Universidade do Minho
Departamento de Informática

ARQUITECTURAS DE SOFTWARE

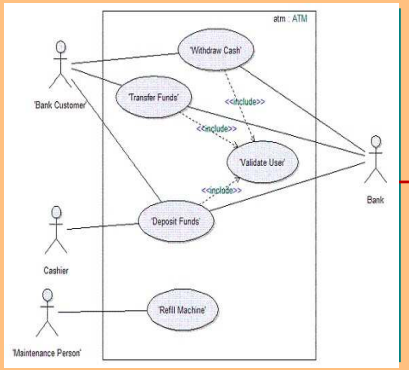
AULAS Nº 5, 6 e 7

16-23-30/11/2007

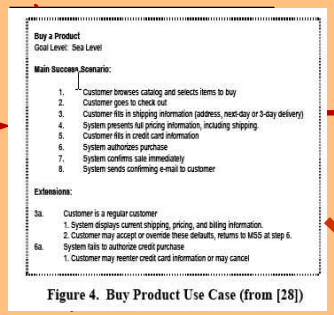
F. Mário Martins

Ligação das partes

Use Case Diagram



Use Case Specification



Buy a Product
Goal Level: Sea Level

Main Success Scenario:

1. Customer browses catalog and selects items to buy
2. Customer goes to check out
3. Customer fills in shipping information (address, next-day or 3-day delivery)
4. System presents full pricing information, including shipping.
5. Customer fills in credit card information
6. System authorizes purchase
7. System confirms sale immediately
8. System sends confirming e-mail to customer

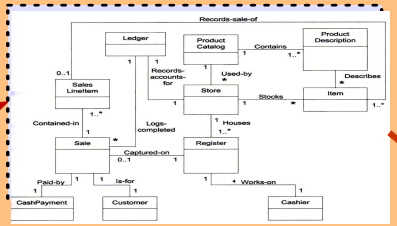
Extensions:

3a. Customer is a regular customer
1. System displays current shipping, pricing, and billing information.
2. Customer may accept or override these defaults, returns to MSS at step 6.

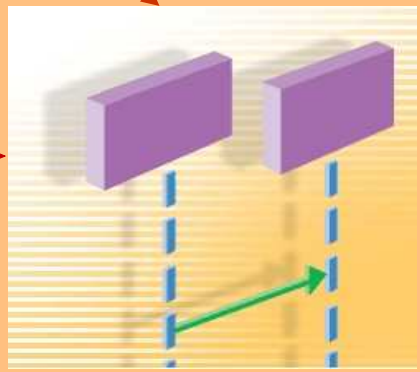
6a. System fails to authorize credit purchase
1. Customer may reenter credit card information or may cancel

Figure 4. Buy Product Use Case (from [28])

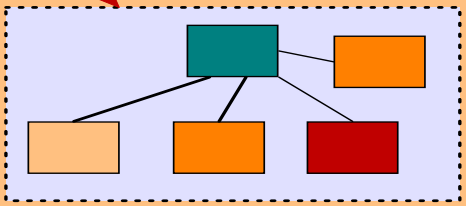
Domain Model

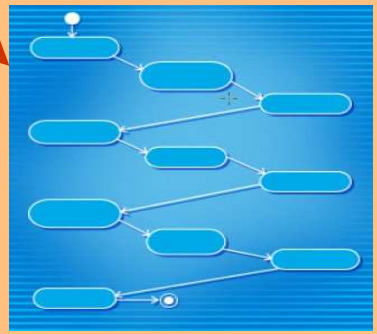


Dynamic Design Models



Static Design Model





A definição coerente de todos estes modelos corresponde à determinação das características funcionais do sistema

CASE STUDY

Estado da "Arte"

Tópicos (aulas 5, 6 e 7):

- **Consolidação** dos Use Cases e suas Realizações;
- **Diagramas de Classes** são bem conhecidos da OOD; Podem ser usados a qualquer momento, a qualquer nível;
- Vamos estudar e usar a técnica de **"responsability-based modelling"** mas modificando os **CRC-cards**;
- **CRC-cards** são interessantes como **Class-Responsability-Collaboration** cards, mas talvez sejam ainda melhores se usados a alto-nível (genéricos) para entidades a detalhar (que talvez venham a ser classes); Chamemos-lhes pois **SCRC-cards** (**"System Component-RC cards"**) para os distinguir;
- Vamos usar todos estes artefactos em 2 case studies.

Use Cases: Revisão

Abstracção : Programadores têm pouca paciência !!

Programador: Como é que eu especifico que numa máquina de café uma pessoa pode inserir só uma moeda, várias de 10 cent. ou de 20 c, etc. ?

Resposta: Especifique apenas que uma pessoa insere moedas.

- Use Cases devem descrever as acções que um actor deve realizar para obter os seus objectivos com o sistema e não que funções o sistema deve realizar para satisfazer os objectivos de um actor;
- Se o sistema conseguir a satisfação de tais objectivos, então o sistema foi realizado de forma a atingir o seu máximo valor de negócio.
- Porém, os objectivos nem sempre podem ser atingidos: falham !!

Use Cases: Revisão

Segundo A. Cockburn (#) bons use cases são:

- ☺ Textuais;
- ☺ Não têm referências à Interface com o Utilizador;
- ☺ Não têm formatos de dados;
- ☺ Possuem, em geral, 3 a 9 passos no cenário de sucesso;
- ☺ São fáceis de ler (depois de fixar vocabulário comum);
- ☺ Referem-se exclusivamente a "User´s Goals";
- ☺ São um registo das decisões sobre requisitos dos sistemas.

Nomes dos Objectivos => Para executivos

Cenário de Sucesso => Para todos

Extensões => Para analistas e programadores

Use Cases: Revisão

e ainda:

- Use Cases não são operações do sistema mas incluem operações do sistema;
- Jacobson diz que: cada use case constitui uma série completa de eventos iniciados por um actor e especifica a interacção que tem lugar entre o actor e o sistema. Um use case é, portanto, uma sequência especial de transacções relacionadas executadas por um actor e pelo sistema em diálogo.

Transacção:

(para o actor é atómica !)

I.- Actor primário envia pedidos e dados para o sistema;

II.- Sistema valida pedido e dados;

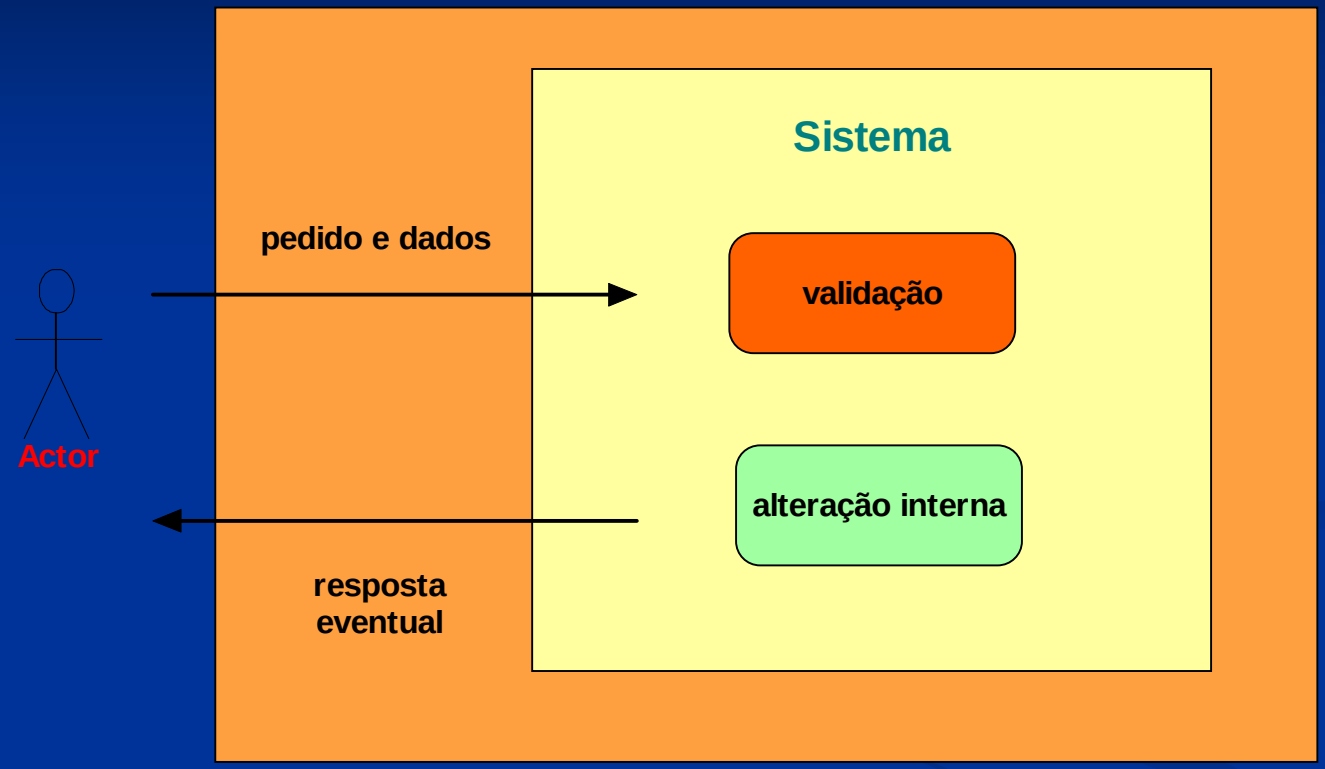
III.- Sistema altera ou não o seu estado interno;

IV.- Sistema responde ao actor com um resultado (ou não !!).

Use Cases: Revisão

Transacção:

(para o actor é atómica !)

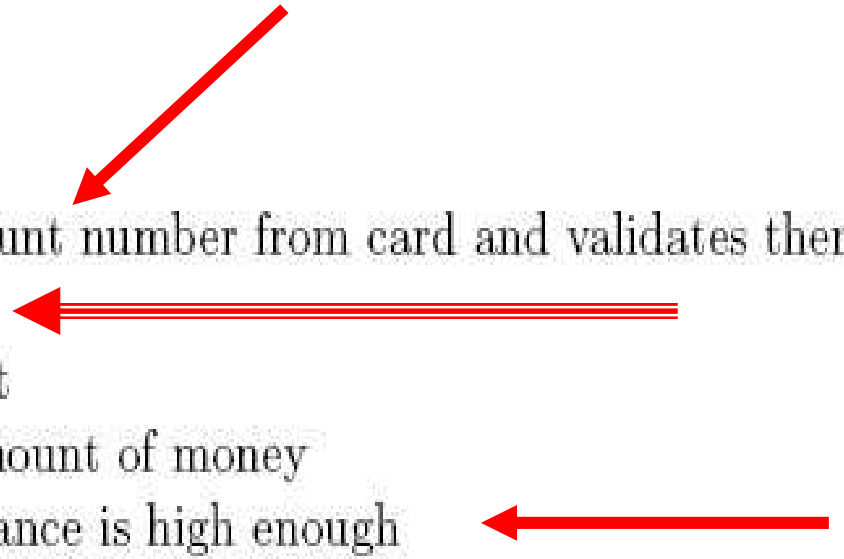


Use Cases: Revisão: Exemplo ATM

LEVANTAMENTO DE UMA ATM: Cenário de sucesso

Use case: Withdraw Money
Main Success Scenario:

1. - user identifies himself by a card
2. system reads the bank ID and account number from card and validates them
3. - user authenticates by PIN
4. system validates that PIN is correct
5. - user requests withdrawal of an amount of money
6. system checks that the account balance is high enough
7. system subtracts the requested amount of money from account balance
8. system returns card and dispenses cash



Onde pode a obtenção deste objectivo falhar ?

Use Cases: Revisão: Exemplo ATM

LEVANTAMENTO DE UMA ATM: Cenário de sucesso (transacções)

Use case: Withdraw Money
Main Success Scenario:

TRANSACÇÕES

1. - user identifies himself by a card
2. system reads the bank ID and account number from card and validates then
3. - user authenticates by PIN
4. system validates that PIN is correct
5. - user requests withdrawal of an amount of money
6. system checks that the account balance is high enough
7. system subtracts the requested amount of money from account balance
8. system returns card and dispenses cash

Algumas são do tipo Acção -> Reacção (ie. possuem apenas 2 passos)

Use Cases: Revisão: Exemplo ATM

LEVANTAMENTO DE UMA ATM: Erros e tratamentos

Extensions:

2-8a. Wrong type of card or destroyed card:
1. System gives back the card

4a. Wrong pin less than 3 times:
1. System updates number of tries
2. start from action step 3

4-8a. Wrong pin 3 times:
1. System keeps the card

6-8a. Not enough money on account:
1. account does not contain enough money
2. system does not change the account
3. system returns card

substituição

Use Cases: Revisão: Exemplo ATM

LEVANTAMENTO DE UMA ATM: Pós-condição do Use Case

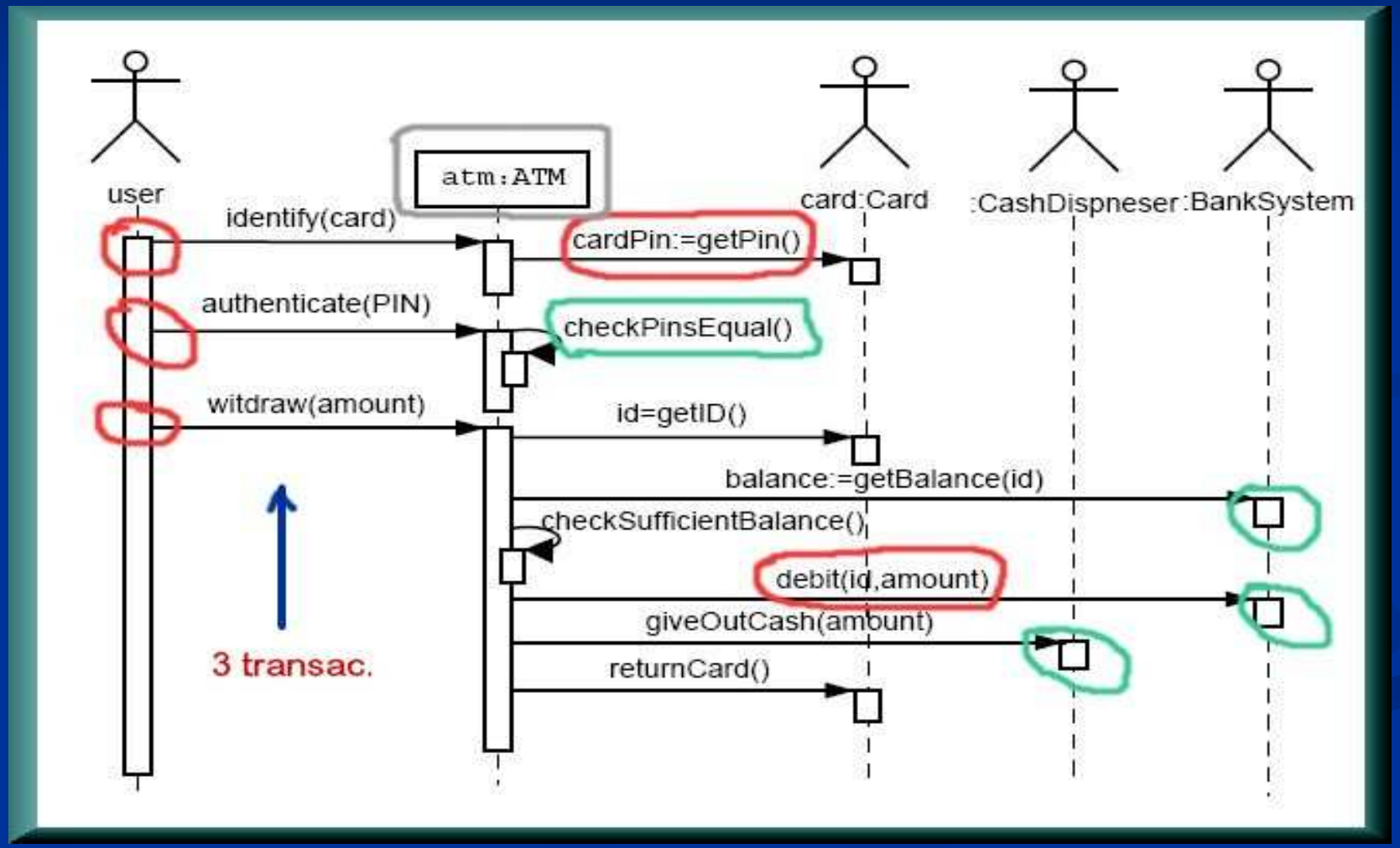
Postcondition:

- If the customer entered the PIN stored on the Card, and the customer's balance was greater than or equal to the requested amount, then the customer got the requested amount and the amount was deducted from the balance.
- If the customer entered the wrong PIN three times, the card was retained.
- If the customer requested too much money, the card was returned to the customer.

Todos os estados finais possíveis, mas nem todos de sucesso !!

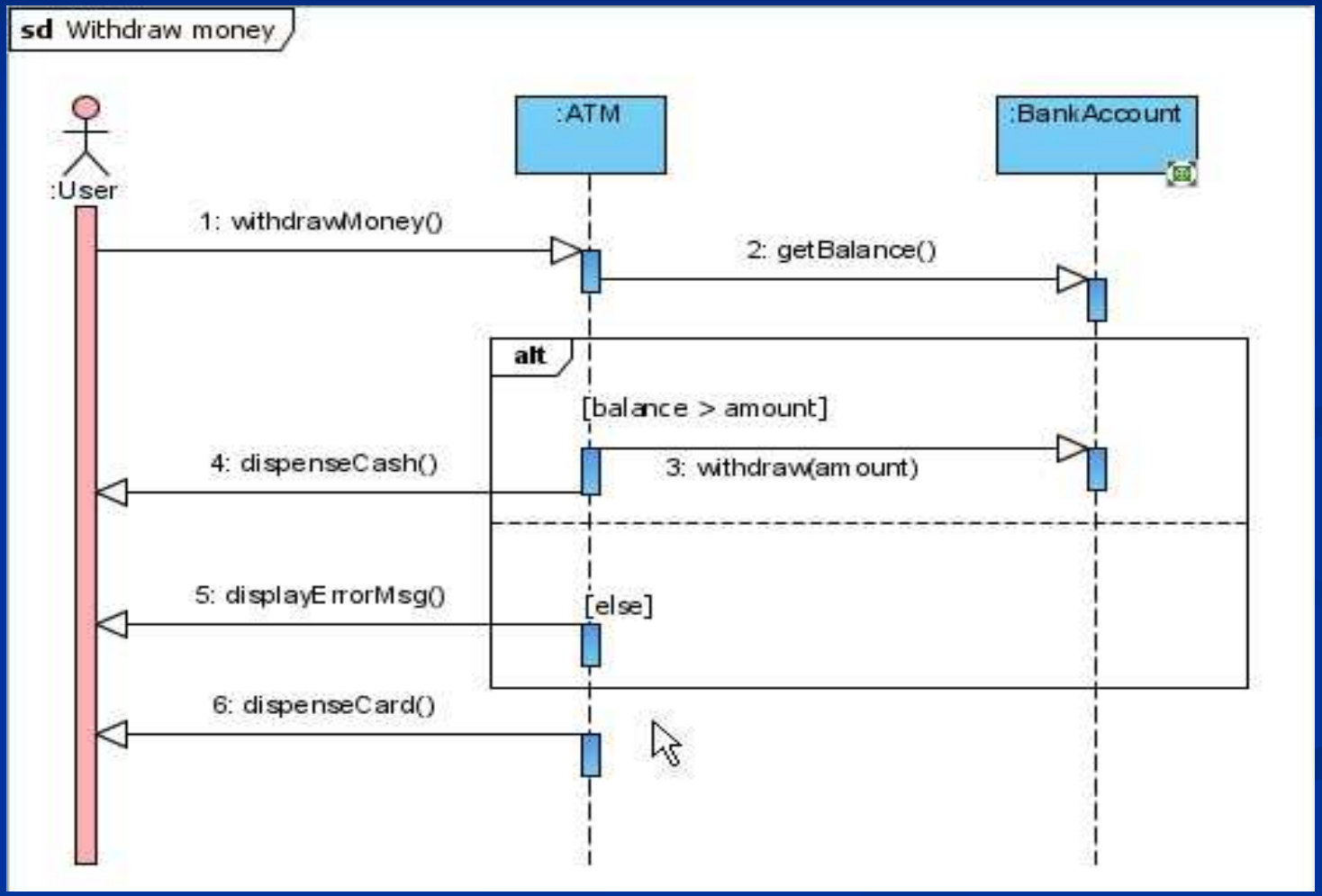
Use Cases: Revisão: Exemplo ATM

LEVANTAMENTO DE UMA ATM: Realização do Use Case (sucesso)



Use Cases: Revisão: Exemplo ATM

LEVANTAMENTO DE UMA ATM (ex. de especificação incompleta!!)



Use Cases: Revisão: Exemplo ATM

ACTIVIDADE DA ATM

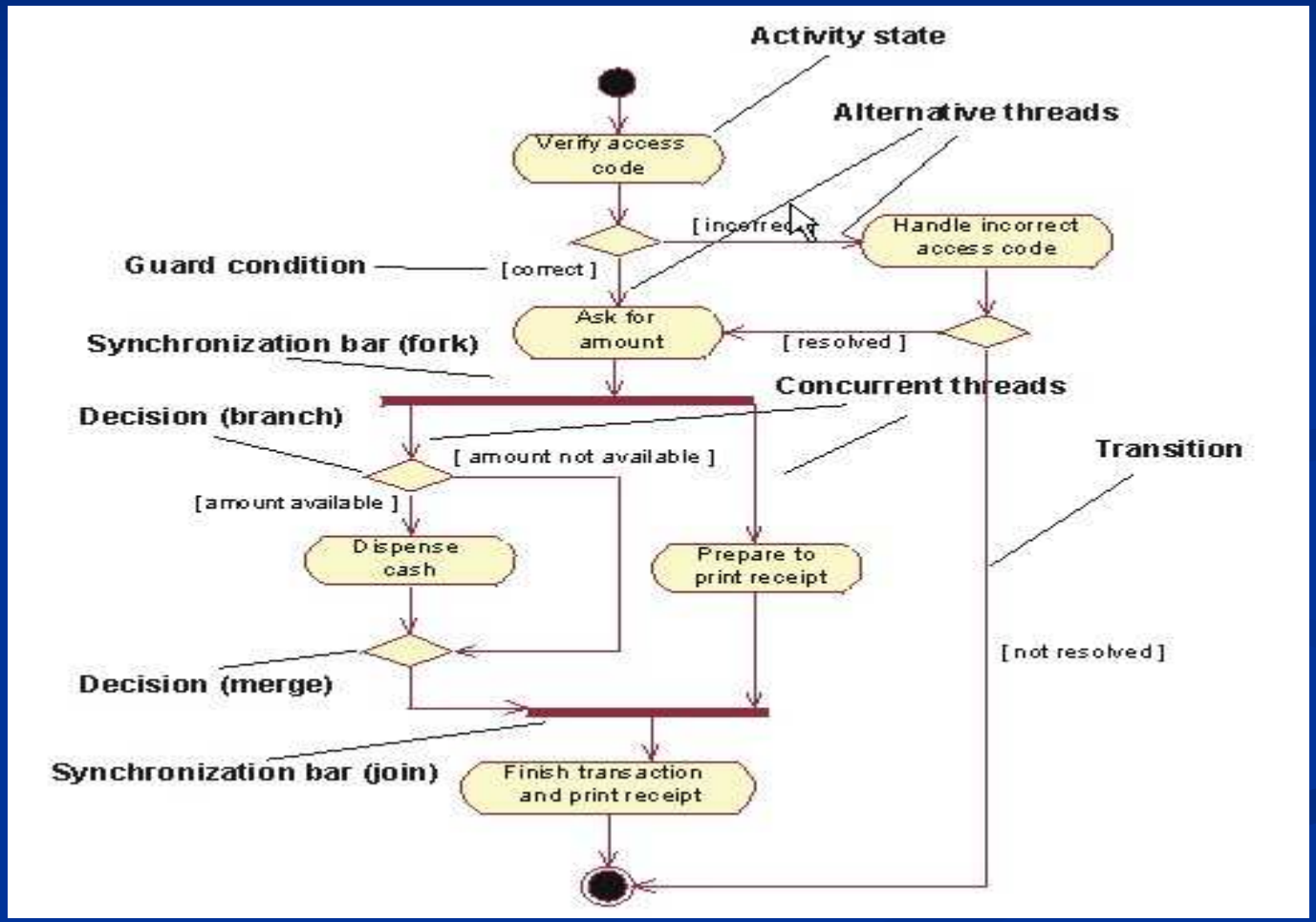
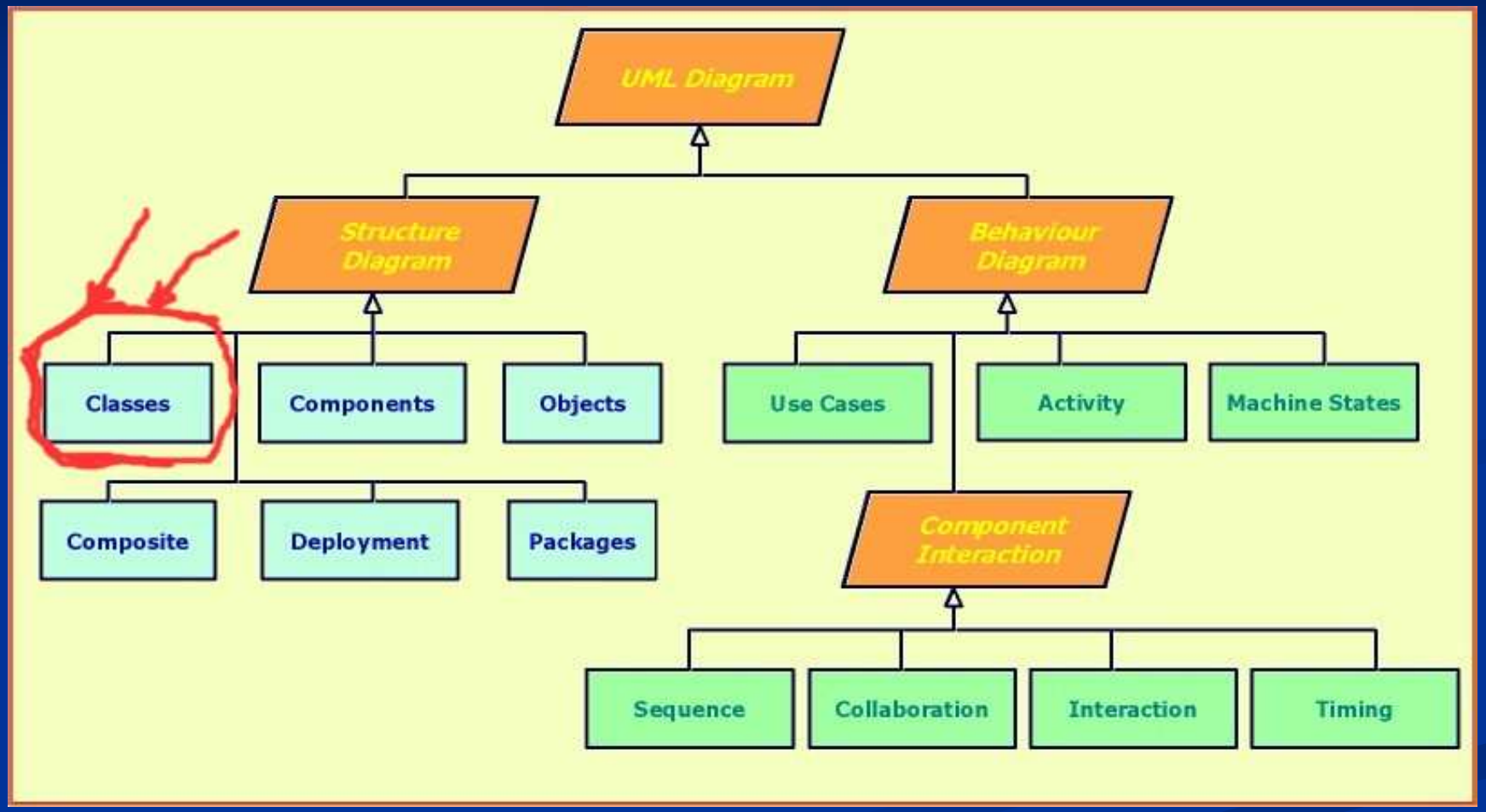


Diagrama de Classes (I)



O nosso 1º diagrama UML de modelção estrutural !

Diagrama de Classes (I)

Objectivo: Modelar a estrutura estática do sistema, em especial os **tipos** das entidades e seus **relacionamentos semânticos**, bem como a sua **estrutura interna** (se aplicável e necessário).

Em UML, os tipos das entidades do sistema estruturam-se em 4 categorias designadas classificadores (**"classifiers"**):

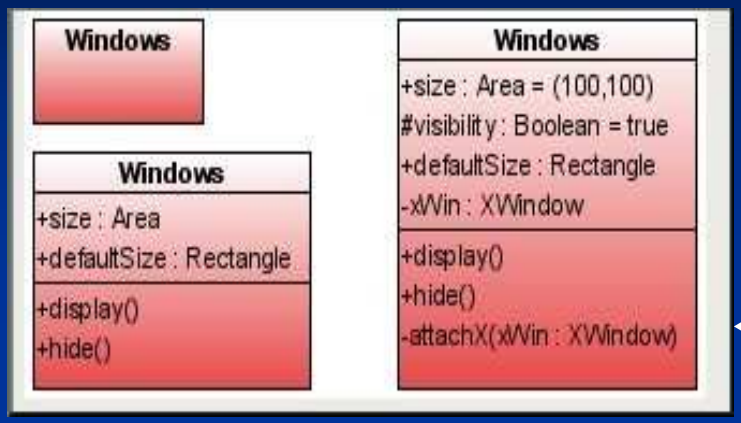
1) **Classe**

2) **Interface**

3) **Tipo de dados**

4) **Componente**.

Diagrama de Classes (I)



Classes podem ser especificadas com vários níveis de detalhe

Métodos, Atributos e suas visibilidades

+	public
#	protected
-	private
~	package

com os significados usuais

ver também em http://www.visual-paradigm.com/VPGallery/diagrams/Class.html

Diagrama de Classes (I)

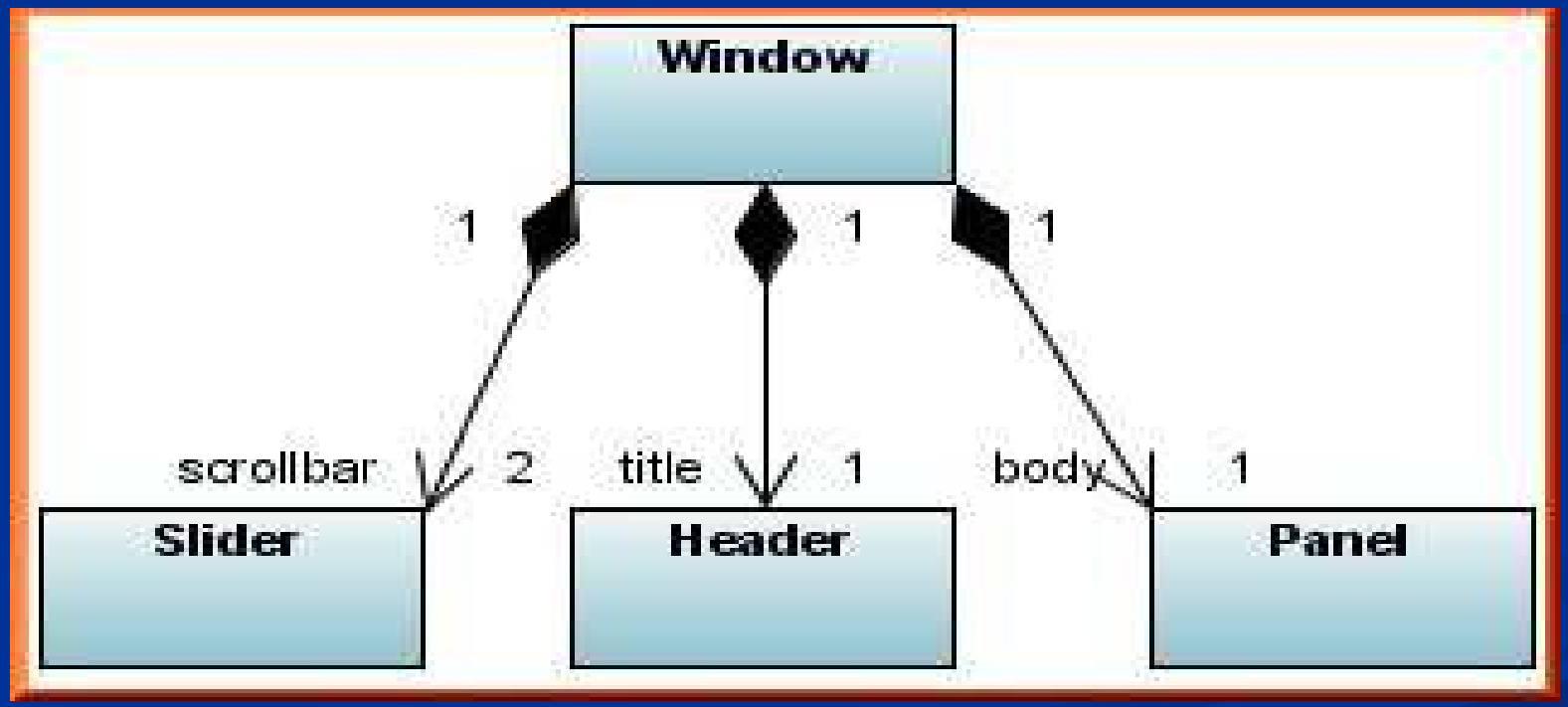
Relação Semântica: Agregação básica
(com multiplicidade e nome)

Relação semântica conhecida como

has ou has_part

Diagrama de Classes (I)

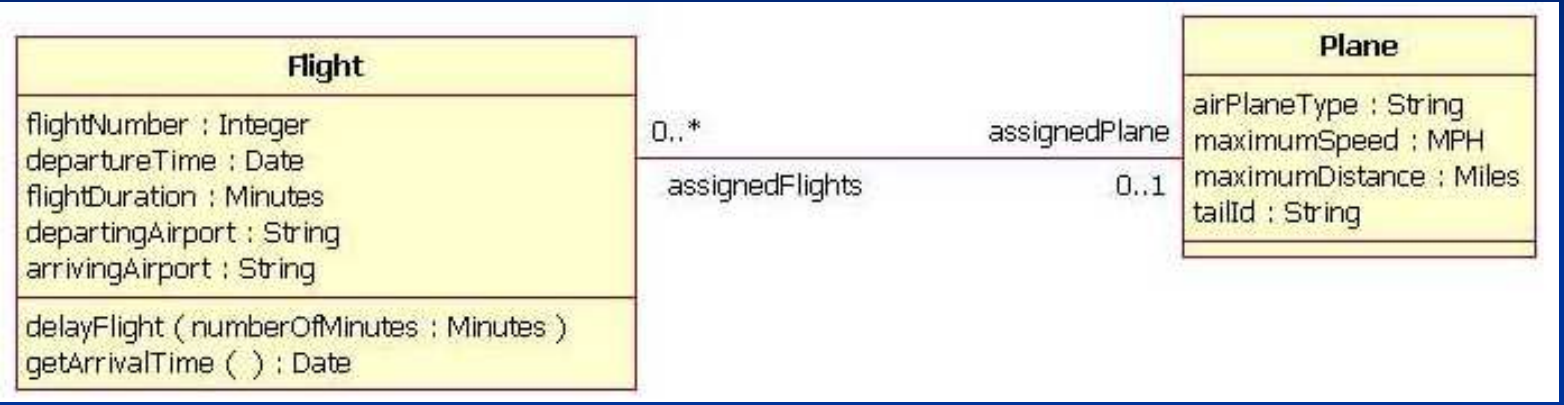
Relação Semântica: Agregação por Composição
(com multiplicidade e nome)



Esta associação é muito forte pois as partes e o todo estão dependentes até mesmo no seu "tempo de vida".

Diagrama de Classes (I)

Associações uni e bi-direccionais



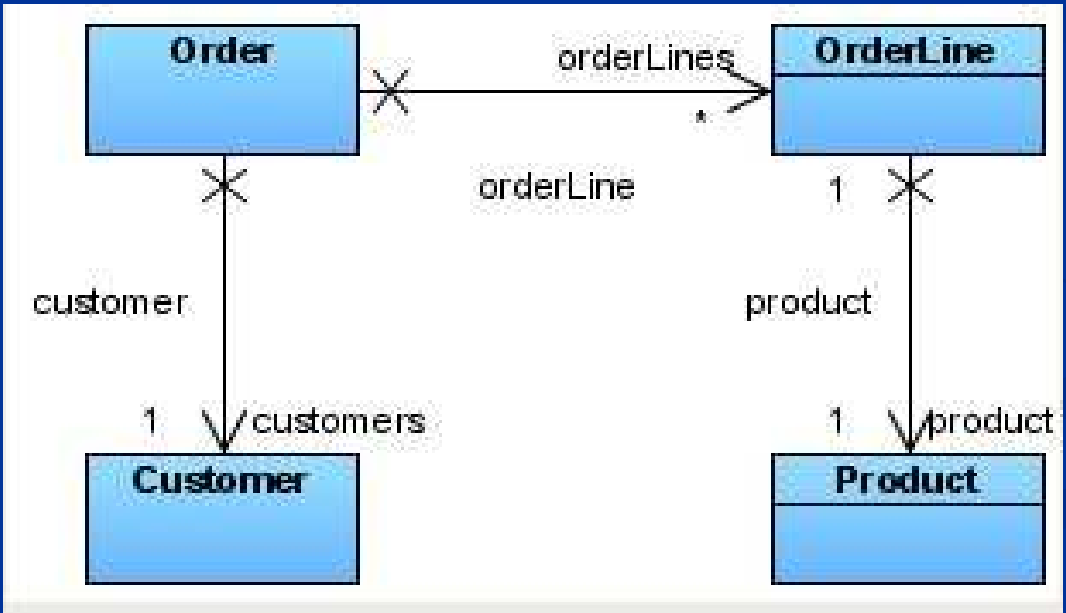
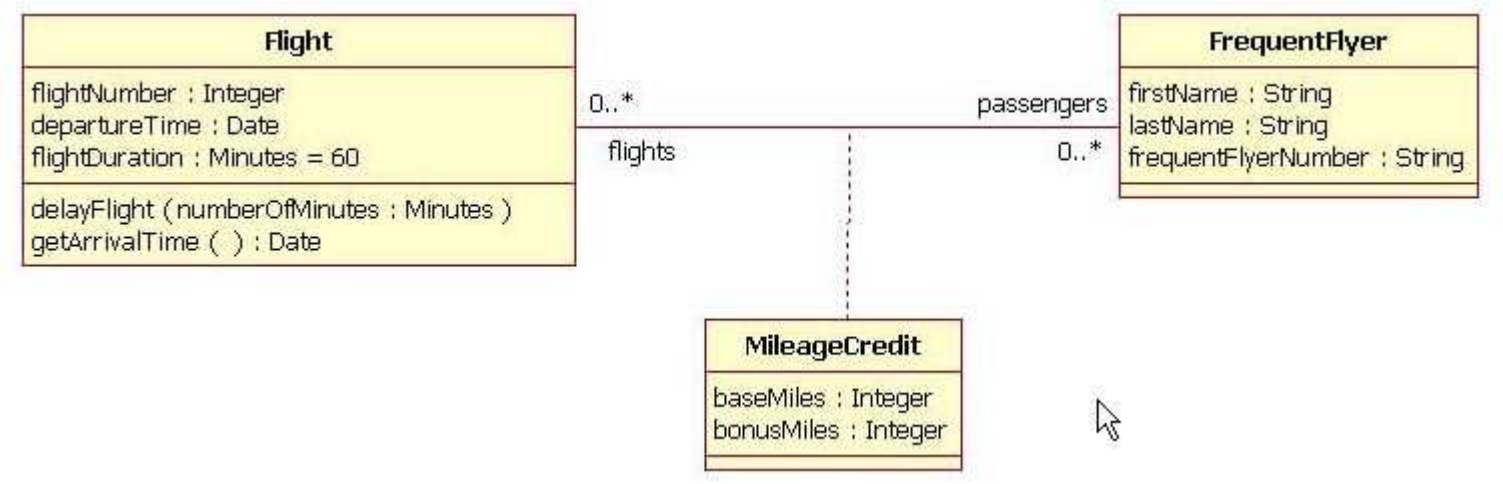


Diagrama de Classes (I)

Classe de Associação (informação adicional - link)



Quando se relaciona uma instância de "Voo" com uma instância de "Passageiro Frequente" deve "juntar-se" uma instância de "Crédito em Milhas".

Diagrama de Classes (I)

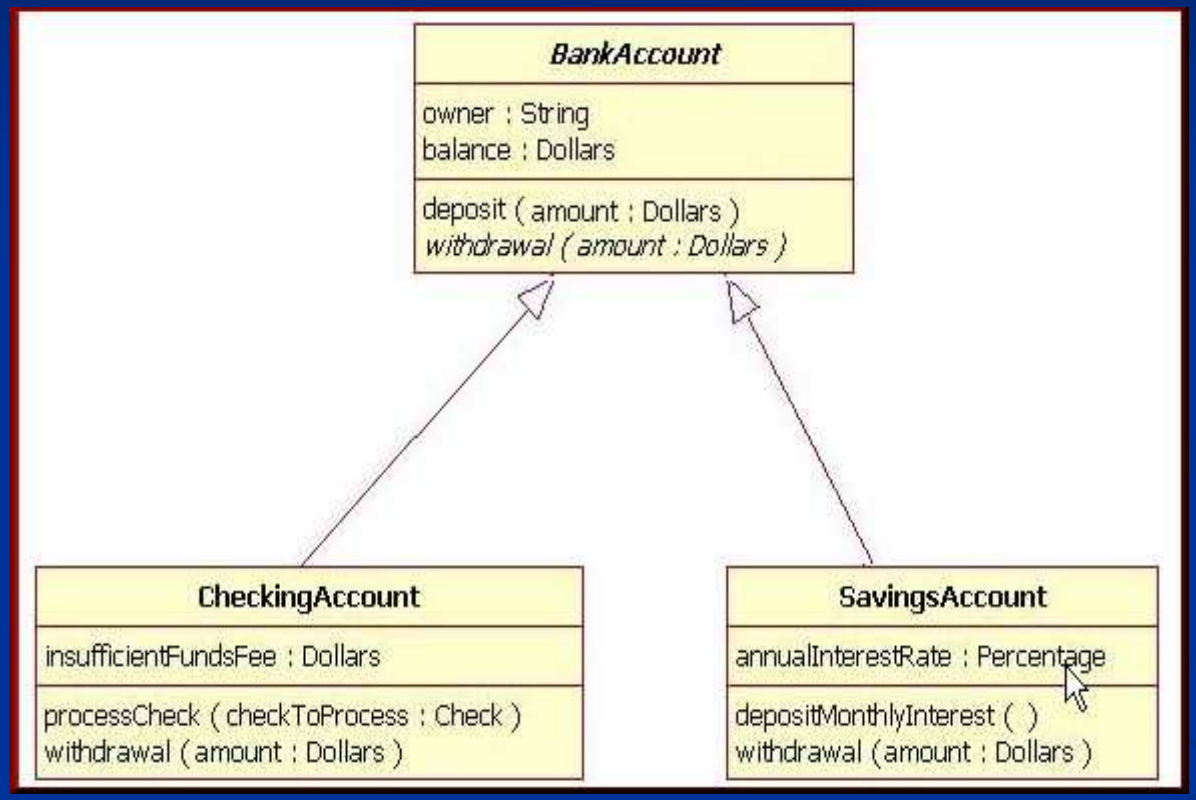
Associações com distinção de Papéis ("roles")

Permitem especificar o "papel" de uma dada classe numa relação semântica, mesmo que seja a própria classe a relacionar-se consigo mesma.

A prática o dirá, mas parece informação redundante. Mas redundância nem sempre é má, pois pode equivaler a maior segurança.

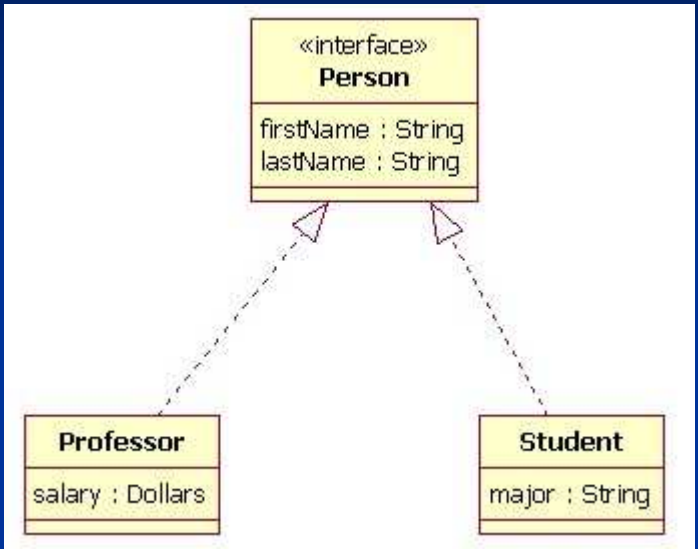
Diagrama de Classes (I)

Generalização - Herança



Nota: A superclasse pode ser concreta ou abstracta.

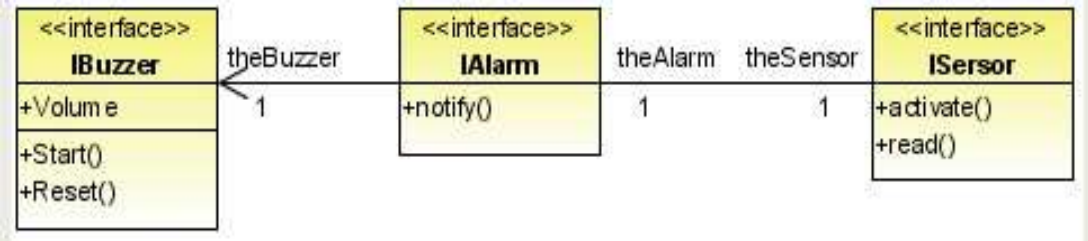
Diagrama de Classes (I)



Relação: implements

Classes implementam interfaces (ou seja, satisfazem as propriedades do tipo especificado); O que é um tipo ?

Mau exemplo ! Porquê?



Bom exemplo mas complexo! Porquê?

Diagrama de Classes (I)

Class Packages como NameSpaces ou Subsystems

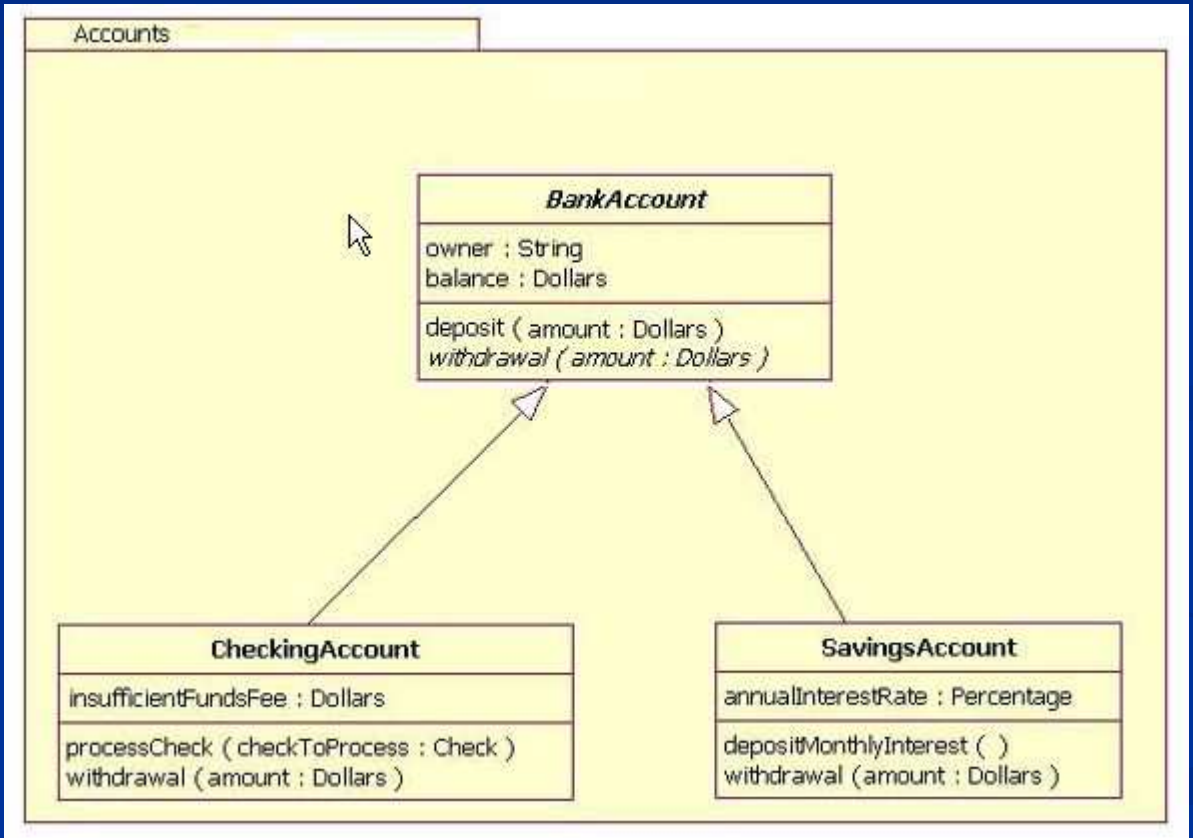


Diagrama de Classes (I)

Packages (UML 2.0)

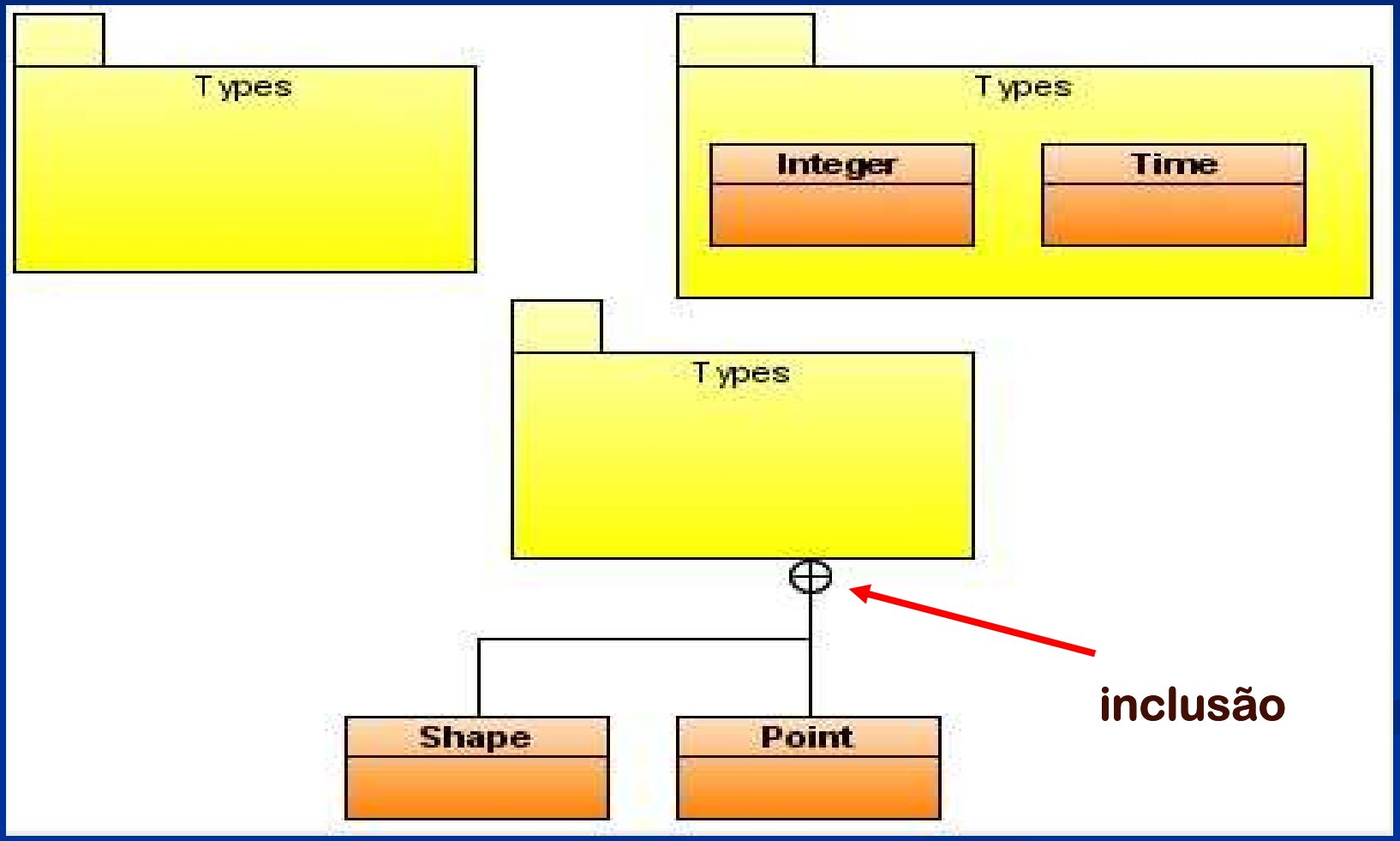


Diagrama de Classes (I)

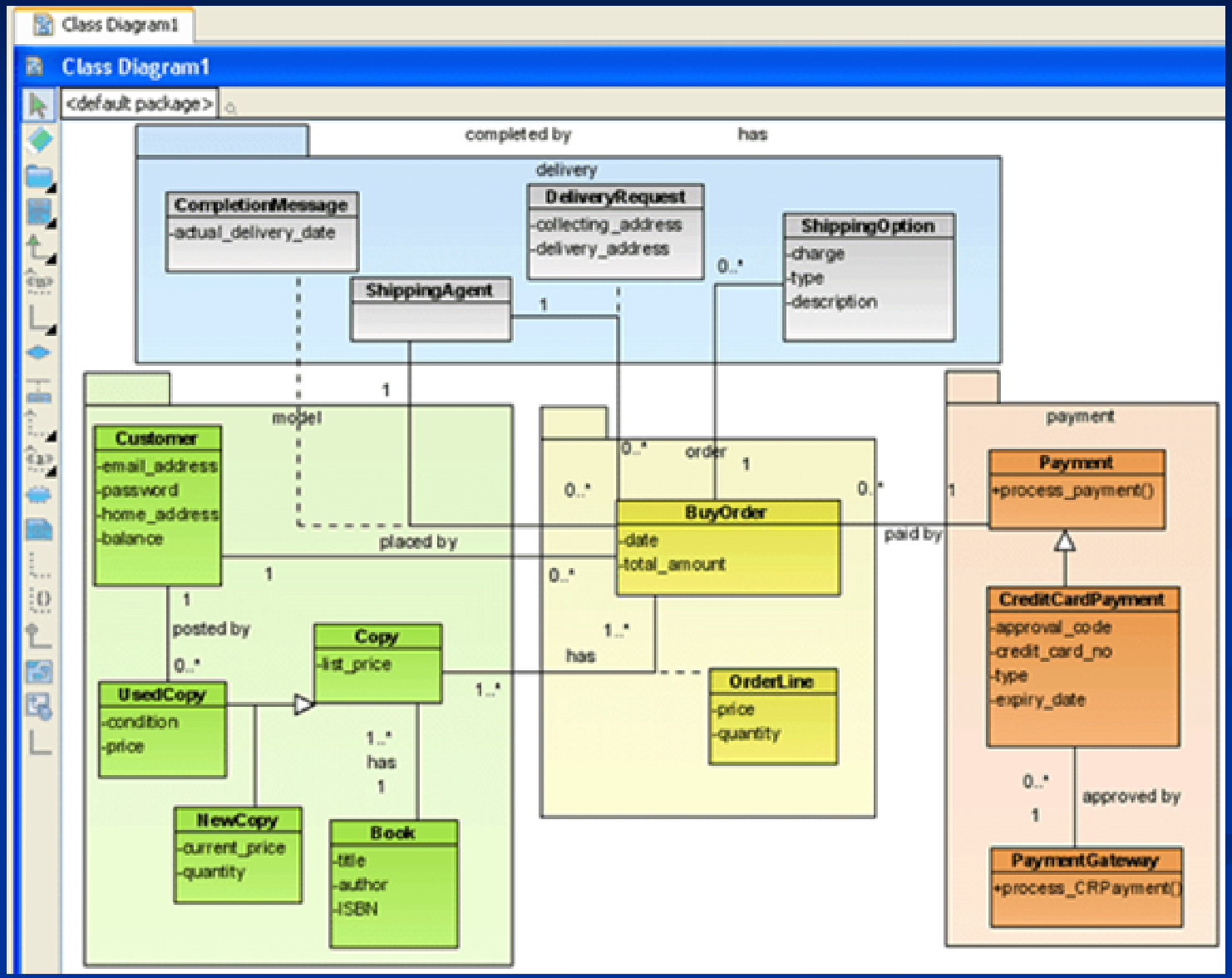


Diagrama de Classes (I)

- Outras construções UML 2.0 a ver posteriormente (porque são orientadas à fase de concepção):
  - Representação de instâncias de classes;
  - Relacionamentos entre instâncias (não devem ser livres !);
  - Estruturas internas de classes ("internal structures").

Outros modelos de domínio

Hospital Billing System

ERD

(continuação)

Responsability-based modelling

A análise da "responsabilidade" (o que deve fazer; o que deve "saber") de cada entidade no funcionamento global de um sistema, permite identificar subsistemas e suas formas de colaboração.

Usaremos SCRC-cards (a introduzir).

Preparação (leituras):

"Object Design – Roles, Responsabilities and Collaborations", Wirfs-Brock & McKean, 2003.

"Reflections on CRC-cards and OO Design", R. Biddle et al., 2002

Case Study 1

PROJECTO 1:

Concepção e programação de uma máquina automática de café de preço único (30 c) mas alterável. Apenas aceita moedas. Tem selecção de café com ou sem açúcar. Tem botão para anular operação. Tem depósito para dar troco.

Questão A) Que subsistemas?

Questão B) Que interacções/colaborações entre subsistemas ?

Questão C) Impacto de modificações de requisitos na concepção;