

Modelagem de Sistemas com UML

(parte 2)

Jobson Massollar
jobson@cos.ufrj.br

Tayana Conte
tayana@cos.ufrj.br

Guilherme Horta Travassos
ght@cos.ufrj.br



COPPE
Instituto Alberto Luiz Coimbra de
Pós-Graduação e Pesquisa de Engenharia

UFRJ
Universidade Federal do Rio
de Janeiro

Sumário

- Paradigma OO
- UML
 - Diagrama de Classes
 - Diagrama de Seqüência
 - Diagrama de Estados
 - Diagrama de Pacotes

- Significa organizar o software como uma coleção de objetos discretos que incorporam a estrutura dos dados e o comportamento.
- O que é um Objeto?

É a representação computacional de um elemento ou processo do mundo real.

- Características de um Objeto:
 - ✓ Identidade
 - ✓ Características ou Estado
 - ✓ Comportamento

➤ Identidade de um Objeto:

- ✓ É o que identifica univocamente um objeto dentre os demais.
- ✓ A identidade permite que um objeto seja referenciado por outros.

➤ Características de um Objeto:

- ✓ Descrevem propriedades do objeto.
- ✓ São mutáveis ao longo do tempo.
- ✓ São chamadas **atributos** do objeto.

➤ Comportamento de um Objeto:

- ✓ Determina como um objeto reage a estímulos do mundo real ou de outros objetos.
- ✓ São chamados **métodos** do objeto.

Paradigma OO

➤ O que é uma Classe?

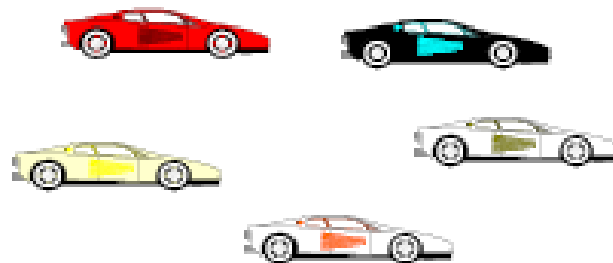
É uma descrição de um grupo de objetos com atributos, comportamentos, relacionamentos e semântica comuns.

- Cada objeto pertence a uma única classe.
- O Objeto é chamado de **instância** de sua Classe.
- A Classe é o bloco básico de construção de programas OO.

Classe Carro

Carro
Número de Rodas Cor Cor Lateral
Anda Para Acelera Estaciona

Objetos da classe Carro



- Alguns tipos de classes que podem ser identificadas:
 - **Classes de Domínio:** modelam objetos associados ao domínio do problema.
 - ✓ aluno, curso, turma, disciplina, docente, etc.
 - ✓ médico, paciente, exame, diagnóstico, etc.
 - ✓ advogado, juiz, processo, julgamento, etc.
 - **Classes de Fronteira:** modelam objetos que realizam a comunicação entre o sistema e a sua vizinhança, ou seja, os atores.
 - ✓ formulário, janela, etc.
 - ✓ catraca, sensor de presença, etc.
 - ✓ sistema de cobrança, administradora de cartão, etc.

- Alguns tipos de classes que podem ser identificadas:
 - **Classes de Controle:** modelam objetos que exercem controle sobre outros objetos.
 - ✓ criação de objetos
 - ✓ controle da concorrência de pedidos dos objetos
 - **Classes Utilitárias:** modelam comportamentos que representam algoritmos de uso comum.
 - ✓ verificação de CPF e CNPJ
 - ✓ cálculo da alíquota do IR
 - **Classes de Exceção:** modelam objetos que representam uma exceção ou falha em um determinado domínio.

➤ Localizando Classes e Atributos:

A partir dos Requisitos/Casos de Uso **extrair** substantivos resulta em uma tentativa de Classes e Atributos

A partir daí **eliminar** Classes estranhas resulta em Classes e Atributos

➤ O que procurar?

- ✓ Estruturas
- ✓ Outros sistemas
- ✓ Dispositivos
- ✓ Coisas ou eventos
- ✓ Papéis executados
- ✓ Lugares
- ✓ Unidades organizacionais
- ✓ ...

➤ Localizando Comportamentos:

A partir dos Requisitos/Casos de Uso **extrair** verbos
resulta em uma tentativa de Comportamentos

A partir daí **eliminar** Comportamentos estranhos
resulta em Comportamentos das Classes

➤ O que procurar?

- ✓ Verbos no imperativo
- ✓ Verbos no passivo
- ✓ Ações
- ✓ Coisas ou eventos
- ✓ Procedimentos operacionais
- ✓ ...

- Exercício 1: extrair classes, atributos e comportamentos da seguinte descrição:
- ✓ No posto de gasolina o cliente tem a opção de ser cobrado automaticamente no ato da compra ou receber uma conta mensal impressa. O pagamento da conta pode ser feito à vista, em cheque ou no CC. Os serviços oferecidos pelo posto são 4: combustível, lavagem e estacionamento rotativo e estacionamento mensal. Os preços de cada serviço são fixos. Algumas vezes, o proprietário do posto pode definir descontos para esses preços. Somente clientes previamente cadastrados com nome, cpf, telefone e endereço podem optar por receber a conta mensal. Para clientes cadastrados também é possível realizar o pagamento via débito automático, sendo que nesses casos também são necessários os dados bancários (numero do banco, agência e conta corrente).

Paradigma OO

- Exercício 1: extrair classes, atributos e comportamentos da seguinte descrição:
- ✓ No **posto de gasolina** o **cliente** tem a opção de ser cobrado automaticamente no ato da **compra** ou receber uma **conta mensal impressa**. O **pagamento** da conta pode ser feito **à vista**, em **cheque** ou no **CC**. Os **serviços oferecidos** pelo **posto** são 4: **combustível**, **lavagem** e **estacionamento rotativo** e **estacionamento mensal**. Os **preços** de cada serviço são fixos. Algumas vezes, o **proprietário** do posto pode definir descontos para esses preços. Somente clientes previamente cadastrados com **nome**, **cpf**, **telefone** e **endereço** podem optar por receber a conta mensal. Para clientes cadastrados também é possível realizar o pagamento via **débito automático**, sendo que nesses casos também são necessários os **dados bancários** (**numero do banco**, **agência** e **conta corrente**).

Paradigma OO

➤ Resposta do exercício:

Candidatos a Métodos/Atributos	
Posto de gasolina	Proprietário
Cliente	Desconto
Compra	Preço
Conta mensal	Nome
Pagamento	CPF
À vista	Telefone
Cheque	Endereço
Cartão de crédito	Débito automático
Serviços	Dados bancários
Combustível	Número do banco
Lavagem	Agência
Estacionamento rotativo	Conta-corrente
Estacionamento mensal	

Candidatos a Comportamentos	
Cobrar	Definir desconto
Imprimir conta	Cadastrar cliente
Realizar pagamento	Enviar conta
Oferecer serviço	

Paradigma OO

➤ Resposta do exercício:

Candidatos a Métodos/Atributos	
Posto de gasolina	Proprietário
Cliente	Desconto
Compra	Preço
Conta mensal	Nome
Pagamento	CPF
À vista	Telefone
Cheque	Endereço
Cartão de crédito	Débito automático
Serviços	Dados bancários
Combustível	Número do banco
Lavagem	Agência
Estacionamento rotativo	Conta-corrente
Estacionamento mensal	

Candidatos a Comportamentos	
Cobrar	Definir desconto
Imprimir conta	Cadastrar cliente
Realizar pagamento	Enviar conta
Oferecer serviço	

Diagrama de Classes

- Mostra a estrutura estática das classes de um sistema.
- É considerado estático porque a estrutura modelada é válida em qualquer ponto do ciclo de vida do sistema.
- É composto pelas classes (com seus atributos e métodos) e pelo relacionamentos entre elas:
 - ✓ Associação
 - ✓ Agregação
 - ✓ Composição
 - ✓ Generalização/Especialização

Diagrama de Classes no Contexto

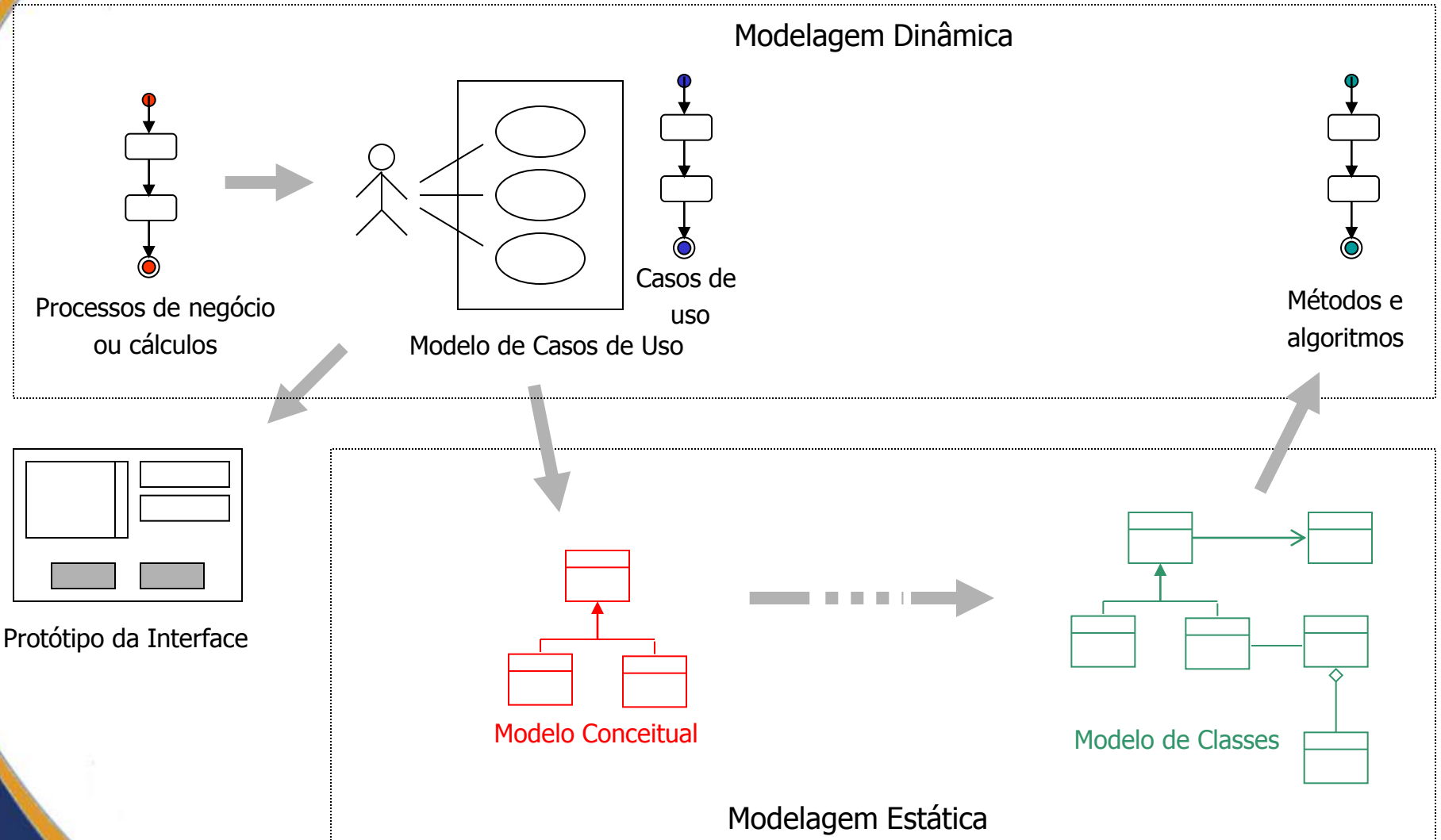
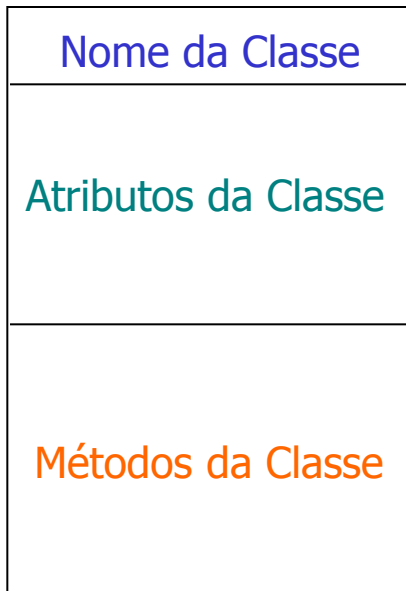


Diagrama de Classes

➤ Representação da Classe:



Sintaxe:

[nome-do-pacote ::] nome-da-classe [{propriedades}]

Sintaxe:

[visibilidade] [/] nome [:tipo] [multiplicidade] [=default] [{propriedades}]



o símbolo / define um atributo **derivado**

Sintaxe:

[visibilidade] nome (lista-de-parametros) [: tipo-de-retorno] [{propriedades}]

Visibilidade de atributos e métodos da classe:

- atributo/método privado (private)
- # atributo/método protegido (protected)
- + atributo/método público (public)
- ~ atributo/método do pacote (package)

Diagrama de Classes

➤ Exemplo de representação da Classe:

AdministracaoEventos::Evento {autor=Joao da Silva}

+ DuracaoMaxima: int = 30 {duracao maxima permitida, frozen}

- nome : String {1 a 50 caracteres, obrigatório}

- dataInicio: Date {deve ser maior que a data atual}

- duracao : integer = 1 {em dias, >= 1}

- /dataFim : Date = dataInicio + duracao

- patrocinadores : Patrocinador[10] {deve ter pelo menos 1 patrocinador}

- local : Local

+ getNome() : String

+ setNome(nome : String)

+ setDuracao(duracao : integer) { nao pode causar um evento sobreposto no mesmo local }

+ getPatrocinadores() : Patrocinador[]

- temConflitoDuracao(duracao : integer) : boolean

Diagrama de Classes

- Diagrama com as classes detectadas no exercício 1:

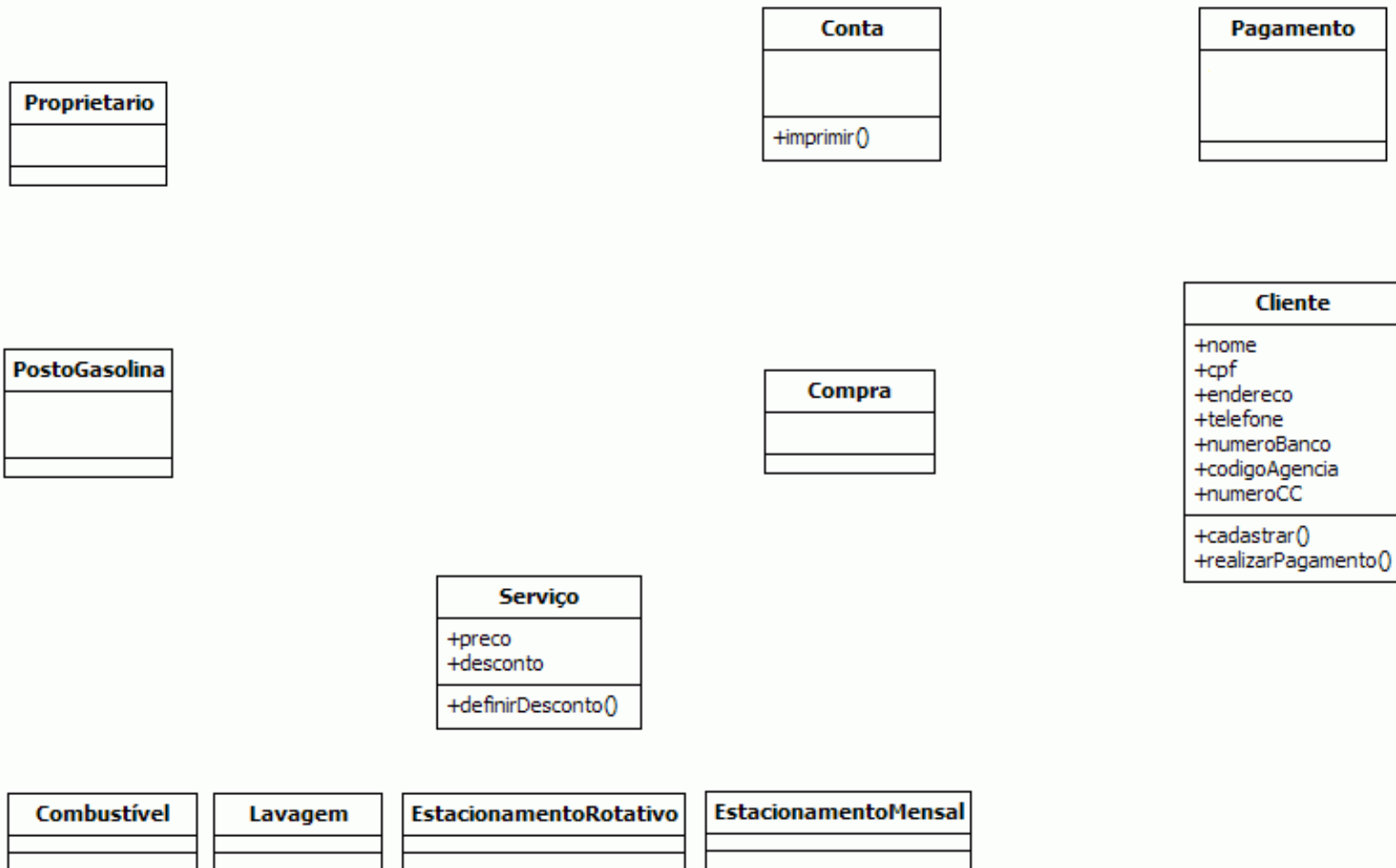


Diagrama de Classes

➤ Associação:

- ✓ Representa um relacionamento entre os objetos de duas ou mais classes.
- ✓ É representada, nos diagramas de classes, por uma linha conectando as classes associadas.
- ✓ O fluxo de dados pode ser uni-direcional ou bi-direcional, através da conexão.
- ✓ Para esclarecer o significado de uma associação, ela é nomeada. No Diagrama de Classes, o nome é apresentado ao longo da linha de associação. Usualmente, esse nome é um verbo ou uma frase verbalizada.
- ✓ Entre duas Classes pode existir mais de uma associação com nomes diferentes.
- ✓ Uma classe pode estar associada a ela mesma (associação reflexiva ou auto-relacionamento).

Diagrama de Classes

- Multiplicidade da Associação:
 - ✓ É o número de instâncias de uma classe relacionada com uma instância de outra classe.
 - ✓ Para cada associação, há uma multiplicidade em cada direção.
- A notação usada pela UML, para os indicadores de multiplicidade, é:

Muitos	*
Apenas um	1
Zero ou muitos	0..*
Um ou muitos	1..*
Zero ou um	0..1
Intervalo especificado	2..3 ou 3,5,7

Diagrama de Classes

- Navegabilidade da Associação:
 - ✓ Define se é possível, a partir de uma extremidade da associação, chegar a outra extremidade.
 - ✓ Existe uma diferença de notação entre a UML 1.4 e a UML 2.0.






Associação	UML 1.4	UML 2.0
Bidirecional		
Unidirecional		
Indefinido	?	

Diagrama de Classes

- Visibilidade da Associação:
 - ✓ Semelhante aos atributos e métodos é possível definir a visibilidade de uma associação:
 - associação privada (private)
 - # associação protegida (protected)
 - + associação pública (public)

Diagrama de Classes

Exemplos de Associação:

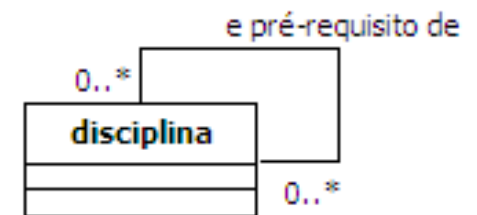
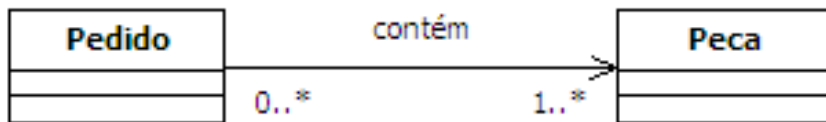
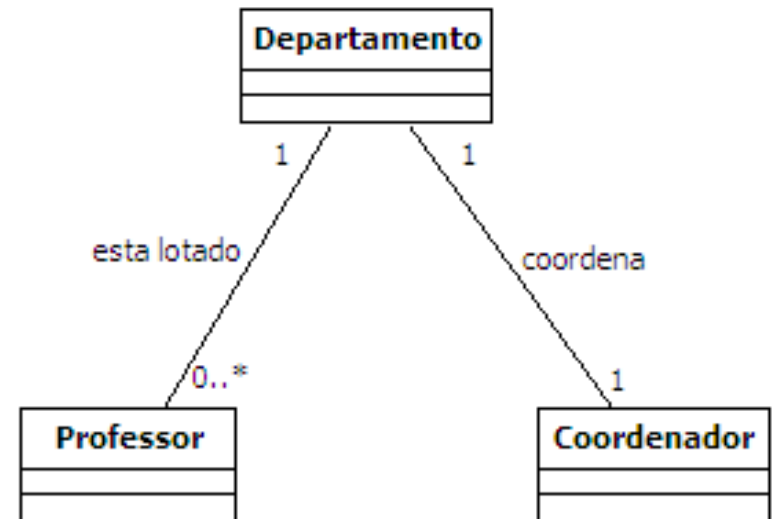
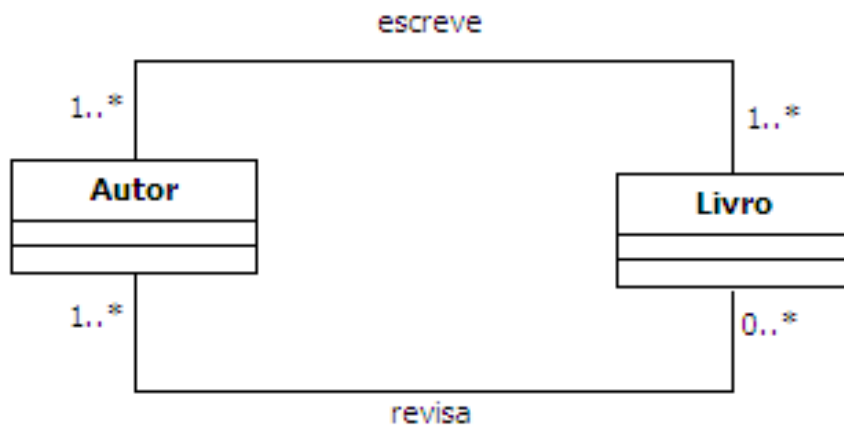


Diagrama de Classes

➤ Agregação:

- ✓ É uma forma especializada de associação na qual um todo é relacionado com suas partes, ou seja, um relacionamento “todo-parte”.
- ✓ Também é conhecida como relação de conteúdo ou posse lógica, pois a classe agregada (parte) é vista como um “atributo” da classe agregadora (todo).
- ✓ Deve ser lida como “é composto de”/“é parte de”.
- ✓ É representada como uma linha de associação com um diamante vazado junto à classe agregadora.
- ✓ Podem existir agregações reflexivas, ou seja, agregações de uma classe com ela mesma.
- ✓ A multiplicidade é representada da mesma maneira que nas associações.

Diagrama de Classes

➤ Exemplos de Agregação:

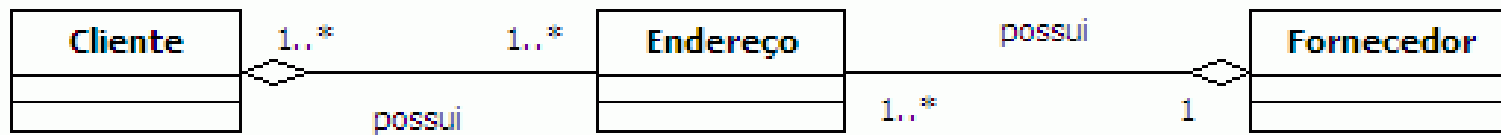


Diagrama de Classes

➤ Composição:

- ✓ É um tipo mais forte de relacionamento "todo-parte".
- ✓ A classe "parte" não tem existência independente da classe "todo".
- ✓ A deleção do "todo" implica obrigatoriamente na deleção da "parte".
- ✓ É representada como uma linha de associação com um diamante preenchido junto à classe "todo".
- ✓ A multiplicidade é representada da mesma maneira que nas associações.

Diagrama de Classes

➤ Exemplo de Composição:

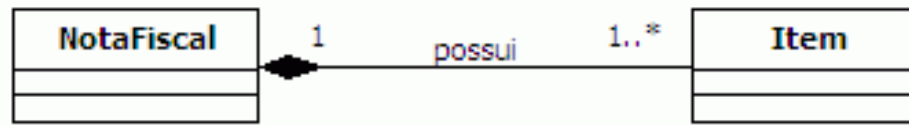


Diagrama de Classes

➤ Generalização/Especialização:

- ✓ É uma hierarquia de abstrações na qual uma subclasse herda a estruturas de dados e comportamentos de uma ou mais superclasses.
- ✓ A subclasse pode acrescentar novas estruturas de dados e comportamentos aos herdados da superclasse (especialização).
- ✓ A subclasse pode alterar o comportamento originalmente herdado da superclasse (polimorfismo).
- ✓ Deve ser lida como “é um tipo de”.
- ✓ **Herança simples** é quando uma subclasse herda estruturas de dados e comportamentos de uma única superclasse.
- ✓ **Herança múltipla** é quando uma subclasse herda estruturas de dados e comportamentos de mais de uma superclasse.
- ✓ **Classes abstratas** podem ser utilizadas para melhorar a legibilidade e manutenibilidade do modelo.

Diagrama de Classes

➤ Exemplos de Generalização/Especialização:

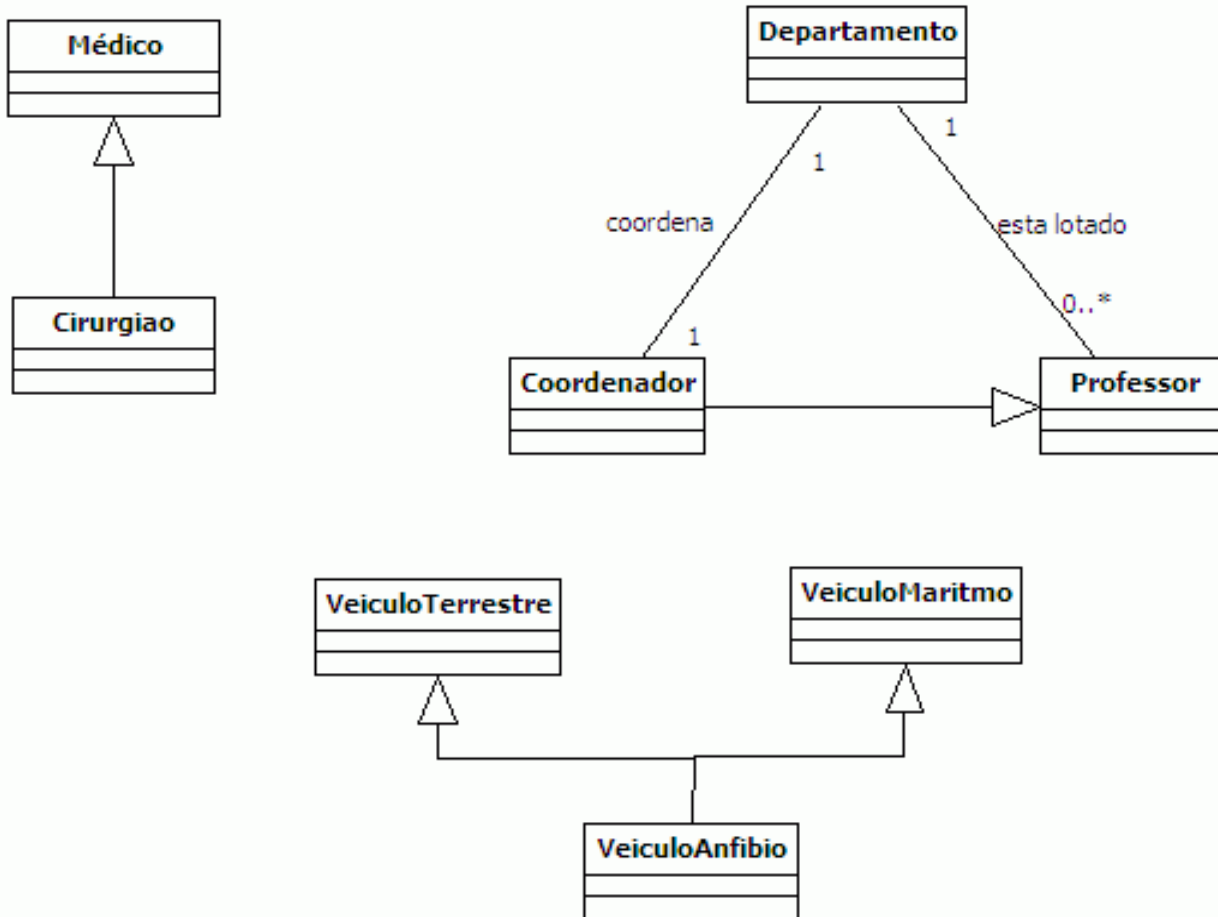


Diagrama de Classes

- Exemplos de Generalização/Especialização com Classe Abstrata:

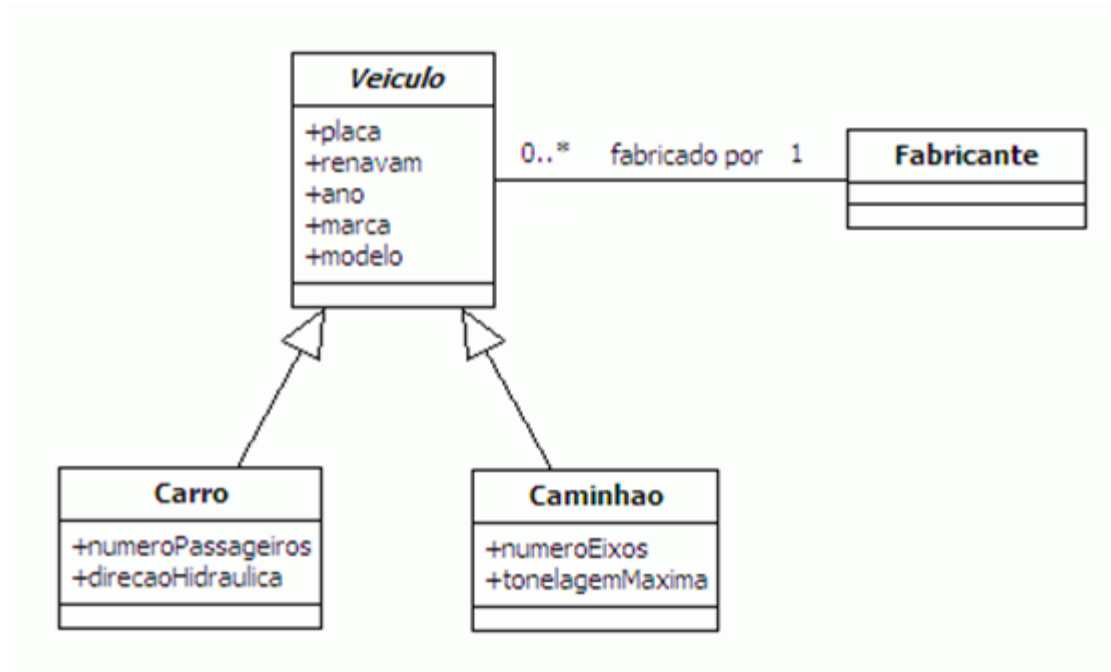


Diagrama de Classes

➤ Classes de Associação:

- ✓ Utilizamos Classes de Associação quando desejamos acrescentar atributos ou métodos a uma associação.
- ✓ Sempre podemos “promover” uma classe de associação em uma classe normal.
- ✓ **IMPORTANTE:** os atributos ou métodos da classe de associação se referem a um e somente um par formado pelas instâncias das classes associadas. Se puder existir mais de um par formado pelas mesmas instâncias das classes associadas, **NÃO** podemos usar classes de associação !

Diagrama de Classes

➤ Exemplos de Classes de Associação:

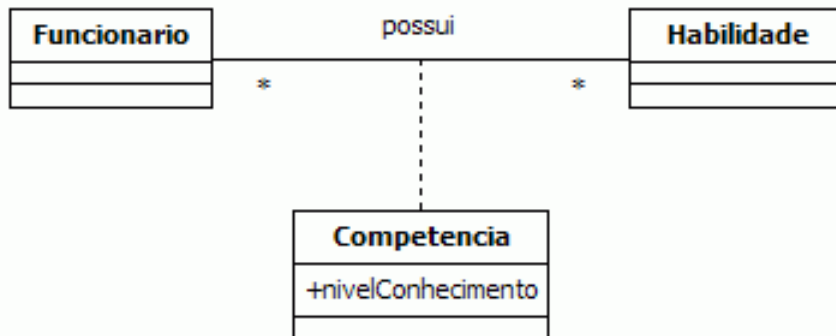
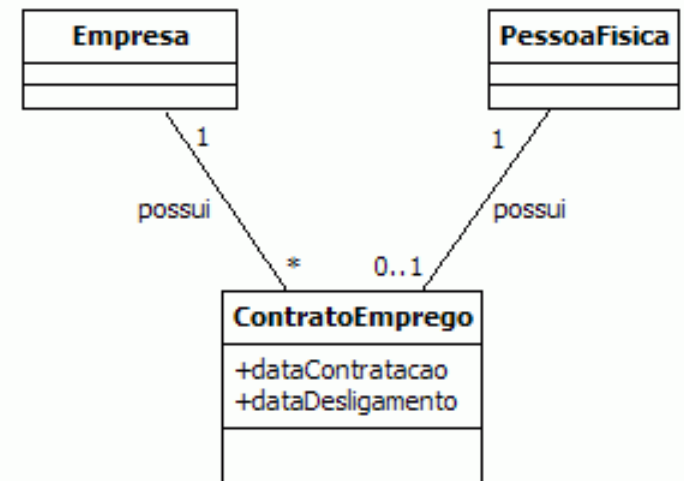
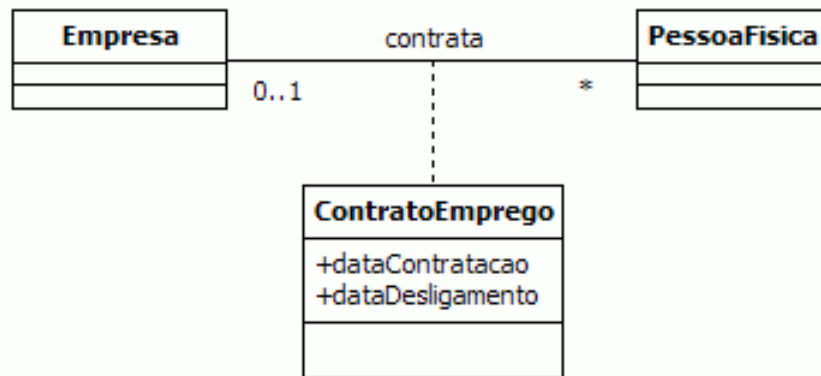


Diagrama de Classes

➤ Associação Exclusiva:

- ✓ Representa uma restrição onde apenas uma associação é válida em dado momento.
- ✓ A restrição pode ser aplicada a duas ou mais associações.

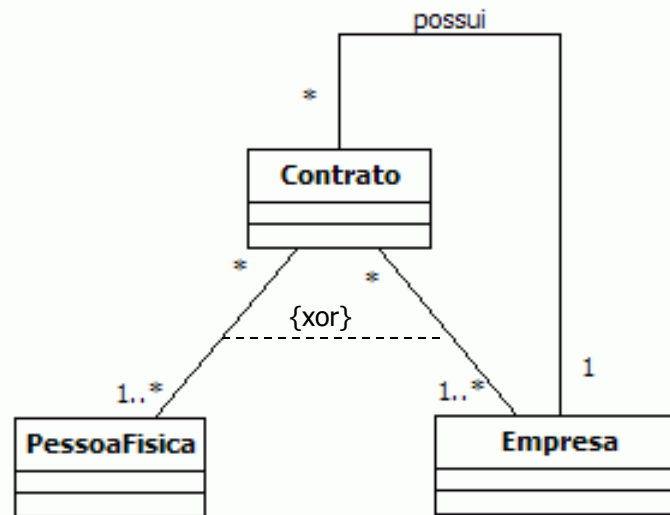


Diagrama de Classes

➤ IMPORTANTE !

- ✓ No início da fase de projeto devemos focar na identificação das classes de domínio e suas associações.
- ✓ No início da fase de projeto **NÃO** devemos nos preocupar com herança, agregação, composição ou classes abstratas.
- ✓ Da mesma forma, no início da fase de projeto **NÃO** devemos nos preocupar com classes relacionadas ao espaço da solução.
- ✓ Agregação, Composição, Herança e classes abstratas são um refinamento do modelo de classes.
- ✓ Eles são utilizados para dar mais robustez ao modelo e adequá-lo aos conceitos do paradigma OO.
- ✓ Entretanto, esse refinamento e adequação são críticos para o processo de evolução do sistema.
- ✓ Na generalização, cuidado para não confundir “é-um-tipo-de” com “pode-ser” ou “às-vezes-é”.
- ✓ Classes do espaço da solução surgirão conforme o projeto do sistema for sendo detalhado. Os diagramas de seqüência nos ajudarão nessa tarefa.

Diagrama de Classes

- ✓ Exercício 2: completar o diagrama de classes do exercício 1 com os relacionamentos entre as classes, classes que porventura ainda não foram definidas e atributos de conhecimento geral necessários para dar consistência ao modelo.
- ✓ No posto de gasolina o cliente tem a opção de ser cobrado automaticamente no ato da compra ou receber uma conta mensal impressa. O pagamento da conta pode ser feito à vista, em cheque ou no CC. Os serviços oferecidos pelo posto são 4: combustível, lavagem e estacionamento rotativo e estacionamento mensal. Os preços de cada serviço são fixos. Algumas vezes, o proprietário do posto pode definir descontos para esses preços. Somente clientes previamente cadastrados com nome, cpf, telefone e endereço podem optar por receber a conta mensal. Para clientes cadastrados também é possível realizar o pagamento via débito automático, sendo que nesses casos também são necessários os dados bancários (numero do banco, agência e conta corrente).

Diagrama de Classes

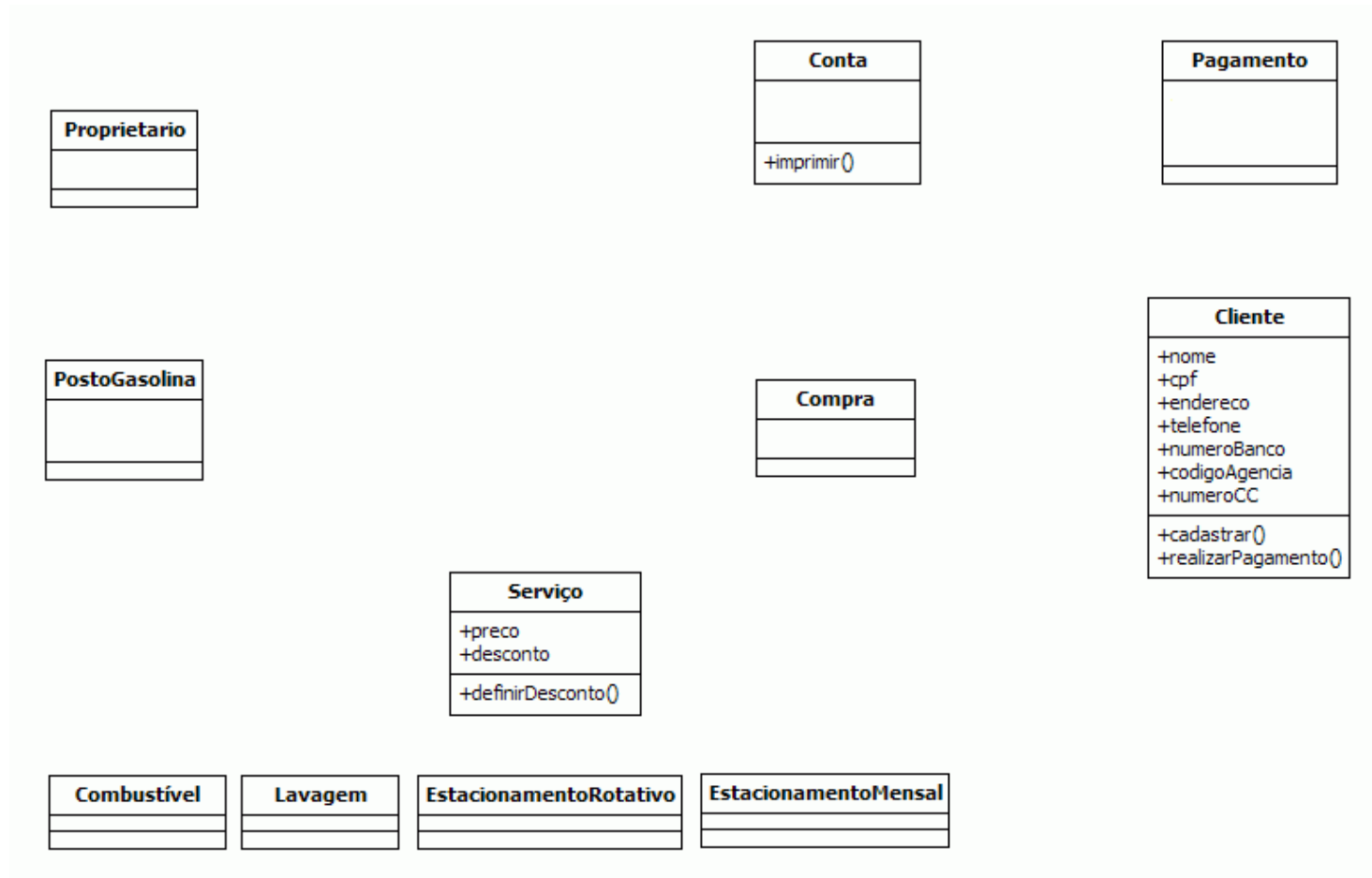


Diagrama de Classes

- Uma solução possível para o exercício 2:

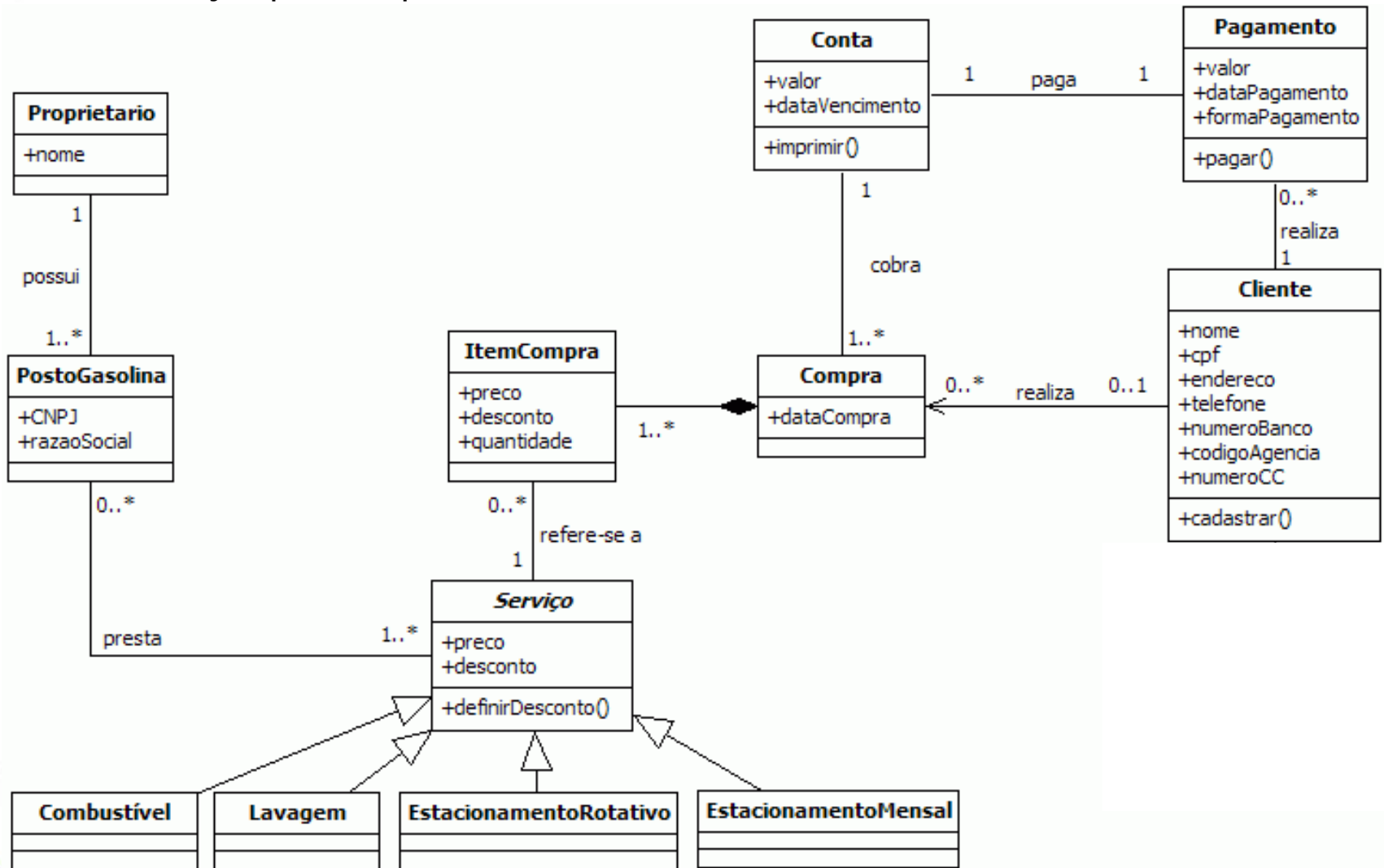


Diagrama de Sequência

- Os diagramas de Sequência e Colaboração/Comunicação são usados para modelar a **interação** entre as diversas classes identificadas nas fases de análise e projeto.
- Em outras palavras, esses diagramas modelam a troca de mensagens entre as classes, ou seja, o **comportamento** das classes.
- Esses diagramas são normalmente criados no início da fase de projeto.
- O diagrama de Sequência modela a troca de mensagens entre as classes tendo por base uma linha de tempo, ou seja, a ordem em que os eventos ocorrem é capturada nesse diagrama.

Diagrama de Sequência no Contexto

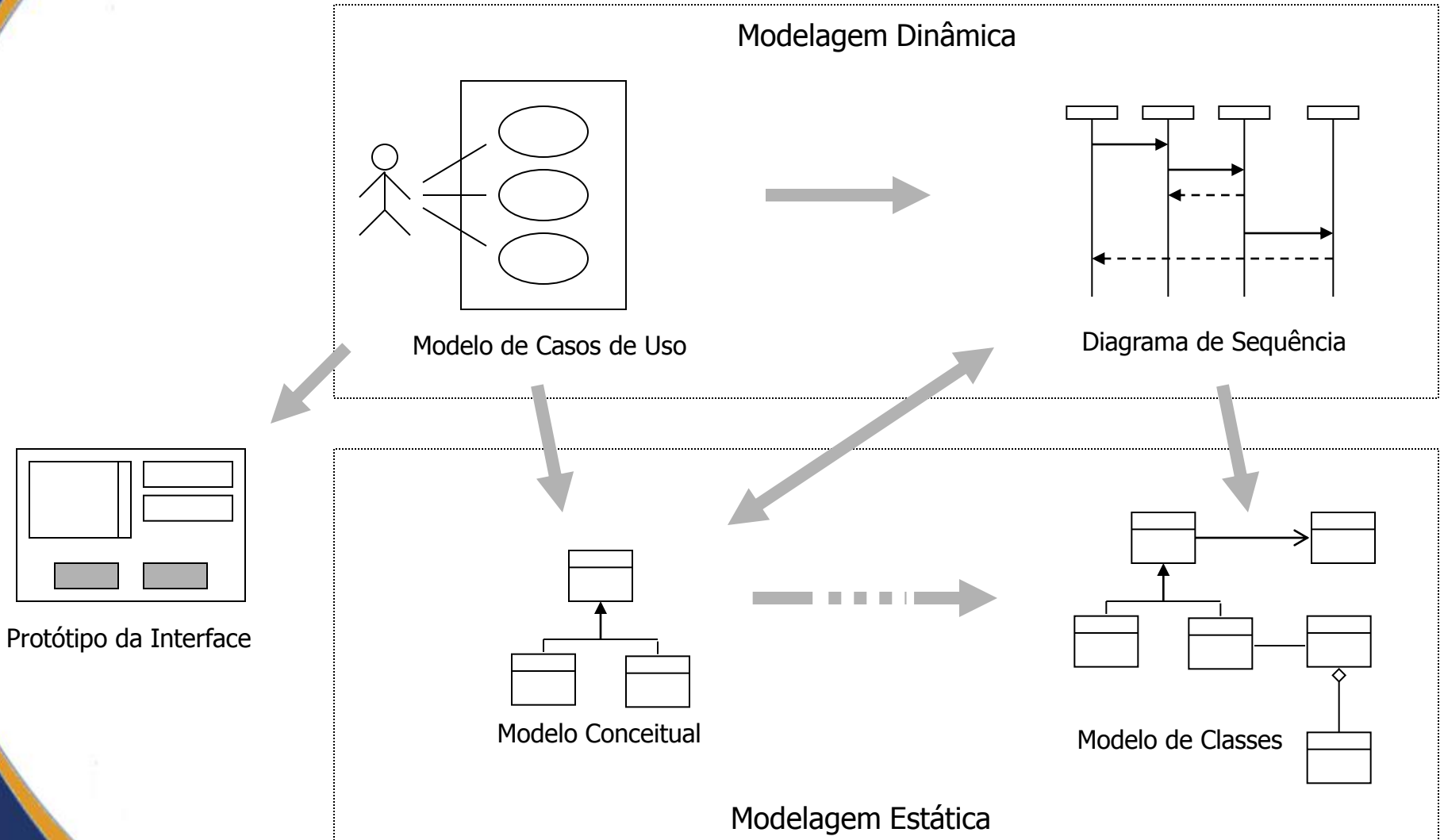


Diagrama de Sequência

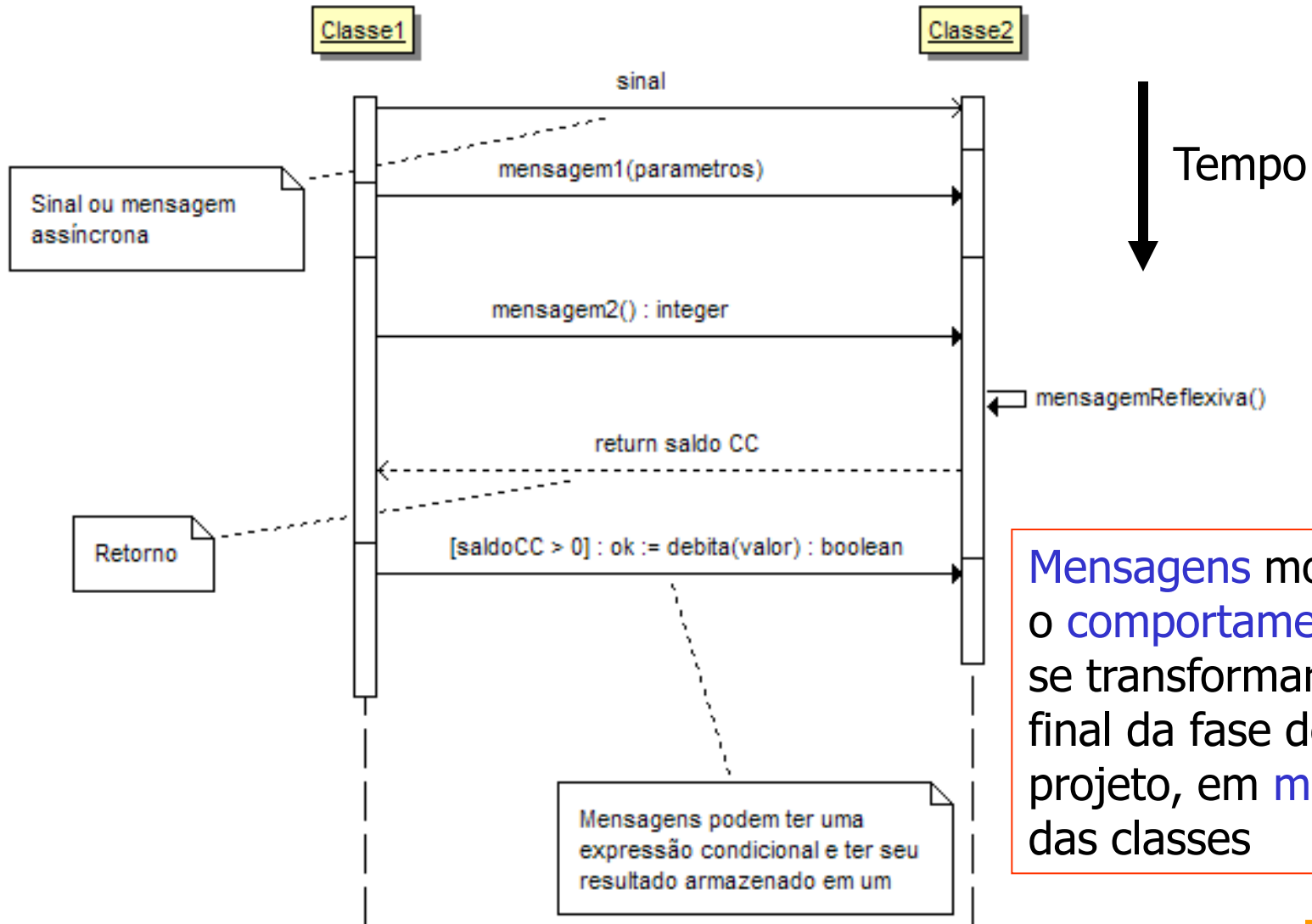


Diagrama de Sequência

- Relembrando: baseados no conjunto de casos de uso podemos usar a técnica de extração de substantivos e verbos para obter uma versão inicial do modelo conceitual.
- Assim, ao final da fase de análise temos somente o modelo conceitual formado pelas classes conceituais extraídas das descrições dos casos de uso.
- A **identificação e alocação dos comportamentos** associados às classes conceituais a partir dos **casos de uso** parece ser uma tarefa **bem mais complexa** do que a identificação dos atributos.
- Esse fato se explica porque:
 - ✓ A descrição dos casos de uso está em um nível de abstração muito alto, o que dificulta a associação dos comportamentos lá identificados à uma classe.
 - ✓ O modelo de classes oferece apenas uma visão estática.

Diagrama de Sequência

- Baseados em que vamos, então, modelar o comportamento?
- Nos Casos de Uso, pois o comportamento do sistema está lá!
- Como ?
- Uma solução: criar **um diagrama de seqüência para cada caso de uso**, ou seja, vamos modelar o comportamento do sistema para cada interação ator x sistema que identificamos anteriormente.
- É importante manter o foco correto:
 - ✓ Concentrar o esforço somente nas classes conceituais nesse momento
 - ✓ A idéia principal é modelar as ações do sistema como mensagens trocadas entre as classes conceituais e não criar um fluxograma detalhado.
 - ✓ As demais classes que farão parte do modelo de classes final serão modeladas nos passos subseqüentes do projeto.

Diagrama de Sequência

- Exercício 1: criar o modelo conceitual e o diagrama de seqüência do seguinte caso de uso:

Objetivo: permitir ao cliente consultar os dados de um produto em uma loja de e-commerce.

Ator: Cliente

Pré-condições: não há

Fluxo Principal:

1. O Cliente acessa a página de consulta de produtos.
2. O Sistema apresenta a lista de categorias de produtos.
3. O Cliente solicita uma categoria.
4. O Sistema apresenta a lista de produtos da categoria selecionada.
5. O Cliente seleciona um produto.
6. O Sistema apresenta o código, a descrição, a foto, o preço e a quantidade em estoque do produto selecionado.

Pós-condições: Cliente obtém os dados do produto desejado.

Diagrama de Sequência

- Exercício 1: criar o modelo conceitual e o diagrama de seqüência do seguinte caso de uso:

Objetivo: permitir ao cliente consultar os dados de um produto em uma loja de e-commerce.

Ator: Cliente

Pré-condições: não há

Fluxo Principal:

1. O Cliente acessa a página de consulta de **produtos**.
2. O Sistema apresenta a lista de **categorias** de produtos.
3. O Cliente solicita uma **categoria**.
4. O Sistema apresenta a lista de produtos da categoria selecionada.
5. O Cliente seleciona um **produto**.
6. O Sistema apresenta o **código**, a **descrição**, a **foto**, o **preço** e a **quantidade** em **estoque** do **produto** selecionado.

Pós-condições: Cliente obtém os dados do produto desejado.

Diagrama de Sequência

- Modelo conceitual (1ª. versão - sem comportamentos)

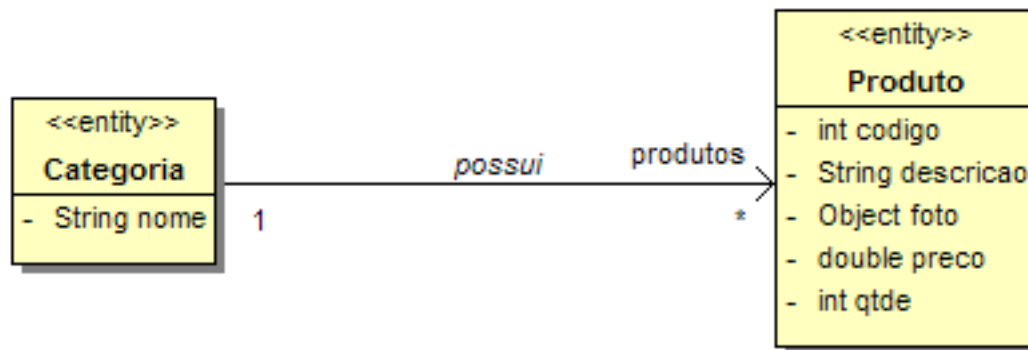

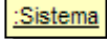
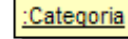


Diagrama de Sequência


:Cliente
|


:Sistema
|


:Categoria
|

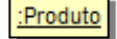

:Produto
|

Diagrama de Sequência

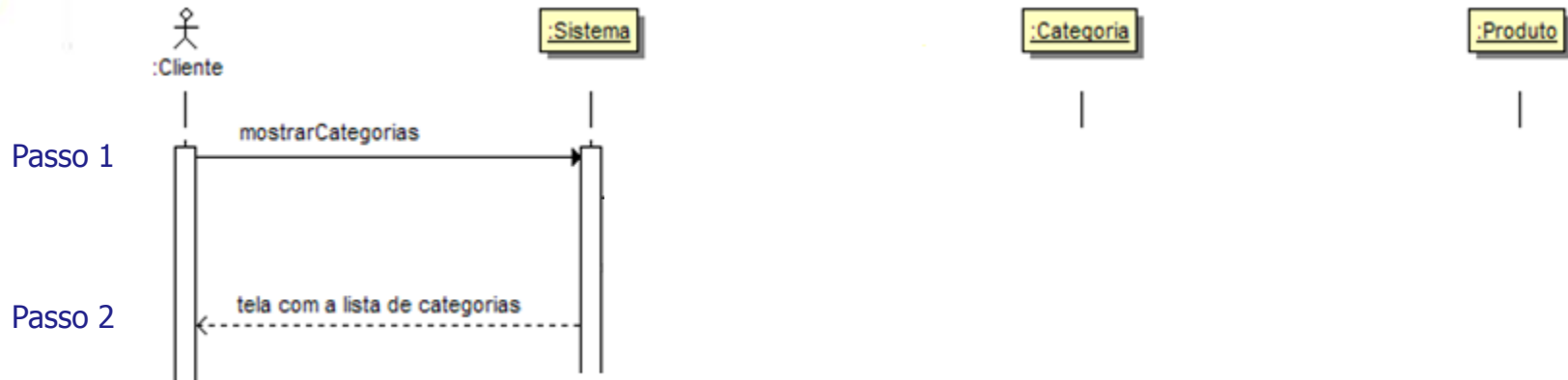


Diagrama de Sequência

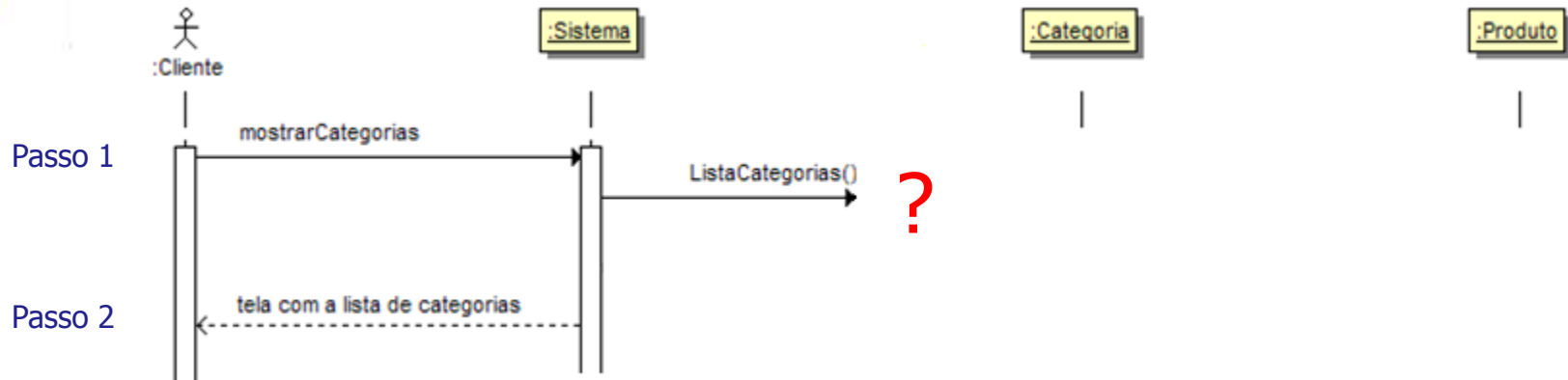


Diagrama de Sequência

- Modelo conceitual (2ª. versão - sem comportamentos)

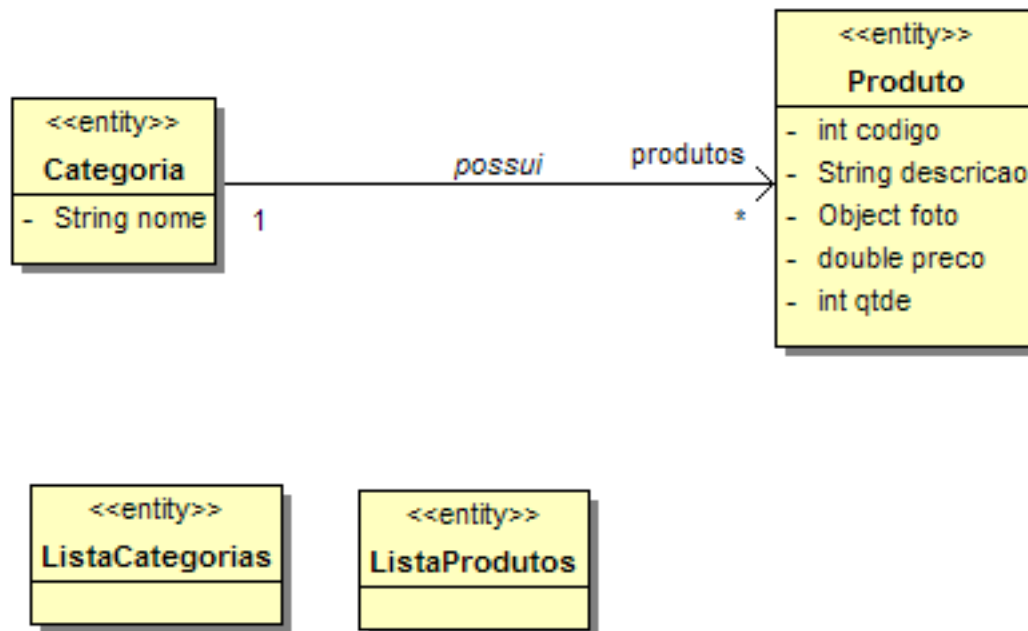


Diagrama de Sequência



Diagrama de Sequência

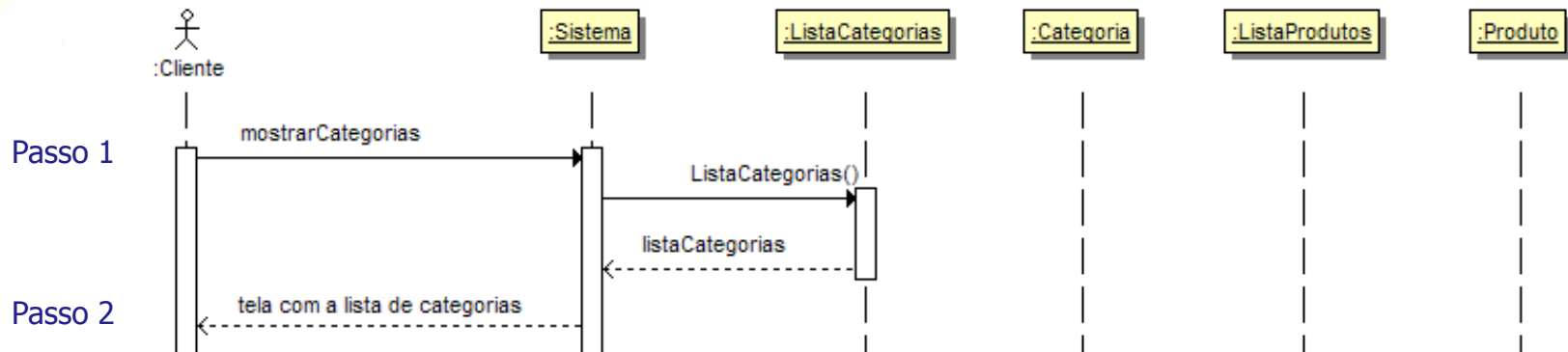


Diagrama de Sequência

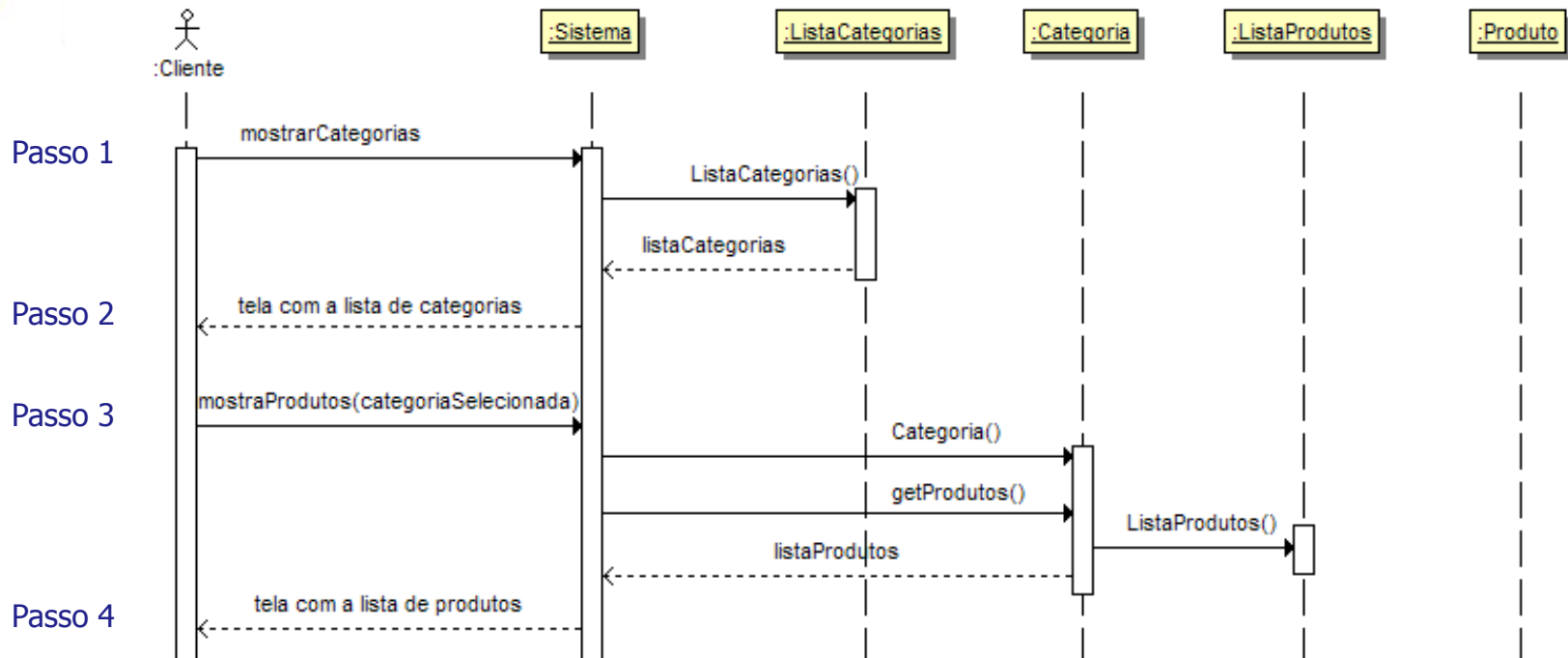


Diagrama de Sequência

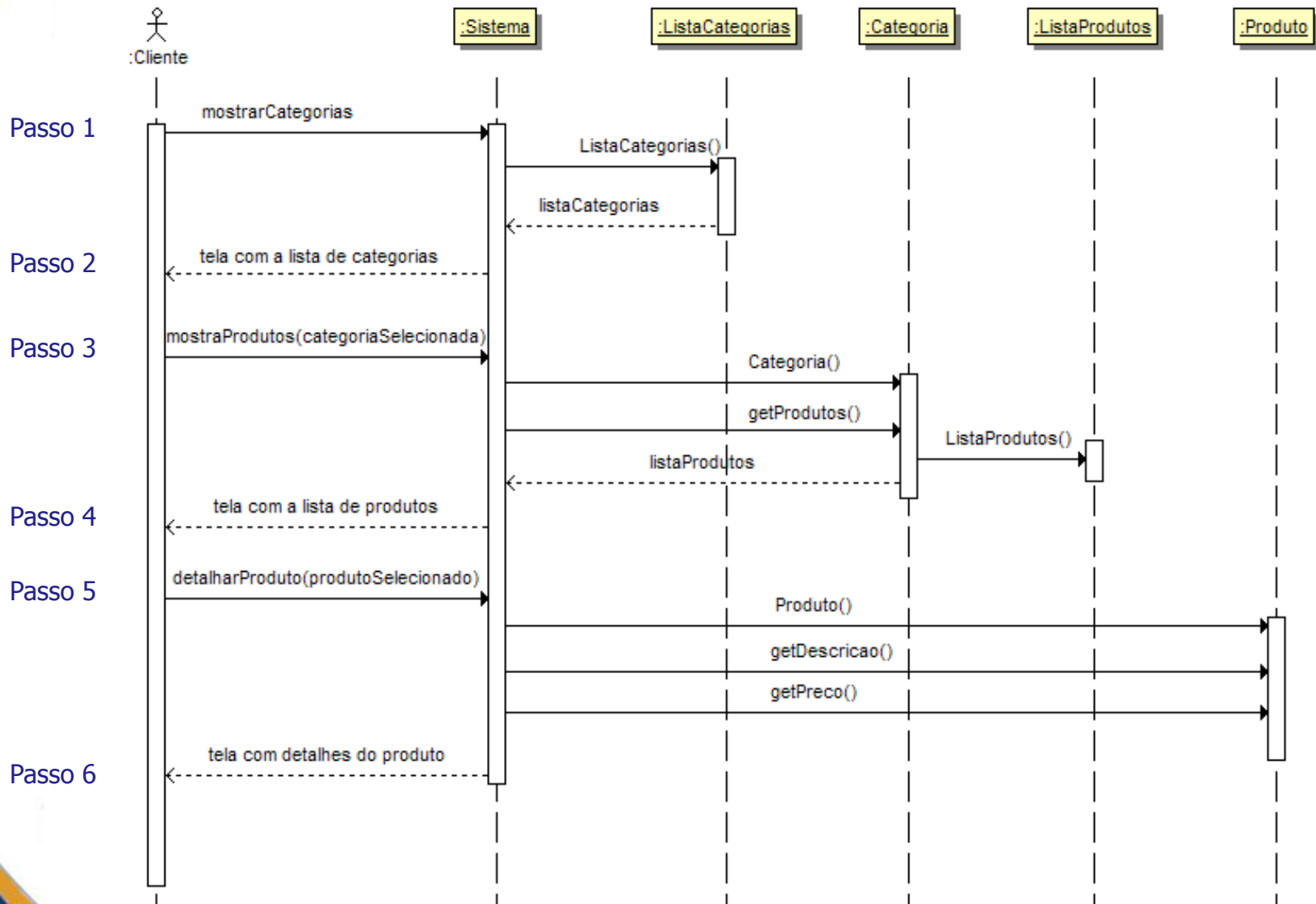


Diagrama de Sequência

- Modelo conceitual (versão final – atributos e métodos)

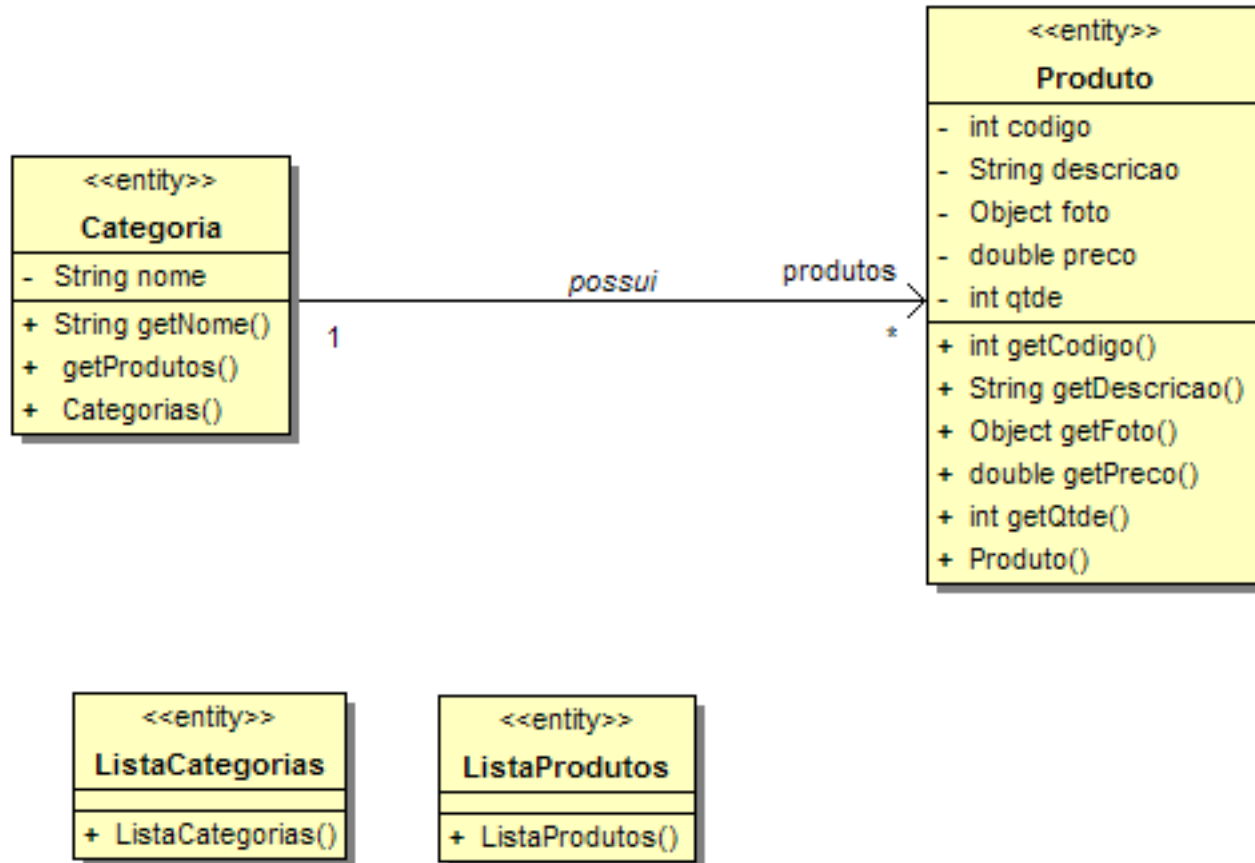


Diagrama de Sequência – UML 2

- Uma alteração importante na UML 2 foi introdução de **frames** e seus **tipos**.
- Frame: define um contexto parametrizável em um diagrama de seqüência, possibilitando também o seu reuso.

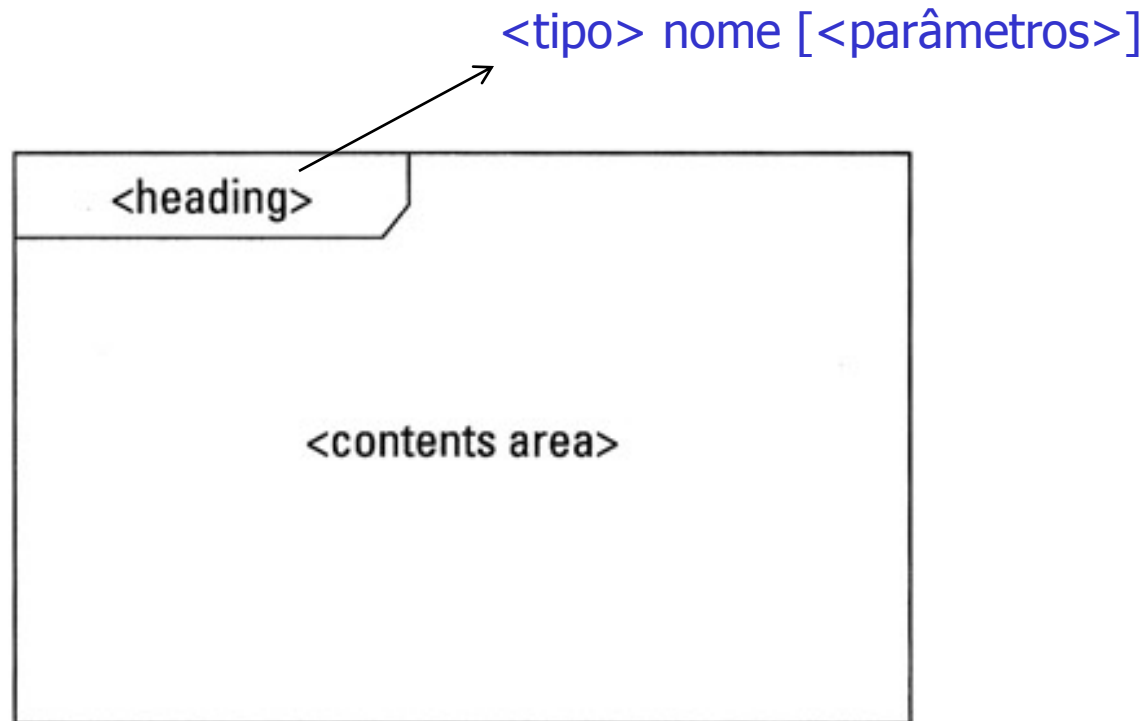


Diagrama de Sequência – UML 2

- Alguns tipos de frame:
 - **optional:** define um bloco de interações que ocorrem somente se uma determinada condição for verdadeira (condição de guarda). Equivalente a um *if-then*.
 - **alternative:** define blocos de interações que ocorrem se as condições associadas a cada um deles for verdadeira. Apenas um conjunto é executado. Equivalente a um *if-then-elseif-then-elseif-then-elseif...*
 - **loop:** define um bloco de interações que ocorrem em *loop*.
 - **break:** equivalente ao opt, só que após a execução do bloco, a execução do diagrama/frame é interrompida.
 - **reference:** define a execução de um frame no local onde a referência é colocada.
 - **parallel:** define blocos de interações que ocorrem em paralelo.

Diagrama de Sequência – UML 2

- **optional:** define um bloco de interações que ocorrem somente se uma determinada condição for verdadeira (condição de guarda). Equivalente a um *if-then*.

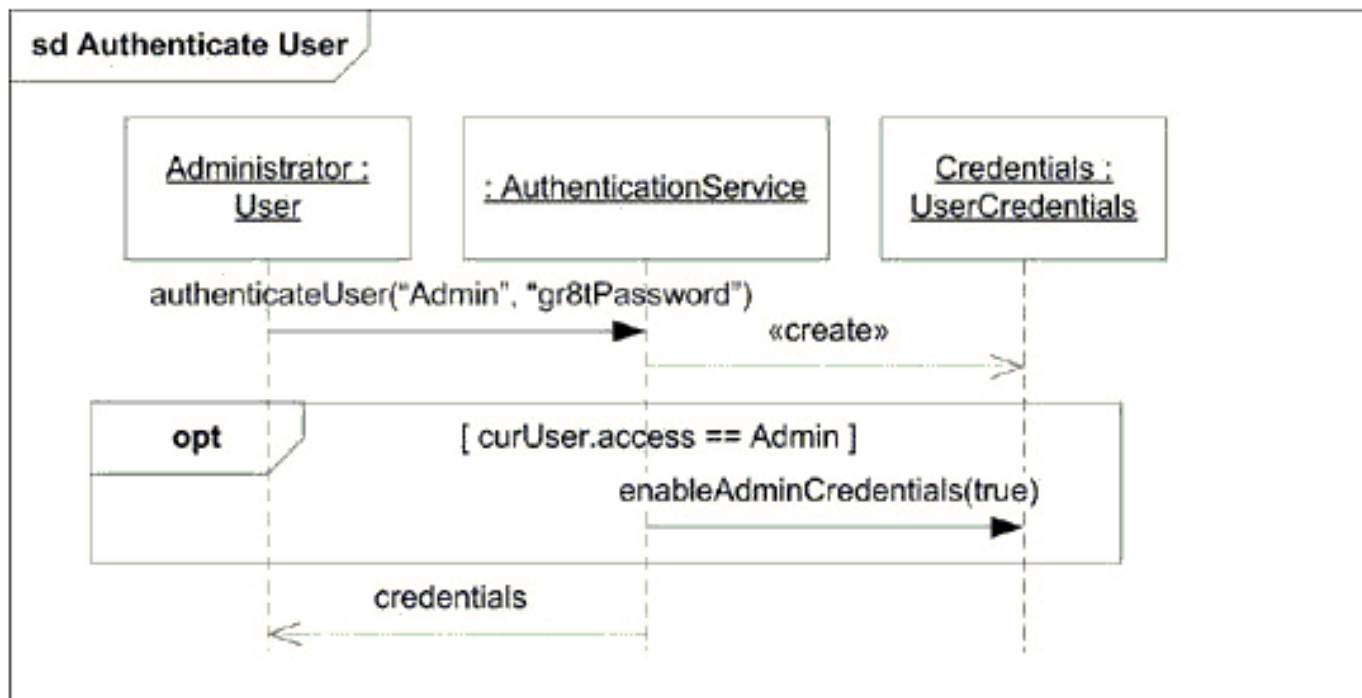


Diagrama de Sequência – UML 2

- **alternative:** define blocos de interações que ocorrem se as condições associadas a cada um deles for verdadeira. Apenas um conjunto é executado. Equivalente a um *if-then-elseif-then-elseif-then-elseif...*

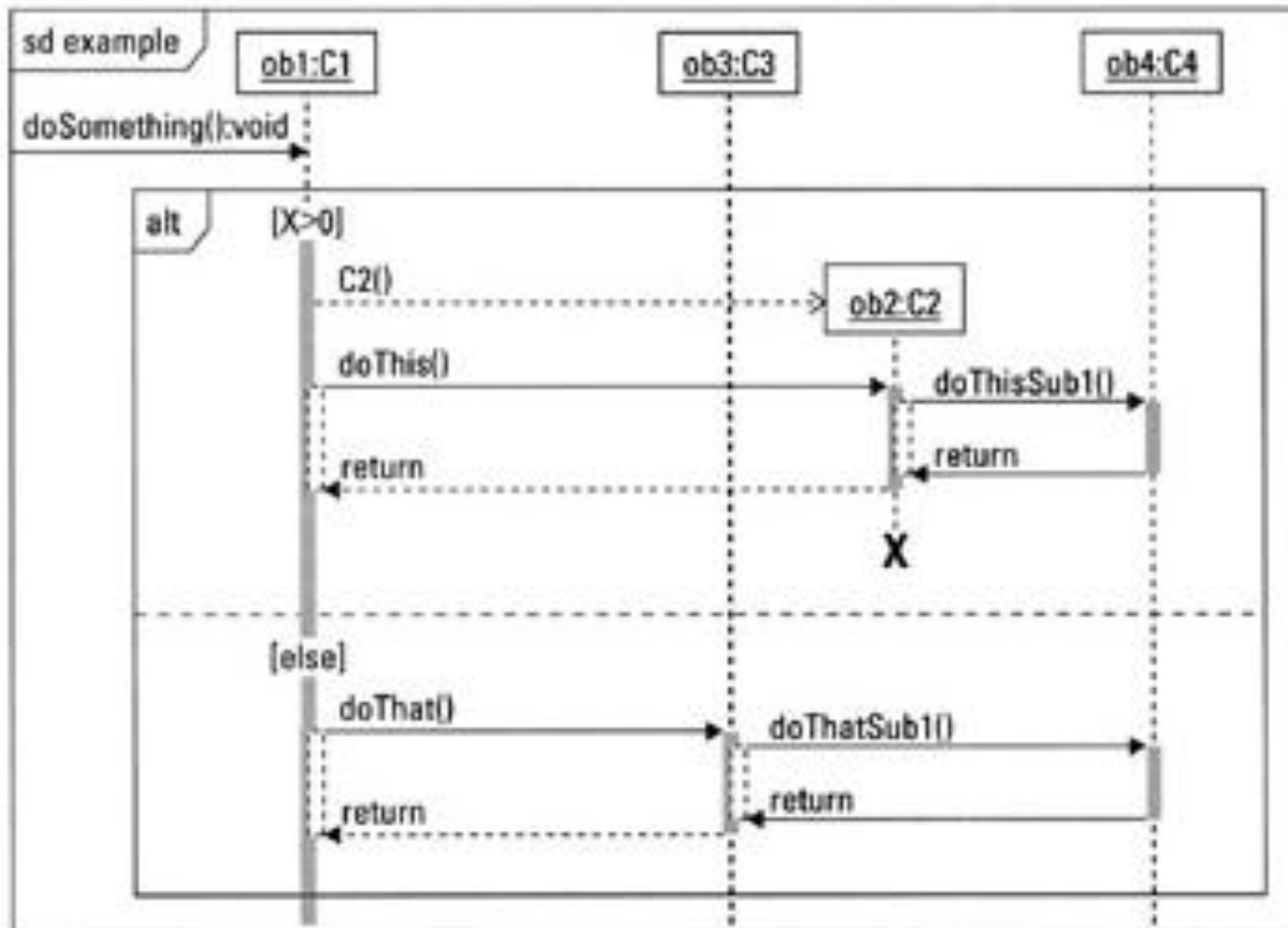


Diagrama de Sequência – UML 2

- **loop:** define um bloco de interações que ocorrem em *loop*.

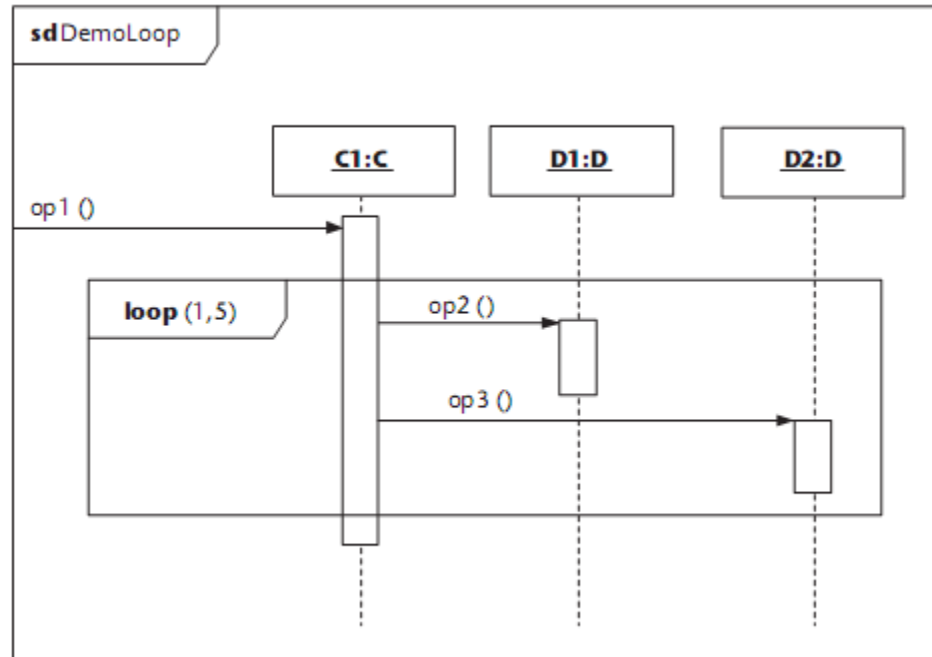


Diagrama de Sequência – UML 2

- **break:** equivalente ao opt, só que após a execução do bloco, a execução do diagrama/frame é interrompida.

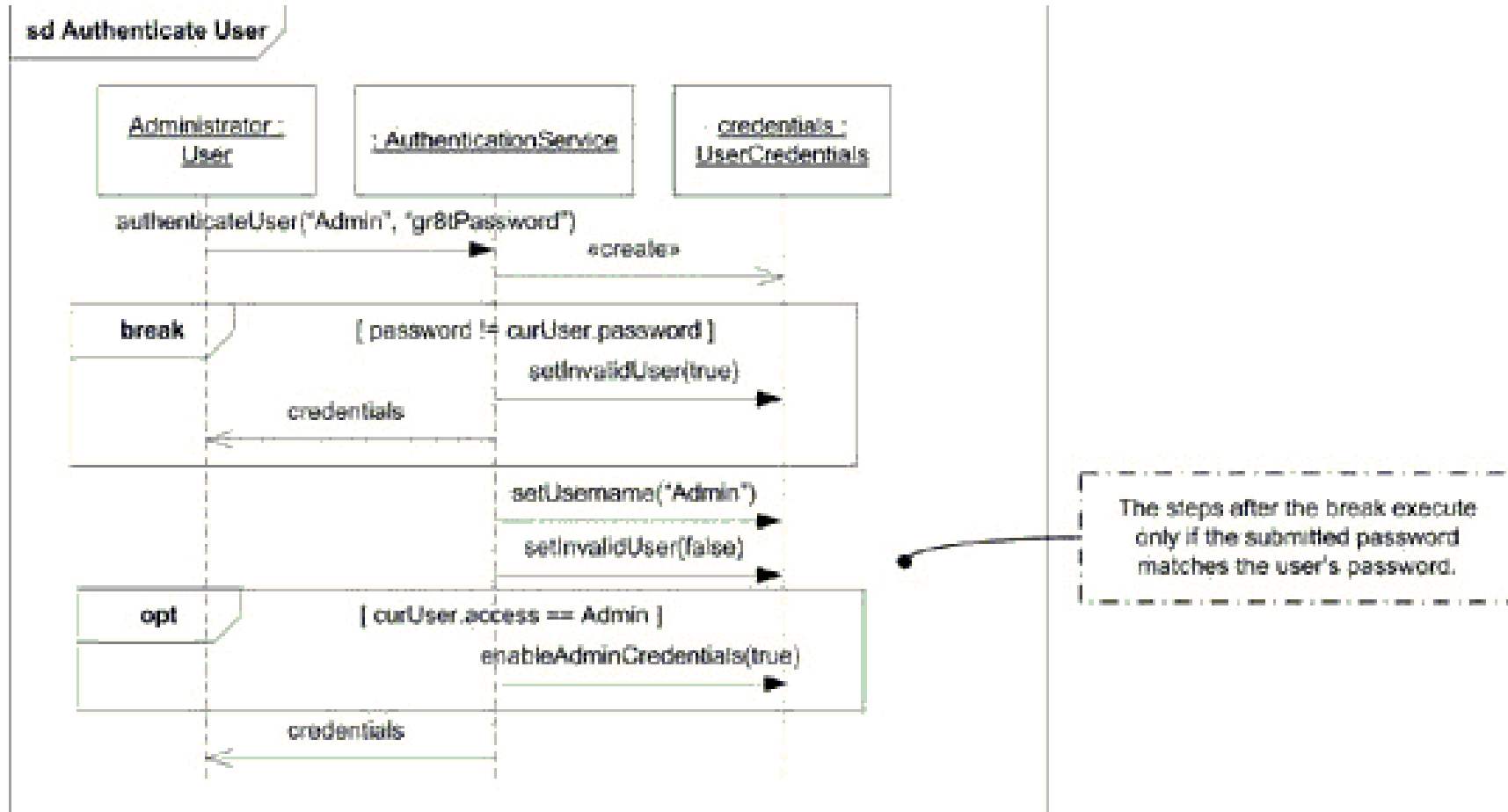


Diagrama de Sequência

➤ Importante:

1. É importante criar UM diagrama de seqüência para cada caso de uso.
2. Extraia as classes dos casos de uso antes de iniciar o diagrama de seqüência.
3. Pode-se colocar a descrição do caso de uso ao lado do diagrama de seqüência a fim de conferir se os passos do caso de uso estão refletidos nas mensagens.
4. Assim como os casos de uso, os diagramas de seqüência devem, inicialmente, se concentrar no fluxo principal. Os fluxos alternativos e de exceção devem ser feitos posteriormente.
5. Não transforme o seu diagrama de seqüência em um grande *flowchart*. O uso principal do diagrama é permitir a definição de responsabilidades (comportamentos).
6. Foque nas mensagens que são realmente relevantes para o comportamento do software como um todo e não perca tempo com métodos de pouca importância.
7. Verifique cuidadosamente se a origem da mensagem está no controle da seqüência no momento em que ela envia a mensagem.
8. Verifique cuidadosamente se o destino da mensagem pode realmente tratar a mensagem enviada.
9. Siga os princípios básicos de OO quando definir uma responsabilidade, ou seja, cada classe deve se concentrar em um conjunto coerente e coeso de métodos.
10. Atualize o modelo conceitual e o modelo de caso de uso, conforme o diagrama de seqüência vai evoluindo.

Diagrama de Sequência – UML 2

- **reference:** define a execução de um frame no local onde a referência é colocada.

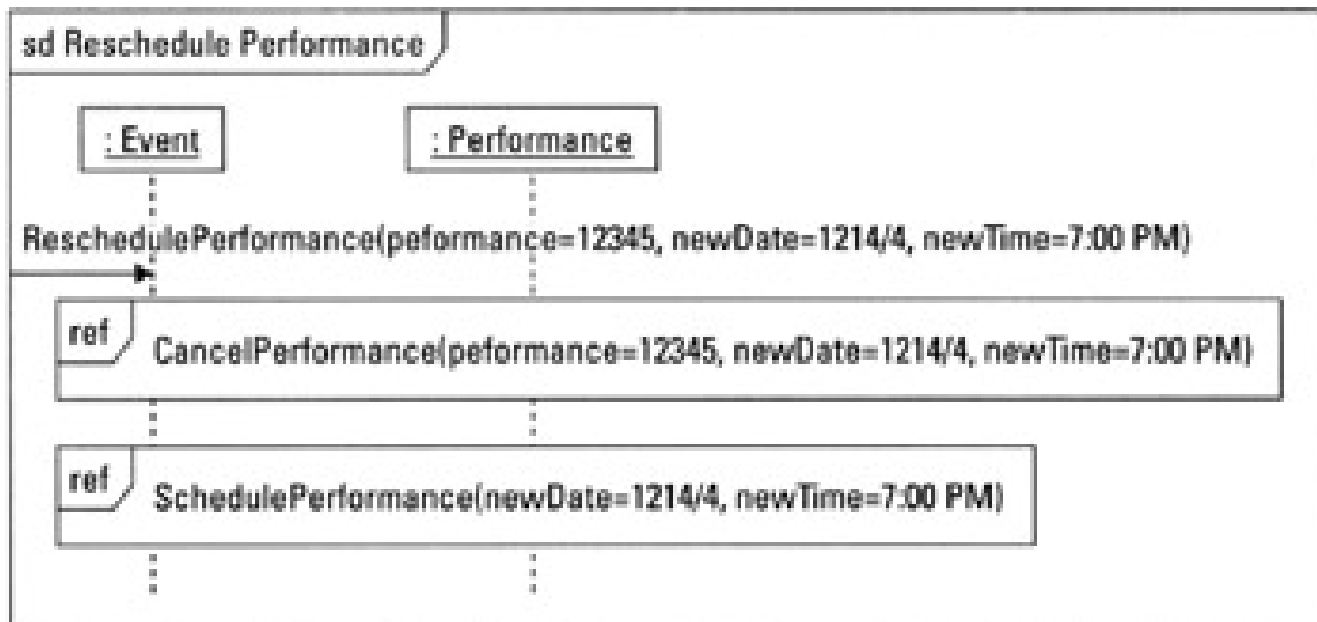


Diagrama de Sequência – UML 2

- **parallel:** define blocos de interações que ocorrem em paralelo.

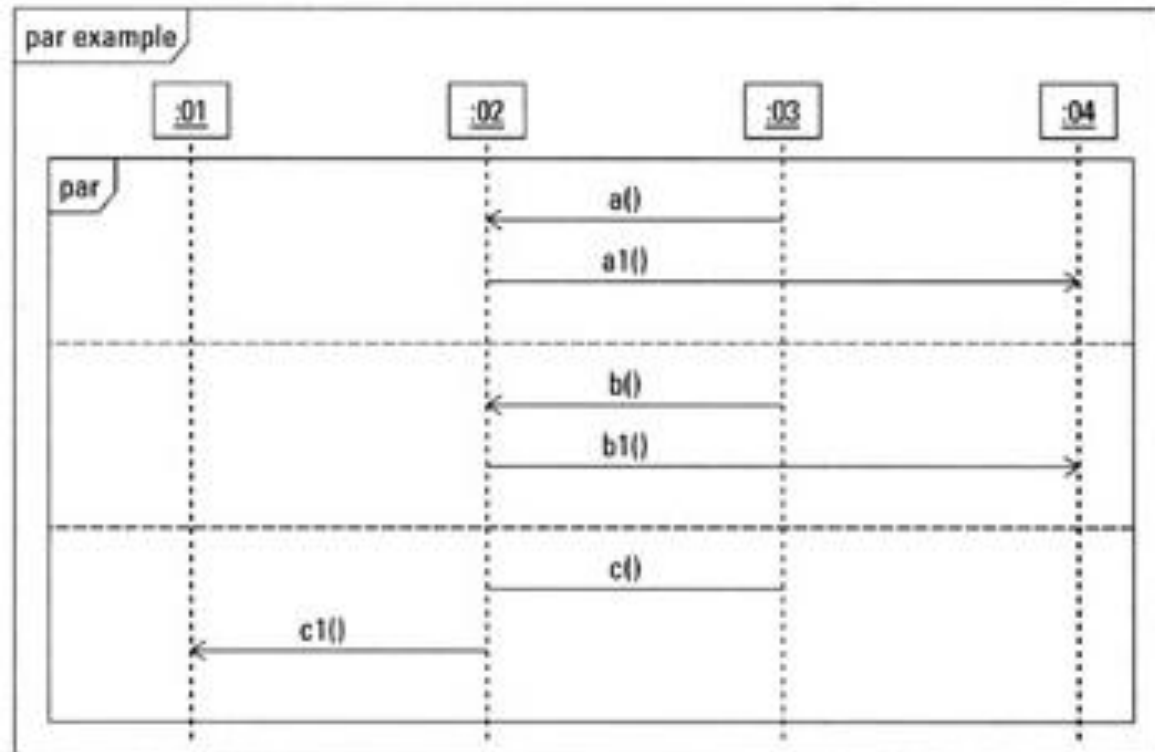


Diagrama de Sequência

- Exercício 2: criar o diagrama de seqüência para o UC01-Consultar Carrinho

Objetivo: permitir ao internauta ou cliente listar os produtos que se encontram atualmente no carrinho de compras em uma loja eletrônica.

Ator: Cliente

Pré-condições: não há

Fluxo Principal:

1. O Sistema apresenta a lista de produtos da seguinte forma:
 - Lista de CDs: título do CD, nome do artista, quantidade, preço unitário e valor total do item (preço x quantidade).
 - Lista de livros: título do livro, ISBN, nomes dos autores, quantidade, preço unitário e valor total do item (preço x quantidade).
 - Valor total do pedido sem o frete.
2. O Sistema também apresenta, para cada item, uma opção de exclusão [A1]

Fluxos Alternativos:

[A1] O ator seleciona a opção de exclusão de item

1. O Sistema solicita a confirmação da exclusão.
2. Se o ator confirmar então o sistema exclui o produto selecionado
3. Retorna para o passo 1 do fluxo principal.

Pós-condições: A lista de produtos existentes no carrinho é apresentada

Diagrama de Sequência

➤ Modelo conceitual inicial:

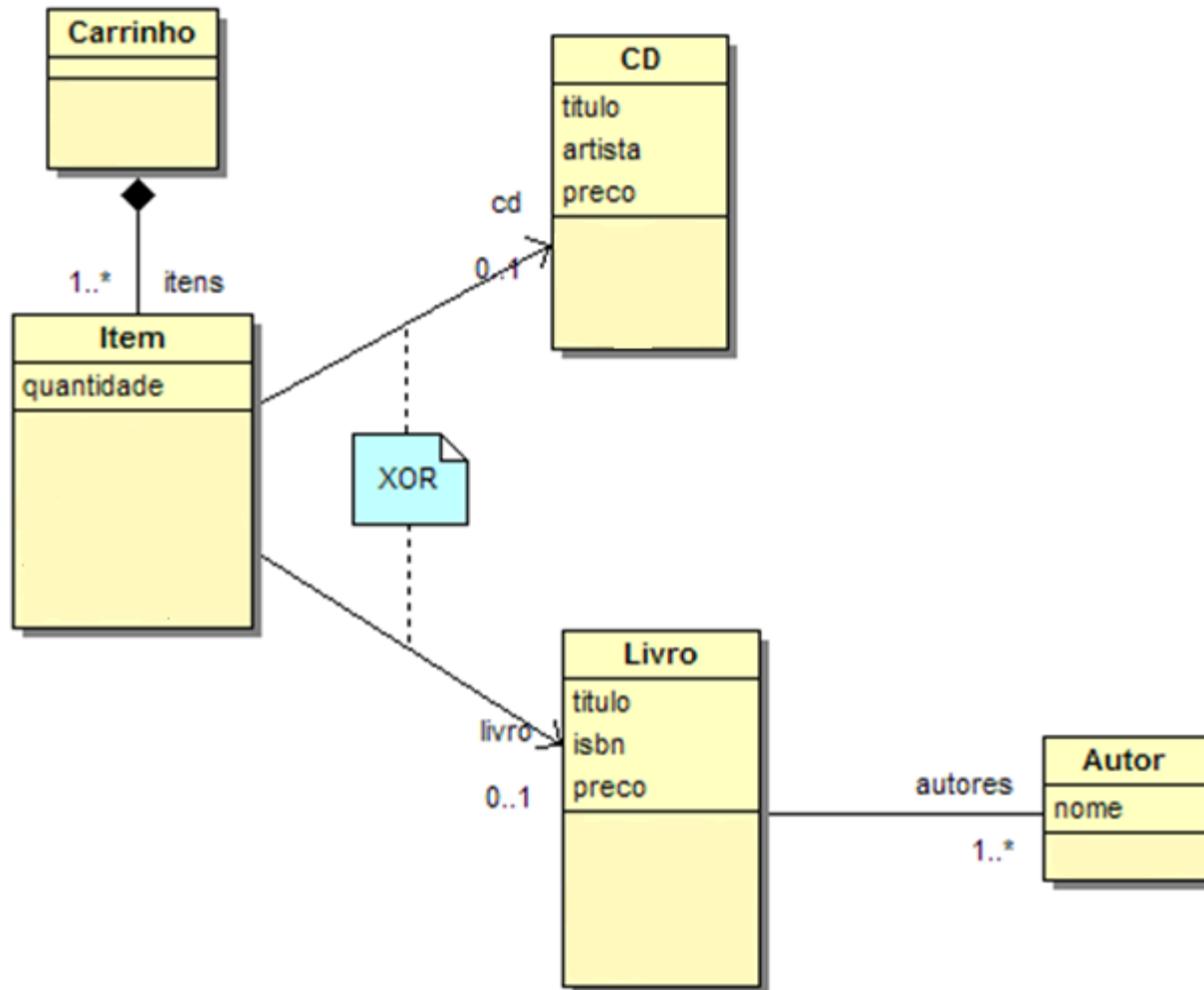


Diagrama de Sequência

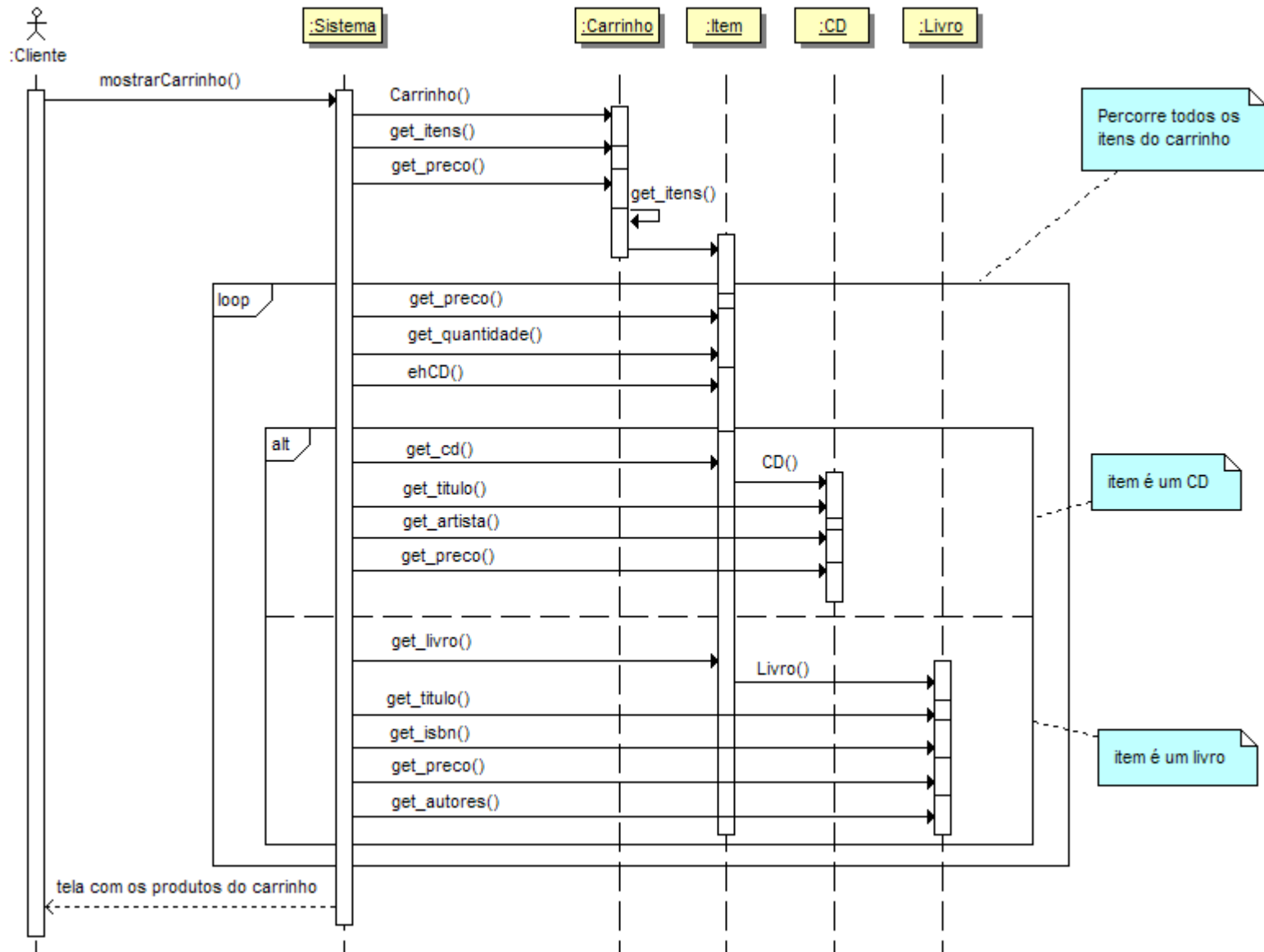


Diagrama de Sequência

- Modelo conceitual final:

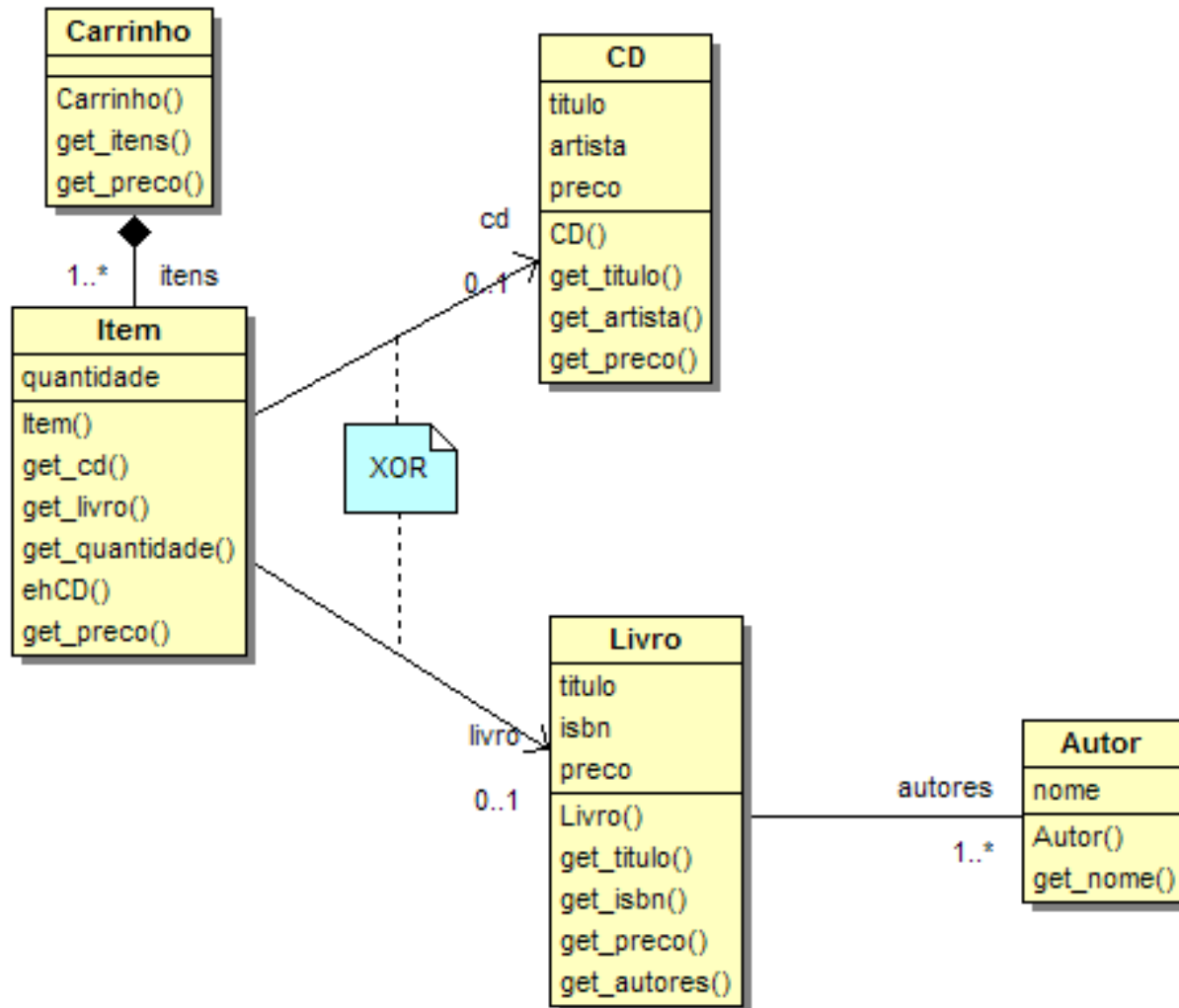


Diagrama de Estado

- Os diagramas de Estado têm como objetivo principal fornecer uma definição formal do comportamento de um objeto, permitindo a descrição dos eventos e transições de estados aos quais ele está sujeito.
- Os diagramas de Estado capturam o ciclo de vida de um objeto em função dos estados que esse objeto pode assumir, dos eventos que provocam essas mudanças e das ações que são tomadas a partir desses eventos.
- Os diagramas de Estado não são escritos para todas as classes de um sistema, mas apenas para aquelas que possuem um número definido de estados conhecidos e onde o comportamento das classes é afetado e modificado por eventos.
- Podem ser utilizados tanto durante a fase de análise quanto durante a fase de projeto.

Diagramas de Estado no Contexto

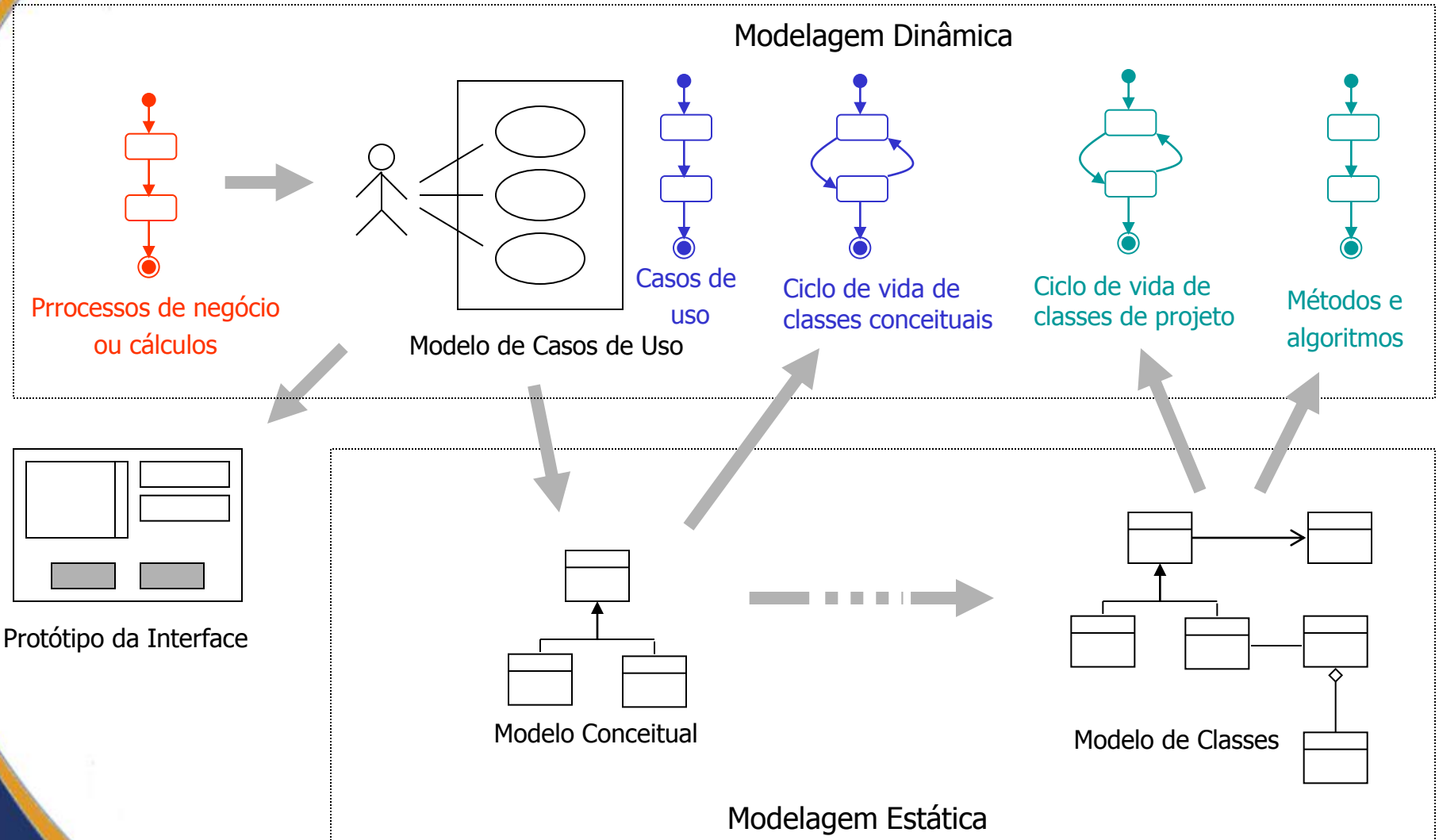


Diagrama de Estado

- O diagrama de Estado pode conter os seguintes componentes básicos:


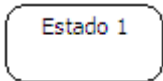
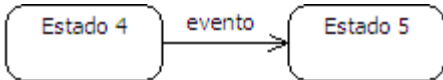
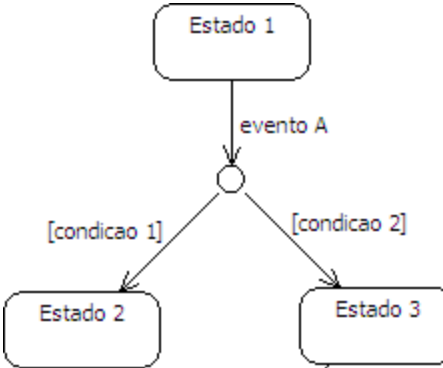
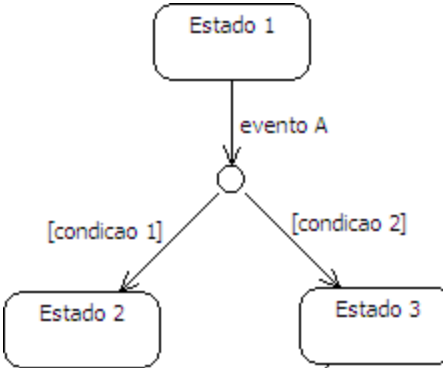
Descrição	Símbolo
Estado inicial: denota o estado em que um objeto é criado. Deve existir apenas um estado inicial.	
Estado: representado por um retângulo com os cantos arredondados com o nome do estado no seu interior.	
Transição: representa uma mudança de estado. Deve estar sempre associada a um evento.	
Evento: sempre aparece associado a uma transição indicando o evento que provoca a mudança de estado. Pode vir acompanhado de uma expressão de guarda e/ou uma ação.	
Escolha: permite definir vários estados alvo a partir de um mesmo evento. O evento deve estar associado ao segmento de entrada, enquanto as condições devem estar associadas aos segmentos de saída. Pode ser representador por um círculo (UML 1.4) ou por um losango (UML 2.0).	

Diagrama de Estado



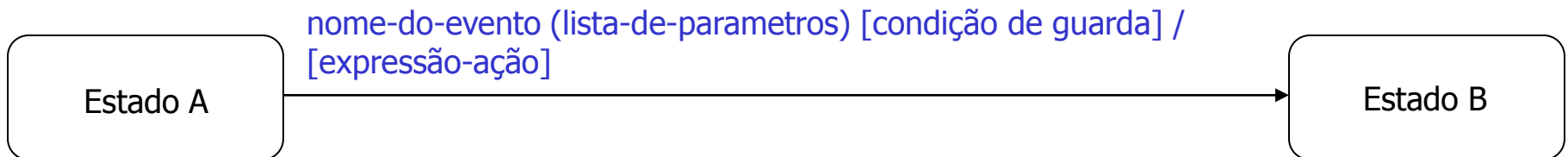
Descrição	Símbolo
Junção: é o ponto comum de junção de vários estados para um mesmo estado alvo.	
Estado final: indica o estado final de um objeto. Pode existir mais de um estado final.	

Diagrama de Estado

- A modelagem completa da **transição** pode ser feita com a seguinte sintaxe:

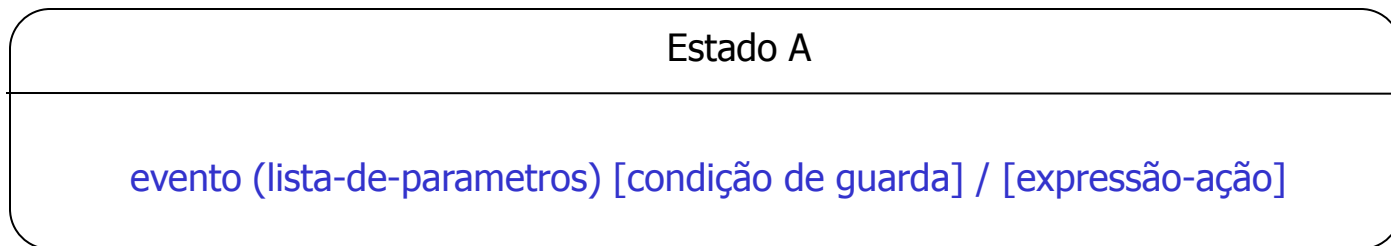


onde:

- **nome-do-evento**: é obrigatório.
- **lista-de-parametros**: parâmetros do evento (opcional).
- **condição-de-guarda**: expressão que define se o evento será ou não aceito (opcional).
- **expressão-ação**: uma ou mais ações separadas por vírgula que são disparadas quando o evento é aceito (opcional).

Diagrama de Estado

- A modelagem completa do **estado** pode ser feita com a seguinte sintaxe:



- Opcionalmente, podemos especificar eventos internos ao estado, ou seja, eventos que ocorrem e não provocam a mudança de estado:
 - **nome-do-evento**: é obrigatório.
 - **lista-de-parametros**: parâmetros do evento (opcional).
 - **condição-de-guarda**: expressão que define se o evento será ou não aceito (opcional).
 - **expressão-ação**: uma ou mais ações separadas por vírgula que são disparadas pelo evento (opcional).
- A UML pré-define 3 tipos de eventos internos:
 - **entry**: a ação associada é disparada quando o objeto entra no estado
 - **do**: a ação associada é disparada enquanto o objeto permanece no estado
 - **exit**: a ação associada é disparada quando o objeto sai do estado

Diagrama de Estado

- Exemplo: diagrama parcial de funcionamento de uma máquina de venda de refrigerante

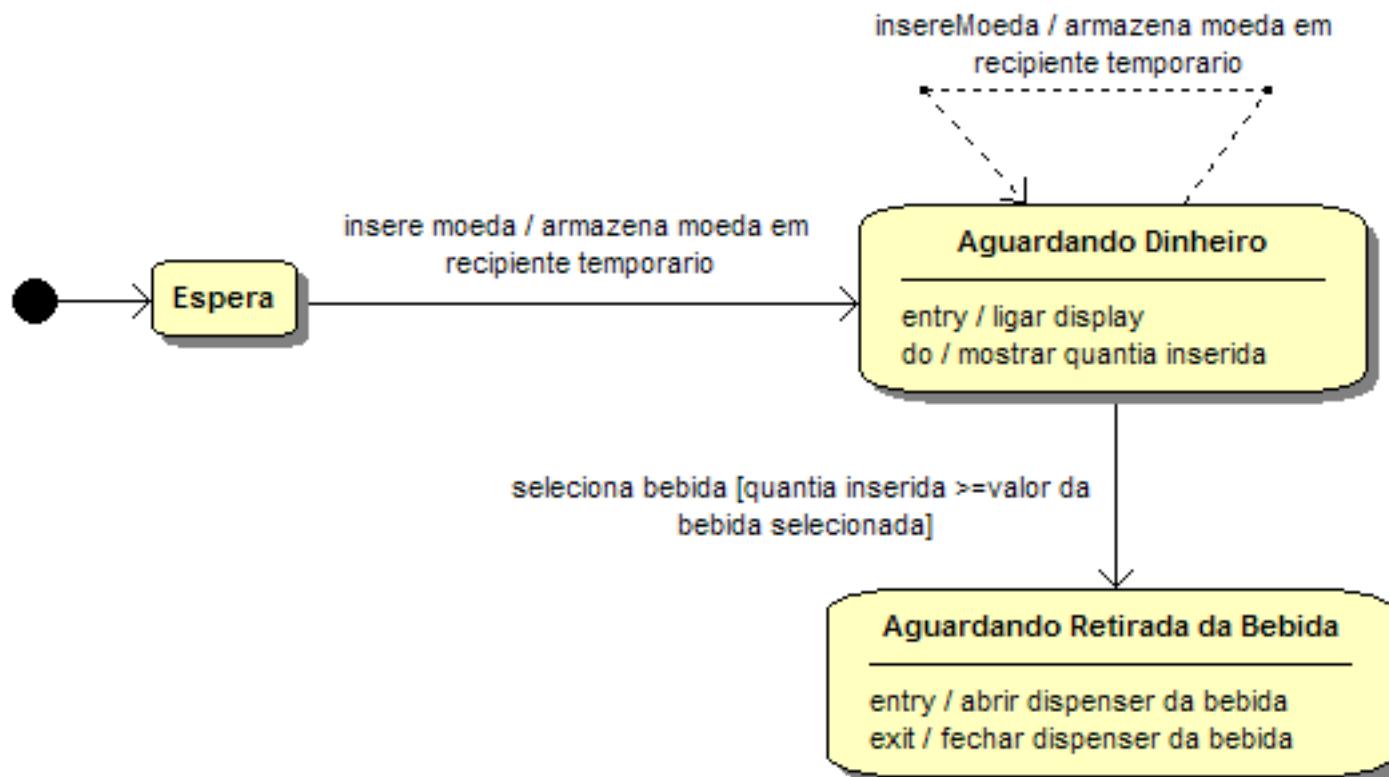


Diagrama de Estado

- Existem 2 eventos que são modelados de uma forma particular:
 - ✓ Evento de tempo decorrido: indica um evento disparado quando ocorre uma determinada quantidade de tempo. É modelado usando a expressão *after(quantidade-de-tempo)*.
 - ✓ Evento de mudança: indica um evento que ocorre no momento em que uma condição é satisfeita. É modelada usando a expressão *when(condição)*.
- Exemplos:

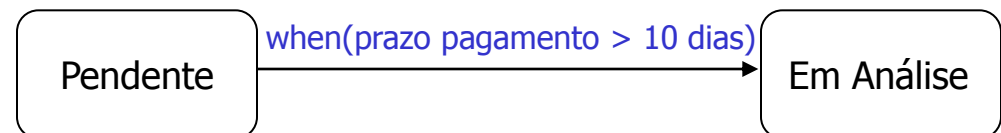
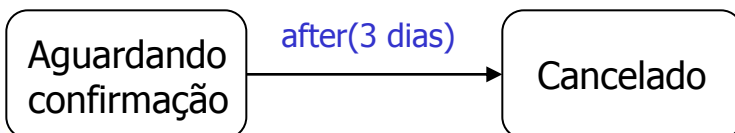
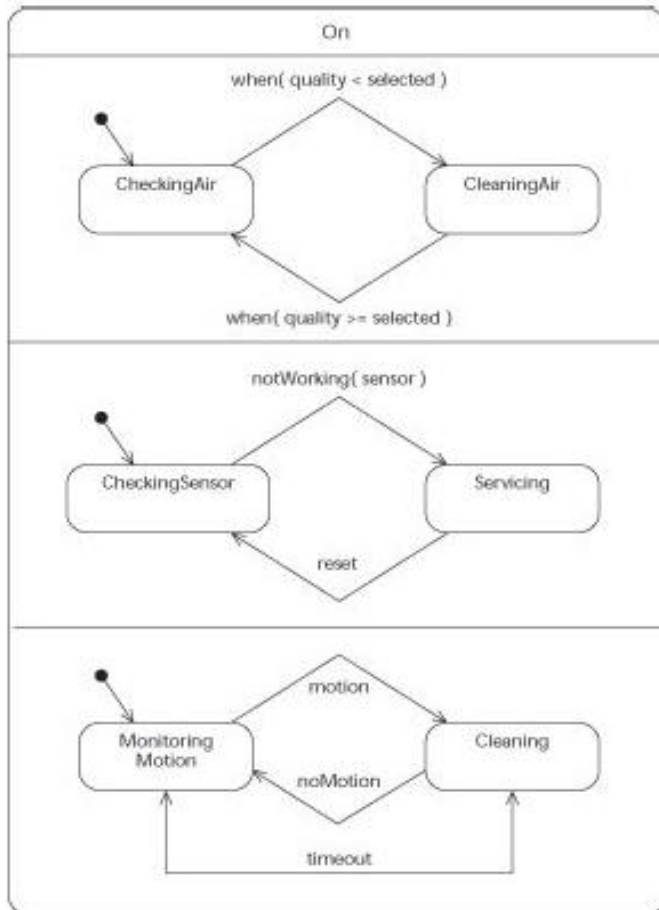


Diagrama de Estado – UML 2

- Alguns conceitos introduzidos com a UML 2:

Estados concorrentes



Estados compostos

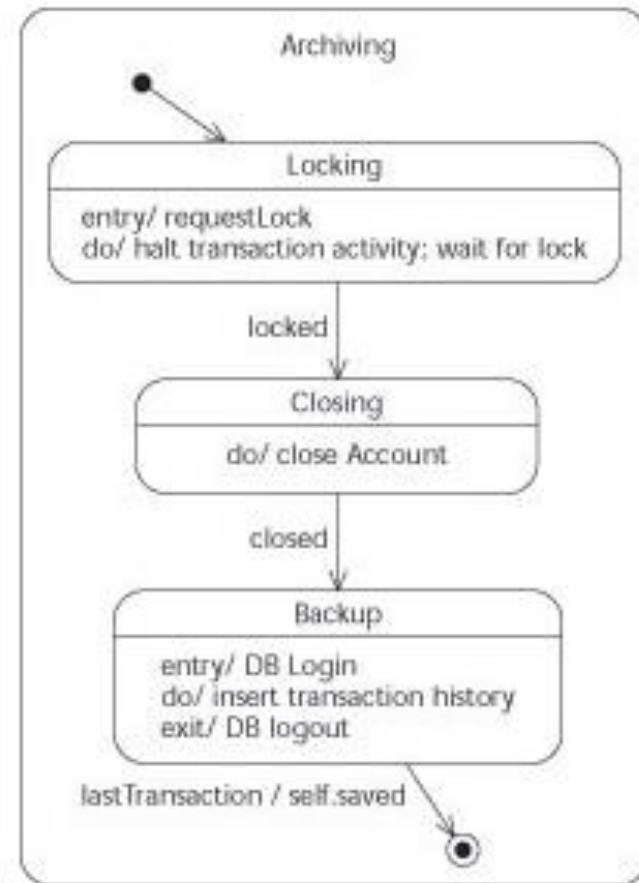


Diagrama de Estado

- Exercício 1: criar um diagrama de estado para representar a requisição no seguinte processo:
- Em uma empresa, quando um gerente deseja conceder um aumento salarial ou uma promoção de cargo a um empregado, ele abre uma requisição no RH. Inicialmente o setor de análise de pessoal verifica se a requisição é somente para aumento salarial, somente para promoção ou ambos. Caso a requisição seja somente de aumento salarial (sem mudança de cargo), a requisição é enviada para o setor de análise financeira que verificará se o departamento ao qual pertence o empregado suporta o aumento requerido. Caso a requisição seja de promoção de cargo (com ou sem aumento salarial), ela é analisada pelo próprio setor de análise de pessoal para verificar se a mesma se encontra dentro das regras de promoção. Caso esteja, ela é enviada para o setor de análise financeira para análise da viabilidade financeira. Caso contrário, ela é enviada para o Gerente de RH que poderá aprová-la mesmo fora das regras de promoção. Nesse caso ela também é enviada para o setor de análise financeira. Caso a requisição seja aprovada ou reprovada, o gerente solicitante toma ciência por email.

Diagrama de Estado

➤ Solução exercício 1:

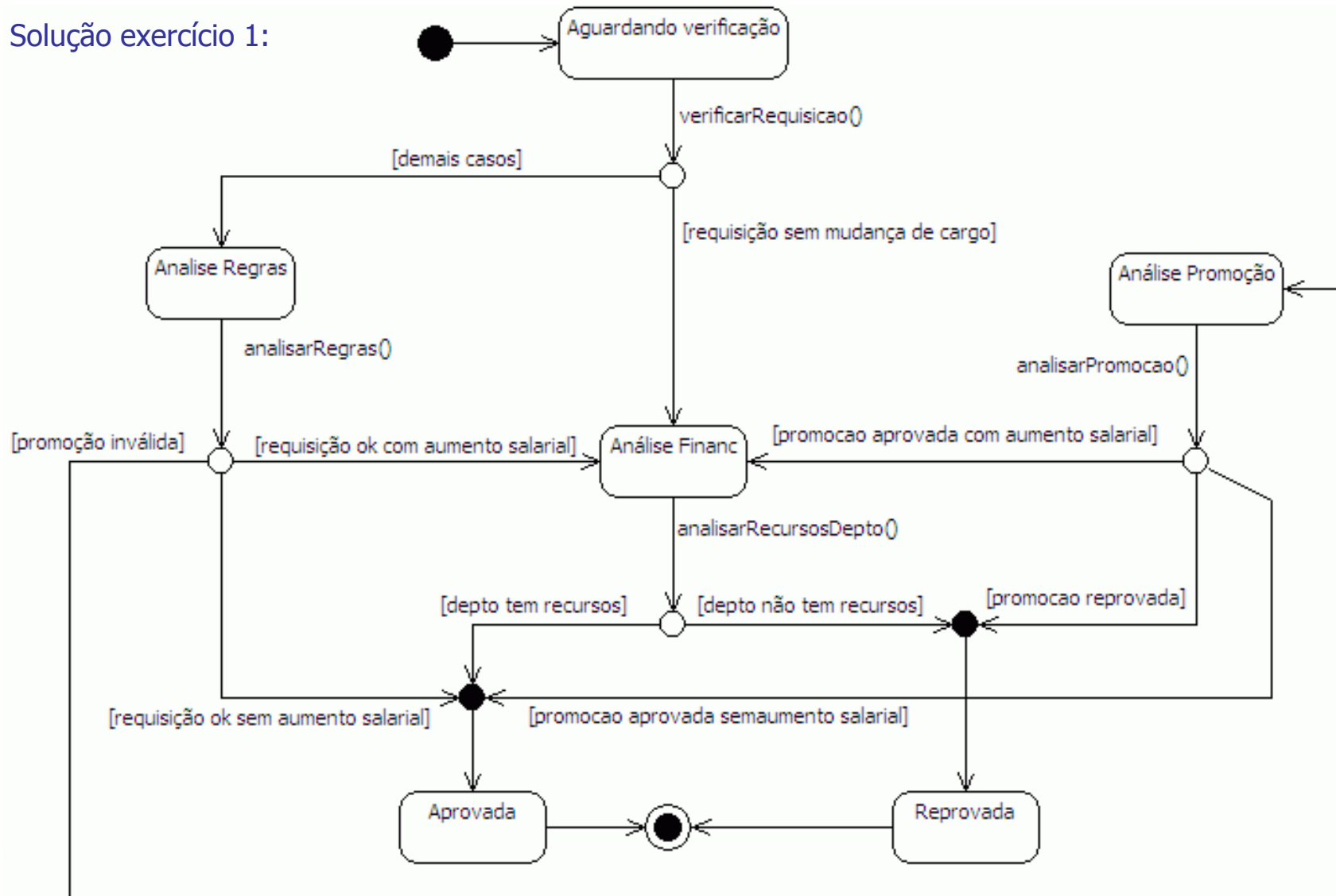


Diagrama de Estado

- Exercício 2: completar o diagrama de estado da máquina de refrigerante seguindo as seguintes regras:
1. A máquina deve devolver o troco caso o valor da bebida seja menor que o dinheiro inserido.
 2. Se não houver dinheiro suficiente para dar o troco ou não existir a bebida selecionada a máquina deve devolver todo o dinheiro inserido e não entregar a bebida.
 3. Quando solicitado, caso o cliente não tome nenhuma ação por 15 segundos, a operação de venda deverá ser cancelada.

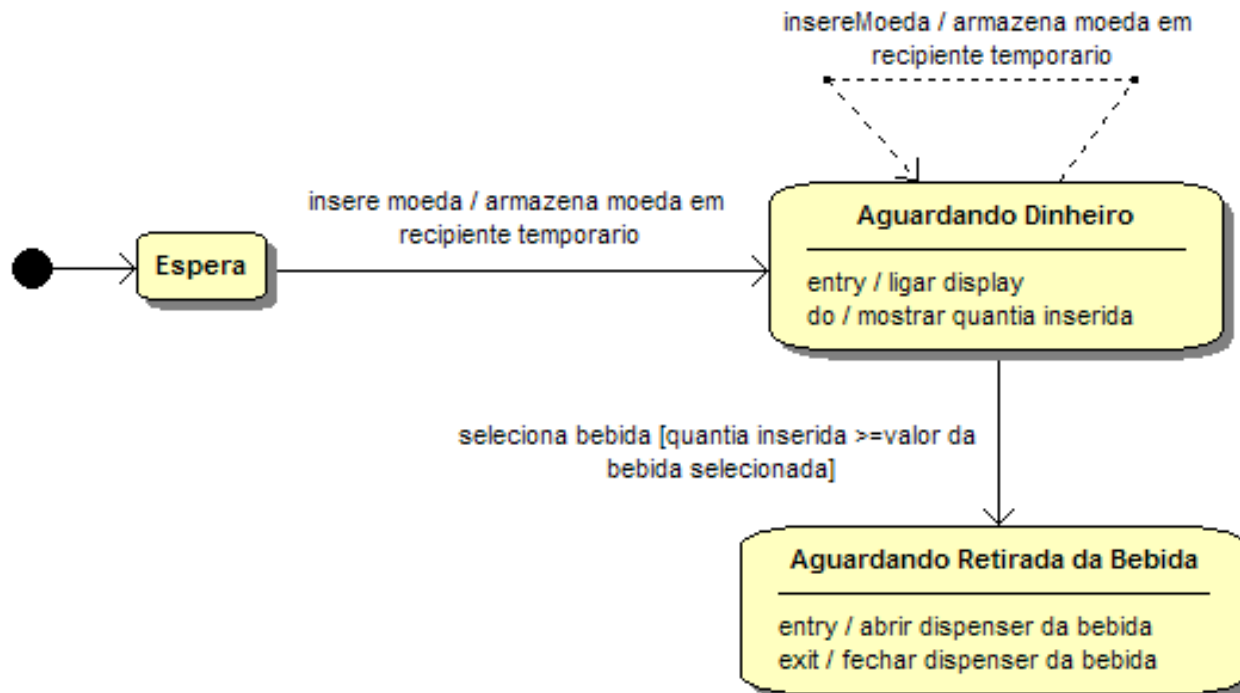


Diagrama de Estado

➤ Solução exercício 2:

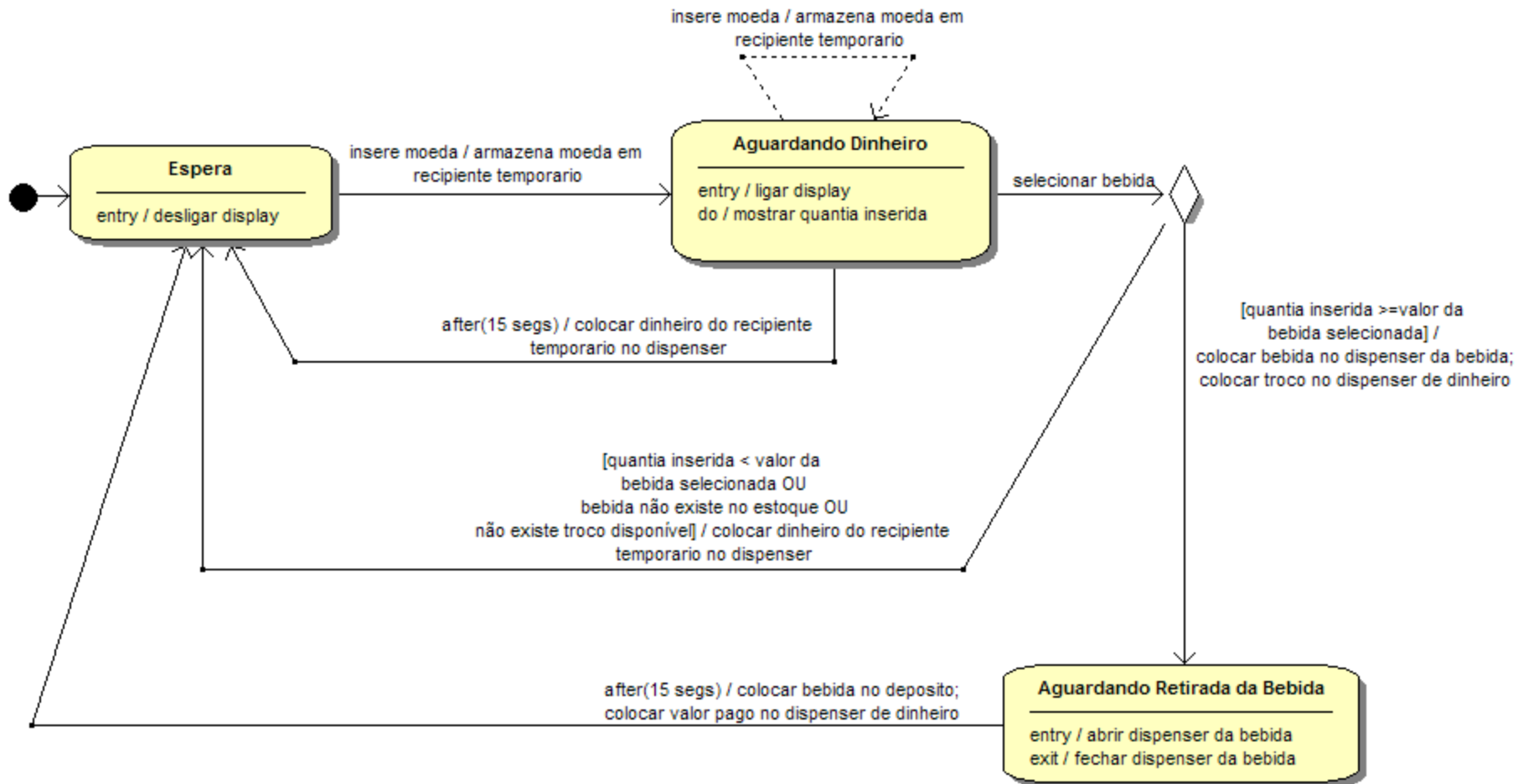


Diagrama de Pacotes

- A utilização do diagrama de pacotes no Modelo e Casos de uso tem como objetivo:
 - ✓ Definir uma representação concisa capaz de orientar a leitura de um modelo complexo.
 - ✓ Útil para a organização de grupos de trabalho em projetos extensos.
 - ✓ Permite dividir a complexidade do problema em unidades menores e auto-contidas.
 - ✓ Identificar sub-sistemas.
 - ✓ Organizar a documentação.

Diagrama de Pacotes

➤ Exemplo: vídeo locadora (1)

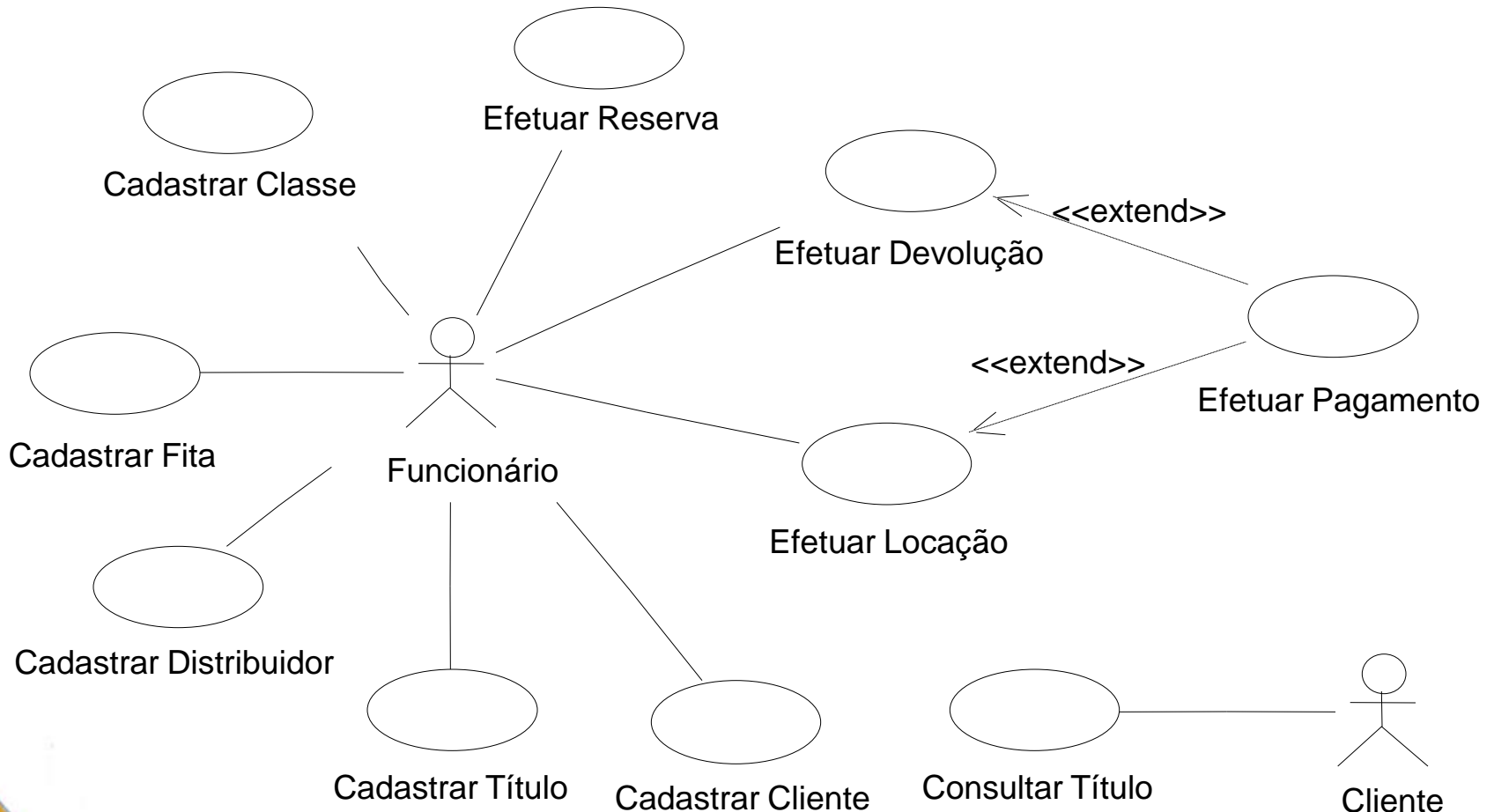


Diagrama de Pacotes

- Exemplo: vídeo locadora (2)
 - ✓ Identificando sub-sistemas com objetivos bem definidos.
 - ✓ A seta pontilhada denota uma relação de dependência.

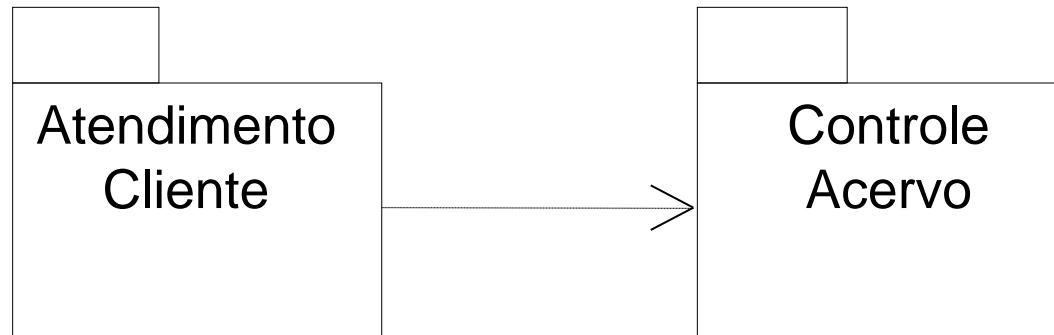
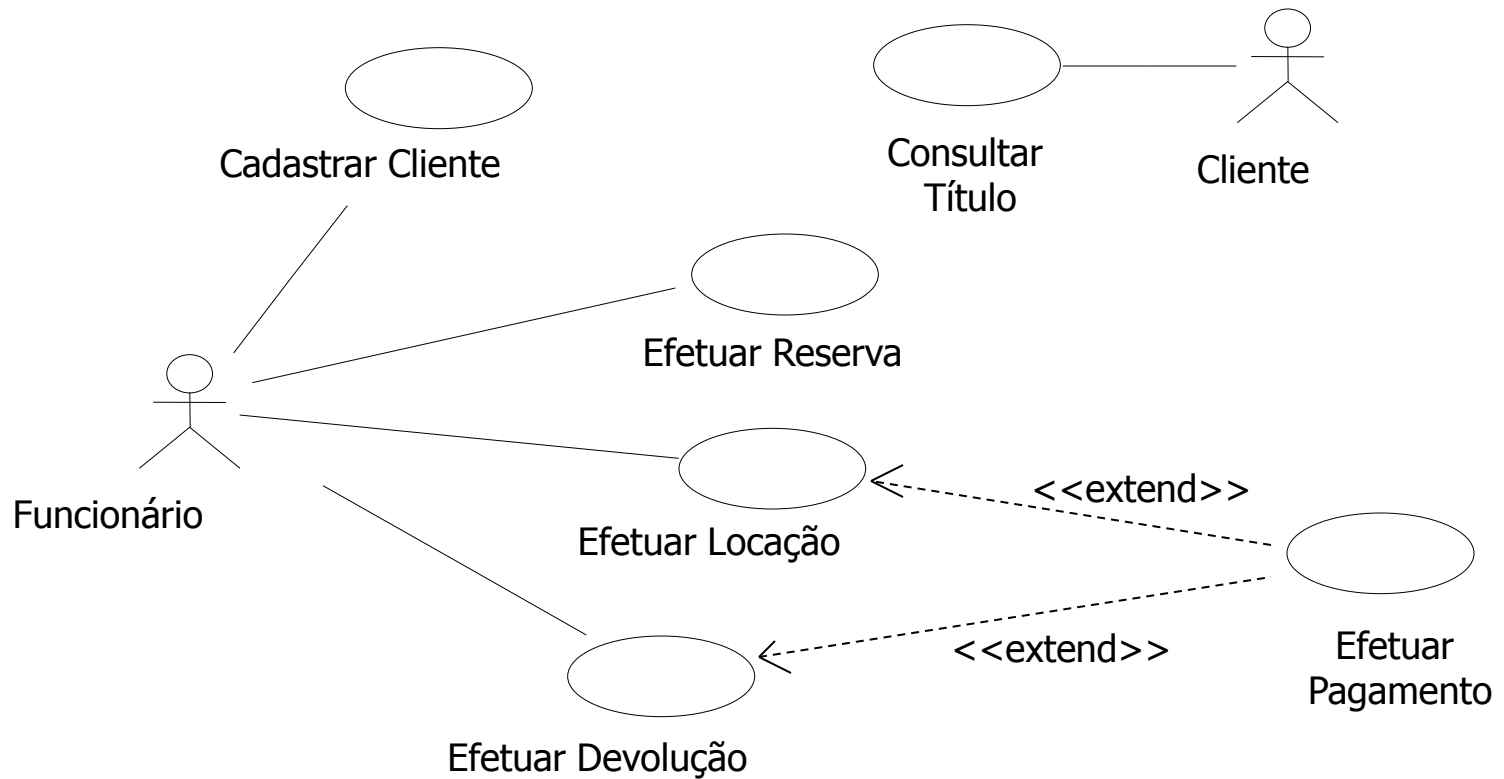


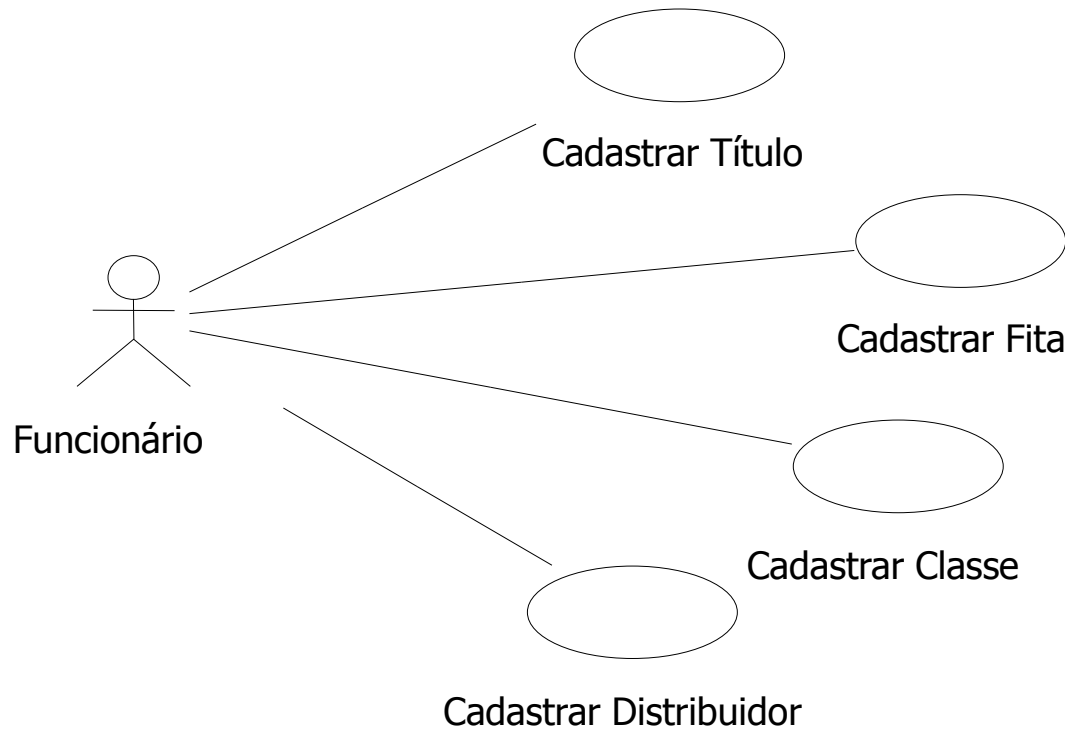
Diagrama de Pacotes

- Exemplo: vídeo locadora (3)
 - ✓ Sub-sistema de atendimento ao cliente.



Modelo de Casos de Uso - Pacotes

- Exemplo: vídeo locadora (4)
 - ✓ Sub-sistema de controle do acervo.



Referências

- Bezerra, E. "Princípios de Análise e Projeto de Sistemas com UML" Editora Campus, 2002.
- Booch, G. et al. "UML, guia do usuário" Campus, 2000.
- Cockburn, A., "Escrevendo Casos de Uso Eficazes", Editora Bookman, 2005.
- Falbo, R., "Modelagem de Objetos usando UML". Mini-Curso XVII Simpósio Brasileiro de Engenharia de Software (SBES), Manaus, 2003.
- Lima, A. S., "UML 2.0 Do Requisito à Solução", Editora Érica, 2005.
- Pender, T., "A Bíblia UML", Editora Elsevier, 2005.
- Pitone, D., Pitman, N. "UML 2.0 In A Nutshell", O'Reilly, 2005.