

**Modelagem de Sistemas Orientado a Objetos com UML.**


---

**Capítulo 6**

**Modelagem Dinâmica**

**Diagrama de Sequência e Padrões**

Ana Paula Gonçalves Serra, Dr.



**Onde Estamos na Disciplina de**

**Modelagem de Sistemas Orientado a Objetos com UML?**

---

- 1 Conceitos fundamentais de orientação a objetos.
- 2 Estruturação e modelagem de sistemas.
- 3 Diagramas de classes – Parte I, II e III.
- 4 Diagrama de sequência.
- 5 Realização de Casos de Uso.
- 6 Diagrama de estados.
- 7 Diagrama de atividades.
- 8 Diagramas de Implementação (Pacote, Componente e Implantação)

---

Pós-Graduação em Eng. de Software      Modelagem de Sistemas  
Universidade São Judas Tadeu      Orientado a Objetos com UML


Ana Paula G. Serra

3

## Objetivos do Capítulo

Este capítulo tem por objetivo apresentar ao alunos os seguintes conceitos:

1. Diagramas de Interação
2. Diagrama de Sequência – Conceitos
3. Diagrama de Sequência – Notação Básica
4. Diagrama de Sequência – Construção
5. Diagrama de Sequência – Notação Complementar
6. Exercício
7. Responsabilidades e Métodos
8. Padrões
9. Modularidade
10. Exercícios
11. Projeto



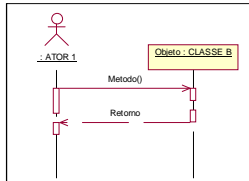
Pós-Graduação em Eng. de Software  
Universidade São Judas Tadeu
Modelagem de Sistemas  
Orientado a Objetos com UML
Ana Paula G. Serra

4

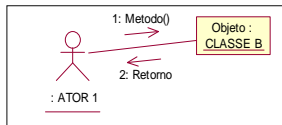
## 1. Diagramas de Interação

Composto pelos diagramas de:

**Sequência**



**Colaboração / Comunicação (UML 2.0)**



Ambos expressam interações de mensagens entre objetos.

Normalmente pode-se escolher entre utilizar o diagrama de colaboração ou o diagrama de sequência.

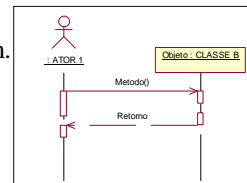
Pós-Graduação em Eng. de Software  
Universidade São Judas Tadeu
Modelagem de Sistemas  
Orientado a Objetos com UML
Ana Paula G. Serra

## 1. Diagramas de Interação

5

### Diagrama de Sequência

- ❑ Mostra a colaboração dinâmica entre os vários objetos de um sistema.
- ❑ Enfatiza a comunicação dos objetos através da passagem de mensagem entre os mesmos.
- ❑ Mostra a interação entre os objetos, alguma coisa que acontecerá em um ponto específico da execução do sistema.
- ❑ Ênfase na ordem temporal que os eventos acontecem.
- ❑ Devem ser construídos por cenários de casos de uso.



Pós-Graduação em Eng. de Software  
Universidade São Judas Tadeu

Modelagem de Sistemas  
Orientado a Objetos com UML

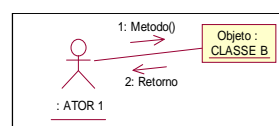
Ana Paula G. Serra

## 1. Diagramas de Interação

6

### Diagrama de Colaboração

- ❑ O diagrama de colaboração mostra de maneira semelhante ao diagrama de sequência, a colaboração dinâmica entre os objetos.
- ❑ No diagrama de colaboração, além de mostrar a troca de mensagens entre os objetos, percebe-se também os objetos com os seus relacionamentos.
- ❑ Não é fácil ver a sequência de mensagens.



Pós-Graduação em Eng. de Software  
Universidade São Judas Tadeu

Modelagem de Sistemas  
Orientado a Objetos com UML

Ana Paula G. Serra

## 1. Diagramas de Interação

## Diagrama de Sequência X Colaboração

- Se a ênfase do diagrama for o decorrer do tempo, é melhor escolher o diagrama de sequência, mas se a ênfase for o contexto do sistema, é melhor dar prioridade ao diagrama de colaboração.
- Notação do diagrama de sequência é mais simples do que a notação do diagrama de colaboração.
- Geralmente as ferramentas CASE de modelagem geram automaticamente o diagrama de colaboração baseado no diagrama de sequência e vice-versa.

## 1. Diagramas de Interação

- ❑ Fazer diagramas de interação, ou em outras palavras decidir detalhes do projeto de objetos é um passo muito criativo.
- ❑ Os padrões, os princípios, as responsabilidades podem ser aplicados para melhorar a qualidade do projeto.

9

## 2. Diagrama de Sequência - Conceitos

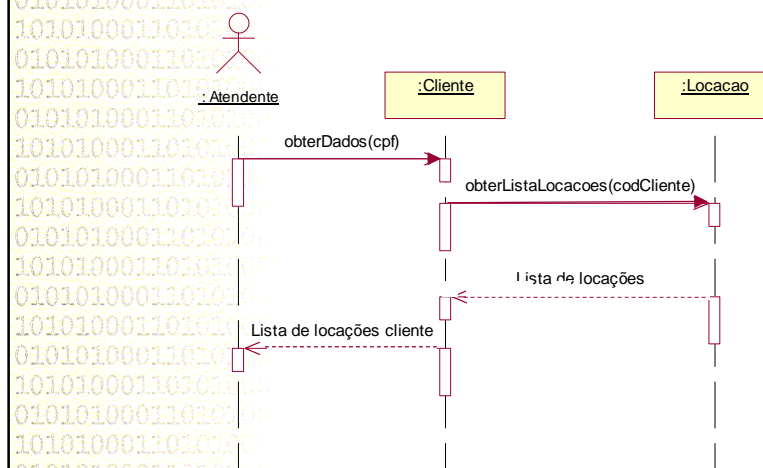
### Em resumo ...

- ❑ Descreve como os objetos interagem e como eles se comunicam.
- ❑ Representa a troca de mensagens entre os objetos.
- ❑ Ênfase na ordem temporal que os eventos acontecem.
- ❑ Devem ser construídos por cenários de casos de uso.

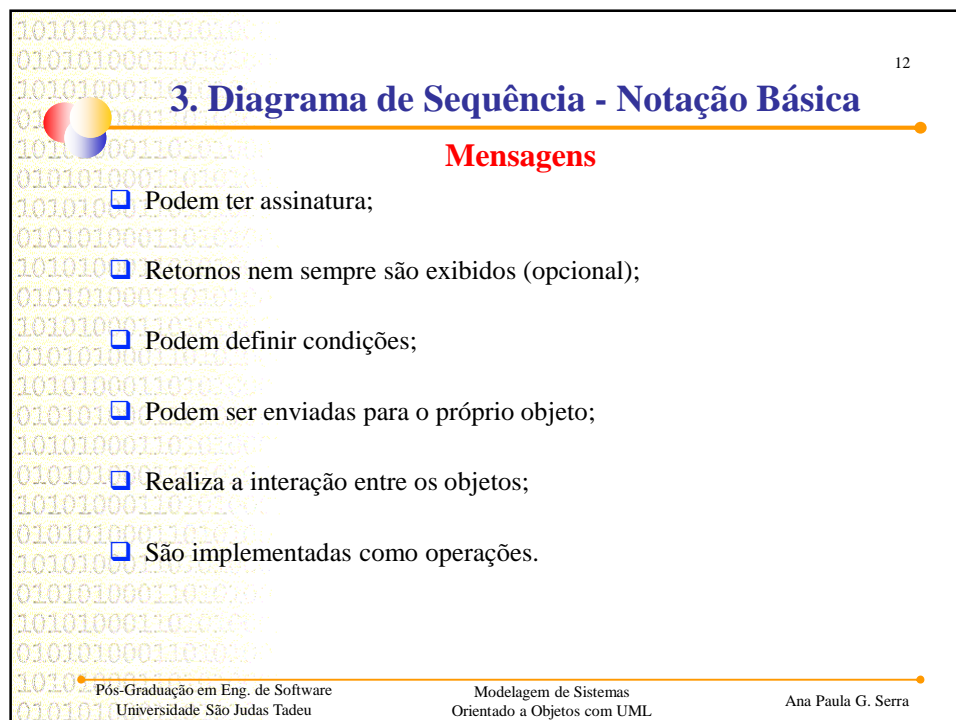
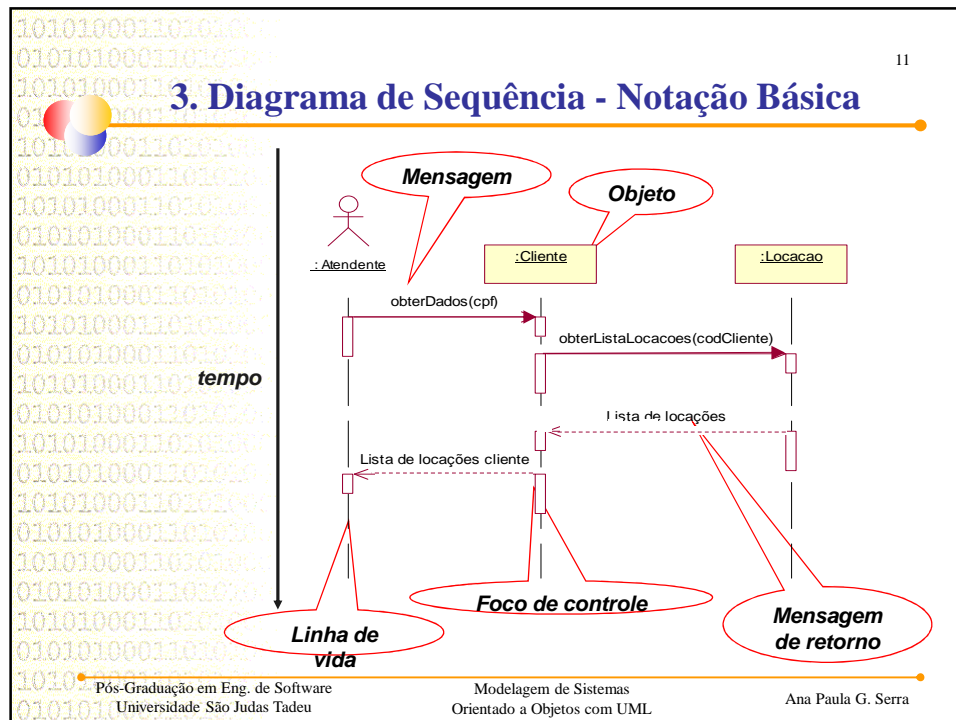
10

## 2. Diagrama de Sequência - Conceitos

### Exemplo







13

## 3. Diagrama de Sequência - Notação Básica

### Mensagens Síncronas e Assíncronas

☐ Mensagem Síncrona:

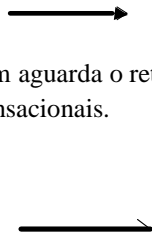
☐ O objeto que envia a mensagem aguarda o retorno;

☐ Mais utilizadas em sistema transacionais.

☐ Mensagem Assíncrona:

☐ Não há retorno ao objeto chamador;

☐ O processamento continua após o envio da mensagem.



Pós-Graduação em Eng. de Software  
Universidade São Judas Tadeu
Modelagem de Sistemas  
Orientado a Objetos com UML
Ana Paula G. Serra

14

## 3. Diagrama de Sequência - Notação Básica

### Mensagens “create” e “destroy”

☐ É a representação de quando os objetos são criados e destruídos durante o cenário modelado;

☐ São enviados através de mensagens;

☐ Utiliza-se os estereótipos:

☐ `<<create>>`

☐ `<<destroy>>`

☐ São implementados nas classes através de:

☐ Método Construtor: *create*

☐ Método Destruidor: *destroy*

Pós-Graduação em Eng. de Software  
Universidade São Judas Tadeu
Modelagem de Sistemas  
Orientado a Objetos com UML
Ana Paula G. Serra

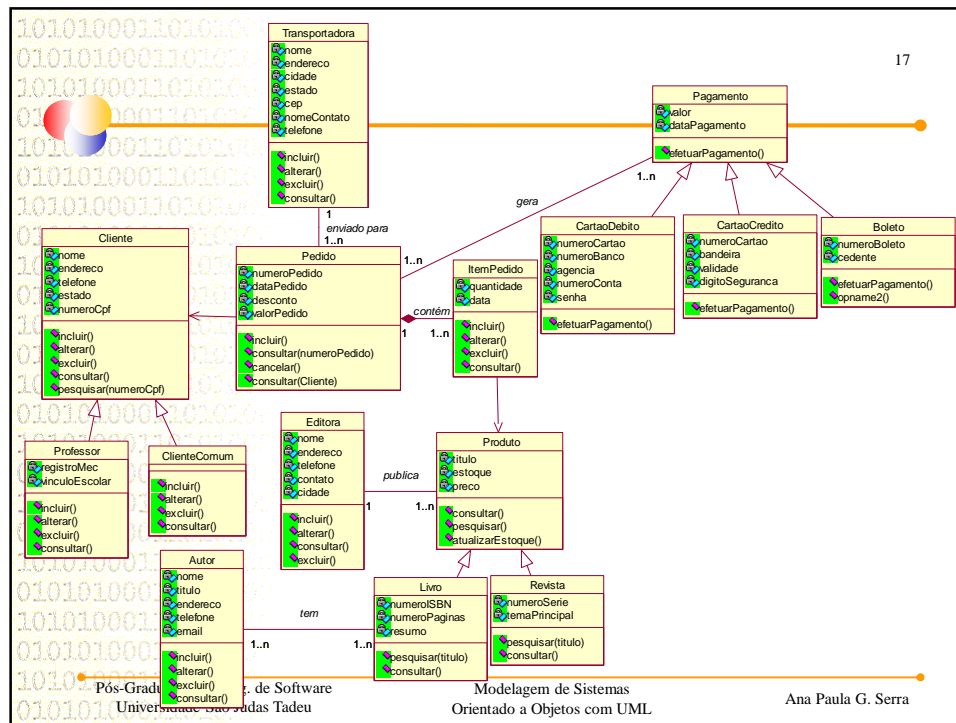
#### 4. Diagrama de Sequência - Construção

- ❑ Podem ser construídos para:
  - ❑ Fluxo Principal dos Casos de Uso;
  - ❑ Fluxos Alternativos;
- ❑ Não é necessário criar o diagrama para todos os cenários. Considere sempre os cenários mais relevantes (necessitam de detalhes);
- ❑ A troca de mensagem auxilia na identificação de novas operações para as classes envolvidas na interação.

#### 4. Diagrama de Sequência - Construção

- ☐ Defina o cenário que irá representar;
- ☐ Identifique os objetos que participam da interação;
- ☐ Coloque o objeto que inicia a interação à esquerda e os demais à direita de acordo com sua participação na interação;
- ☐ Coloque as mensagens que os objetos enviam e recebem na ordem crescente de tempo e de cima para baixo;
- ☐ Coloque as mensagens de retorno relevantes para o cenário;





18

## 6. Exercício

Elaborar o diagrama de sequência baseado na solução do diagrama de classes da transparência anterior.

**Caso de Uso: Efetuar Pedido**

- ☐ Cenário: Efetuar Pedido com sucesso.

**Caso de Uso: Cancelar Pedido**

- ☐ Cenário: Cancelar Pedido com sucesso.

**Caso de Uso: Efetuar Pagamento**


- ☐ Cenário: Erro no Pagamento por Cartão de Crédito (sem crédito).

Pós-Graduação em Eng. de Software  
Universidade São Judas Tadeu

Modelagem de Sistemas  
Orientado a Objetos com UML

Ana Paula G. Serra


19



### Algumas afirmações...

- ❑ Decidir onde os métodos devem ser alocados e como os objetos devem interagir entre si **é muito importante** e **nada fácil**.
- ❑ Os diagramas de interação **são muito importantes** – sob o ponto de vista do desenvolvimento de um bom projeto.

Mas, o que fazer para criar uma boa solução de projeto OO?



**Criar diagramas de interação utilizando princípios de atribuição de responsabilidade e uso de princípio e padrões de projeto, ajudam a garantir a qualidade do projeto.**


---

Pós-Graduação em Eng. de Software  
Universidade São Judas Tadeu

Modelagem de Sistemas  
Orientado a Objetos com UML

Ana Paula G. Serra

20



## 7. Responsabilidades e Métodos

- ❑ UML define responsabilidade como “um contrato ou obrigação de um classificador”. [OMG]
- ❑ As responsabilidades estão relacionadas com as obrigações de um objeto em termos de comportamento.

---

Pós-Graduação em Eng. de Software  
Universidade São Judas Tadeu

Modelagem de Sistemas  
Orientado a Objetos com UML

Ana Paula G. Serra

21

## 7. Responsabilidades e Métodos

O comportamento pode ser de:

### ☐ Responsabilidade (FAZER)

- ☐ Fazer algo, como criar um objeto ou executar um cálculo.
- ☐ Iniciar uma ação em outros objetos.
- ☐ Controlar e coordenar atividades em outros objetos.

### ☐ Colaboração (SABER)

- ☐ Ter conhecimento de dados privados encapsulados.
- ☐ Conhecer objetos relacionados.
- ☐ Ter conhecimento sobre coisas que pode derivar, calcular, ...

☐ Pode-se utilizar cartões CRC (*Class Responsibility Collaboration*) – não faz parte da UML.

22

## 7. Responsabilidades e Métodos

### Exemplo de Cartão CRC

ContaBancária (entidade)	
Responsabilidades	Colaboradores
1. Conhecer o seu cliente. 2. Conhecer o seu número. 3. Conhecer o seu saldo. 4. Manter um histórico de transações. 5. Aceitar saques e depósitos.	Cliente HistóricoTransações

23

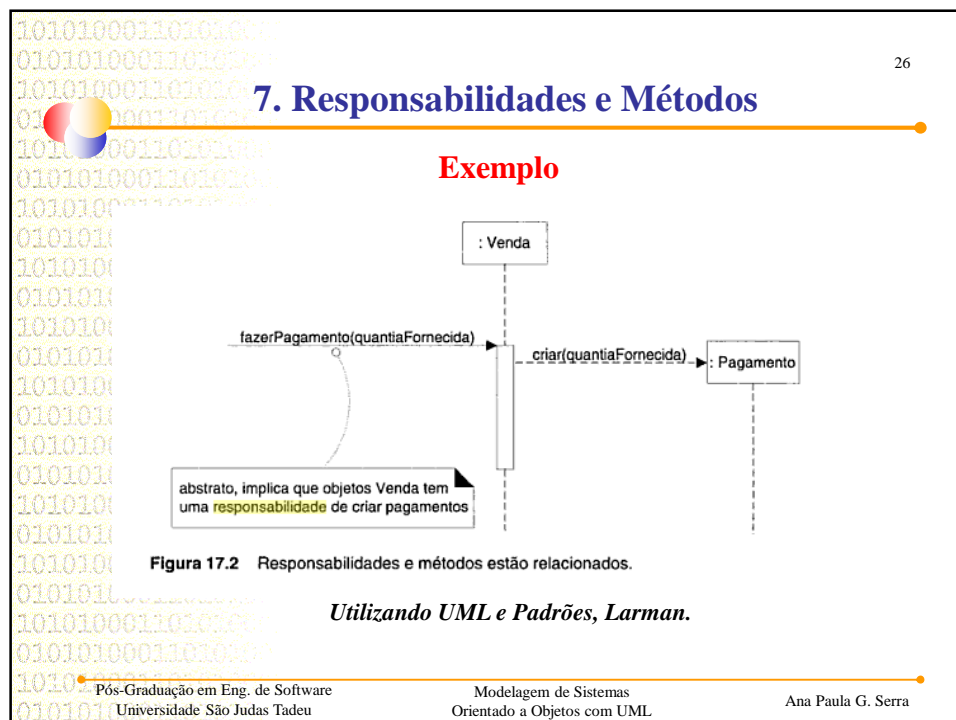
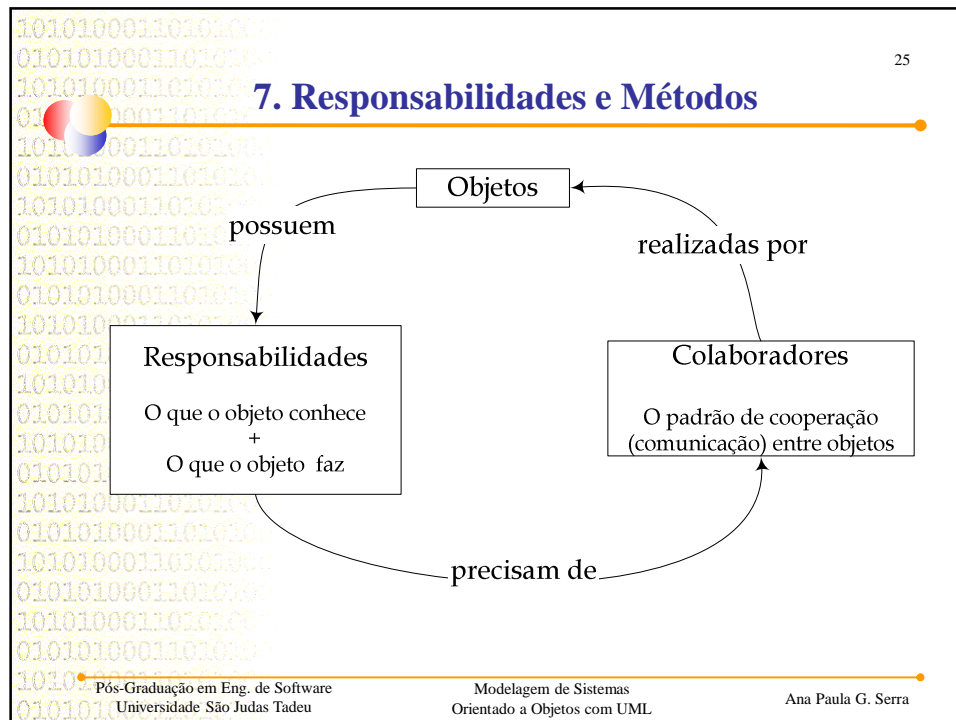
## 7. Responsabilidades e Métodos

- ❑ Na prática, uma responsabilidade é alguma coisa que um objeto **conhece** ou **faz**. (**sozinho** ou **não**).
- ❑ Um objeto Cliente conhece seu nome, seu endereço, seu telefone, etc.
- ❑ Um objeto Pedido conhece sua data de realização e sabe fazer o cálculo do seu total.
- ❑ Se um objeto tem uma responsabilidade a qual não pode cumprir sozinho, ele deve requisitar **colaborações** de outros objetos.

24

## 7. Responsabilidades e Métodos

- ❑ Um exemplo: quando a impressão de uma fatura é requisitada em um sistema de vendas, vários objetos precisam colaborar:
- ❑ um objeto Pedido pode ter a responsabilidade de fornecer o seu valor total
- ❑ um objeto Cliente fornece seu nome
- ❑ cada ItemPedido informa a quantidade correspondente e o valor de seu subtotal
- ❑ os objetos Produto também colaboraram fornecendo seu nome e preço unitário.





27

## 8. Padrões

---

- ☐ São princípios gerais e de soluções que guiam a criação de software, que fornecem orientação sobre sua aplicação.
- ☐ Possui um nome, descrição do problema e uma solução que pode ser aplicada em novos contextos.
- ☐ Alguns padrões fornecem orientação sobre a atribuição de responsabilidade a objetos.
- ☐ Compreender e ser capaz de aplicar padrões de projeto durante a criação de diagramas de interação é importante para a qualidade do projeto.

Pós-Graduação em Eng. de Software  
 Universidade São Judas Tadeu

Modelagem de Sistemas  
 Orientado a Objetos com UML

Ana Paula G. Serra

28

## 8. Padrões

---

Padrões básicos que tratam questões comuns e problemas fundamentais de projeto.

- ☐ Especialista na Informação
- ☐ Criador
- ☐ Acoplamento Fraco
- ☐ Coesão Alta

Pós-Graduação em Eng. de Software  
 Universidade São Judas Tadeu

Modelagem de Sistemas  
 Orientado a Objetos com UML

Ana Paula G. Serra

29

## 8. Padrões

### Especialista na Informação

#### Problema

Qual é o princípio básico de atribuição de responsabilidades a objetos?

#### Solução

Atribuir uma responsabilidade ao especialista da informação, ou seja, a classe que tem a informação necessária para satisfazer a responsabilidade.

**Padrões Relacionados:** Acoplamento Fraco e Coesão Alta

Pós-Graduação em Eng. de Software  
Universidade São Judas Tadeu

Modelagem de Sistemas  
Orientado a Objetos com UML

Ana Paula G. Serra

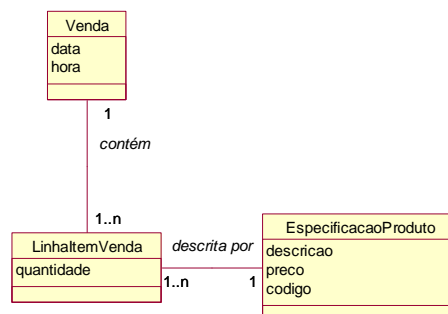
30

## 8. Padrões

### Especialista na Informação - Exemplo

#### Problema

Qual classe deve conter a responsabilidade de total geral de venda?



Pós-Graduação em Eng. de Software  
Universidade São Judas Tadeu

Modelagem de Sistemas  
Orientado a Objetos com UML

Ana Paula G. Serra

31

## 8. Padrões

### Especialista na Informação - Exemplo

Classe de Projeto	Responsabilidade
Venda	Sabe o total da venda
LinhaItemVenda	Sabe o subtotal da linha de item
EspecificacaoProduto	Sabe o preço do produto

As responsabilidades foram atribuídas segundo o princípio do especialista – atribuir a responsabilidade ao objeto que tem a informação necessária para satisfazê-la.

Pós-Graduação em Eng. de Software  
 Universidade São Judas Tadeu

Modelagem de Sistemas  
 Orientado a Objetos com UML

Ana Paula G. Serra

32

## 8. Padrões

### Criador

☐ **Problema**  
 Quem deve ser responsável pela criação de uma nova instância de uma classe?

☐ **Solução**  
 Atribuir a classe B a responsabilidade de criar uma instância da classe A se uma das seguintes condições for verdadeira:

- ☐ B agrega objetos de A.
- ☐ B contém objetos de A.
- ☐ B registra instâncias de objetos de A.
- ☐ B usa de maneira muito próxima objetos de A.
- ☐ B tem os dados de iniciação que serão passados para A quando ele for criado.

**Padrões Relacionados:** Acoplamento Fraco

Pós-Graduação em Eng. de Software  
 Universidade São Judas Tadeu

Modelagem de Sistemas  
 Orientado a Objetos com UML

Ana Paula G. Serra

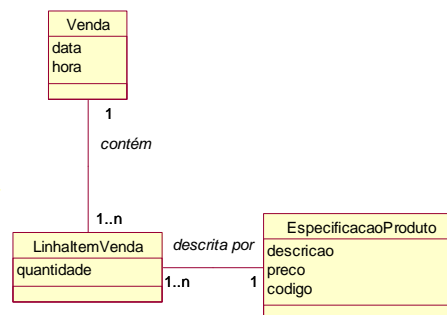
33

## 8. Padrões

### Criador - Exemplo

#### Problema

Qual classe deve ser responsável por criar uma instância de **LinhaItemVenda**?



34

## 8. Padrões

### Criador - Exemplo

Classe candidata **Venda**, pois contém (agrega) muitos objetos **LinhaItemVenda**.

35

## 8. Padrões

### Acoplamento Fraco

- ☐ **Problema**  
Como favorecer a dependência baixa, o pequeno impacto à mudança e aumentar a reutilização?
- ☐ **Solução**  
Atribuir uma responsabilidade de maneira que o acoplamento permaneça fraco.

Acoplamento Fraco é um princípio a ser levado em consideração em todas as decisões de projeto, mas não deve ser considerado isoladamente de outros padrões, como Especialista e Coesão Alta.

Pós-Graduação em Eng. de Software  
 Universidade São Judas Tadeu

Modelagem de Sistemas  
 Orientado a Objetos com UML

Ana Paula G. Serra

36

## 8. Padrões

### Acoplamento - Conceitos

- ☐ Mede o quanto um elemento está conectado a outro.
- ☐ Um elemento com acoplamento fraco não depende de muitos outros elementos.
- ☐ Se uma classe A conhece (e depende) da interface de uma classe B, e a interface da classe B é modificada, a classe A terá que ser modificada também.
- ☐ Minimizando o acoplamento entre as classes (e objetos) de um sistema, nós facilitamos as modificações no sistema.

Pós-Graduação em Eng. de Software  
 Universidade São Judas Tadeu

Modelagem de Sistemas  
 Orientado a Objetos com UML

Ana Paula G. Serra



## 8. Padrões

37

### Acoplamento Forte em Classes - Problemas

- ❑ As mudanças em classes relacionadas forçam mudanças locais.
- ❑ São difíceis de compreender isoladamente.
- ❑ São de difícil reutilização, porque exige a presença das classes das quais ela depende.

Pós-Graduação em Eng. de Software  
Universidade São Judas Tadeu

Modelagem de Sistemas  
Orientado a Objetos com UML

Ana Paula G. Serra

## 8. Padrões

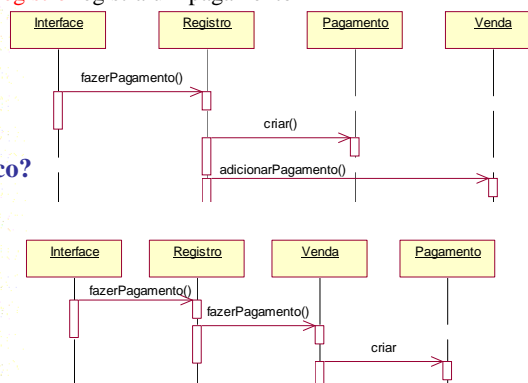
38

### Acoplamento Fraco - Exemplo

#### ❑ Problema

- ❑ Considere 3 classes: **Pagamento**, **Registro** e **Venda**
- ❑ Criar uma instância de **Pagamento** e associá-la a **Venda**. A classe **Registro** registra um pagamento

Qual melhor  
Acoplamento fraco?



Pós-Graduação em Eng. de Software  
Universidade São Judas Tadeu

Modelagem de Sistemas  
Orientado a Objetos com UML

Ana Paula G. Serra

39

## 8. Padrões

### Coesão Alta

- ☐ **Problema**  
Como manter a complexidade sob controle?
- ☐ **Solução**  
Atribuir uma responsabilidade de maneira que a coesão permaneça alta.

Na prática, o nível de coesão não pode ser considerado isoladamente de outras responsabilidades e de outros princípios, como Especialista e Acoplamento Fraco.

Pós-Graduação em Eng. de Software  
 Universidade São Judas Tadeu

Modelagem de Sistemas  
 Orientado a Objetos com UML

Ana Paula G. Serra

40

## 8. Padrões

### Coesão - Conceitos

- ☐ Mede o quanto as responsabilidades de um elemento são fortemente relacionadas.
- ☐ Um elemento com responsabilidades altamente relacionadas e que não executa um grande volume de trabalho, tem coesão alta.

Pós-Graduação em Eng. de Software  
 Universidade São Judas Tadeu

Modelagem de Sistemas  
 Orientado a Objetos com UML

Ana Paula G. Serra

41

## 8. Padrões

### Coesão Baixa em Classes - Problemas

- ☐ São delicadas, constantemente afetada pelas mudanças.
- ☐ São difíceis de compreender.
- ☐ São de difícil reutilização.
- ☐ São difíceis de manter.

Pós-Graduação em Eng. de Software  
Universidade São Judas Tadeu

Modelagem de Sistemas  
Orientado a Objetos com UML

Ana Paula G. Serra

42

## 8. Padrões

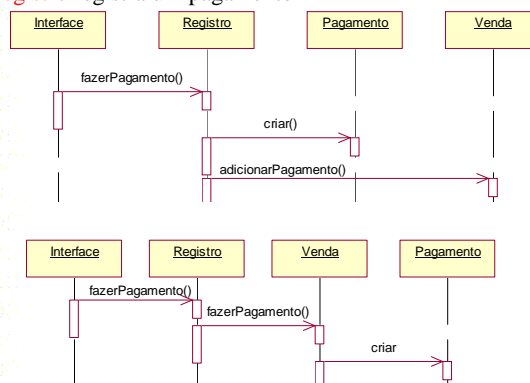
### Coesão Alta - Exemplo

#### ☐ Problema

- ☐ Considere 3 classes: **Pagamento** **Registro** **Venda**
- ☐ Criar uma instância de **Pagamento** e associá-la a **Venda**. A classe **Registro** registra um pagamento

Qual melhor  
Coesão Alta?

Qual melhor  
Coesão Alta e  
Acoplamento  
Baixo?



Pós-Graduação em Eng. de Software  
Universidade São Judas Tadeu

Modelagem de Sistemas  
Orientado a Objetos com UML

Ana Paula G. Serra

43

## 8. Padrões

- ❑ Uma frase comum em Engenharia de Software é: "Qualidade de software aumenta-se com baixo acoplamento e alta coesão".
- ❑ Encapsulamento
  - ❑ Reforça a alta coesão e baixo acoplamento;
  - ❑ Consiste em esconder os detalhes de implementação da classe;
  - ❑ A interação da classe com outras classes deve ser feita através de sua interface (métodos);
  - ❑ Deve-se evitar o acesso público a atributos da classe, baixando o acoplamento com outras classes;
  - ❑ A lógica é que os atributos fornecem o estado de um objeto e se este estiver acessível para o mundo externo, então fica complexo avaliar e prever a mudança do estado dos objetos.
  - ❑ Na prática, garante-se encapsulamento por meio de:
    - atributos privados (private).
    - caso seja necessário, fornecer uma interface de métodos não-privado.

Pós-Graduação em Eng. de Software  
Universidade São Judas Tadeu

Modelagem de Sistemas  
Orientado a Objetos com UML

Ana Paula G. Serra

44

## 9. Modularidade

- ❑ Além de Acoplamento e Coesão, o projeto dele ser modular.

### Algumas constatações...

- ❑ Modularidade é a propriedade de um sistema ser decomposto em um conjunto de módulos coesos e fracamente acoplados. [Booch]
- ❑ Modularizar ajuda a lidar com a complexidade de sistemas (divida para conquistar), facilitando o projeto, o entendimento, os testes, ... através de encapsulamento e abstração.
- ❑ Modularizar ajuda a manter a coesão de cada subsistema.
- ❑ Modularizar ajuda a diminuir o acoplamento geral do sistema (há comunicação entre alguns subsistemas e usando algumas interfaces bem definidas na "borda" dos subsistemas).

Pós-Graduação em Eng. de Software  
Universidade São Judas Tadeu

Modelagem de Sistemas  
Orientado a Objetos com UML

Ana Paula G. Serra

45

## 9. Modularidade

**Algumas constatações...**

- ❑ Modularizar permite escolher como desenvolver cada subsistema: se cada subsistema será desenvolvido internamente, contratado para desenvolvimento externo ou comprado
- ❑ Modularizar facilita a divisão de trabalho, permitindo desenvolver em paralelo, uma equipe por subsistema.
- ❑ Modularizar permite reutilizar subsistemas em várias aplicações.
- ❑ Modularizar permite que várias aplicações compartilhem um mesmo subsistema.
- ❑ Modularizar permite construir sistemas distribuídos em que subsistemas podem estar espalhados fisicamente.

Pós-Graduação em Eng. de Software  
 Universidade São Judas Tadeu


Modelagem de Sistemas  
 Orientado a Objetos com UML

Ana Paula G. Serra

46

## 10. Exercícios

1. Analisar a possível solução do Diagrama de Classes – Sistema Comércio Eletrônico da transparência 17 e identificar.
2. Omissões e ambiguidades referentes a classes, atributos, operações e relacionamentos.
3. Possíveis melhorias considerando os padrões apresentados neste material
4. De acordo, com a sua avaliação os seus diagramas de sequência poderiam ser melhorados?



Pós-Graduação em Eng. de Software  
 Universidade São Judas Tadeu

Modelagem de Sistemas  
 Orientado a Objetos com UML

Ana Paula G. Serra



47


## 11. Projeto

---

**Elaborar diagramas de sequência do Sistema de Controle de Matrículas de Cursos Livres pela Internet**

***(Projeto) – Item 3***

☐ Caso de Uso: Realizar Matrícula(2 cenários)



• Pós-Graduação em Eng. de Software  
Universidade São Judas Tadeu

Modelagem de Sistemas  
Orientado a Objetos com UML


Ana Paula G. Serra

48

## Referências Bibliográfica

---

☐ Larman, Craig; Utilizando UML e Padrões, 2a. Edição. Bookman, 2003.  
ISBN: 85-363-0358-1. Capítulo 15 e 16.




• Pós-Graduação em Eng. de Software  
Universidade São Judas Tadeu

Modelagem de Sistemas  
Orientado a Objetos com UML

Ana Paula G. Serra

49



---

Copyright © 2012-2013 Profa. Dra. Ana Paula Gonçalves Serra.

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).

---

Pós-Graduação em Eng. de Software	Modelagem de Sistemas	Ana Paula G. Serra
Universidade São Judas Tadeu	Orientado a Objetos com UML	