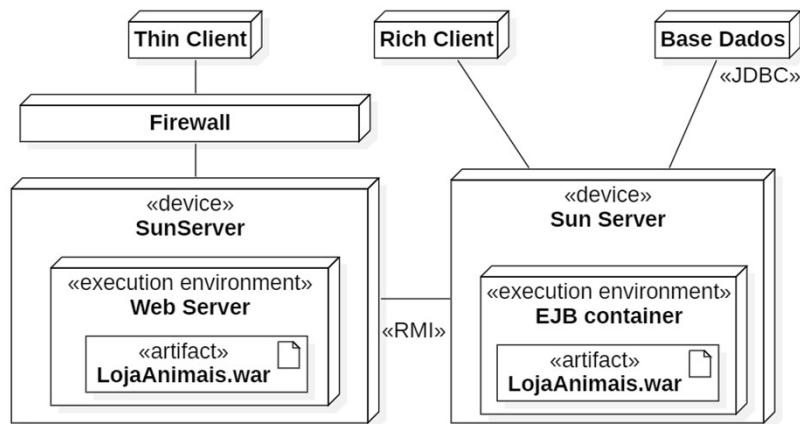
25 Diagrama de Instalação

25

25

## Fases mais avançadas

- Mais detalhes: tecnologias, protocolos de comunicação, artefactos de software



Leonor Melo

25 Diagrama de Instalação

26

26

# Modelação e Design

## 26 e 27: Processos de Desenvolvimento de software

Leonor Melo

leonor@isec.pt

1

## Construir software

- Atividades desenvolvidas durante a construção do software:
  - Definição do problema
  - Desenvolvimento dos requisitos
  - Planeamento da construção
  - Arquitetura do software ou visão de alto-nível
  - Design detalhado
  - Codificação e *debugging*
  - Testes unitários
  - Teste de integração
  - Integração
  - Teste de sistema
  - Correções de manutenção

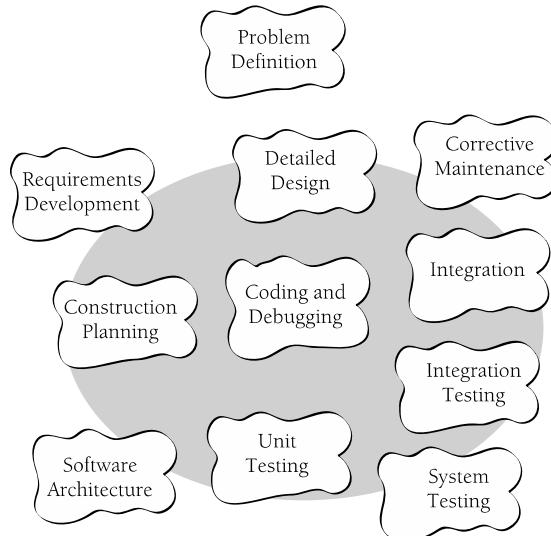
Leonor Melo

26 e 27 Processos de desenvolvimento de Software

2

## Construir software

- Manter em mente as diversas atividades permite criar uma perspetiva mais rica do que deve ser feito e como



Leonor Melo

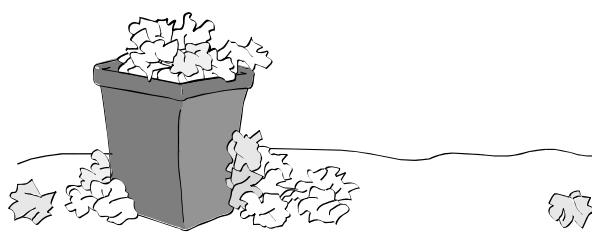
26 e 27 Processos de desenvolvimento de Software

3

3

## Construir software

- Desenvolver o sistema como quem escreve uma carta:
  - Sentar ao teclado e começar a escrever do início ao fim
    - Só funciona para projetos muito pequenos desenvolvidos por uma única pessoa
    - Processo “tentativa e erro”



Leonor Melo

26 e 27 Processos de desenvolvimento de Software

4

4

## Construir software

- Construir implica planificação, preparação e execução
- Quanto mais complexo o sistema mais importante se torna a planificação e preparação



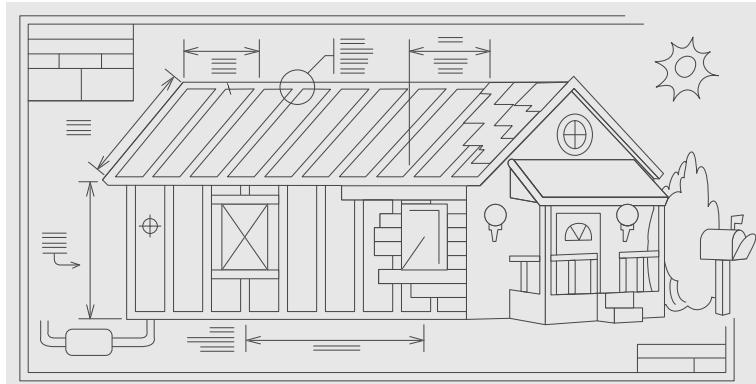
Leonor Melo

26 e 27 Processos de desenvolvimento de Software

5

## Construir software

- As consequências de um design infeliz num projeto mais complexo de construir são maiores



Leonor Melo

26 e 27 Processos de desenvolvimento de Software

6

## Construir software

- Construir edifício vs construir software:
  - Definir estilo de edifício
    - (definição problemas)
  - Plantas que apresentam várias perspetivas
    - (Design da arquitetura até design do software)
  - Feito por determinada ordem: fundações, paredes, instalações elétricas/água/gás, acabamentos
    - (várias fases do desenvolvimento e otimização)
  - Inspeções técnicas
    - (revisões do software)

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

7

## Construir software

- Construir edifício vs construir software:
  - Nem tudo é construído de raiz: fogão, forno, exaustor, lavatórios, banheiras, etc. são comprados
    - (linguagens de alto-nível, bibliotecas, classes de interface com o utilizador e manipulação de BD pré-existentes)
  - Certas partes podem ser adaptadas à medida: armários da cozinha , portas, janelas
    - (classes que sirvam de adaptadores de classes pré-existentes para uma utilização mais uniforme)

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

8

## Construir software

- Construir edifício vs construir software:
  - Nem todas as alterações têm o mesmo custo: mover certas paredes compromete a integridade do edifício
    - (nem todas as classes podem ser alteradas com a mesma facilidade)
- A importância da planificação, design e verificação variam de acordo com a dimensão e complexidade do projeto

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

9

9

## Método waterfall

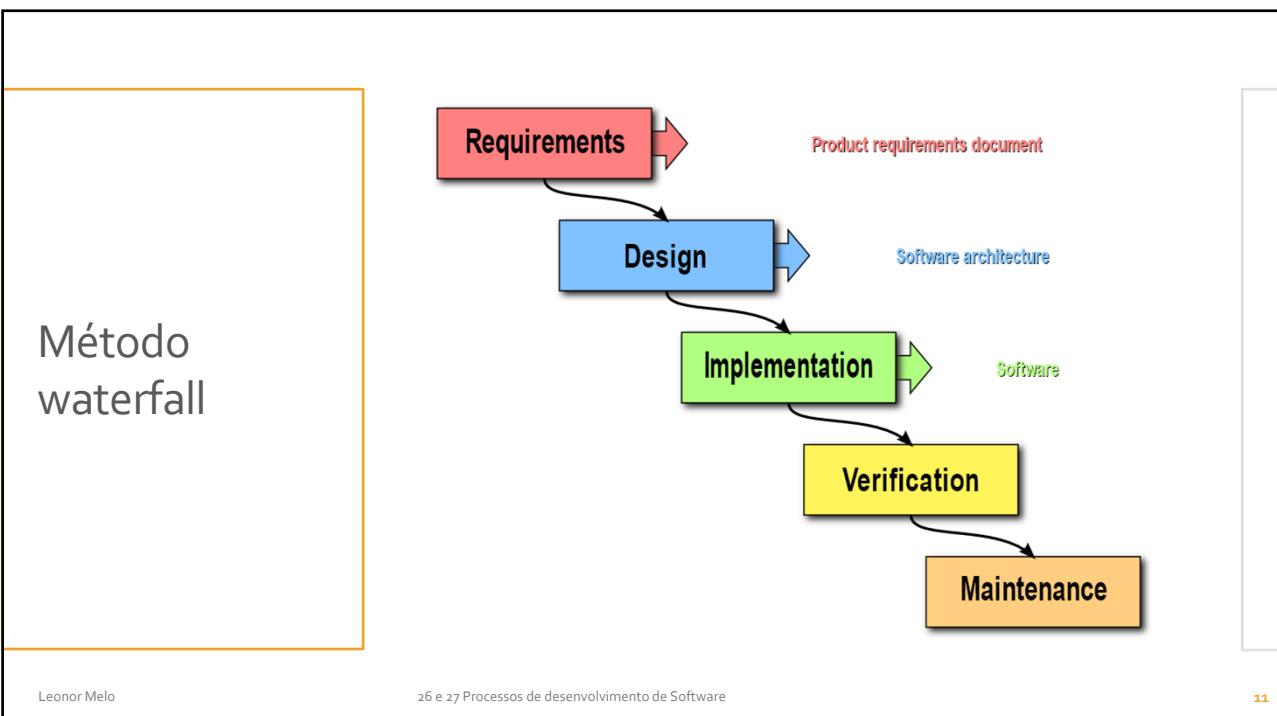
- Método de desenvolvimento de software waterfall (desenvolvimento em cascata)
  - Dividido por fases
  - Abordagem sequencial
    - Uma fase não pode começar antes da anterior terminar
  - Cada fase existe de forma isolada
    - O cliente deixa de ter qualquer papel no desenvolvimento uma vez terminada a fase inicial

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

10

10



11



12

## Método waterfall

- Verificação
  - Apresentação do produto terminado ao cliente.
- Manutenção
  - Utilização do produto por parte do cliente.
  - Cliente reporta bugs, características indesejadas e erros. A equipa de produção vai fazendo correções até ao cliente ficar satisfeito

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

13

13

## Suposições do método waterfall

- Suposições do método waterfall:
  - Requisitos são conhecidos desde o início e com exatidão
  - Requisitos nunca (ou muito raramente) mudam
  - Os clientes sabem exatamente o que pretendem e não necessitam de ajuda a visualizar o sistema
  - O design pode ser feito de forma puramente abstrata e a tentativa raramente leva a erro

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

14

14

## Suposições do método waterfall

- Suposições do método waterfall (cont.):
  - A tecnologia irá manter-se estável e responder como o esperado quando for feita a integração
  - O sistema não é assim tão complexo (e bonecos são para todos)
- Método waterfall:
  - Limita a habilidade de reagir a qualquer mudança ou de fazer correções ao longo do desenvolvimento

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

15

15

## Motivo de falha de projetos de software

- Alguns dos motivos mais comuns que levam a falhas nos projetos de software são:
  - Falta de comunicação com o cliente
  - Requisitos mal definidos
  - Objetivos pouco realistas ou mal articulados
  - Falta de competência para lidar com a complexidade do projeto
  - Falta de organização
  - Práticas de desenvolvimento desleixadas
  - Estimativas pouco rigorosas

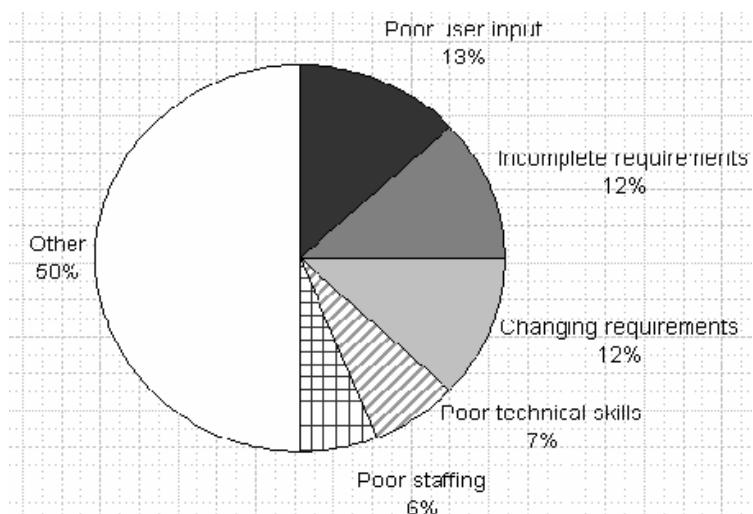
Leonor Melo

26 e 27 Processos de desenvolvimento de Software

16

16

## Motivo de falha de projetos de software



Leonor Melo

26 e 27 Processos de desenvolvimento de Software

17

17

## Motivo de falha de projetos de software

- (alguns) motivos de falha podem ser controlados:
  - Estimativas devem ser tão próximas da realidade quanto possível
  - Datas de entrega devem ser realistas
  - Riscos devem ser reconsiderados, controlados e geridos
  - Os trabalhadores podem ser recompensados por trabalhos extra

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

18

18

## Necessidade de feedback e adaptação

- Sistemas complexos e sujeitos a mudança necessitam de feedback e capacidade de adaptação:
  - Feedback das implementações iniciais, da interpretação das especificações e das demonstrações ao cliente, para refinar os requisitos
  - Feedback dos testes e dos developers, para refinar o design e os modelos
  - Feedback do progresso da equipa desde as primeiras etapas, para refinar o planeamento e as estimativas
  - Feedback do cliente e do mercado, para re-prioritizar as features a desenvolver na fase seguinte

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

19

19

## Métodos ágeis

- Métodos ágeis
  - Não tem definição exata: práticas específicas variam bastante
- Normalmente:
  - aplicam desenvolvimento iterativo e incremental, de iterações curtas
  - usam planeamento adaptativo
  - promovem entregas incrementais
  - incluem práticas que promovem uma resposta rápida e flexível à mudança

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

20

20

## Métodos ágeis

- Métodos ágeis (continuação)
  - Princípios e práticas que promovem:
    - Inclusão do cliente em todo o processo
    - Simplicidade
    - Leveza
    - Comunicação
    - Equipas auto-organizadas
  - Não está suposto de ser interpretado de forma extremista
    - Organização e reflexão continuam a ser necessárias

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

21

21

## Unified Process

- Unified Software Development Process
- Processo de desenvolvimento de software
  - para criação e manutenção de sistemas orientados a objetos
  - iterativo
  - incremental
- Conjunto de boas práticas
  - Lifecycle iterativo
  - Guiado pelo risco
  - Descrição coesa e bem documentada

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

22

22

## Unified Process

- Desenvolvimento iterativo e evolucionário
  - Permite começar o desenvolvimento com conhecimento incompleto e imperfeito
- Vantagens do desenvolvimento iterativo e evolucionário:
  - Progride de forma lógica para uma arquitetura robusta
  - Gestão eficaz de requisitos que vão mudando
  - Integração continua
  - Compreensão precoce do sistema
  - Verificação do risco contínua

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

23

23

## Unified Process

Iterações iniciais com maiores desvios em relação aquilo que o sistema deve ser. Através do feedback e adaptação o sistema converge para os requisitos e design mais apropriados

Em iterações finais uma modificação significativa dos requisitos é rara mas não impossível. Alterações tardias podem dar uma vantagem competitiva à organização.



Uma iteração de design, implementação, integração e teste

Leonor Melo

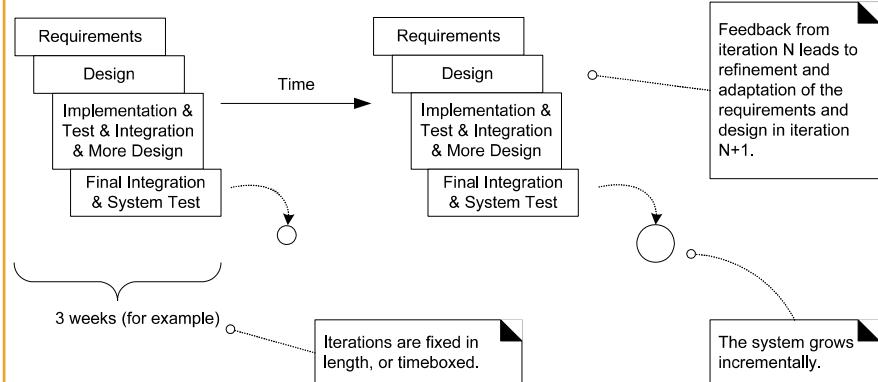
26 e 27 Processos de desenvolvimento de Software

24

24

## Unified Process

- Desenvolvimento iterativo e evolucionário:



Leonor Melo

26 e 27 Processos de desenvolvimento de Software

25

25

## Unified Process

- O UP fornece

- Orientação quanto à ordem de execução das tarefas da equipa
- Integra o trabalho individual com o trabalho da equipa
- Especifica os artefactos a produzir
- Fornece critérios de medição e monitorização do progresso

- Quem faz o quê, quando e como?

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

26

26

## RUP

- Rational Unified Process
  - Possivelmente versão mais conhecida do UP
    - Criada por uma divisão da IBM, que vende as ferramentas de apoio à metodologia
  - Promove a produtividade da equipa
  - Configurável
    - Nenhum processo de desenvolvimento é adequado a todos os sistemas de software
    - Adaptações para equipas grandes e pequenas
  - Documentação
    - Artefactos baseados em modelos
    - UML

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

27

27

## RUP

- Peças fundamentais
  - Papéis (quem)
    - responsabilidades
  - Tarefas (como)
    - Unidades de trabalho
    - Orientado a resultados – tem de ser útil
  - Produtos do trabalho (o quê)
    - Produto resultante
  - Workflows (quando)
    - Quando é que as atividades devem ser executadas

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

28

28

## Boas práticas do RUP

- Desenvolver o software de forma iterativa
  - Não é possível
    - Definir exaustivamente todo o problema antes de tudo o resto
    - Desenhar a solução completa
  - Cada iteração termina com uma release
- Gerir os requisitos
  - Casos de uso para capturar os requisitos funcionais
  - Devem ser claramente identificados e poder ser verificados

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

29

29

## Boas práticas do RUP

- Incentiva arquiteturas baseadas em componentes
- Cria modelos visuais do software
  - Diversos tipo de modelo para comunicar
    - Diferentes aspectos do sistema, dependendo do leitor a que se destinam
    - UML
- Controlo das alterações feitas ao software
  - Integração contínua

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

30

30

## Boas práticas do RUP

- Verificação da qualidade do software
- Revisões
  - Requisitos funcionais
  - Requisitos não funcionais
- Fazem parte do processo

Leonor Melo

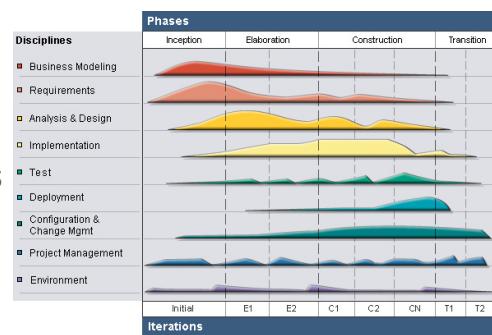
26 e 27 Processos de desenvolvimento de Software

31

31

## Processo RUP

- Eixo horizontal representa o tempo e mostra o aspecto dinâmico do processo
  - Expresso em termos de ciclos, fases, iterações e milestones
- Eixo vertical representa a vertente estática do processo
  - Descreve o processo em termos de atividades, artefactos, trabalhadores e workflows



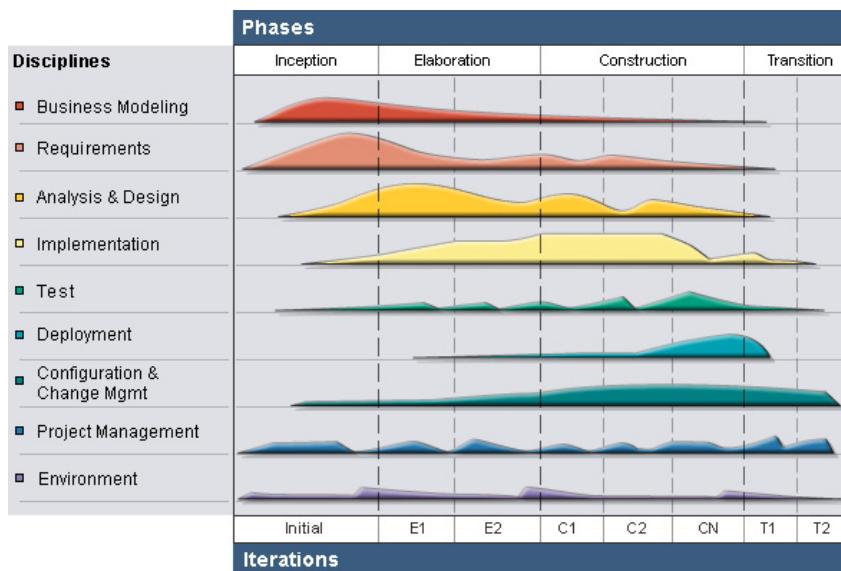
Leonor Melo

26 e 27 Processos de desenvolvimento de Software

32

32

## Processo RUP (exemplo para um pequeno projeto)



Leonor Melo

26 e 27 Processos de desenvolvimento de Software

33

33

## Fases e iterações

- Fases e iterações – o aspecto dinâmico do processo
  - O desenvolvimento é dividido em ciclos
    - Cada ciclo trabalha numa nova geração do produto
  - 4 fases:
    - *Inception* (ponto de início)
    - Elaboração
    - Construção
    - Transição

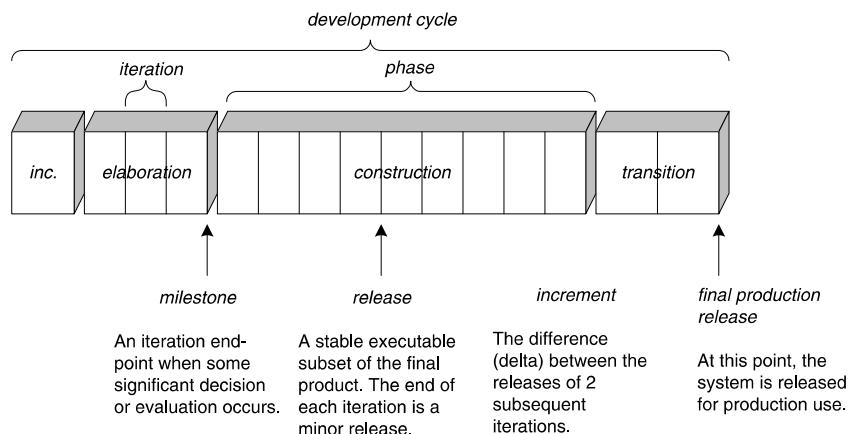
Leonor Melo

26 e 27 Processos de desenvolvimento de Software

34

34

## Fases e iterações



Leonor Melo

26 e 27 Processos de desenvolvimento de Software

35

35

## Fases do desenvolvimento

- Inception (início)
  - Âmbito do sistema
  - Intervenientes mais importantes
  - Risco, custo, etc.
- Elaboração
  - Identificação dos riscos
  - Domínio do problema
  - Análise e arquitetura
  - Continuação da identificação dos riscos

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

36

36

## Fases do desenvolvimento

- Construção
  - Maior parte do software é construído nesta fase
  - Possivelmente dividido em subfases
- Transição
  - Transição de desenvolvimento para produção
  - Testes finais
  - Documentação

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

37

37

## RUP principais workflows

- Modelação do negócio (business modeling)
  - O objetivo é compreender os conceitos e dinâmicas fundamentais do negócio desenvolvido pela organização (e que o sistema irá apoiar)
- Requisitos
  - O objetivo é definir o âmbito: o que será e o que não será desenvolvido
- Análise e Design
  - O objetivo é analisar os requisitos e desenhar uma solução

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

38

38

## RUP principais workflows

- Implementação
  - O objetivo é criar o código baseado no design: desenvolvimento dos componentes
- Teste
  - O objetivo é garantir a qualidade através da verificação todos os aspectos do sistema
- Instalação (Deployment)
  - O objetivo é planejar e entregar o sistema em funcionamento ao cliente

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

39

39

## RUP workflows de suporte

- Gestão de projeto
- Gestão de configuração
- Envolvente (*environment*):
  - Selecionar e adquirir ferramentas
  - Construir ferramentas
  - Administração de Sistema
  - Treino

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

40

40

## Exemplo de artefatos UP

| Discipline         | Artifact Iteration-*        | Incep. 11 | Elab. El. .En | Const. CL.Cn | Trans. T1..T2 |
|--------------------|-----------------------------|-----------|---------------|--------------|---------------|
| Business Modeling  | Domain Model                |           | s             |              |               |
| Requirements       | Use-Case Model              | s         | r             |              |               |
|                    | Vision                      | s         | r             |              |               |
|                    | Supplementary Specification | s         | r             |              |               |
|                    | Glossary                    | s         | r             |              |               |
| Design             | Design Model                |           | s             | r            |               |
|                    | SW Architecture Document    |           | s             |              |               |
|                    | Data Model                  |           | s             | r            |               |
| Implementation     | Implementation Model        |           | s             | r            | r             |
| Project Management | SW Development Plan         | s         | r             | r            | r             |
| Testing            | Test Model                  |           | s             | r            |               |
| Environment        | Development Case            | s         | r             |              |               |

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

41

41

## inception

- Estabelecer o âmbito do software e as condições limites
  - Conceito operacional
  - Critério de aceitação
  - Descrição do que deve (e não deve) ser incluído
- Discriminar os casos de uso críticos do sistema
  - Comportamento dos cenários principais
- Propor (pelo menos) uma arquitetura candidata
- Estimar o custo global
- Estimar o risco

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

42

42

## inception

- Atividades
  - Formular o âmbito do projeto
  - Planear e preparar o aspeto comercial e avaliar alternativas para gestão de risco, gestão de pessoal, planificação do projeto
  - Sintetizar uma proposta de arquitetura

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

43

43

## inception

- Resultados produzidos:
  - Documento de visão
    - Visão geral dos requisitos fundamentais
    - Características principais
    - Restrições principais
  - Todos os caso de uso, atores e *stakeholders* que podem ser identificados até ao momento
  - Glossário do projeto inicial
  - Proposta comercial incluindo
    - Contexto de negócio
    - Critério de sucesso
    - Previsão financeira
  - Avaliação inicial de risco

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

44

44

## inception

- Outros artefatos produzidos
  - Plano do projeto
    - Várias fases e iterações
  - Modelo de casos de uso inicial (10%-20% completo)
  - Modelo do domínio
  - Workflow do modelo de negócio (identificação dos processos de negócio, realizações, papéis e responsabilidades)
  - Descrição preliminar do processo de desenvolvimento
  - Um ou mais protótipos
    - Comportamento, estrutura, ...

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

45

45

## inception

- Milestone da inception:
  - Concordância dos stakeholders quanto à definição do âmbito e estimativa inicial de custo e calendário (que serão refinados em fases posteriores)
  - Concordância quanto a que o conjunto certo de requisitos foi capturado e que existe uma conceptualização partilhada sobre esses requisitos
  - Concordância quanto à adequação das estimativas de custo e calendário, prioridades, risco e processo de desenvolvimento
  - Concordância quanto à identificação dos riscos iniciais e à existência de uma estratégia de mitigação para cada um deles

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

46

46

## Elaboração

- Objetivos
  - Analisar o domínio do problema
  - Desenvolver o plano do projeto
  - Consolidar e detalhar os casos de uso mais críticos (os que terão maior impacto para que o projeto seja considerado um sucesso)
  - Mitigar os elementos de risco elevado e produzir um calendário e estimativa de custo mais rigoroso

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

47

47

## Elaboração

- Objetivos
  - Elaborar um plano para a fase de construção
  - Estabelecer uma fundação sólida para a arquitetura
    - Definir, validar e chegar a consenso sobre a arquitetura tão cedo quanto possível
    - Demonstrar que a arquitetura irá suportar o sistema delineado no documento de visão, mantendo um custo e um gasto de tempo razoáveis

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

48

48

## Elaboração

- Atividades

- Desenhar arquitetura e selecionar componentes de forma iterativa:
  - Avaliar potenciais componentes
  - Integrar os componentes e re-avaliar
- Decisões de fazer / comprar / reutilizar vão afetar o custo e duração da fase de construção

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

49

49

## Elaboração

- Resultados produzidos

- Modelo de caso de uso (80% completo)
  - Todos os casos de uso identificados
  - Todos os atores identificados
  - A maior parte das descrições de caso de uso escritas
- Requisitos suplementares
  - Não funcionais ou não associados com um caso de uso
- Descrição da arquitetura do software

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

50

50

## Elaboração

- Resultados produzidos
  - Protótipo de arquitetura executável
  - Lista de riscos e caso de negócio revistos
  - Plano de desenvolvimento para todo o projeto
    - Projeto de plano genérico com as iterações e critérios de avaliação para cada iteração
  - Especificação do processo de desenvolvimento atualizado
  - Manual do utilizador preliminar (opcional)

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

51

51

## Elaboração

- Milestones da elaboração
  - A visão e os requisitos estão estáveis?
  - A arquitetura está estável?
  - As abordagens a usar para teste e avaliação estão confirmadas?
  - Os testes e avaliação dos protótipos executáveis demonstraram que os elementos de maior risco foram considerados e resolvidos?
  - O plano de iterações para a fase de construção é suficientemente detalhado e realista para que o trabalho possa prosseguir?

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

52

52

## Elaboração

### • Milestones da elaboração

- Todos os stakeholders concordam que a visão documentada no documento de visão pode ser alcançada se o plano atual for executado de forma a desenvolver o sistema de acordo com a arquitetura atual?
- Os custo atuais são aceitáveis face aos custos planeados?

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

53

53

## Construção

- Todos os componentes e características da aplicação restantes são desenvolvidas e integradas no produto
- Todas as características são testadas de forma minuciosa
- O ênfase é na gestão dos recurso e controlo das operações de forma a otimizar o custo, calendário e qualidade
- Construção paralela pode acelerar a disponibilização de releases

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

54

54

## Construção

- Objetivos:
  - Minimizar o custo de desenvolvimento
  - Otimizar recursos
  - Evitar deita-fora-e-recomeça desnecessário
- Alcançar qualidade adequada tão rapidamente quanto possível
- Alcançar versões úteis (alfa, beta, ou outras) tão rapidamente quanto possível

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

55

55

## Construção

- Atividades:
  - Gestão de recursos, controlo de recursos, otimização do processo
  - Desenvolvimento de componentes completos e testados de acordo com os critérios de avaliação definidos antes
  - Verificação do produto lançado em relação aos critérios de aceitação documentados na visão

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

56

56

## Construção

- Resultados produzidos:
  - Um produto pronto a ser entregue ao utilizador final
  - Um produto integrado nas plataformas adequadas
  - Os manuais do utilizador
  - A descrição da versão atual

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

57

57

## Construção

- Milestones da construção
  - Esta versão do produto está estável e madura o suficiente para ser lançada na comunidade de utilizadores?
  - Estão todos os stakeholders prontos para a transição para a comunidade de utilizadores?
  - Os custos atuais quando comparados com os custos planeados continuam aceitáveis?
- A transição pode ter de ser adiada uma versão se alguma destas perguntas não for respondida afirmativamente

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

58

58

## Transição

- Disponibiliza o projeto de software à comunidade de utilizadores
- Depois do lançamento, normalmente aparecem problemas que requerem novas versões ou o terminar de características do software que foram sendo adiadas
- Esta fase começa quando a versão base está madura o suficiente para ser lançada no domínio do utilizador final
  - Uma versão utilizável e razoavelmente completa foi construída com um nível aceitável de qualidade e existe documentação disponível

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

59

59

## Transição

- Objetivos
  - Garantir que os utilizadores conseguem usar o sistema de forma autónoma
  - Obter a concordância dos stakeholders de que o deployment do sistema está completo e de acordo com os critérios de avaliação da visão
  - Alcançar o produto final base de forma tão rápida e económica possível

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

60

60

## Transição

- Faz parte da fase de transição
  - Teste à versão beta para validar o sistema em relação às expetativas dos utilizadores
  - Funcionamento em paralelo com o legacy system que o projeto irá substituir
  - Conversão de base de dados operacionais
  - Treino de utilizadores e responsáveis pela manutenção
  - Envio do produto para as equipas de marketing, distribuição e vendas
- Termina quando deployment completa a visão

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

61

61

## Transição

- Objetivos
  - Garantir que os utilizadores se conseguem usar o sistema de forma autónoma
  - Obter a concordância do stakeholders de que o deployment do sistema está completo e de acordo com os critérios de avaliação da visão
  - Alcançar o produto final base de forma tão rápida e económica possível

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

62

62

## Transição

- Atividades
  - Ações específicas do deployment em questão: produção e empacotamento comercial, lançamento de vendas, treino do pessoal
  - Afinação de detalhes, incluindo correção de erros e melhorias de performance e usabilidade
  - Verificação do deployment em relação à visão e critérios de aceitação do produto

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

63

63

## Transição

- Atividades (cont.)
  - As atividades dependem do objetivo:
    - Para corrigir bugs
      - Implementação e teste são geralmente suficientes
    - Para novas features
      - A iteração é semelhante a uma da fase de construção

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

64

64

## Transição

- Milestones da transição:
  - Os clientes estão satisfeitos?
  - Os custos atuais quando comparados com os planeados são aceitáveis?
    - Se não, que ações executar em projetos futuros para evitar esse problema?

Leonor Melo

26 e 27 Processos de desenvolvimento de Software

65