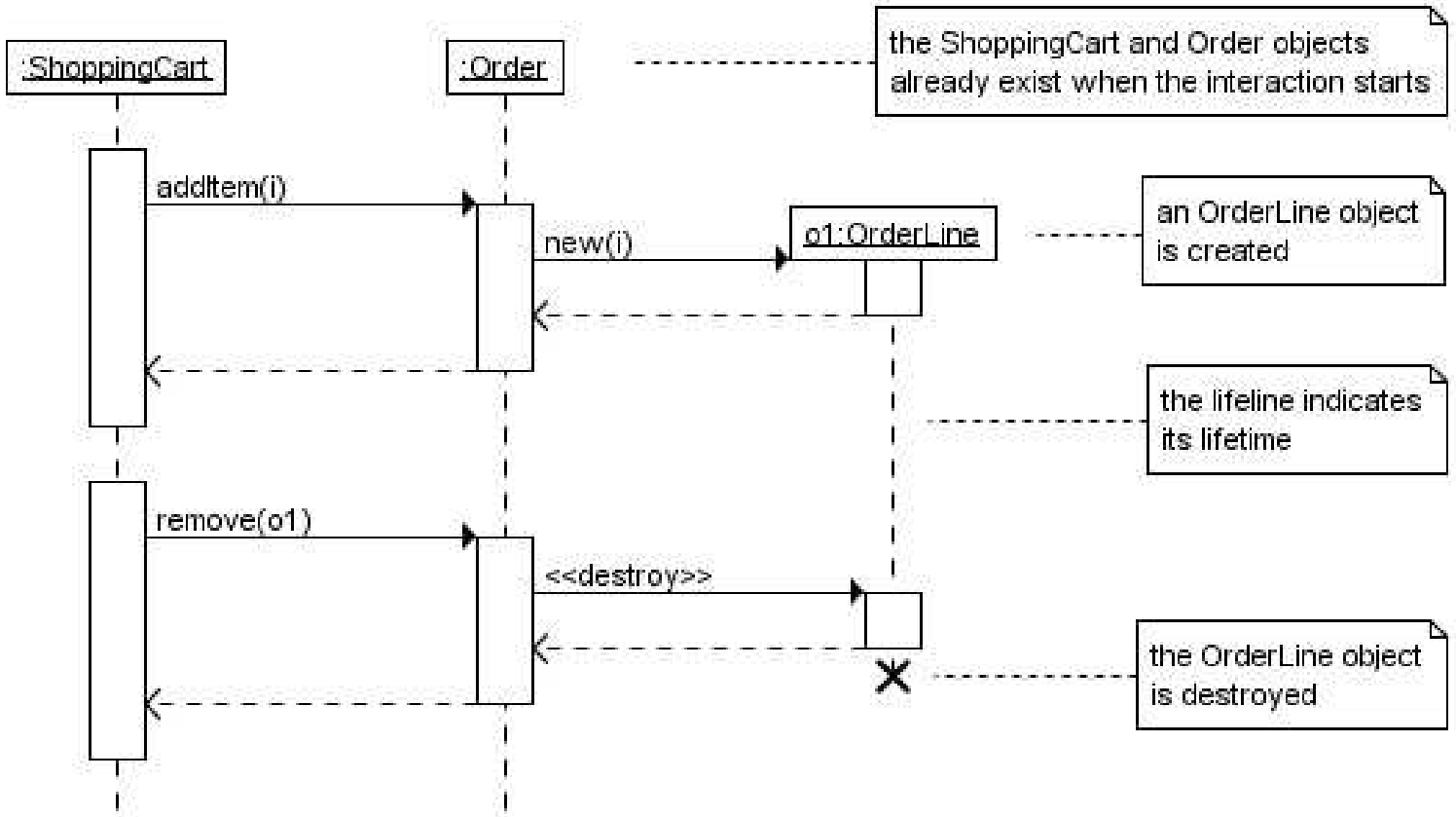
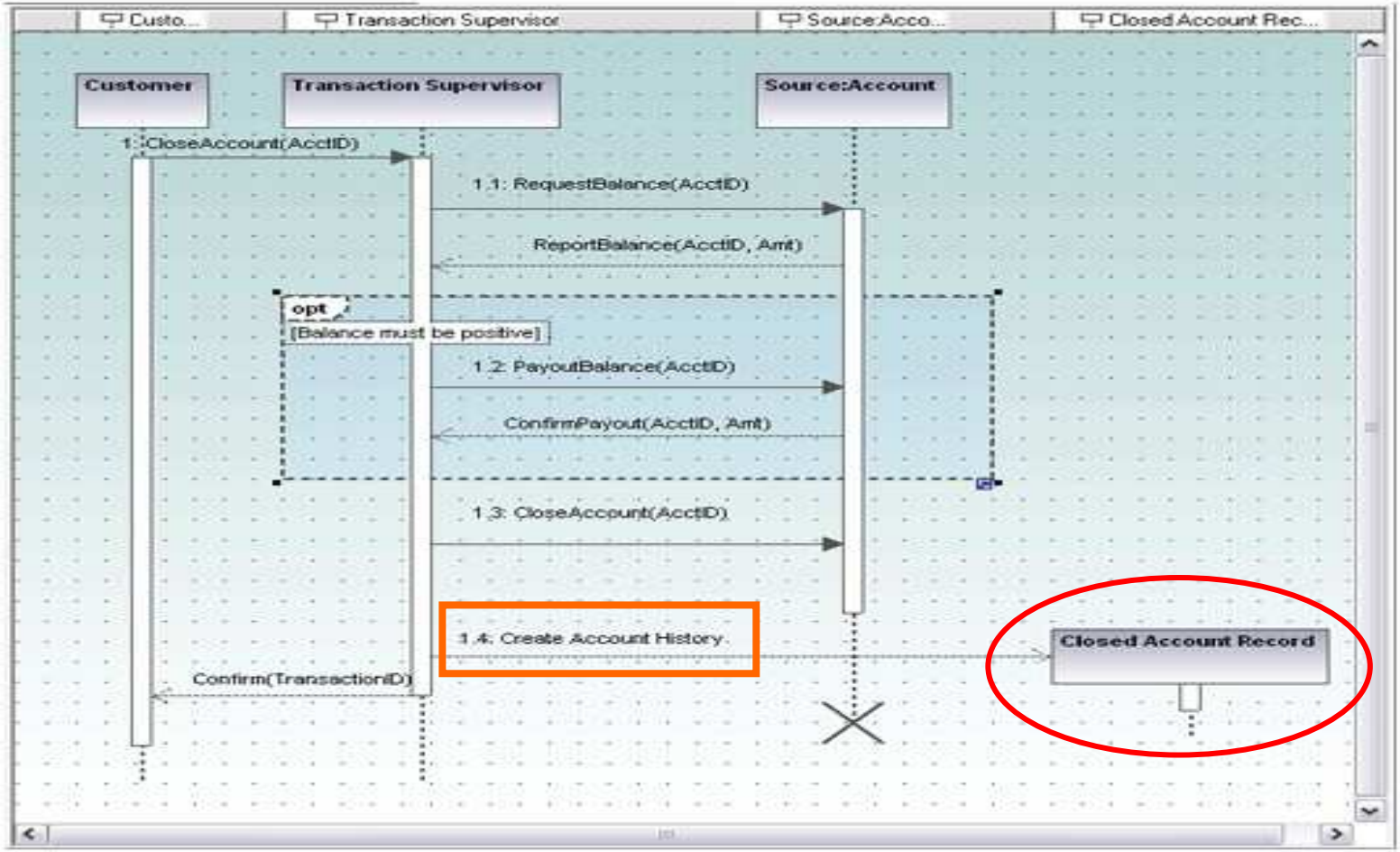
DS: notação

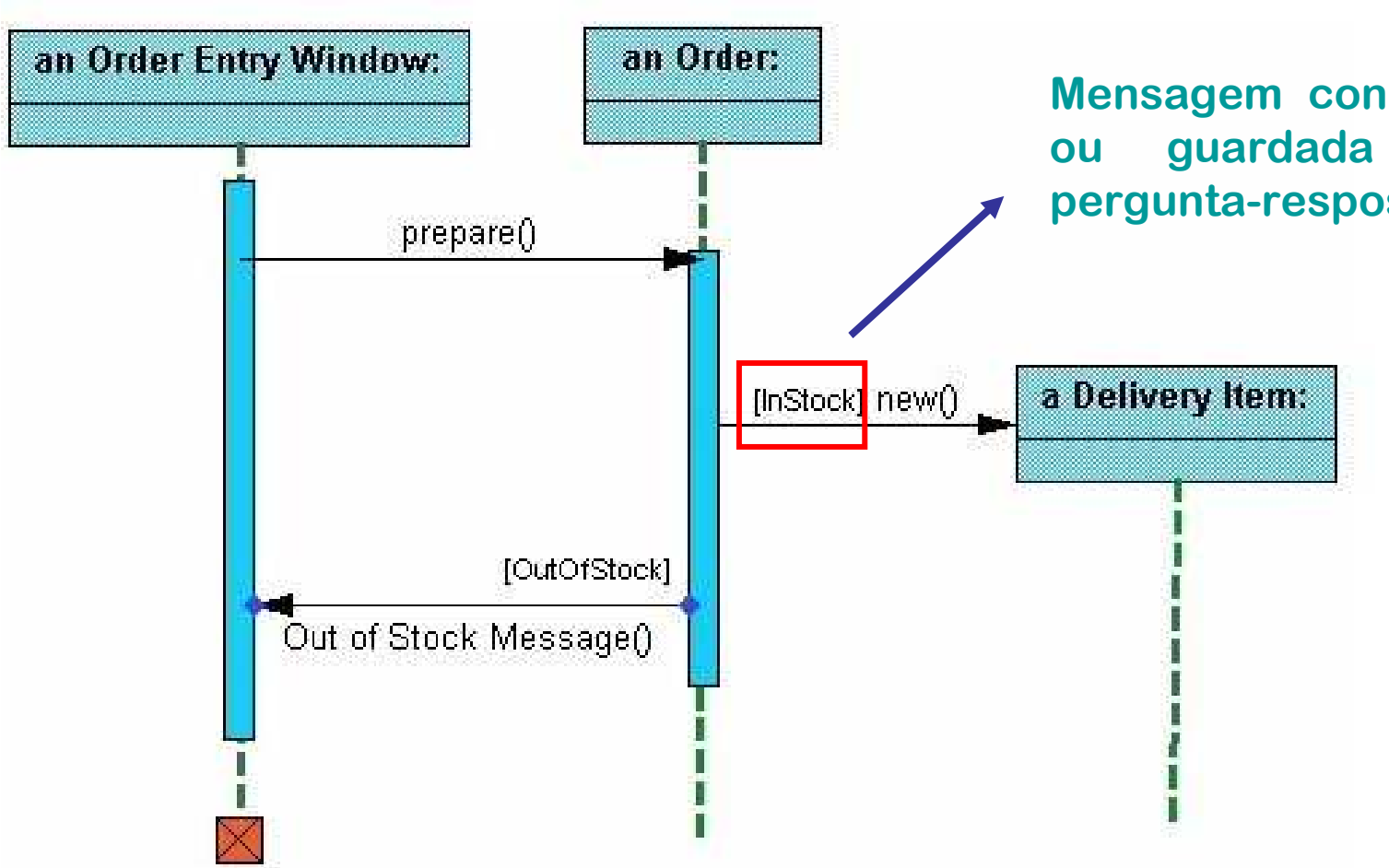
Falta-nos apenas dar exemplos de DSS que contenham a criação de objectos temporários e sua posterior destruição.



DS: notação


Custo...
Transaction Supervisor
Source:Acco...
Closed Account Rec...
Customer
Transaction Supervisor
Source:Account
1: CloseAccount(AcctID)
1.1: RequestBalance(AcctID)
ReportBalance(AcctID, Amt)
opt
[Balance must be positive]
1.2: PayoutBalance(AcctID)
ConfirmPayout(AcctID, Amt)
1.3: CloseAccount(AcctID)
1.4: Create Account History
Closed Account Record
Confirm(TransactionID)

DS: notação



Mensagem condicional ou guardada (evita pergunta-resposta)

DS: notação

- Em geral, para a criação de objectos é usado na mensagem o estereótipo **<<create>>** ou a designação **new()** ou **new(parâmetros)**;

- Em geral, para a destruição ou se usa o estereótipo **<<destroy>>** ou se assinala o fim do ciclo de vida do objecto com um **X**.

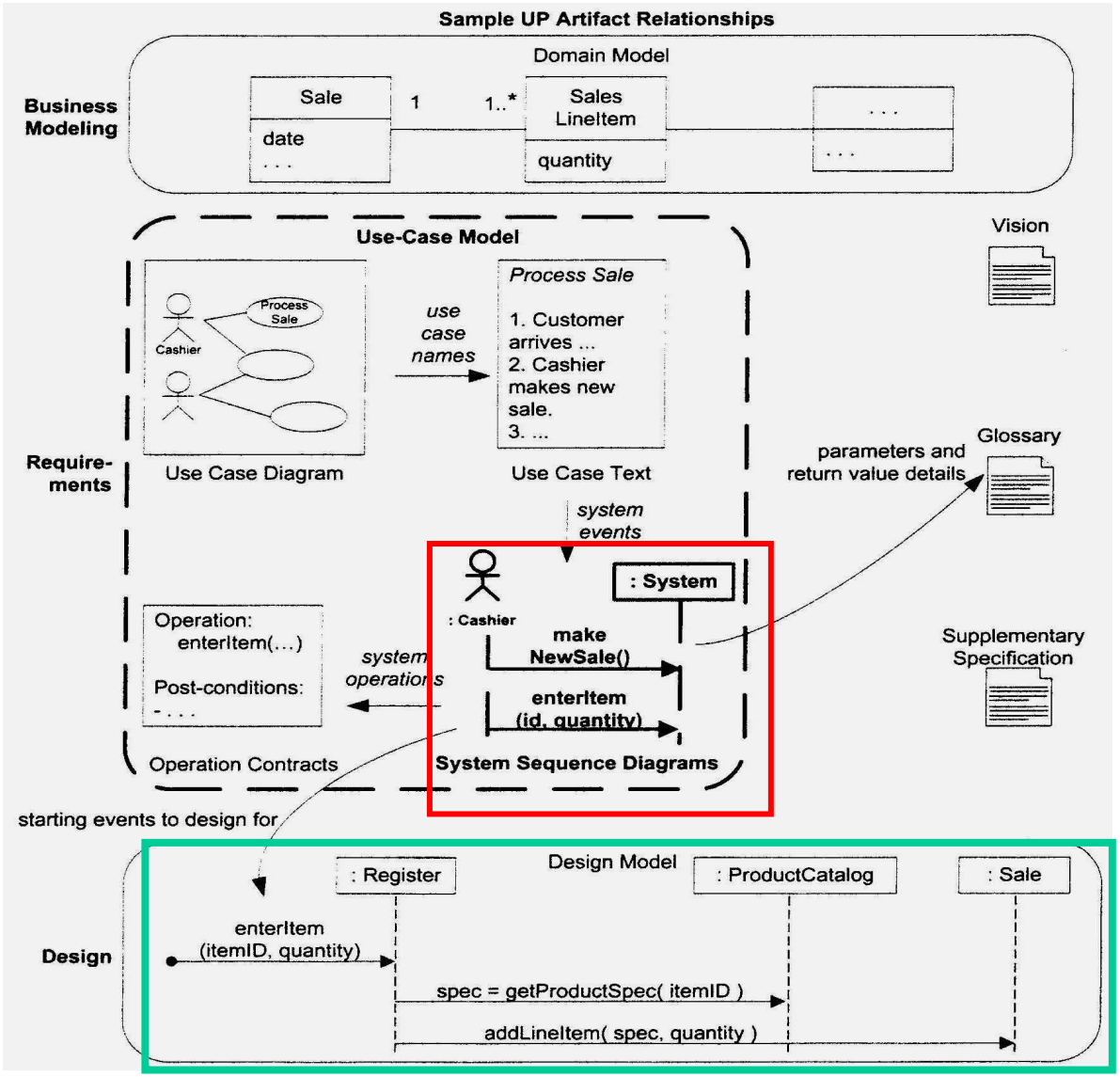
- Estes objectos são portanto objectos auxiliares, que são criados durante a realização da tarefa, e que podem ou não continuar a existir. Por exemplo, o objecto pode ser criado e posteriormente ser guardado num sistema persistente de dados.

DS e Fases do Projecto

Tal como a tabela seguinte procura mostrar, os Diagramas de Sequência (DS) podem e devem ser usados em diferentes fases do processo, desde os requisitos ao desenvolvimento.

Fase	Aplicação
Análise	Na fase de análise e captura de requisitos, os Diagramas de Sequência são usados para modelar as interações entre as instâncias de certas classes, ou seja, o comportamento necessário à realização dos UC. Tipicamente, um DS ilustra o fluxo principal de eventos do UC, e diagramas adicionais modelam alternativas importantes e excepções. As instâncias podem manter os estereótipos <<boundary>>, <<control>> ou <<entity>> das suas classes, assim tornando claro que são classes da fase de análise. Os DS, são excelentes auxiliares na identificação das classes que são necessárias a um sistema, e o que os respectivos objectos devem ser capazes de fazer nas interacções com outros.
Concepção lógica	Na fase de concepção, os DS devem ser refinados de forma a modelarem como o sistema é capaz de completar de facto as interações. Por exemplo, todas as mensagens dos DS devem ser associadas a métodos concretos definidos nas classes. Algumas decisões de arquitectura podem ser definidas nesta fase, por exemplo, o número de camadas do sistema, formas de persistência dos dados, necessidade de "web services", etc. Assim, na fase de concepção os DS servem para explicar como o sistema vai funcionar para responder às interacções especificadas.
Desenvolvimento da Arquitectura	Outros diagramas UML existem mais adequados para definir a arquitectura do sistema. No entanto, certos componentes podem necessitar de ter o seu comportamento especificado com DS.

DS: Refinamento

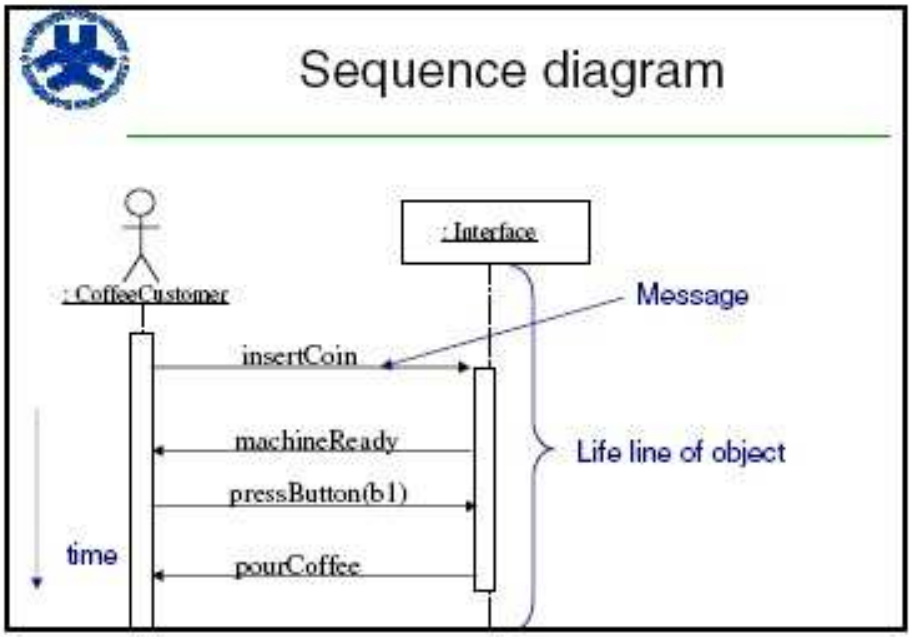


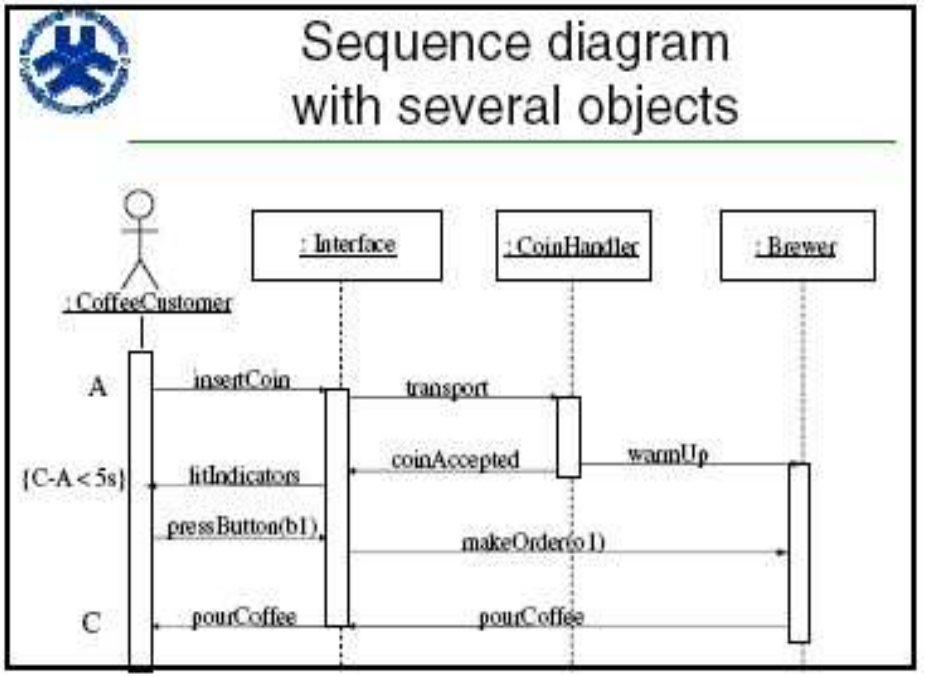
Os DS a nível de Sistema (DSS) são o ponto de partida para os DS de concepção.

Porém, temos que saber dividir o objecto :System nos subsistemas, ou seja, nas entidades funcionais e estruturais que irão fornecer a funcionalidade especificada e pretendida.

DS: Níveis – Exemplo 1

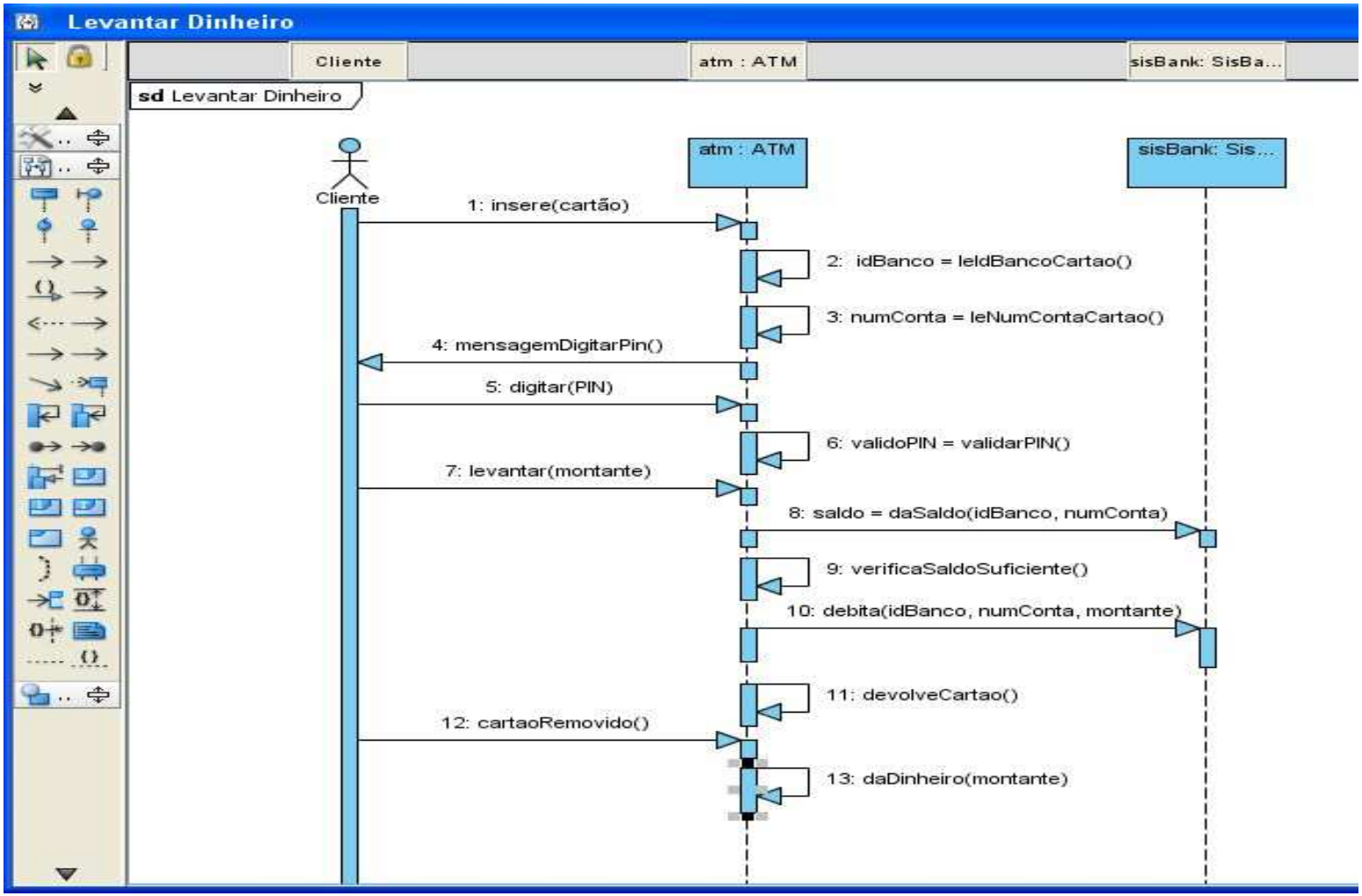
Um DSS com :System deverá ser refinado num DS com :Subsytem1, etc…





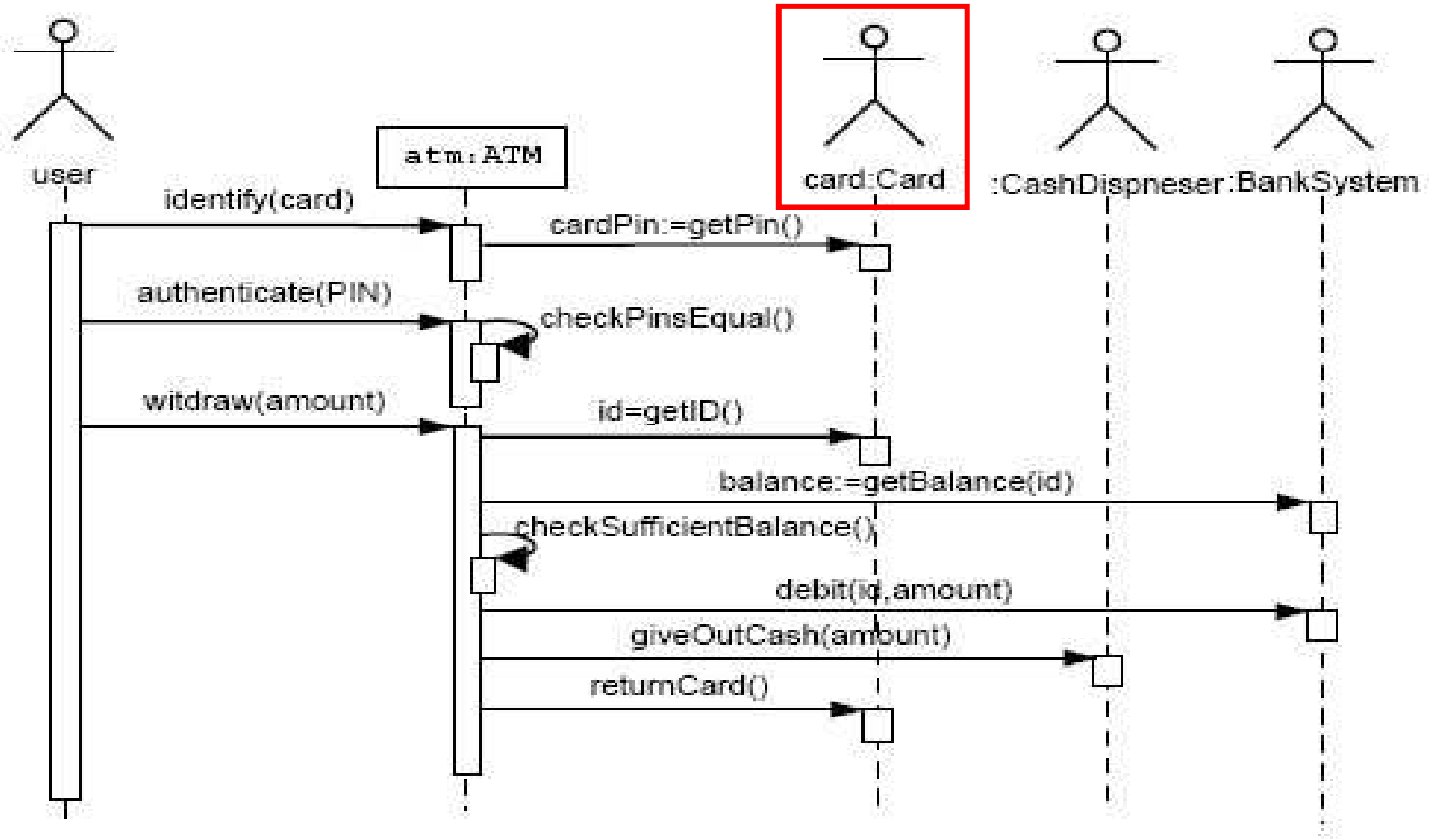
DS: Níveis – Exemplo 2

Um DSS com :System deverá ser refinado num DS com Entidades



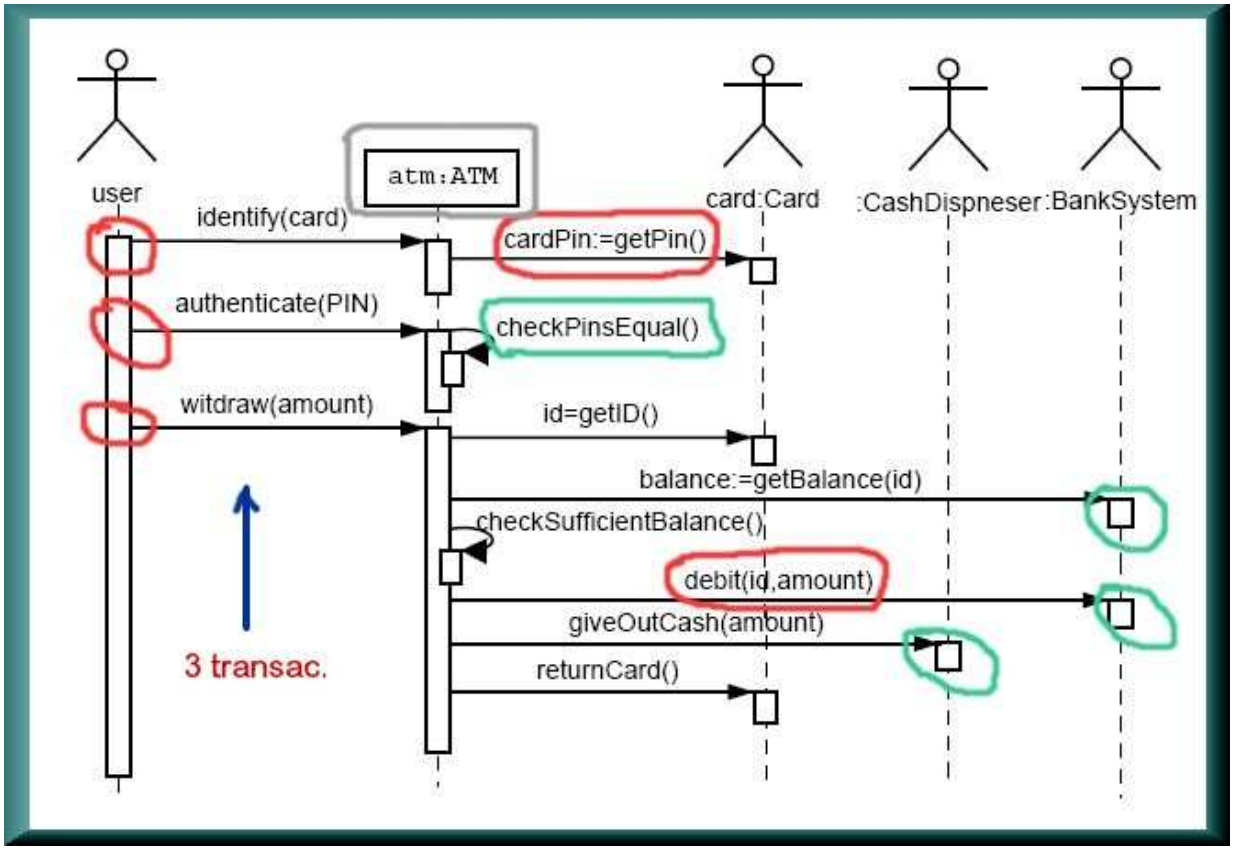
DS: Níveis – Exemplo 3

Um DSS com :System deverá ser refinado num DS com Entidades



DS: Níveis – Exemplo 4

Um DSS com :System deverá ser refinado num DS com Entidades



O número de transacções determinadas mantém-se. Mas há novas mensagens !

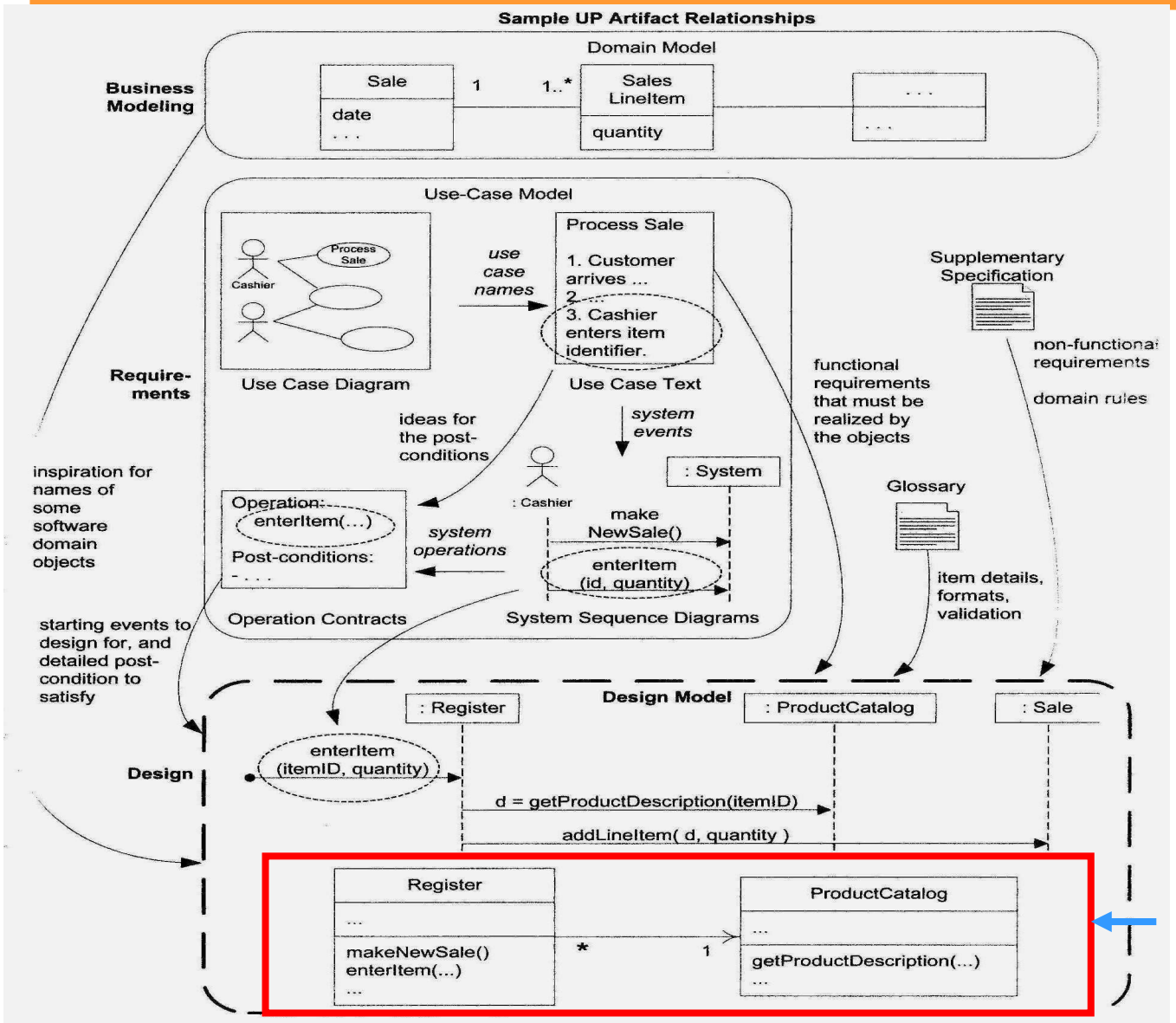
DS: Nível Concepção

Um DSS com :System deverá ser refinado num DS com :Subsytem1, etc…

Isto quer dizer que depois de termos especificado fundamentalmente comportamento (ou seja o **"yin"** do UML), cf. Use Cases, Diagramas de Actividade e Diagramas de Sequência, devemos agora preocuparmo-nos com o **"yang"**, ou seja, a estrutura.

Numa metodologia OO, as entidades que compõem um sistema e que representam a sua estrutura são modeladas, naturalmente, usando **CLASSES**, pois são estas que criam os objectos, que são as entidades computacionais.

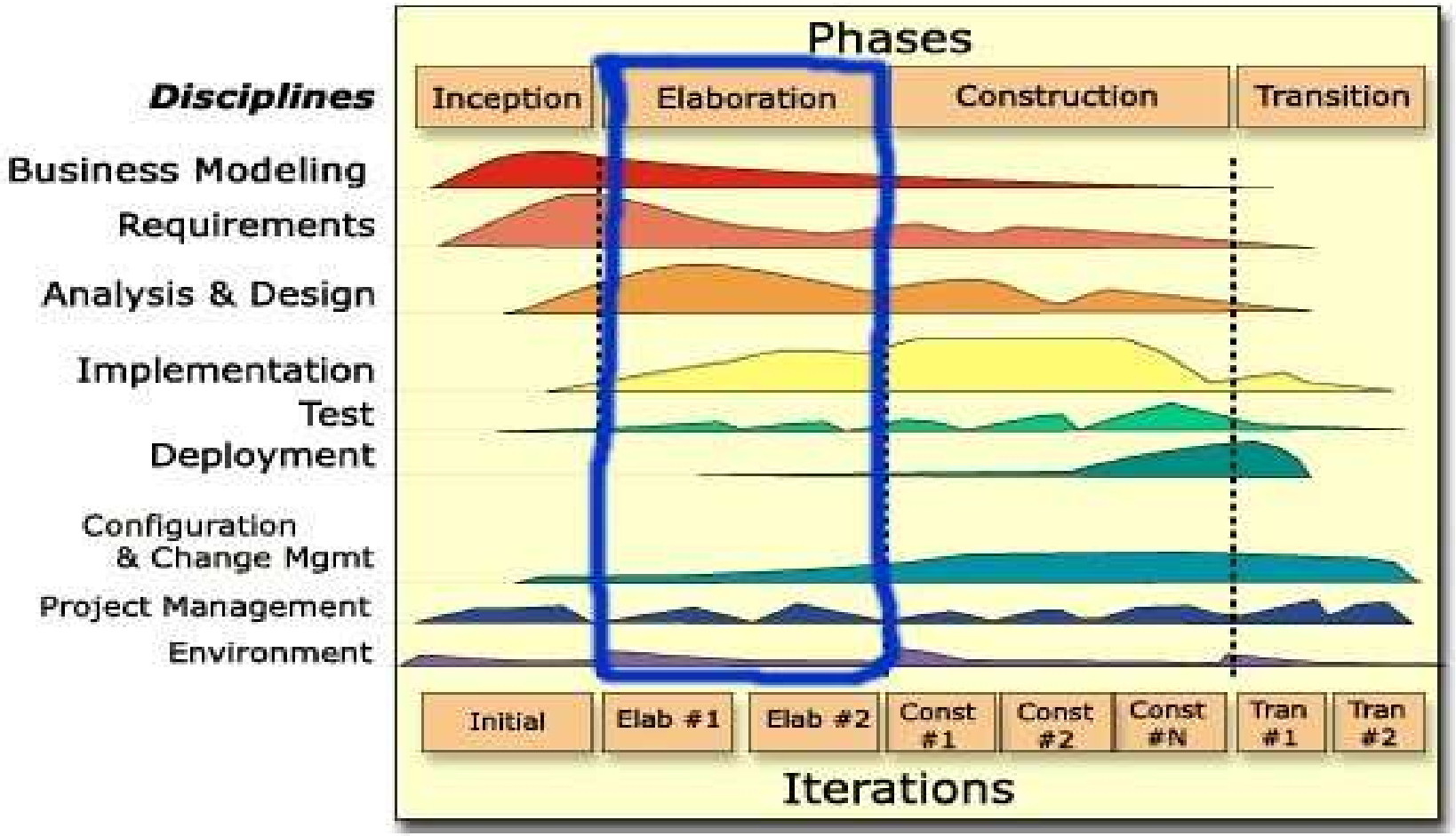
PASSO SEGUINTE: Concepção



Numa abordagem OO, é natural que o Mega-Objecto :System seja em seguida dividido e estruturado em objectos menores com responsabilidades particulares inicialmente não clarificadas mas que agora temos que "desenhar" com maior detalhe.

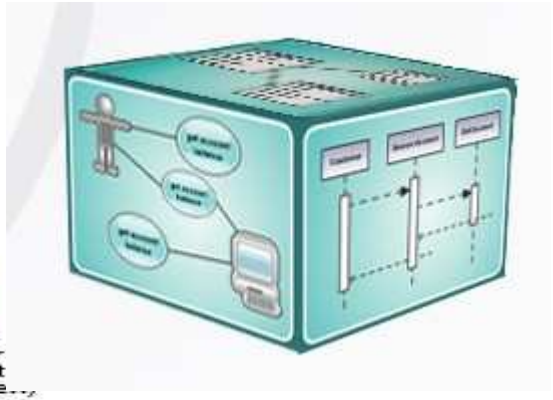
Diagramas de Classe e/ou Packages

PASSO SEGUINTE: Concepção

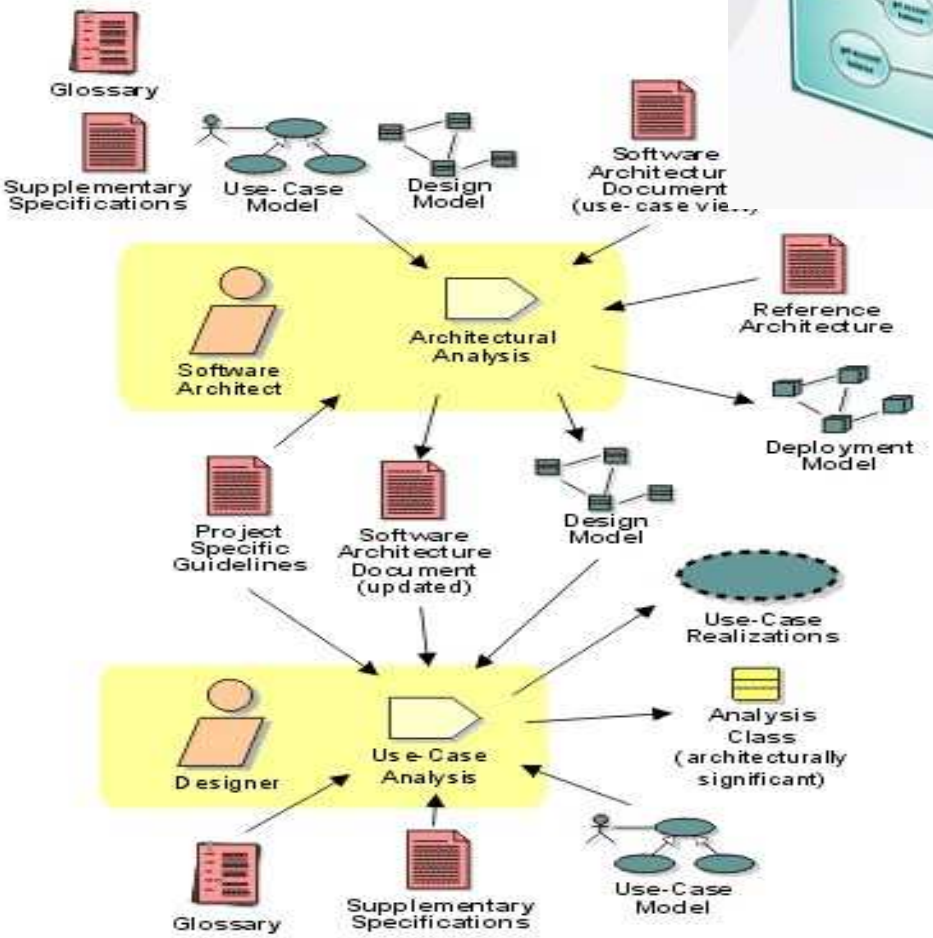


VIEWS e MISSÃO

- Os vários modelos desenvolvidos dão-nos várias "views" do sistema a desenvolver



- Agora temos de as refinar e integrar de modo coerente (análise detalhada), com vista ao desenvol-vimento de um sistema software com arquitectura e desempenho correctos.



DSS: Refinamento

Questão: Qual o critério ou método para se passar de um DSS para um DS de concepção ?

Resposta: Dividindo a "blackbox" :System em subsistemas que possuam a "competência" (o que deve saber) e a "responsabilidade" (o que deve fazer), para responder a certas operações especificadas ao nível do Sistema;

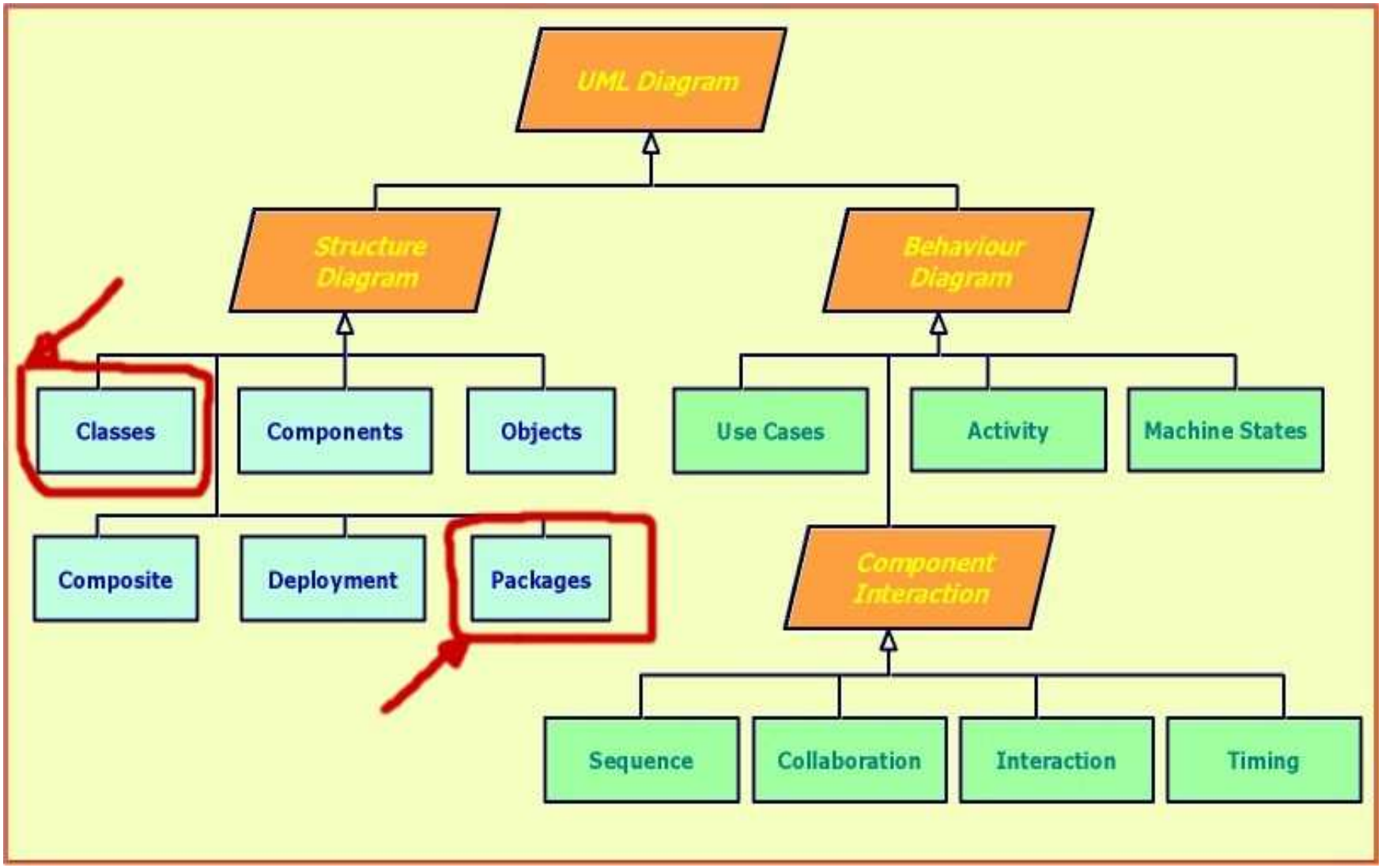
Questão: O que "são" tais subsistemas num DS ?

Resposta: Dado que num DS de UML só há "objectos", serão objectos, ainda que possam ser de "tipos" diferentes, ie., de classes diferentes;

Questão: Como especificar então as competências e as responsabilidades de um objecto a este nível ?

Resposta: Há várias técnicas. Os cartões de competência-responsabilidade (CRC-cards) são uma boa técnica, que numa abordagem OO se transformam de imediato em Classes importantes

DIAGRAMAS DE CLASSES



DIAGRAMAS DE CLASSES

▣ Vamos para já estudar o que os Diagramas de Classes (DC) nos permitem especificar em UML, agora que, cf. o RUP, deixámos as preocupações de captura de requisitos para trás, e vamos passar para a análise e concepção de uma solução, na sua arquitectura e lógica de funcionamento, nas formas da sua implementação e, posteriormente, nas formas de instalação ("deployment").

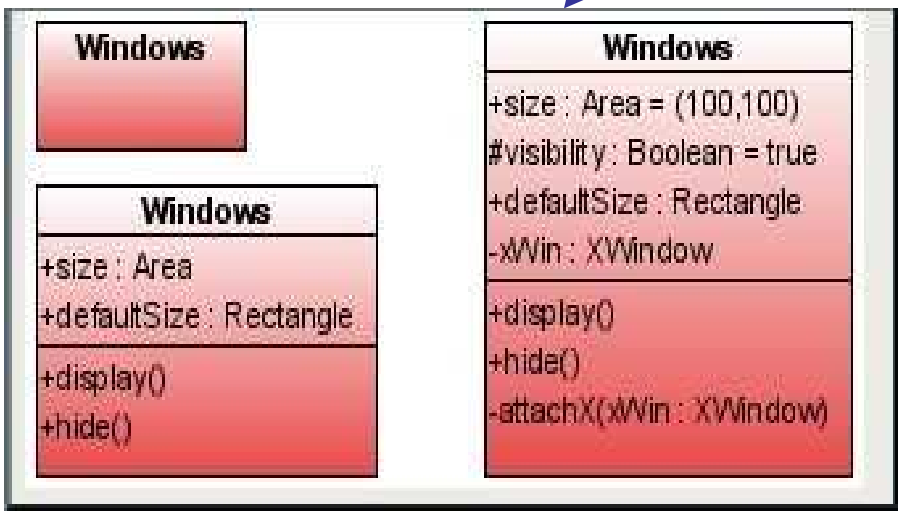
▣ Todos acreditamos, alunos e docentes, que explicadas as técnicas de captura de requisitos e análise de alto nível (cf. Diagramas de Use Cases, Use Cases e Diagramas de Sequência), e, desenvolvida e apresentada de forma consistente uma metodologia para a passagem sistemática de UCs para Diagramas de Sequência, ainda que os DSS sejam apenas diagramas de sequência de Sistema como "black-boxes", o que falta agora é uma "correcta" definição do que se deve entender, e como tal especificar em UML, o que é, de facto, um "subsistema", ou seja, uma correcta partição do que ao nível da análise se considerou como sendo o "sistema".

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.**

Classes podem ser especificadas com vários níveis de detalhe

Métodos, Atributos e suas visibilidades



+	public
#	protected
-	private
~	package

DIAGRAMAS DE CLASSES

Quando se modela um **sistema**, alguns objectos estarão certamente relacionados com outros, através de relacionamentos diferentes, mas que semanticamente devem ser bem definidos.

Há 5 tipos de relacionamentos expressáveis em Diagramas de Classes de UML:

1) Associação bi-direccional;

2) Associação uni-direccional;

3) Agregação de Classes (Agregação básica e Composição);

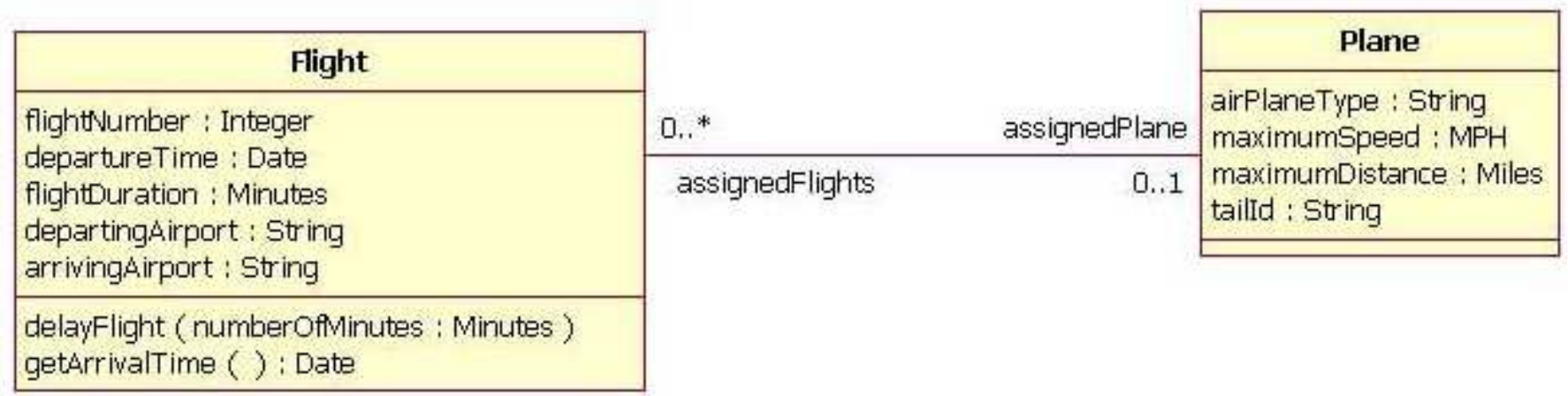
4) Associações Reflexivas;

5) Associação com Classes de Associação;

Para além destas associações, podem ser usados mecanismos de classificação tais como interfaces, packages, classes abstractas, e, como é óbvio em OO, o mecanismo de herança entre classes e entre interfaces.

Associação bi-direccional

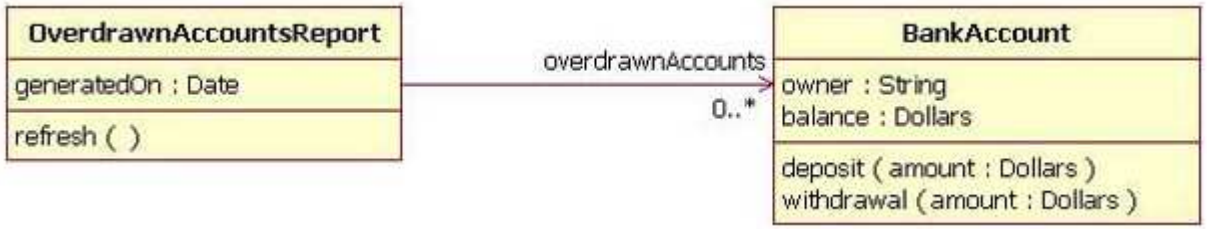
Uma associação bidireccional é especificada por uma linha sólida que une duas classes, tendo no seu início e fim atributos de multiplicidade. Os papéis de cada classe na associação são descritos textualmente.

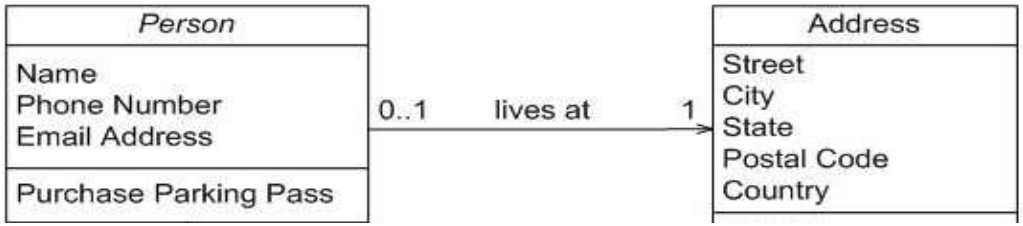


Ler: Cada instância de Voo tem a si "atribuído" (assigned), num dado momento, ou 0 ou 1 instância de Avião. Cada instância de Avião tem a si atribuídos 0 ou mais voos.

Associação uni-direccional

Uma associação unidireccional é especificada por uma linha sólida com seta, indicando que apenas uma das classes "conhece" o seu relacionamento com a outra (e o seu significado).



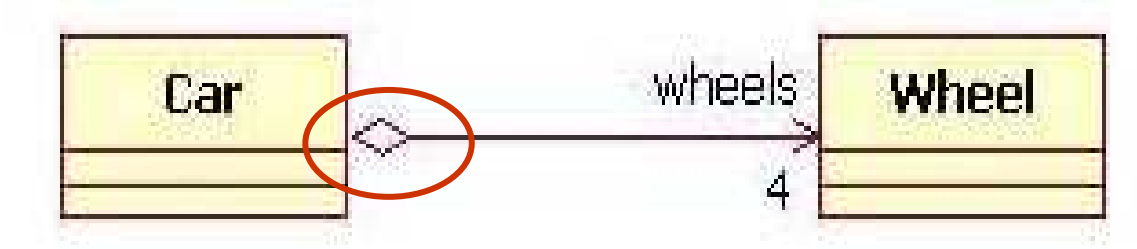


Em geral é preferível definir relações bi-direccionais, porque, em tais casos, tal implica que há a "questão inversa" à qual é possível responder, ou seja, de uma entidade sabemos da outra e vice-versa.

DC: Agregação Básica

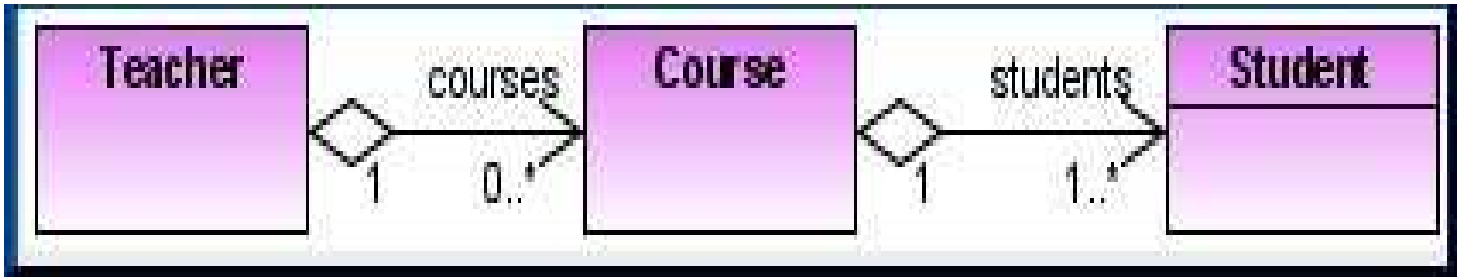
Agregação básica

Numa agregação básica, especifica-se que há uma associação entre duas classes tão forte que uma (ou mais delas) fazem parte da definição e estrutura interna da outra, e até em certo número (recordar que uma classe define objectos com dada estrutura representada por variáveis de instância que são objectos de outras classes).



Aspecto importante da semântica: Embora estando cada instância de Carro associada por agregação básica (ou agregação) a 4 instâncias de Roda, se tal instância de Carro for destruída as instâncias de Roda são independentes e, portanto, permanecem "vivas", não sendo destruídas.

DC: Agregação Básica



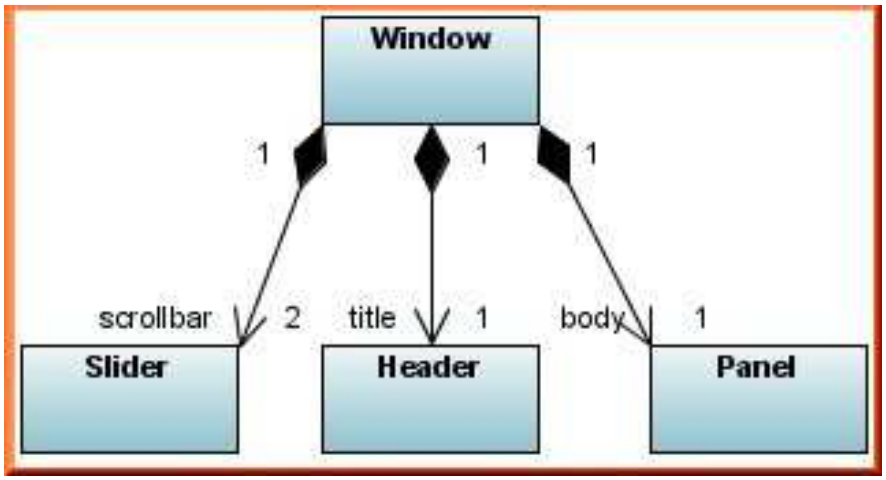
Alunos existem no modelo independentemente dos cursos, cursos existem independentemente dos professores que os possam leccionar, mas há relações bem definidas entre estas entidades "independentes" quando se relacionam entre si.

Um professor lecciona 0 ou mais disciplinas (UC), cada disciplina é "estudada"/"frequentada" por 1 ou mais alunos.

DC: Composição

Agregação por Composição

Numa **composição**, especifica-se que há uma associação entre duas classes tão forte que uma (ou mais delas) fazem parte da vida da outra e, portanto, quando a instância da classe superior é destruída, todas as instâncias das classe compostas devem também ser destruídas. As classes que compõem não têm sentido sem a sua class superior.

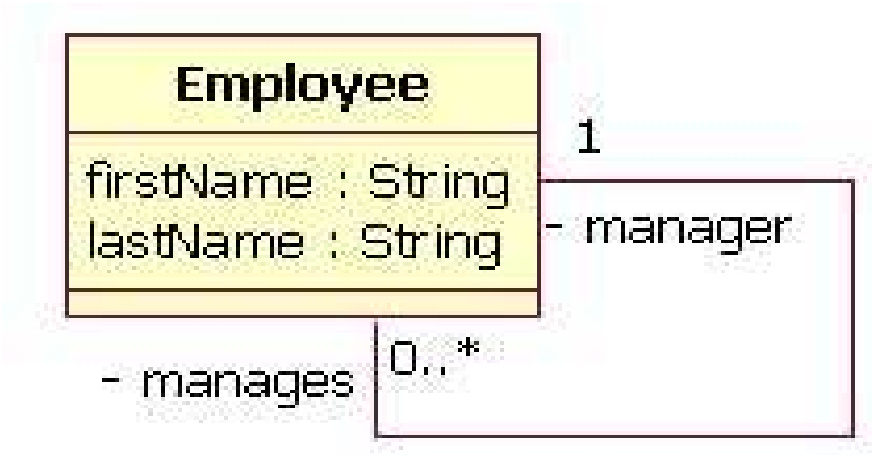


Decisões do tipo:

Se uma Biblioteca for fechada os livros que lhe estavam associados são destruídos ou não ?

Associação Reflexiva

Numa associação reflexiva, especifica-se que uma instância de uma classe pode referenciar uma outra da mesma classe, ainda que entre elas haja uma dada semântica de relacionamento que pode derivar de papéis semânticos diferentes (cf. Actores e generalização).



Uma instância de Empregado pode ser "manager" de 0 ou mais "empregados". Quer tenha ou não sentido, é o que está especificado.

DC: Exemplo - Interpretação