

Engenharia de Software I

Rogério Eduardo Garcia
(rogerio.garcia@unesp.br)

Aula 06

*To reach a port, we must sail – sail,
not tie at anchor - sail, not drift.*
Franklin D. Roosevelt


14/04/2025 Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia 1




Bacharelado em
Ciência da
Computação
2025

1

Cronograma



14/04/2025 Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia 2



BCC
2025

2

Engenharia de Software I – Aula 6

unesp

BCC

2025

Revisão

Revisão Geral

Introdução ao Método Larman

Planejar e Elaborar

Construir

Analisar

Projeto

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

3

3

Problema X Solução

unesp

BCC

2025

	Problema	Solução
Abstrato		
Concreto		

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

4

4

unesp

BCC

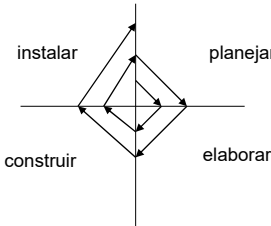
2025

Desenvolvimento Iterativo

Um ciclo de vida iterativo (CVI) envolve a repetição dos ciclos de planejamento, elaboração, construção e instalação

O sistema cresce pela adição de novas funções (e refinamento das existentes) em cada ciclo iterativo

Cada ciclo ataca um pequeno conjunto de requisitos



14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

5

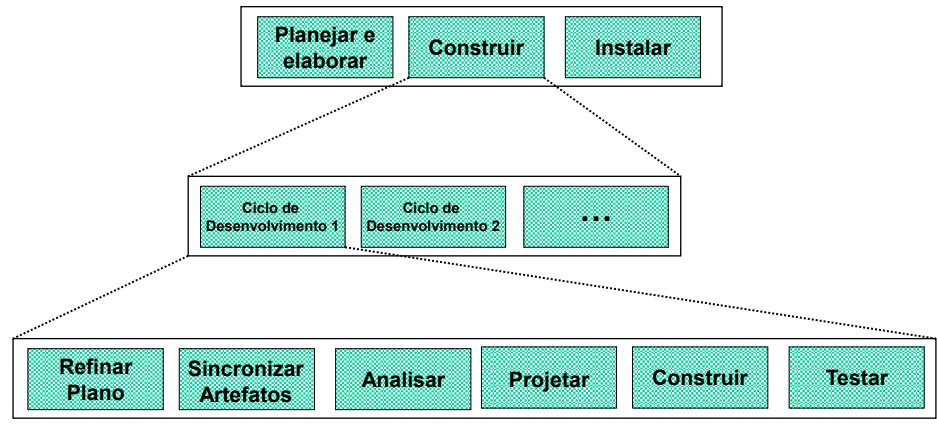
5

unesp

BCC

2025

Desenvolvimento Iterativo



14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

6

6

Analisar: Atividades



Definir/Refinar os casos de uso essenciais (se ainda não foi feito)

Definir/Refinar os diagramas de casos de uso

Definir/Refinar o modelo conceitual

Definir/Refinar o glossário

Definir/Refinar os diagramas de sequência do sistema

Definir/Refinar os contratos de operação

Definir/Refinar os diagramas de estado

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

7

7

Projetar: Atividades



Definir casos de uso reais

Definir Relatórios, Interface e Storyboards, ...

Refinar a arquitetura do sistema

Definir diagramas de interação

Definir diagramas de classe do projeto

Definir esquema da base de dados

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

8

8

Construir: Atividades



Implementar definições de classes e interfaces

Implementar métodos

Implementar interfaces com usuário

Implementar relatórios

Implementar esquema da base de dados (SQL)

Escrever casos de teste

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

9

9

Diagrama de Sequência



Um Diagrama de Sequência mostra interações de **objetos** ordenados numa sequência de tempo

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

10

10

Diagrama de Sequência - Exemplo

Como chegar aqui?

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

11

11

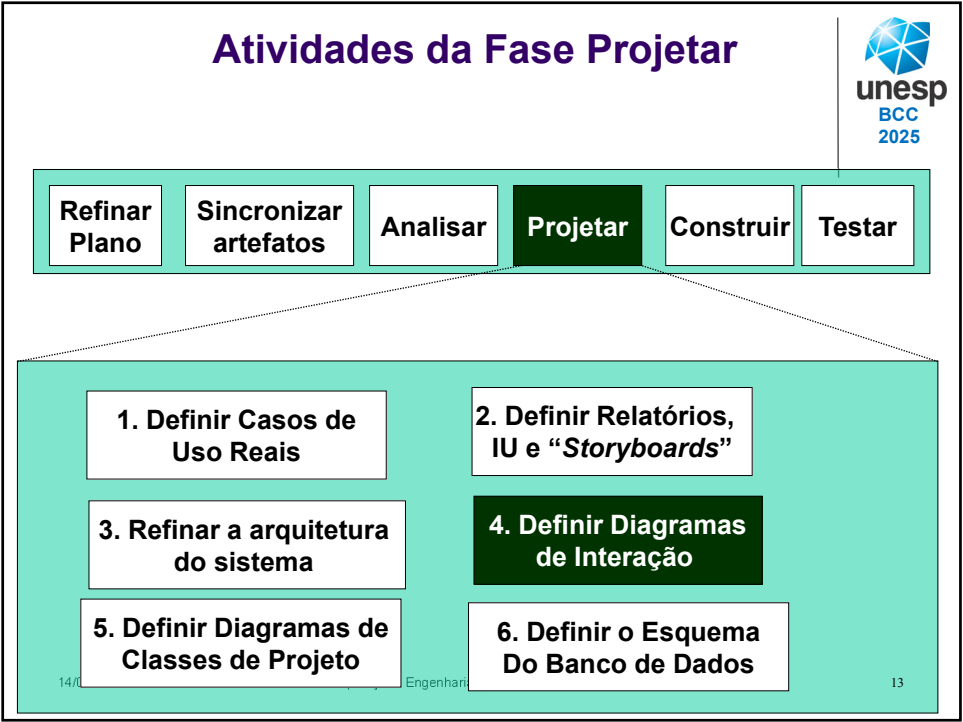
Desenvolvimento Iterativo

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

12

12



13

Início da fase Projetar

Nesta fase é desenvolvida uma solução lógica baseada no paradigma de orientação a objetos – objetos, mensagens, classes, métodos,

“Fazer Certo a Coisa” – projetar de maneira competente uma solução que satisfaça os requisitos

Os dois artefatos principais a serem desenvolvidos são:

- Diagramas de Interação
- Diagramas de Classe de Projeto

14/04/2025 Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia 14

14

Diagramas de Interação



Diagramas de Interação – apresentam como os objetos interagem, por meio de mensagens, para responder a um determinado evento

são importantes para o desenvolvimento de um bom projeto
exigem criatividade

A UML fornece dois tipos de diagramas de interação que permitem representar interação (colaboração) entre classes (ou objetos):

diagramas de colaboração – formato de grafo

diagramas de sequência – formato de cerca



14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

15

15

Diagrama de Colaboração



Diagrama de Colaboração foca tanto a interação quanto os *links* entre um conjunto de objetos que “colaboram” entre si;

O Diagrama de sequência e o de Colaboração mostram as interações, mas o de sequência foca o tempo e o de colaboração foca o espaço.

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

16

16

Diagrama de Colaboração

```
graph LR; msg1[mensagem1()] --> A[":Instância da Classe A"]; A -- "1:mensagem2()" --> B[":Instância da Classe B"]; A -- "2:mensagem3()" --> B;
```

- Os diagramas de colaboração têm melhor capacidade de expressar informações contextuais e podem ser mais econômicos em termos de espaço

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

17

17

Diagrama de Sequência

```
sequenceDiagram; participant A as :instância de Classe A; participant B as :instância de Classe B; participant C as :instância de Classe C; A->>A: mensagem1(); A->>B: 1:mensagem2(); A->>B: 2:mensagem3(); B->>C: 3:mensagem4();
```

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

18

18


Diagramas de Colaboração

Diagrama de Colaboração - um dos artefatos mais importantes criados no projeto OO

o tempo gasto na sua criação deveria absorver um percentual de tempo significativo do tempo gasto no projeto

Elaborar diagramas de colaboração exige conhecimento de:

- princípios de atribuição de responsabilidades
- padrões de projeto




14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

19

19

Classes e Instâncias



Classe	Instância	Instância nomeada
Venda	: <u>Venda</u>	<u>venda1:Venda</u>

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

20

20

Mensagens



Sintaxe UML:

retorno := mensagem (parâmetro : tipoParâmetro) : tipoRetorno

O retorno pode não existir

Os tipos podem ser omitidos se forem óbvios ou não forem importantes

Exemplo:

especificacao := obterEspecificacaoProduto(id)

especificacao := obterEspecificacaoProduto(id:IdItem)

especificacao := obterEspecificacaoProduto(id:IdItem) : EspecificacaoProduto

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

21

21

Mensagens



No paradigma de orientação a objetos:

Mensagem é o mecanismo de comunicação entre objetos
invoca as operações desejadas

“O processo de invocar um método é chamado de envio de uma mensagem ao objeto”

Ex: quando uma mensagem **façaAlgo()** é enviada a um objeto *obj*, o método **façaAlgo()** definido na classe de *obj* é executado:

obj . façaAlgo ()

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

22

22

unesp
BCC
2025

Ligações

- Ligação: conexão em dois objeto que indica a possibilidade de alguma forma de navegação ou de visibilidade entre eles
 - permite que mensagens fluam de um objeto para outro

```
graph LR; TPV[":TPV"] -- "total:=totalizarVenda():float" --> Venda[":Venda"]
```

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

23

23

unesp
BCC
2025

Mensagens Múltiplas

Várias mensagens, em ambos os sentidos, podem fluir ao longo de uma mesma ligação

- usar seta para indicar direção da mensagem
- usar números de sequência para indicar ordem de execução das mensagens

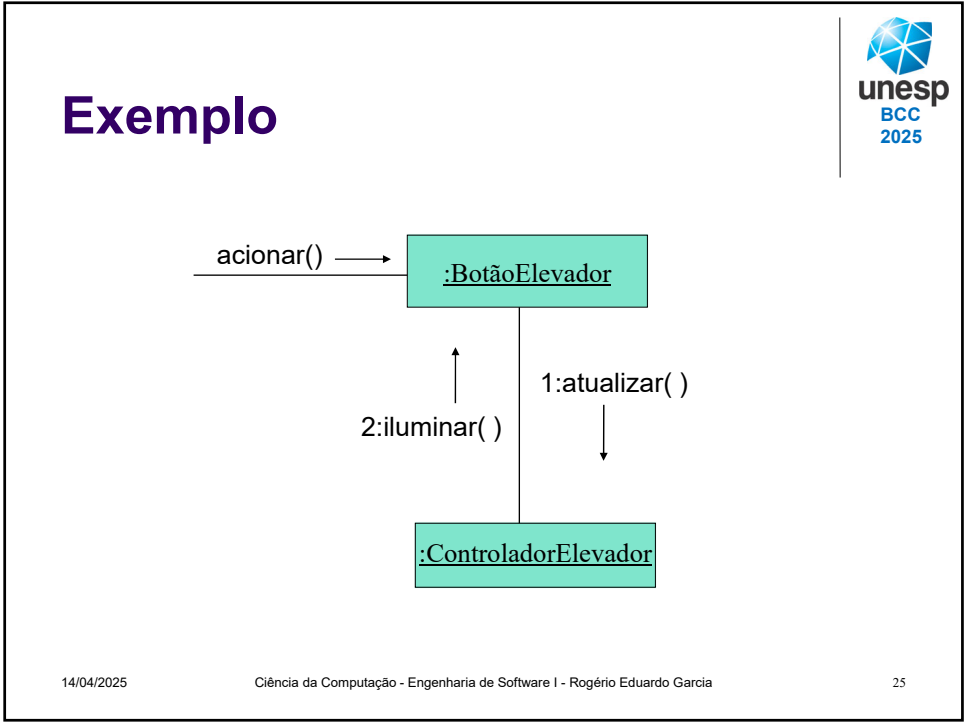
```
graph LR; A[":Instância  
Classe A"] -- "1:mensagem1()  
2:mensagem2()  
3:mensagem3()" --> B[":Instância  
Classe B"]; B -- "4:mensagem4()" --> A
```

14/04/2025

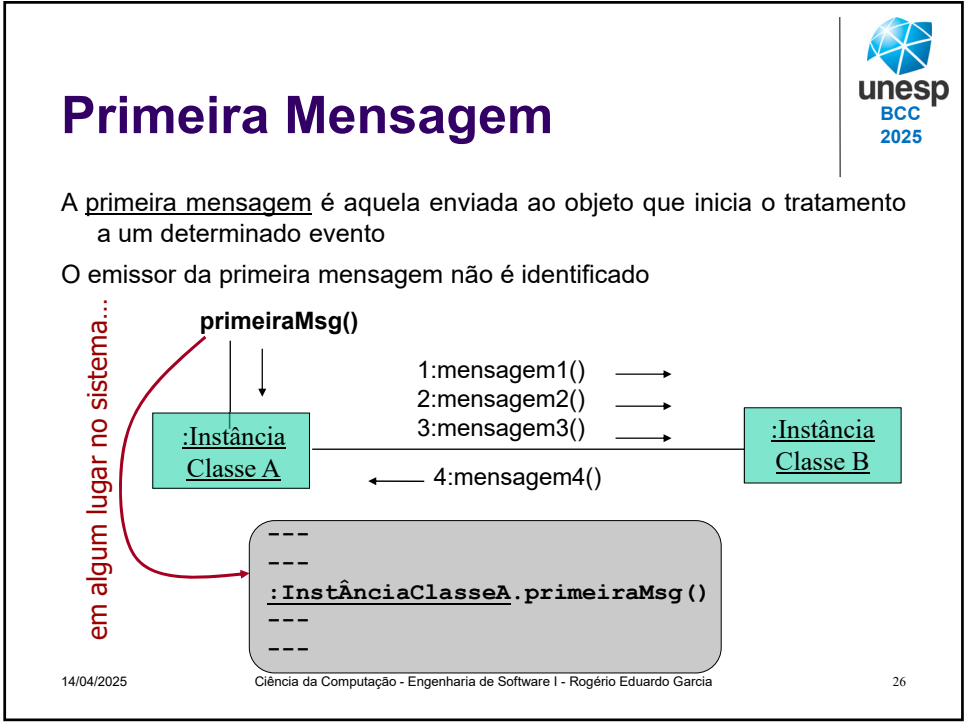
Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

24

24



25



26

Exemplo

```
sequenceDiagram
    participant Aeronave as :Aeronave
    participant Flape as :Flape
    Aeronave->>Flape: 1:ok :=posicionarAngulo(angulo):Booleano
    note over Aeronave: Aterrissar()
```

unesp
BCC
2025

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

27

27

Número de Sequência da Mensagem

Primeira mensagem ⇒ não é numerada

Numeração formal e agregada indica a ordem e o aninhamento das mensagens

```
sequenceDiagram
    participant A as :ClasseA
    participant B as :ClasseB
    participant C as :ClasseC
    A->>A: msg1()
    A->>B: 1:msg2()
    B->>C: 1.1:msg3()
    note over B,C: aninhamento de mensagens
```

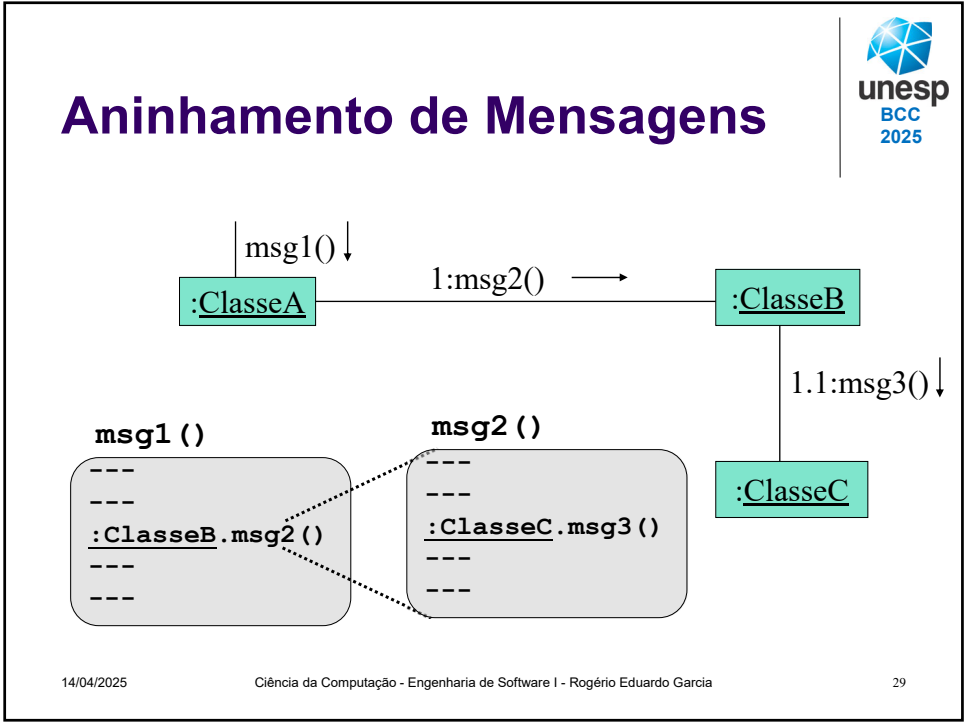
unesp
BCC
2025

14/04/2025

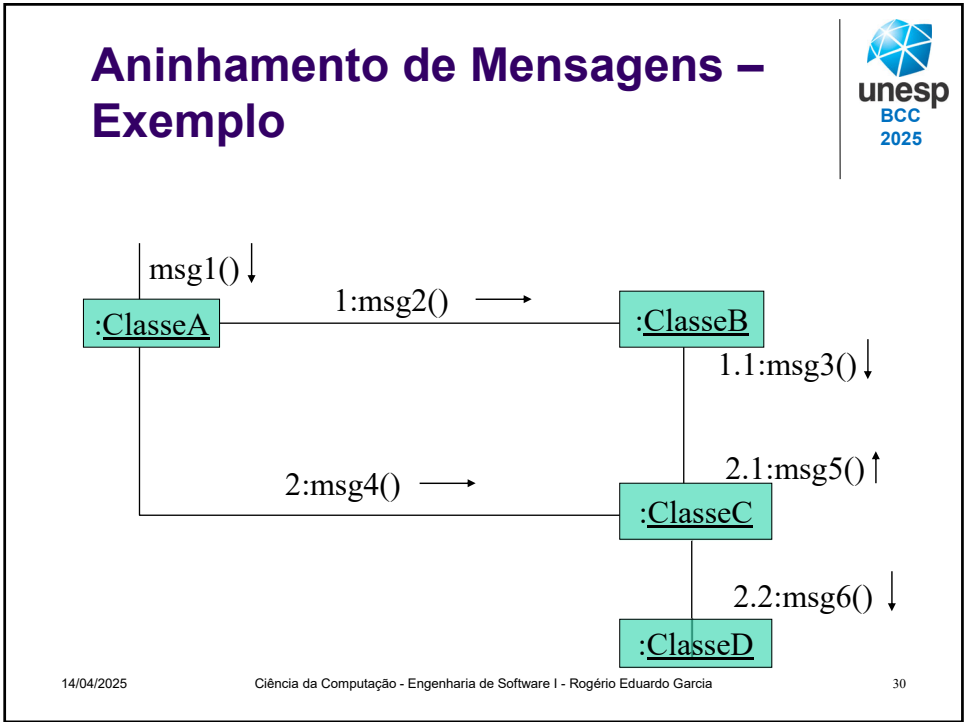
Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

28

28




29



30

Aninhamento de Mensagens – Exemplo (cont.)



em algum lugar no sistema... `:ClasseA.msg1()`

msg1 ()

```
---
---
:ClasseB.msg2 ()
---
---
:ClasseC.msg4 ()
---
---
```

msg2 ()

```
---
---
:ClasseC.msg3 ()
---
---
```

msg4 ()

```
---
---
:ClasseB.msg5 ()
---
---
:ClasseD.msg6 ()
---
---
```


14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

31

31

Auto-Mensagem (this)



msg1() ↓

:ClasseA

1: msg2() ↓

iniciarSistema() ↓

:TPV

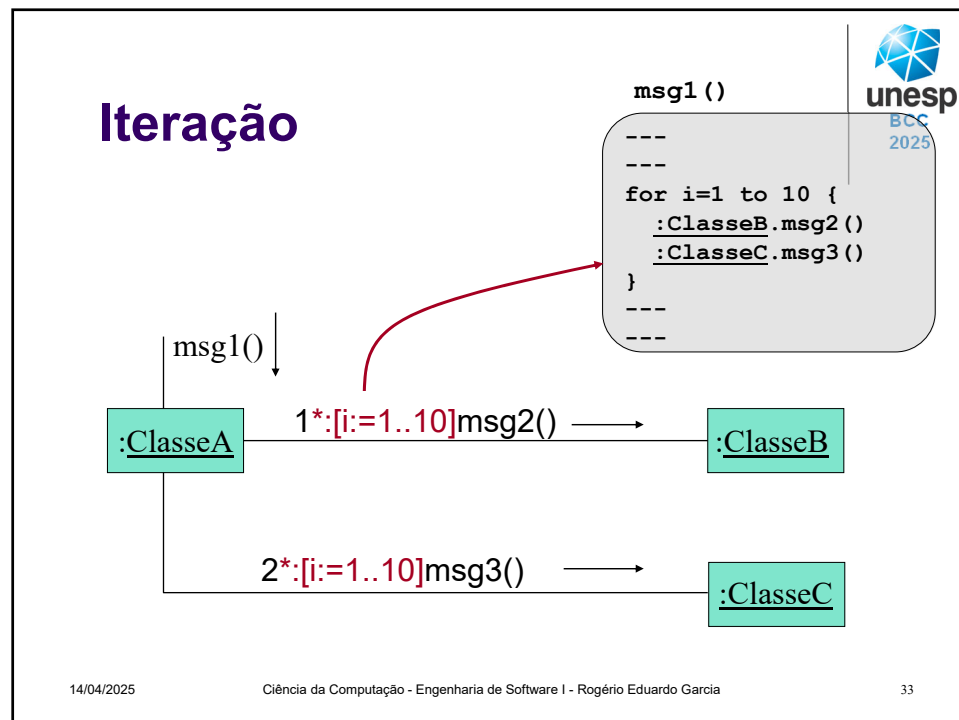
1: iniciar() ↓

14/04/2025

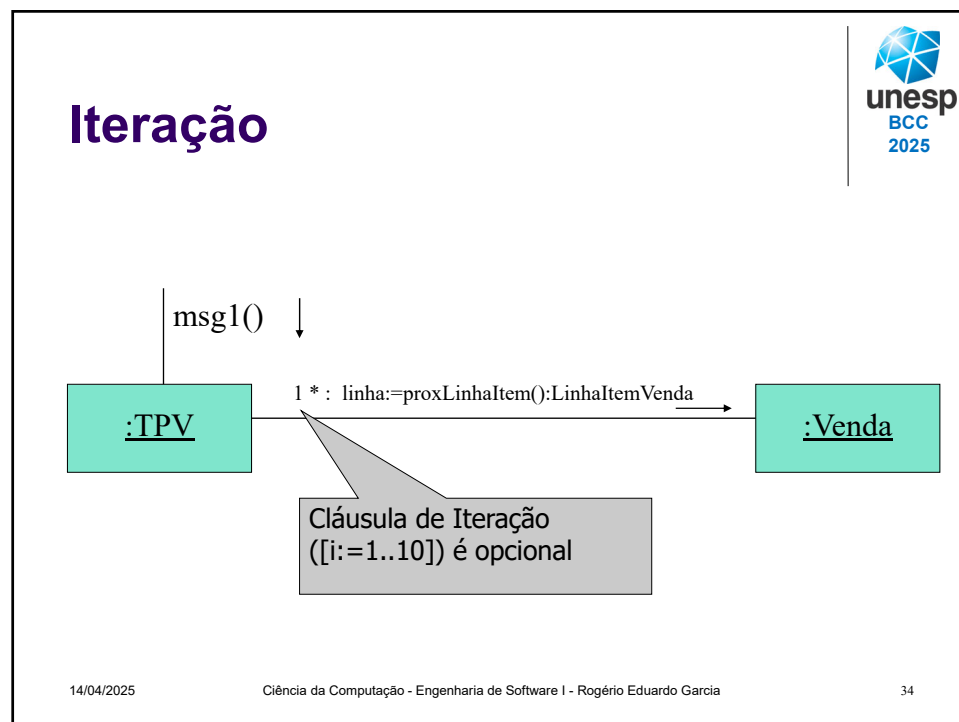
Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

32

32



33



34

unesp
BCC
2025

Criação de Instância

- Uma mensagem pode ser usada para criar uma instância
 - nomear mensagem como *criar()*

```
sequenceDiagram
    participant TPV as :TPV
    TPV->>TPV: iniciarVenda()
    TPV->>Venda as 1:criar(caixa)
    activate Venda
```

mensagem *criar()*, com parâmetros *opcionais*, normalmente é interpretada, na implementação, como chamada a um construtor da classe de software

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

35

35

unesp
BCC
2025

Mensagens Condicionais

- Uma mensagem condicional só será enviada se a cláusula entre [] tiver valor *true*

```
sequenceDiagram
    participant ClasseA as :ClasseA
    ClasseA->>ClasseA: msg1()
    ClasseA->>ClasseB as 1: [condição] msg2()
    activate ClasseB
```

msg1 ()

```
---
if (condicao == true) {
    :ClasseB.msg2 ()
}
---
```

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

36

36

Mensagens Condicionais

```
sequenceDiagram
    participant TPV as :TPV
    participant Venda as :Venda
    TPV->>Venda: 1: [nova venda] criar()
    activate TPV
    activate Venda
    deactivate Venda
    deactivate TPV
```

unesp
BCC
2025

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

37

37

Caminhos Condicionais Mutuamente Exclusivos

- Apenas uma ou outra mensagem é enviada dependendo da condição ser verdadeira ou falsa

```
sequenceDiagram
    participant ClasseA as :ClasseA
    participant ClasseB as :ClasseB
    participant ClasseC as :ClasseC
    ClasseA->>ClasseB: 1a: [condição] msg2()
    ClasseA->>ClasseC: 1b:[not condição] msg3()
    activate ClasseA
    activate ClasseB
    deactivate ClasseB
    activate ClasseC
    deactivate ClasseC
    deactivate ClasseA
```

```
msg1()
---
if (condicao = true)
    :ClasseB.msg2()
else :ClasseC.msg3()
---
---
```

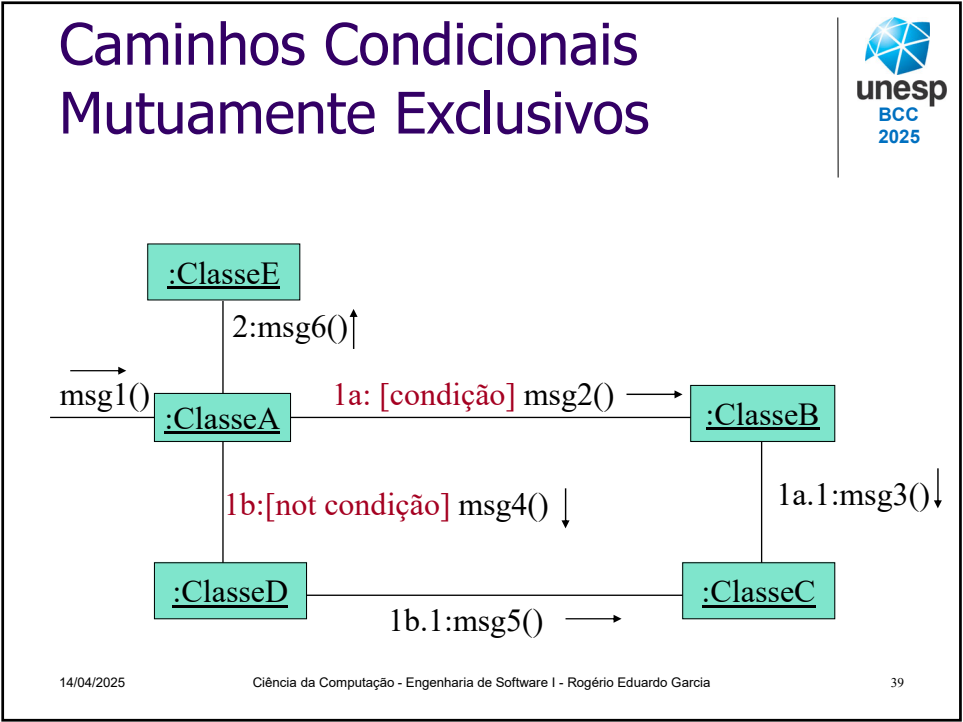
unesp
BCC
2025

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

38

38



39

Multiobjetos

Um multiobjeto é uma coleção de instâncias, representada por um único ícone

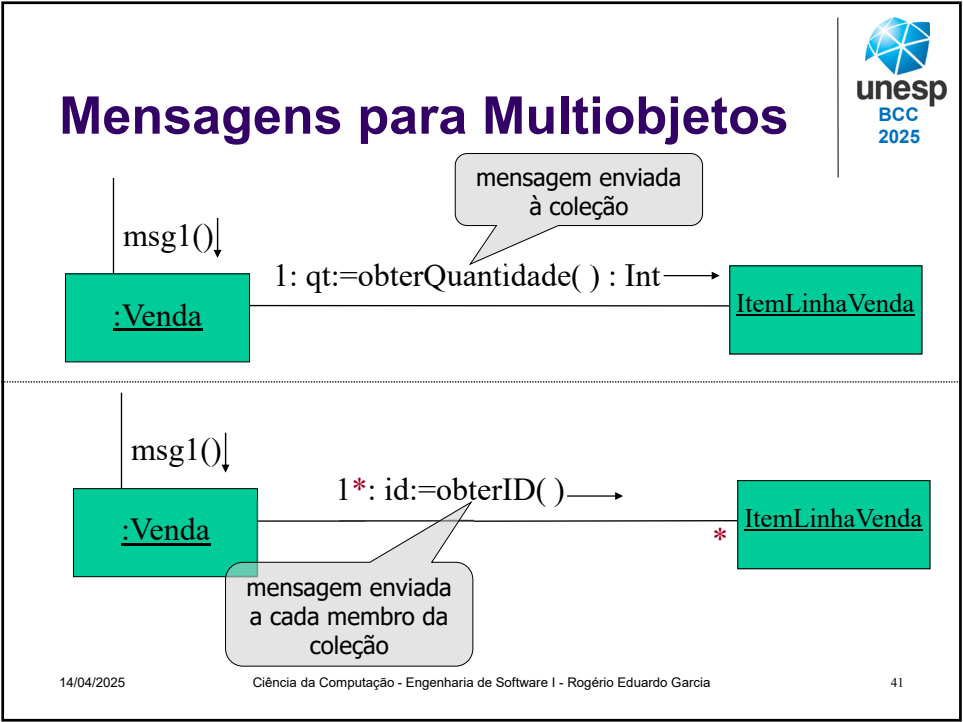
Uma mensagem pode ser enviada ao multiobjeto ou a cada membro da coleção

14/04/2025

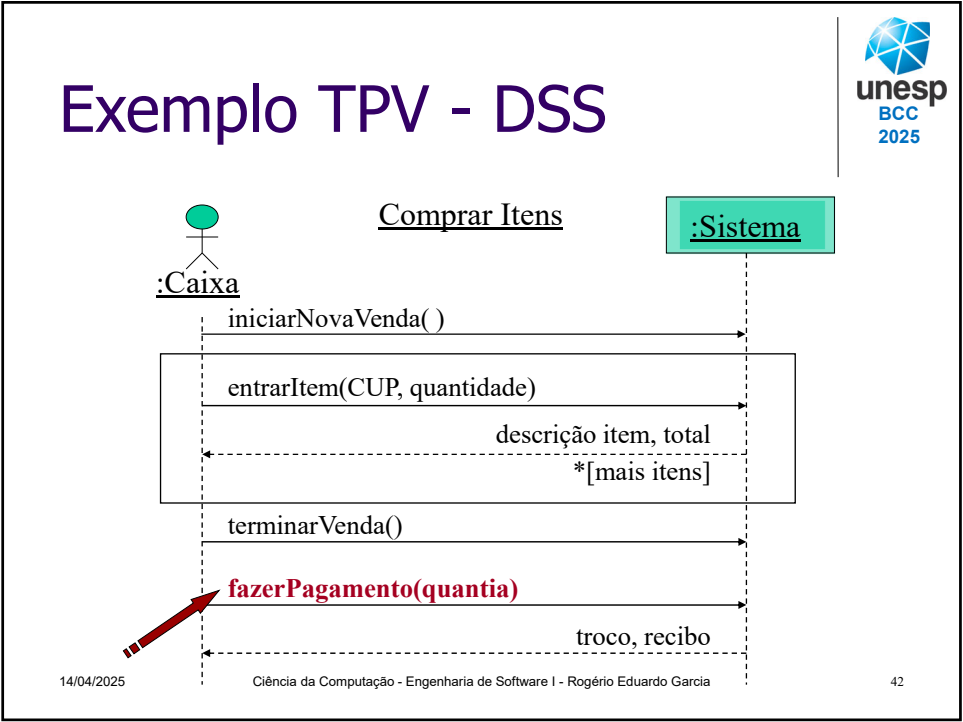
Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

40

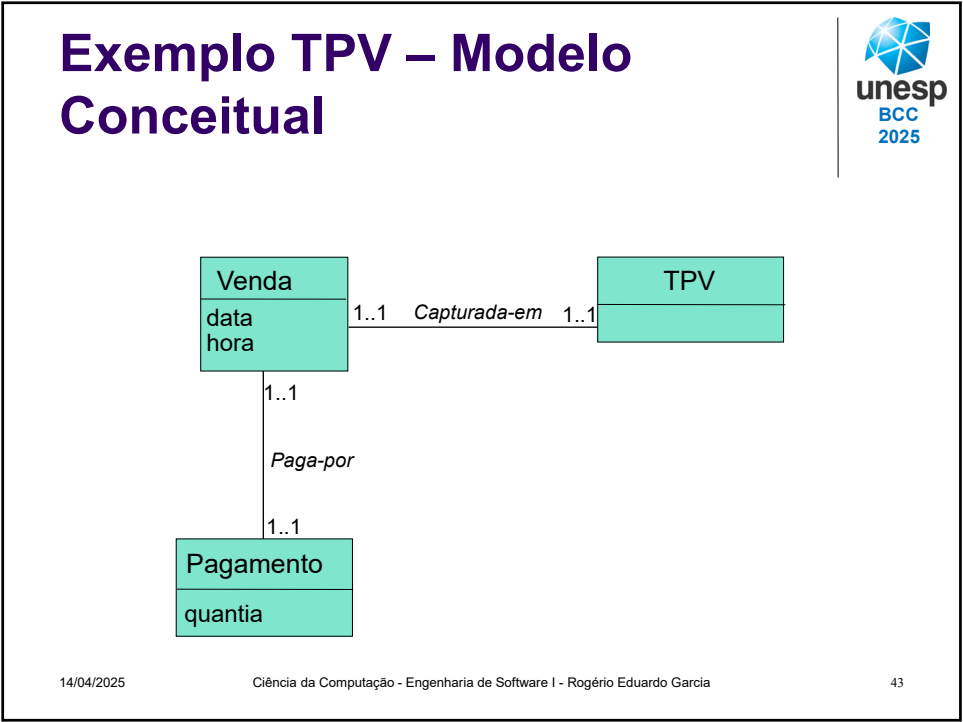
40



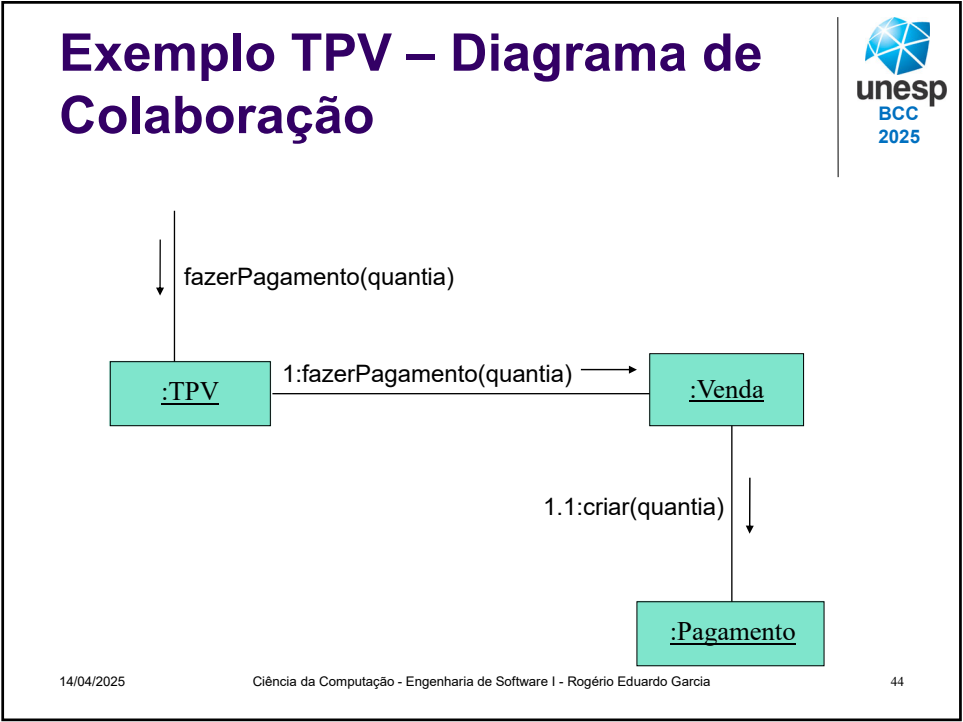
41



42



43



44

Responsabilidade



Responsabilidade

- um contrato ou obrigação de um tipo ou classe
- serviços fornecidos por um elemento (classe ou subsistema)
- incorpora os objetivos de um elemento

Dois tipos de responsabilidades básicas:

Fazer

- fazer algo (criar um objeto, executar uma operação,...)
- iniciar ações em outros objetos
- coordenar e controlar atividades em outros objetos

Saber

- conhecer dados privados encapsulados
- conhecer objetos relacionados
- conhecer coisas que podem ser derivadas ou calculadas

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

45

45

Responsabilidade



Exemplos:

“uma *Venda* é responsável por criar *ItemLinhaVenda* ” – FAZER

“uma *Venda* é responsável por conhecer o seu total”- SABER

OBS: responsabilidades do tipo saber frequentemente podem ser deduzidas do modelo conceitual (atributos e associações)

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

46

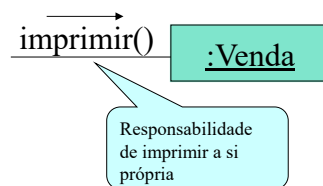
46

Responsabilidades e Diagramas de Colaboração



Diagramas de colaboração mostram escolhas de atribuição de responsabilidade a objetos

Exemplo: atribuir aos objetos do tipo *Venda* a responsabilidade de imprimirem a si próprios.



14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

47

47

Responsabilidade



A “tradução” de responsabilidade em classes e métodos é influenciada pela granularidade da responsabilidade. Ex:

“Fornecer acesso a bancos de dados relacionais” – pode envolver muitas classes e métodos

“Imprimir uma venda” – pode envolver apenas um ou alguns poucos métodos

Os métodos são implementados para satisfazer uma responsabilidade

Um objeto pode colaborar com outros objetos para atender a uma responsabilidade

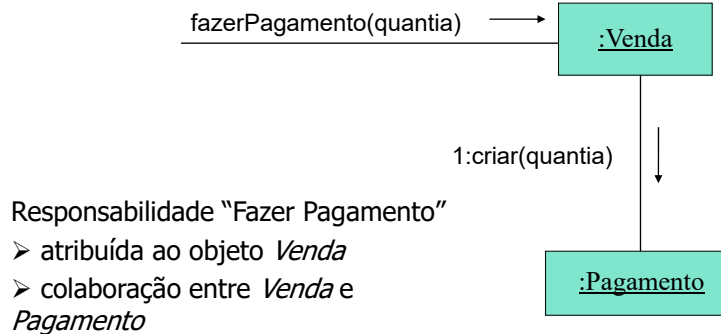
14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

48

48

Exemplo



14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

49

49

Visibilidade entre Objetos



Visibilidade: capacidade de um objeto ver ou fazer referência a outro

Para que um objeto A envie uma mensagem para o objeto B, é necessário que B seja visível para A

Tipos de visibilidade

por atributo: B é um atributo de A

por parâmetro: B é um parâmetro de um método de A

localmente declarada: B é declarado como um objeto local em um método de A

global: B é, de alguma forma, globalmente visível.

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

50

50


Visibilidade por atributo

Persiste por muito tempo

É a forma mais comum

Geralmente se deve às associações existentes no modelo conceitual

Ex: TPV tem um atributo para poder enviar mensagens ao Catálogo de Produtos.




14/04/2025

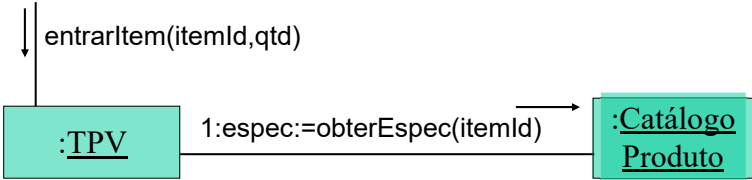
Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

51

51

Exemplo





```
sequenceDiagram
    participant TPV as :TPV
    participant Catálogo as :Catálogo Produto
    TPV->>TPV: entrarItem(itemId, qtd)
    TPV->>Catálogo: 1:espec:=obterEspec(itemId)
```

```
class TPV
{
    ...
    private CatálogoProduto catalogo;
    ...
    public void entrarItem(...);
    ...
}
```


```
public void entrarItem(ID itemId, int qtd)
{
    EspecificacaoProduto espec;
    ...
    espec = catalogo.obterEspecificacao(itemId);
    ...
}
```

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

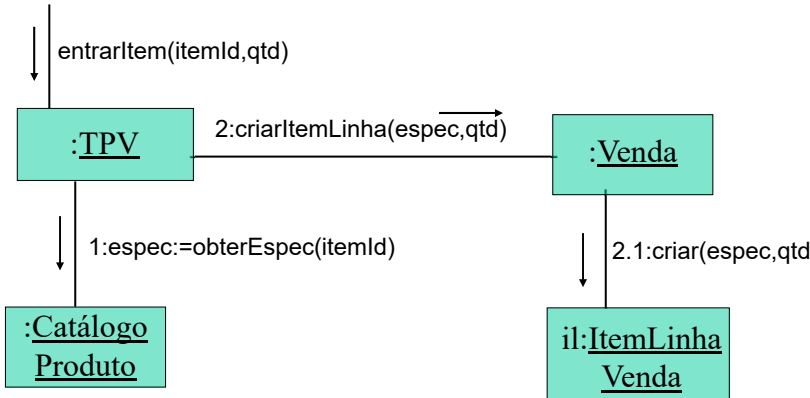
52

52



Visibilidade por parâmetro

É relativamente temporária, persiste enquanto persistir o método.



```
sequenceDiagram
    participant TPV as :TPV
    participant Catálogo as :Catálogo Produto
    participant Venda as :Venda
    participant ItemLinha as il:ItemLinha Venda


    TPV->>Catálogo: 1:espec:=obterEspec(itemId)
    TPV->>Venda: 2:criarItemLinha(espec, qtd)
    Venda->>ItemLinha: 2.1:criar(espec, qtd)
```

14/04/2025

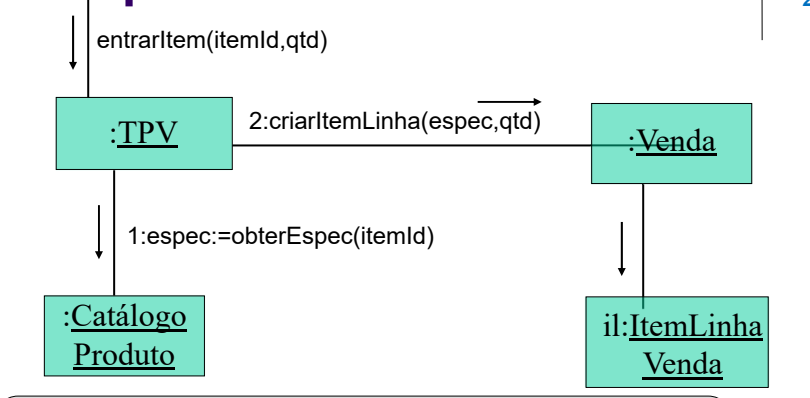
Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

53

53



Exemplo



```
sequenceDiagram
    participant TPV as :TPV
    participant Catálogo as :Catálogo Produto
    participant Venda as :Venda
    participant ItemLinha as il:ItemLinha Venda

    TPV->>Catálogo: 1:espec:=obterEspec(itemId)
    TPV->>Venda: 2:criarItemLinha(espec, qtd)
    Venda->>ItemLinha: 2.1:criar(espec, qtd)
```

```
public void criarItemLinha(EspecificaçãoProduto espec, int qtd)
{
    ...
    li = new ItemLinhaVenda(espec, qtd);
    ...
}
```

espec é passada como parâmetro para que *Venda* tenha visibilidade de *EspecificaçãoProduto*.

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

54

54

Visibilidade localmente declarada



Visibilidade relativamente temporária

Duas formas:

- criar uma nova instância local e atribuí-la a uma variável local
- atribuir o objeto retornado pela invocação de um método a uma variável local.

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

55

55

Exemplo

criação de instância local

```
public void criarItemLinha(EspecificacaoProduto espec, int qtd)
{
    ...
    ItemLinhaVenda li = new ItemLinhaVenda(espec, qtd);
    ...
}
```

atribuição de objeto a variável local

```
public void entrarItem(ID itemId, int qtd)
{
    EspecificacaoProduto espec;
    ...
    espec = catalogo.obterEspecificacao(itemId);
    ...
}
```

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

56


56

Visibilidade Global

Menos comum

Relativamente permanente (persiste enquanto A ou B existirem)

Forma óbvia e menos desejável: atribuir uma instância de objeto a uma variável global.



14/04/2025

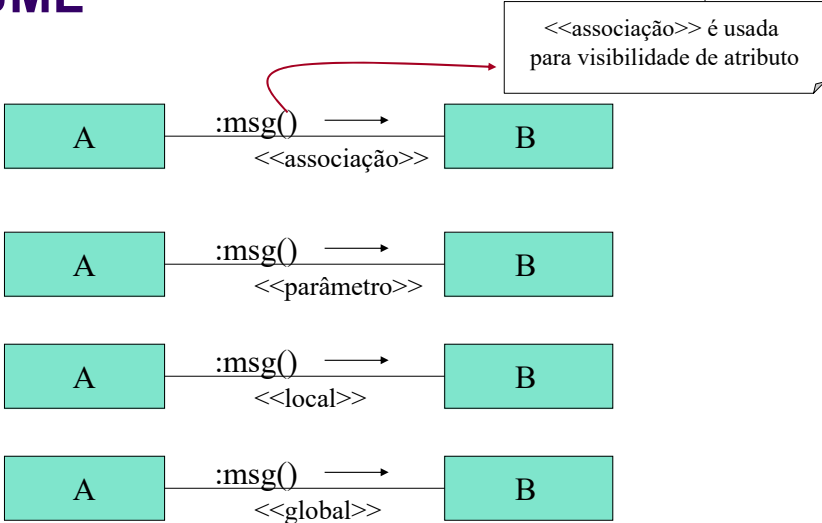
Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

57

57

Notação de Visibilidade em UML

<<associação>> é usada para visibilidade de atributo



14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

58

58

Criando Diagramas de Colaboração



Atribuir responsabilidades e criar os diagramas de colaboração são as atividades mais criativas da fase de projeto.

Não há soluções mágicas ou não justificadas. Elas devem ser baseadas em raciocínio lógico.

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

59

59

Criando Diagramas de Colaboração



O caso de uso sugere eventos do sistema → diagramas de sequência do sistema

Os eventos de sistema representam mensagens que iniciam diagramas de colaboração

Os diagramas de colaboração ilustram como os objetos interagem para realizar tarefas

interação por mensagens de objetos de software, cujos nomes podem ser inspirados pelos nomes dos conceitos (objetos) do Modelo Conceitual

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

60

60

Criando Diagramas de Colaboração



Crie um diagrama separado para cada operação do sistema
para cada evento do sistema, crie um diagrama com o evento como a primeira mensagem

Se o diagrama se tornar complexo, separe-o em diagramas menores

Use os padrões GRASP para apoiar as decisões sobre atribuição de responsabilidades

Use o modelo conceitual como apoio
conceitos podem inspirar criação de classes de software com organização similar

Comece escolhendo a classe controladora do evento

Exemplo sistema TPV...

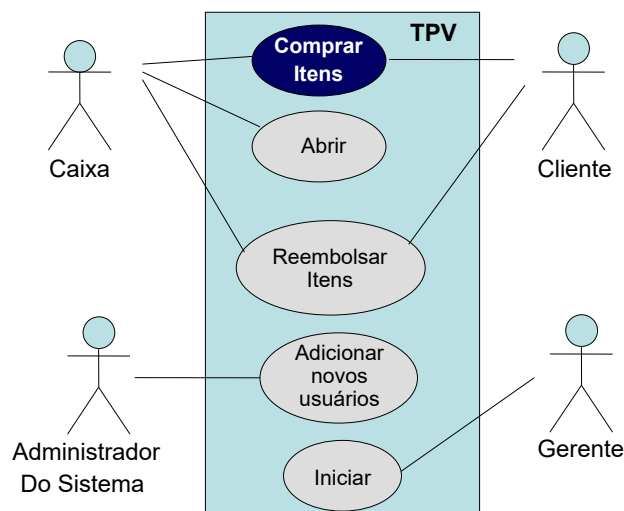
14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

61

61

Diagrama de Casos de Uso

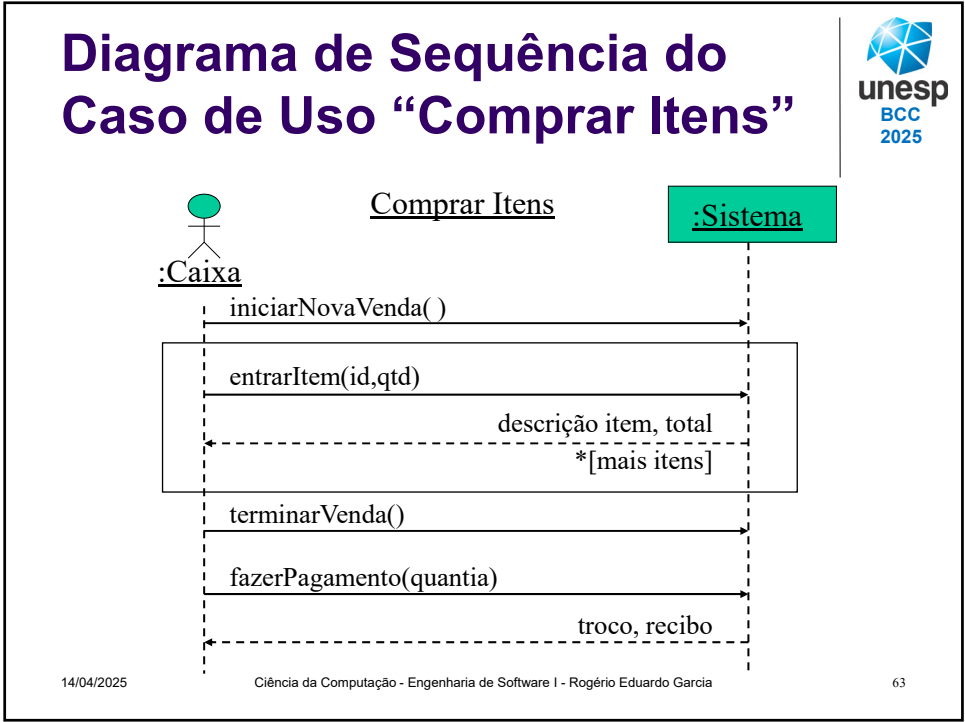


14/04/2025

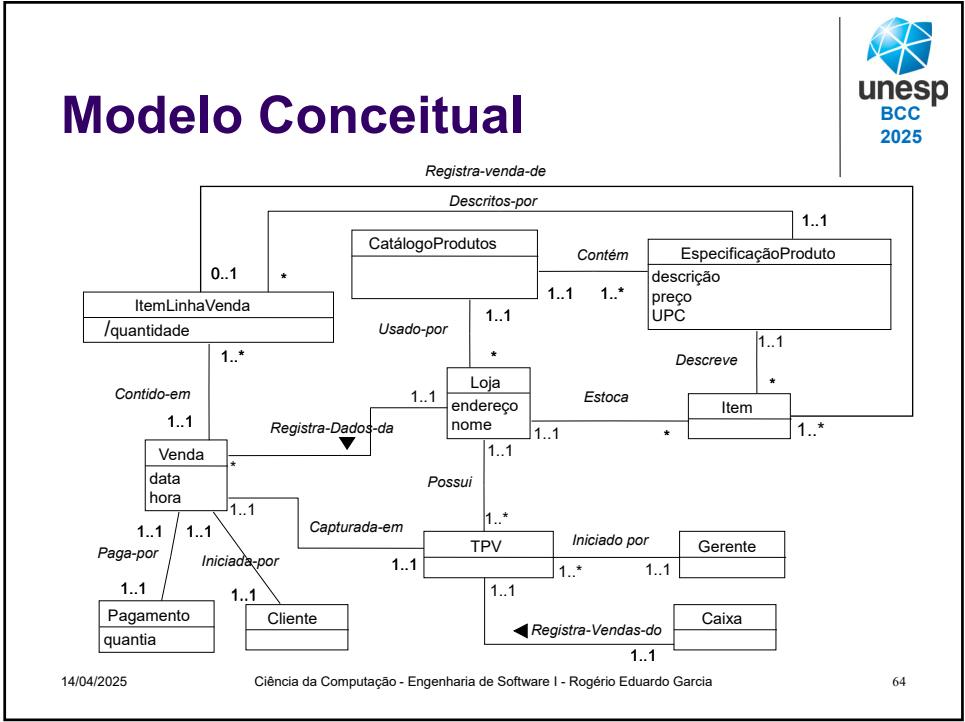
Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

62

62

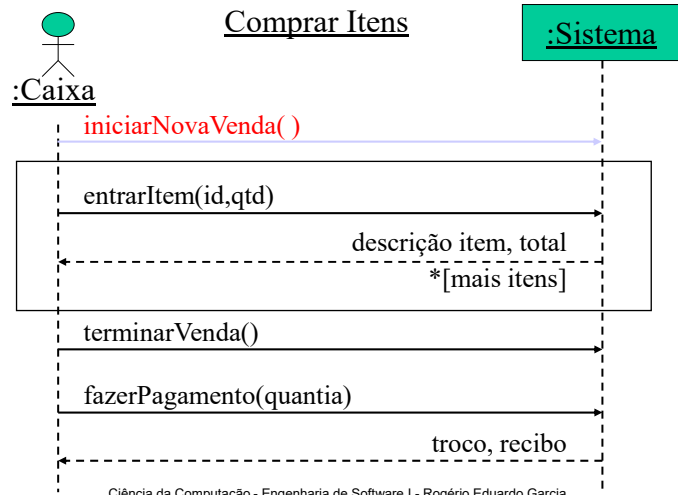


63



64

Diagrama de Sequência do Caso de Uso “Comprar Itens”



14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

65

65

Diagrama de Colaboração – iniciarNovaVenda



Quem é a classe controladora?

Opções:

representando todo o sistema, um dispositivo ou um subsistema

TPV, SistemaTPV

representando um tratador de todos os eventos de um cenário de caso de uso

ControladorDeComprarItens

TratadorDeComprarItens


14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

66

66

Diagrama de Colaboração – iniciarNovaVenda



Classe controladora (de acordo com padrão Controlador): TPV

`iniciarNovaVenda()` → `:TPV`

Observação:

- na fase de projeto, o TPV é um objeto no “mundo do software”, e não o registrador físico (como no Modelo Conceitual)
- mesmo nome foi escolhido para diminuir gap semântico


14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

67

67

Diagrama de Colaboração – iniciarNovaVenda



Criar objeto Venda

- padrão Criador sugere atribuir a responsabilidade à classe que agrega, registra ou contém o objeto a ser criado
- pelo Modelo Conceitual: TPV registra Venda
- TPV é bom candidato para criar Venda

Pelo Modelo Conceitual: Venda possui vários ItemLinhaVenda. Então, quando uma Venda é criada, uma coleção vazia de ItemLinhaVenda deve ser criada para armazenar futuras instâncias

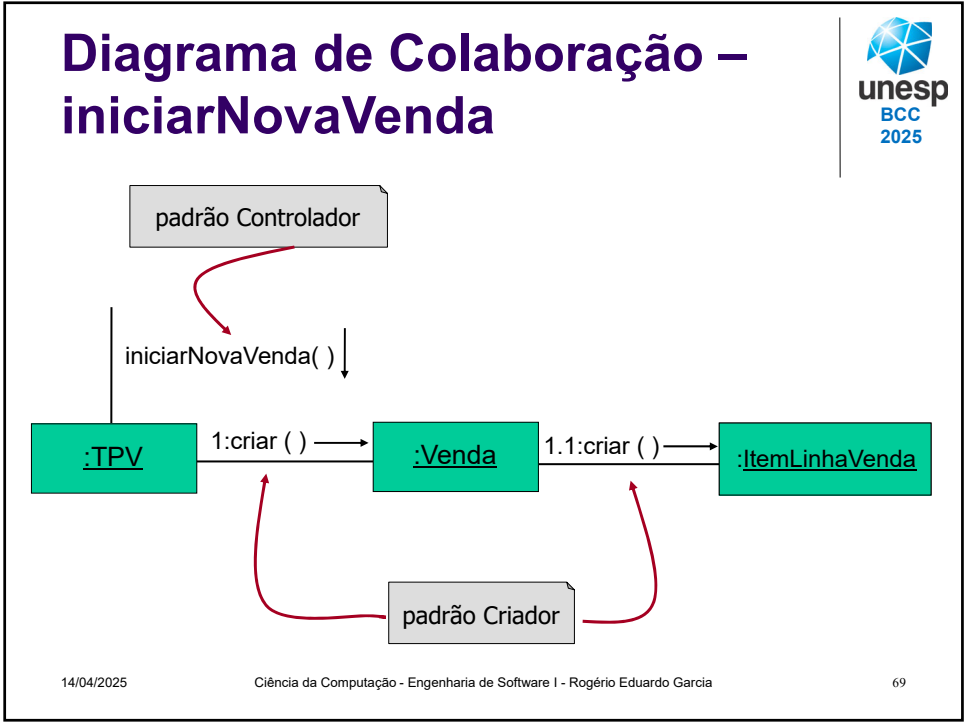
- pelo padrão Criador, Venda é um bom candidato para criar a coleção

14/04/2025

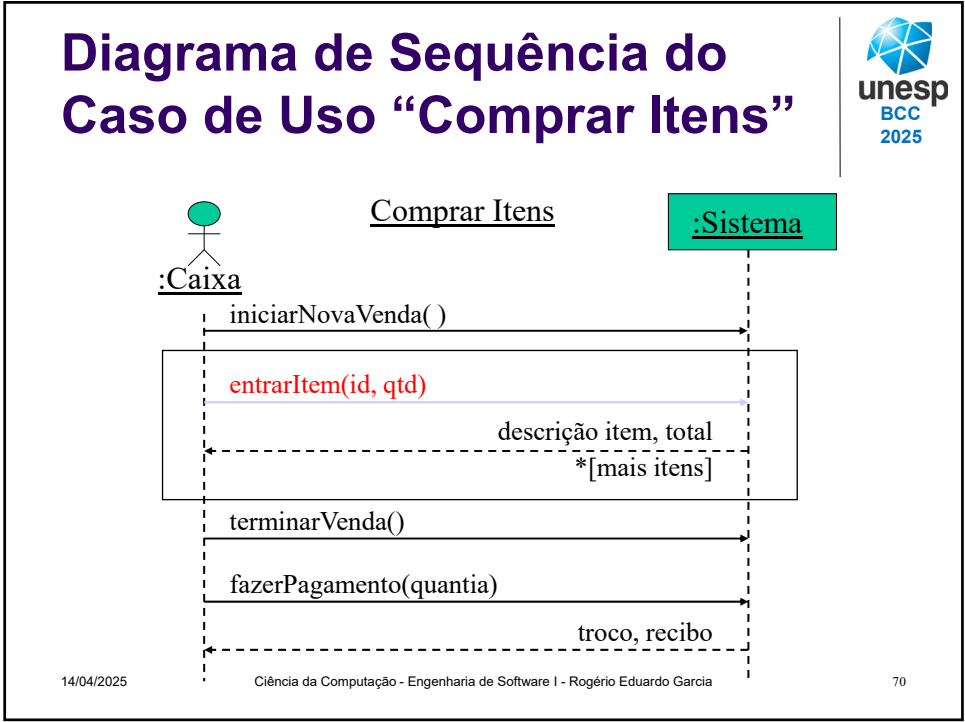
Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

68

68




69



70

Diagrama de Colaboração – entrarItem



Classe controladora (de acordo com padrão Controlador): TPV

```
graph TD
    TPV[TPV] -- "entrarItem(id, qtd)" --> TPV
```

A cada tipo de item vendido está associado um ItemLinhaVenda na Venda

padrão Criador: Venda é candidata para criar ItemLinhaVenda

cada linha de venda criada será inserida na coleção de ItemLinhaVenda associada à Venda


14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

71

71

Diagrama de Colaboração – entrarItem



Pelo Modelo Conceitual: *ItemLinhaVenda* tem quantidade e está associado a *EspecificaçãoProduto*

como obter a informação de especificação de produto por meio do ID do item?

quem é o especialista nesta informação??

Modelo Conceitual: o *CatálogoProdutos* contém todas as especificações → pelo padrão Especialista, o *CatálogoProduto* é bom candidato para a responsabilidade de obter a especificação

quem vai enviar a mensagem *obterEspecificação* para o objeto *CatálogoProduto*???

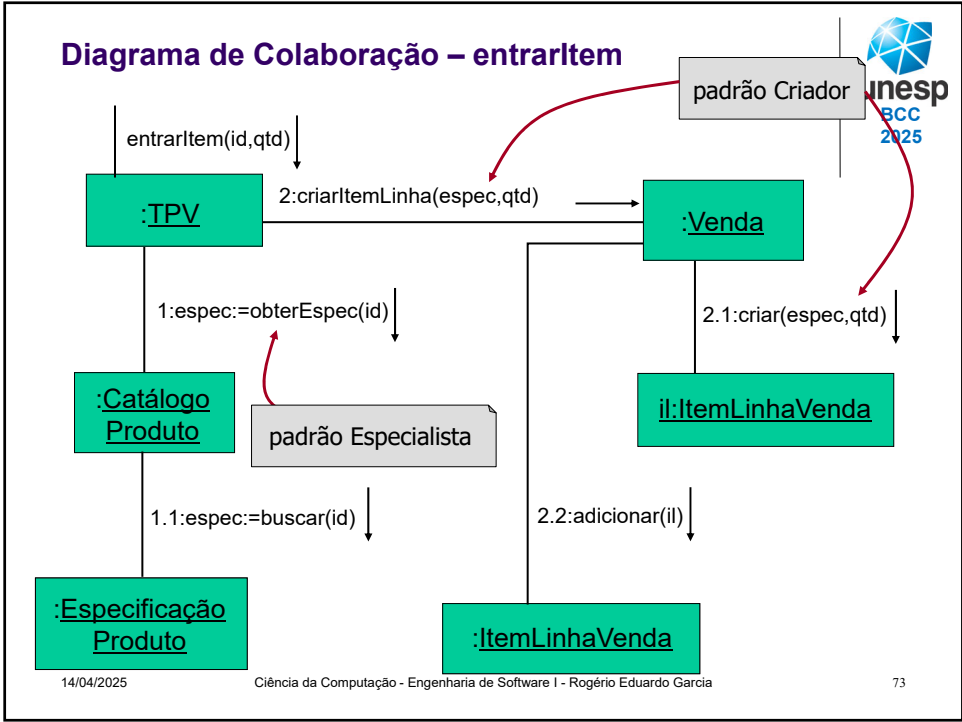
é razoável assumir conexão permanente entre *CatálogoProduto* e *TPV* ?

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

72

72



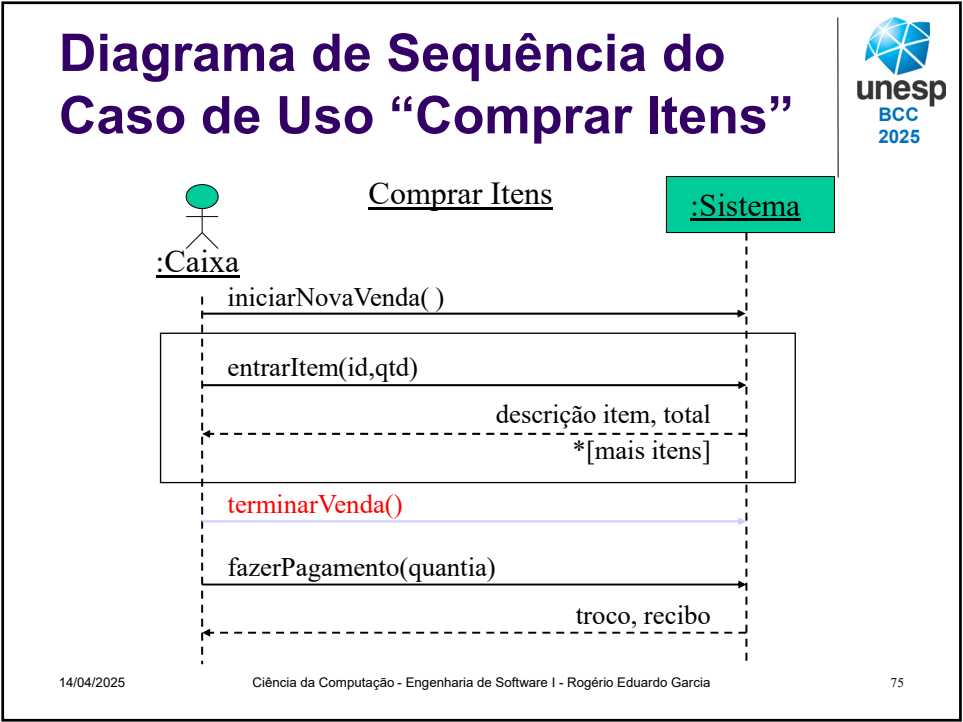
73

Diagrama de Colaboração – entrarItem

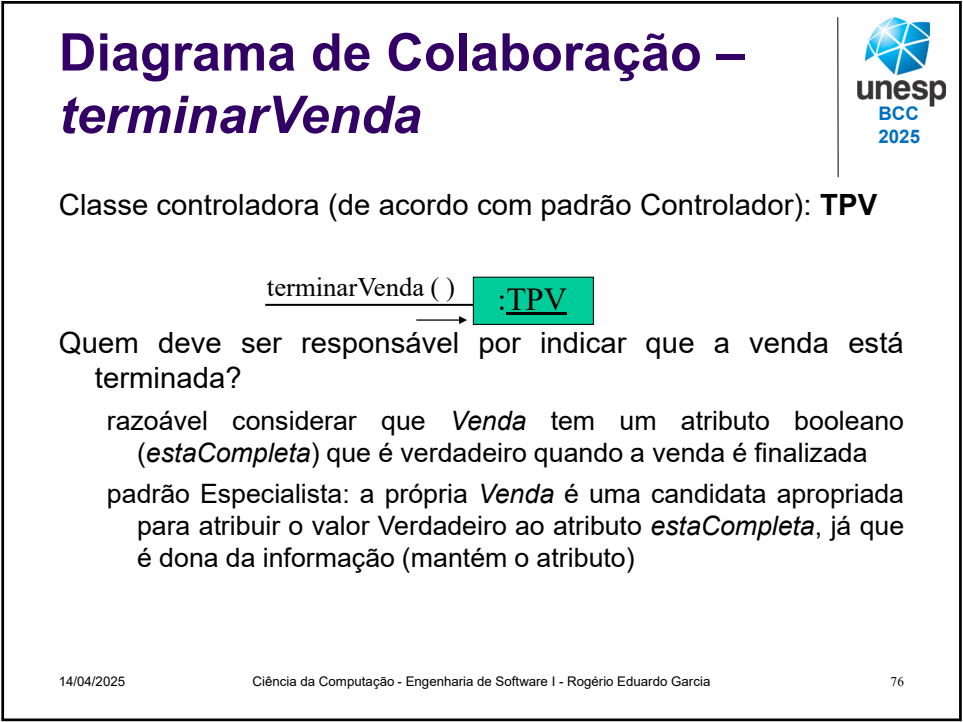
E quanto às informações descrição e preço do item que, segundo o diagrama de sequência do sistema, devem ser exibidas?

- não são responsabilidade dos objetos do domínio as tarefas relacionadas a operações de saída do sistema
- isso deverá ser feito pela camada de apresentação
- tudo o que é necessário com relação às responsabilidades de exibição de informação é que esta seja conhecida e esteja disponível nos objetos do domínio

74

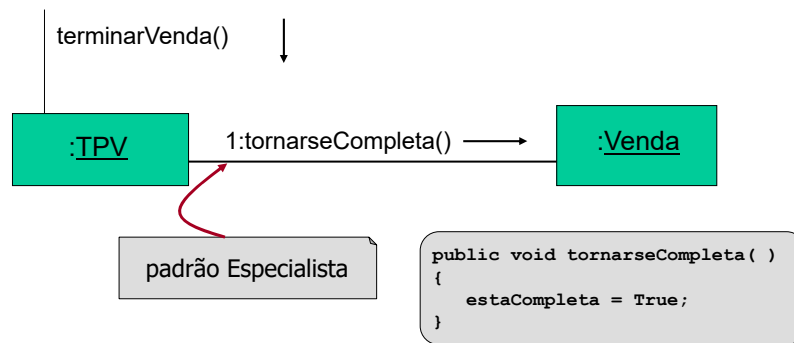


75



76

Diagrama de Colaboração – terminarVenda



14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

77

77

Diagrama de Colaboração – *obterTotal*



Terminada a venda, o sistema deve apresentar o total
(de acordo com a sequência típica de eventos do
caso de uso *Comprar Itens*)

Análise Lógica:

1. definir a responsabilidade: quem deve ser responsável por saber o total da venda?
2. resumir informações requeridas
 - total da venda é a soma dos subtotais de todos os itens de linha de venda
 - o subtotal de cada item de linha de venda é dado por: quantidade de itens X preço item


14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

78

78

Diagrama de Colaboração – *obterTotal*



Análise Lógica (cont.):

3. listar classes que conhecem as informações necessárias

Informação requerida	Especialista
preço do produto	<i>EspecificaçãoProduto</i>
quantidade de itens	<i>ItemLinhaVenda</i>
todos os <i>ItemLinhasVendas</i> da <i>Venda</i>	<i>Venda</i>


14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

79

79

Diagrama de Colaboração – *obterTotal*



Quem deve ser responsável por calcular o total da *Venda*?

pelo padrão Especialista: *Venda* (conhece todos os *ItemLinhaVenda*)

O cálculo do total requer cálculo do subtotal de cada *ItemLinhaVenda*. Quem deve ser responsável por calcular esse subtotal?

pelo padrão Especialista: *ItemLinhaVenda* (conhece quantidade e *EspecificaçãoProduto*)

O cálculo do subtotal requer o preço do produto. Quem deve ser responsável por fornecer o preço?

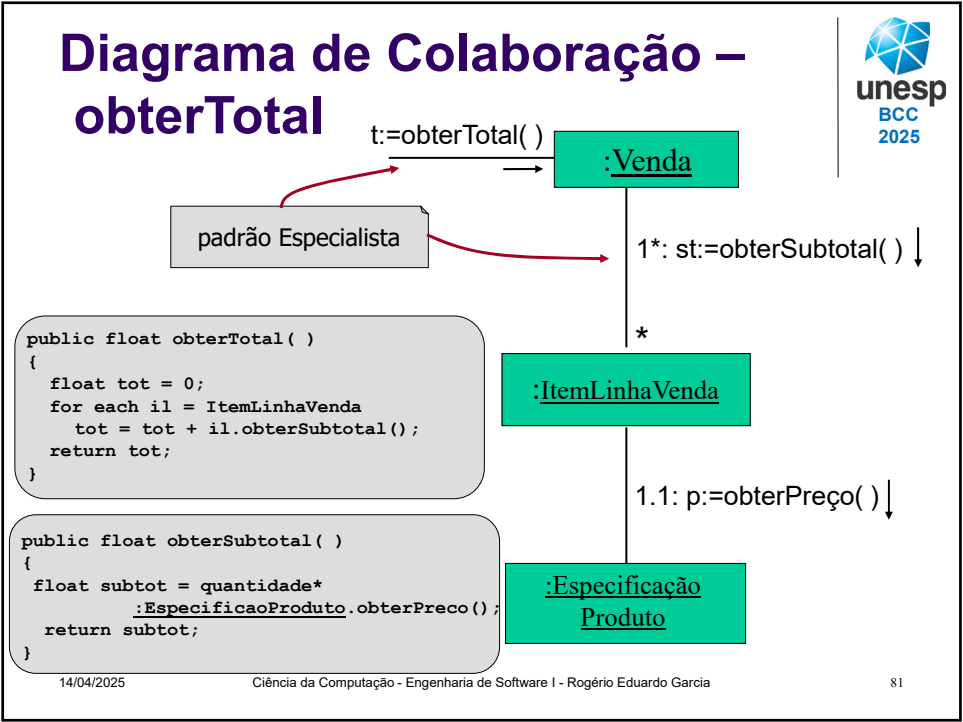
pelo padrão Especialista: *EspecificaçãoProduto* (conhece preço)

14/04/2025

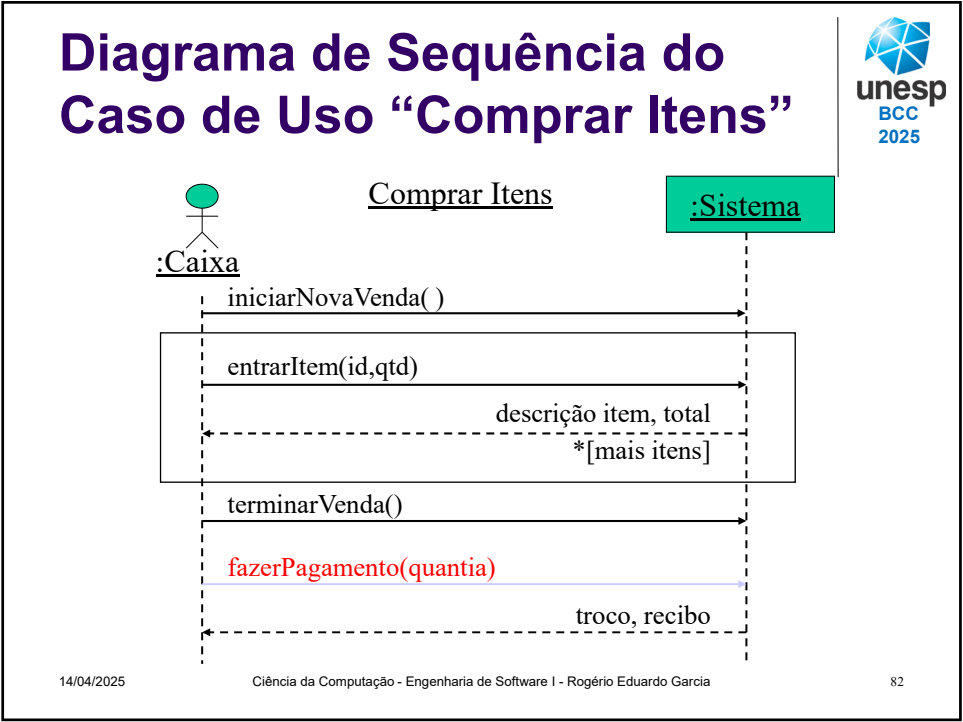
Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

80

80



81



82

Diagrama de Colaboração – *fazerPagamento*



Classe controladora (de acordo com padrão Controlador): **TPV**

fazerPagamento(quantia) → **:TPV**

Quem deve ser responsável pela criação de *Pagamento*? Opções:

padrão Criador

TPV – registra *Pagamento* e tem o valor inicial (quantia) para criação da instância

Venda – usa, de maneira bem próxima, *Pagamento*

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

83

83

Diagrama de Colaboração – *fazerPagamento*



Avaliando as opções – considerar Coesão e Acoplamento

escolha de *Venda* :

pelo Modelo Conceitual - já existe uma associação entre *Venda* e *Pagamento*

trabalho de *TPV* fica mais leve → aumenta coesão

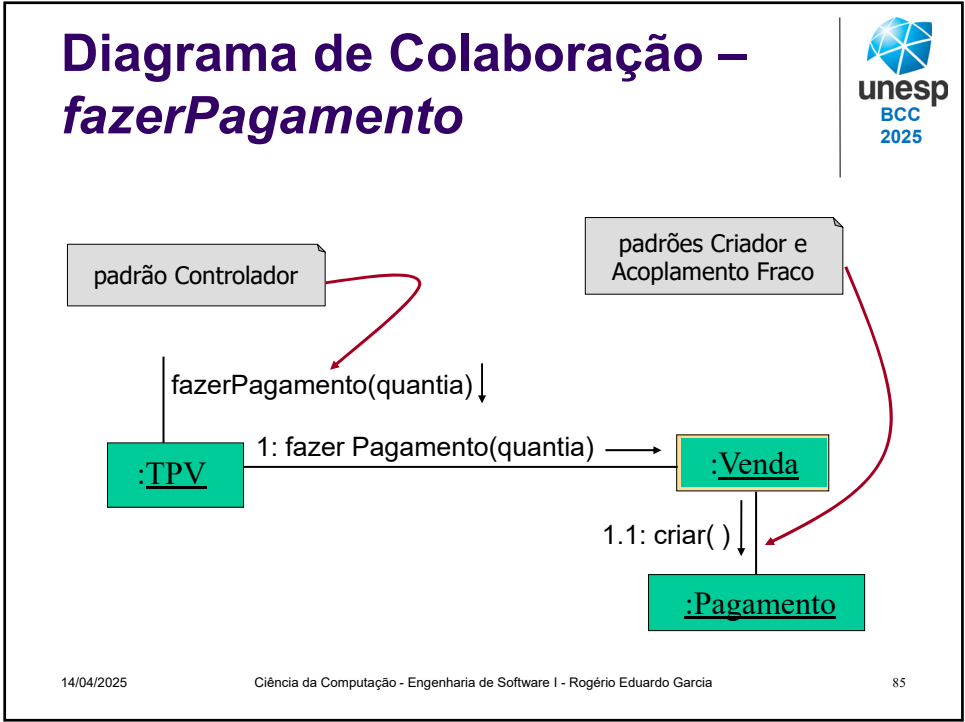
TPV não toma conhecimento da existência de *Pagamento* → favorece baixo acoplamento

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

84

84



85

Diagrama de Colaboração – *fazerPagamento*

The diagram illustrates the collaboration for the *fazerPagamento* use case. It features three lifelines: **:TPV**, **:Venda**, and **:Pagamento**. The process begins with a message from **:TPV** to **:Venda** labeled "1: fazer Pagamento(quantia)". Subsequently, **:Venda** sends a message to **:Pagamento** labeled "1.1: criar()". Two notes are present: "padrão Controlador" with a red arrow pointing to the initial message, and "padrões Criador e Acoplamento Fraco" with a red arrow pointing to the "1.1: criar()" message. The UNESP BCC 2025 logo is in the top right corner.

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

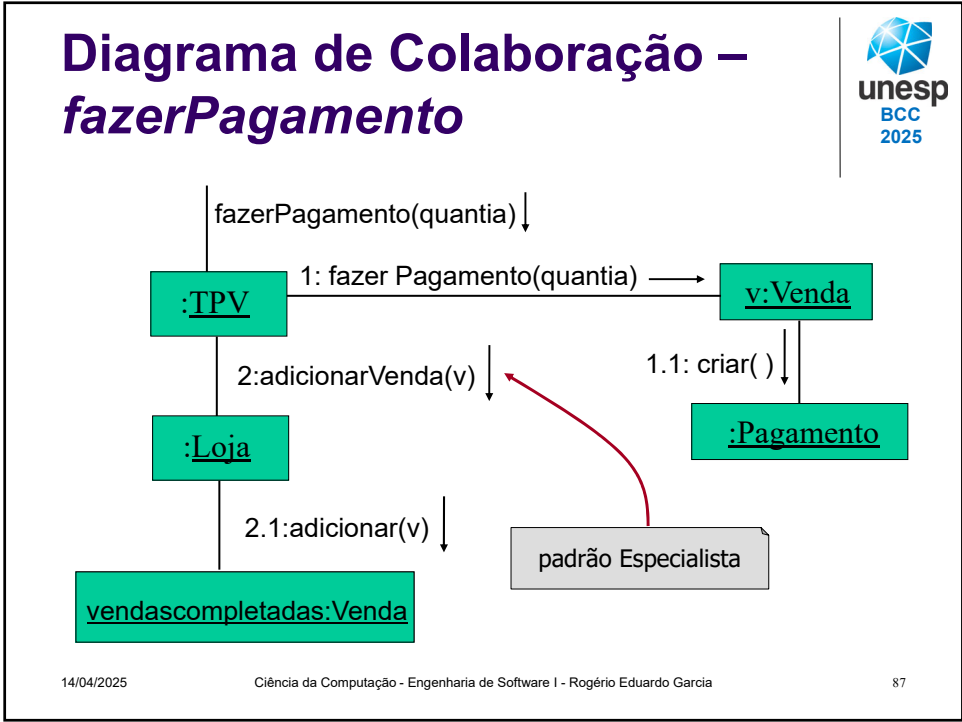
86

A venda completada deve ser registrada, por exemplo, num arquivo histórico da loja (ver Modelo Conceitual)

Quem deve ser responsável por conhecer todas as vendas registradas e por registrá-las?

padrão Especialista: a *Loja* conhece e registra todas as vendas

86



87

Diagrama de Colaboração – calcularTroco

Caso de Uso *Comprar Itens* requer que seja devolvido o troco e apresentado o recibo

a forma de exibição do troco e impressão do recibo não cabe aqui, mas a informação tem que ser conhecida por objetos do domínio


Quem deve ser responsável por saber/calcular o troco?

padrão Especialista: *Venda* (conhece total da compra) e *Pagamento* (conhece quantia paga) são especialistas parciais

14/04/2025 Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia 88

88

Diagrama de Colaboração – calcularTroco



Avaliando as opções:

Pagamento : possui quantia paga, mas precisa pedir total da compra para *Venda*

Pagamento precisará ter visibilidade de *Venda* (o que ainda não tem) → aumento no acoplamento

Venda : possui total da compra, mas precisa solicitar quantia paga a *Pagamento*

Venda já tem visibilidade de *Pagamento* (pois é seu criador) → não causa aumento no acoplamento, e portanto é a solução mais desejável


14/04/2025

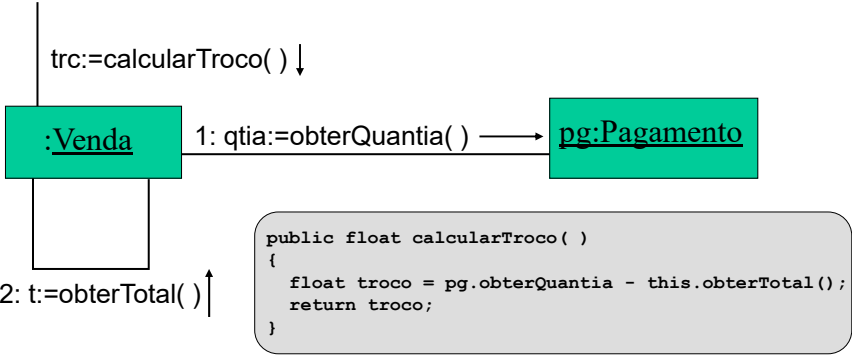
Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

89

89

Diagrama de Colaboração – calcularTroco





```
trc:=calcularTroco() ↓

:Venda
1: qtia:=obterQuantia() → pg:Pagamento
2: t:=obterTotal() ↑

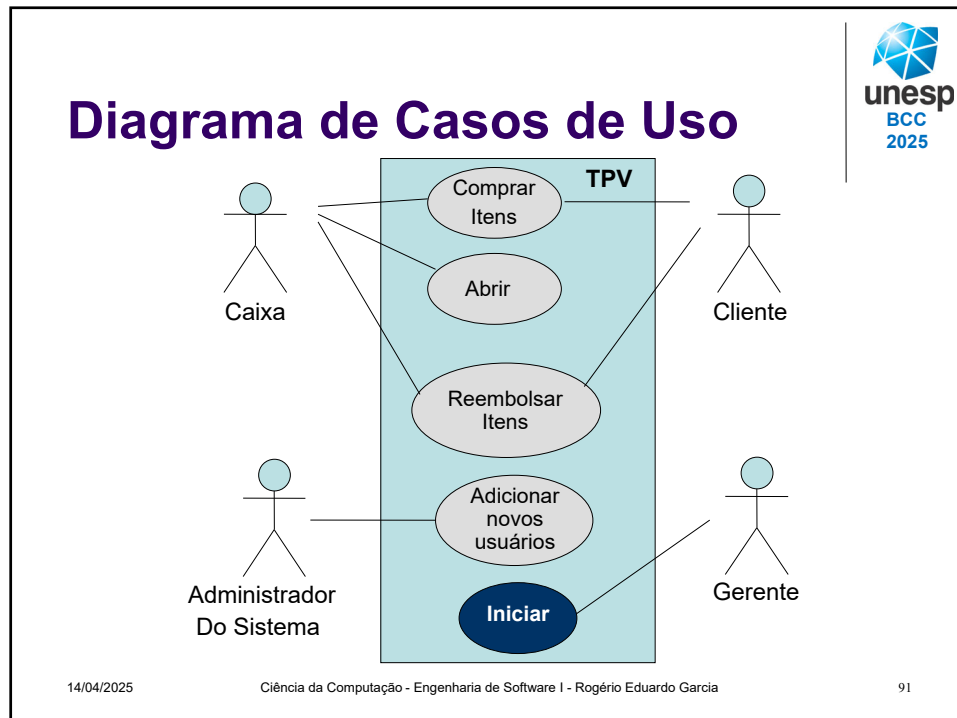
public float calcularTroco( )
{
    float troco = pg.obterQuantia - this.obterTotal();
    return troco;
}
```

14/04/2025

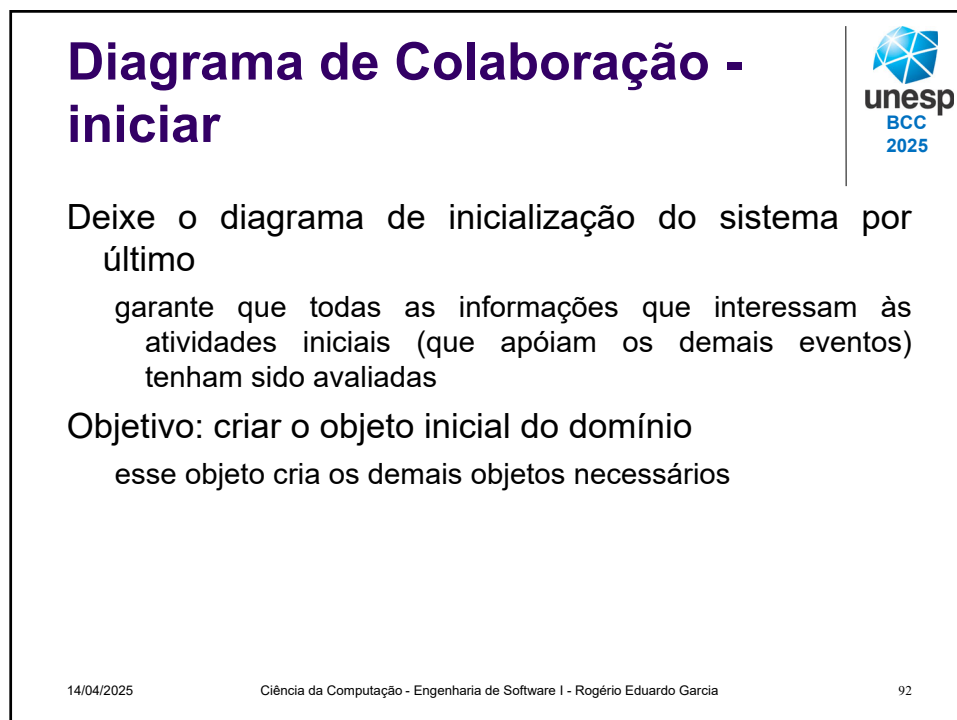
Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

90

90



91



92

Diagrama de Colaboração - *iniciar*



Objeto inicial pode ou não assumir o controle do processo

em geral, objeto inicial não assume o controle do processo quando existe uma interface de usuário gráfica

Então, para operação *iniciar*:

crie um diagrama de colaboração com primeira mensagem *criar()* para o objeto inicial do domínio

(opcional) se o objeto inicial está assumindo o controle do processo, crie um outro diagrama de colaboração e envie uma mensagem *executar()* ao objeto inicial

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

93

93

Diagrama de Colaboração - *iniciar*



No TPV:

iniciar ocorre quando *Gerente* liga o sistema

objeto inicial não assume controle do processo – apenas um diagrama para *iniciar*

Quem deve ser o objeto inicial do domínio?

escolha um objeto que intuitivamente (e conceitualmente) contenha todos os (ou a maioria dos) objetos do domínio

Ex: *Loja*

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia


94

94

Diagrama de Colaboração - *iniciar*

A partir dos diagramas anteriores, pode-se identificar as tarefas iniciais:

- criar *Loja*, *TPV*, *CatálogoProdutos*, *EspecificaçãoProduto*
- associar *CatálogoProduto* a *EspecificaçãoProduto*
- associar *Loja* a *CatálogoProduto*
- associar *Loja* a *TPV*
- associar *TPV* a *CatálogoProduto*




14/04/2025

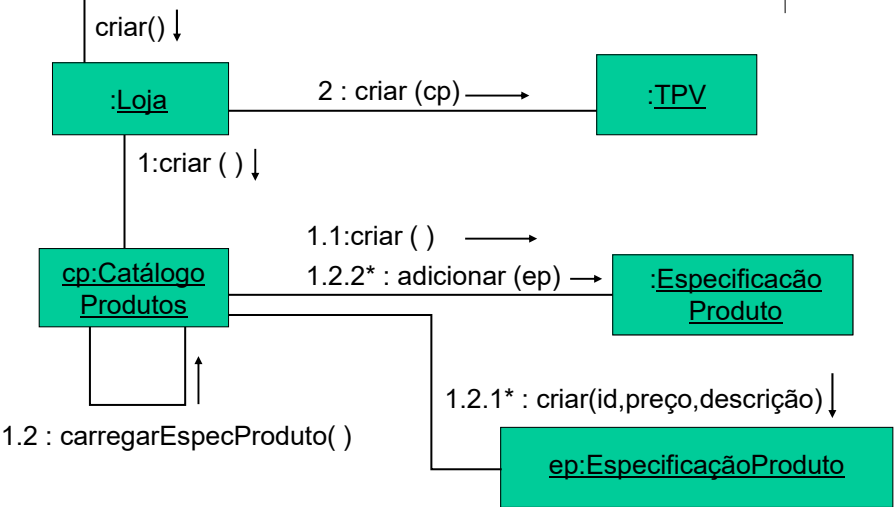
Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

95

95

Diagrama de Colaboração - *iniciar*





```
sequenceDiagram
    participant Loja as :Loja
    participant TPV as :TPV
    participant Catálogo as cp:CatálogoProdutos
    participant Especificação as :EspecificaçãoProduto
    participant EspecificaçãoEp as ep:EspecificaçãoProduto

    Loja->>Loja: criar()
    Loja->>TPV: 2 : criar (cp)
    Loja->>Catálogo: 1:criar ( )
    Catálogo->>Especificação: 1.1:criar ( )
    Catálogo->>Especificação: 1.2.2* : adicionar (ep)
    Catálogo->>EspecificaçãoEp: 1.2.1* : criar(id,preço,descrição)
    Catálogo->>Catálogo: 1.2 : carregarEspecProduto( )
```


14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

96

96

Comentários



Multiplicidade : observe que a loja só cria um objeto *TPV*

a multiplicidade no Modelo Conceitual e no projeto de objetos pode não ser a mesma

Persistência: num sistema real, *EspecificaçãoProduto* residirá num meio de armazenamento permanente (arquivo em disco ou base de dados).

portanto, a criação de objetos dessa classe é temporária e não será refletida na implementação


14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

97

97

Atividades da Fase Projetar



Refinar Plano	Sincronizar artefatos	Analisar	Projetar	Construir	Testar
---------------	-----------------------	----------	----------	-----------	--------

1. Definir Casos de Uso Reais

2. Definir Relatórios, IU e “Storyboards”

3. Refinar a arquitetura do sistema

4. Definir Diagramas de Interação

5. Definir Diagramas de Classes de Projeto

6. Definir o Esquema Do Banco de Dados

14/04/2025

Ciência da Computação - Engenharia de Software I - Rogério Eduardo Garcia

98

98