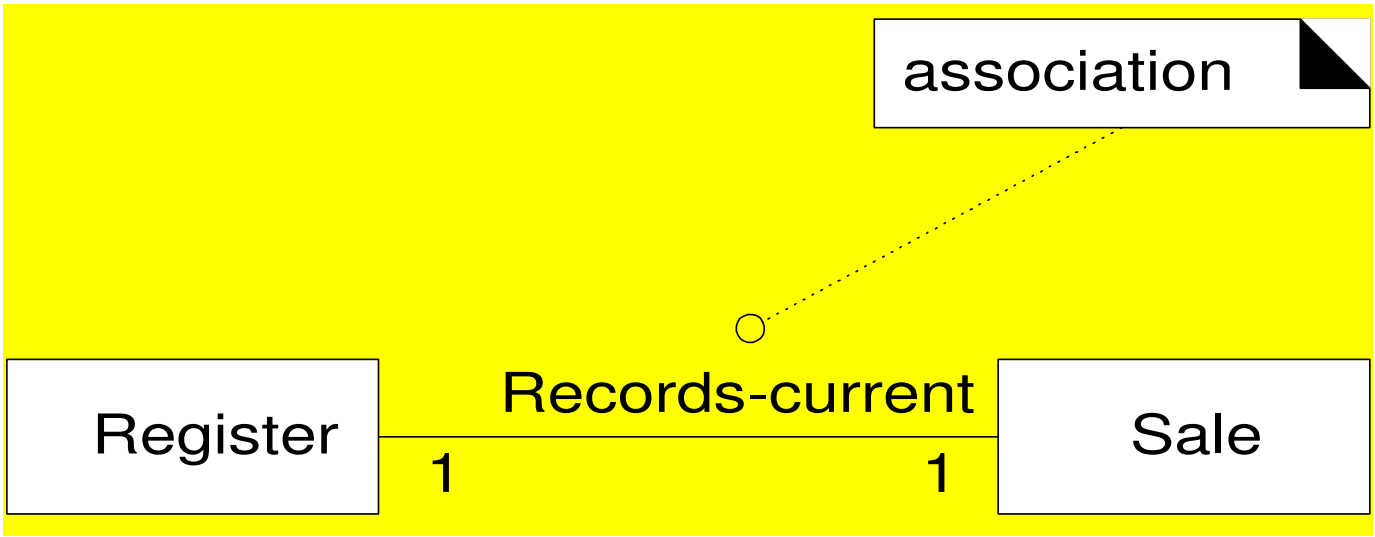
Modelo de Domínio: Adicionando Associações e Atributos

Uma Associação

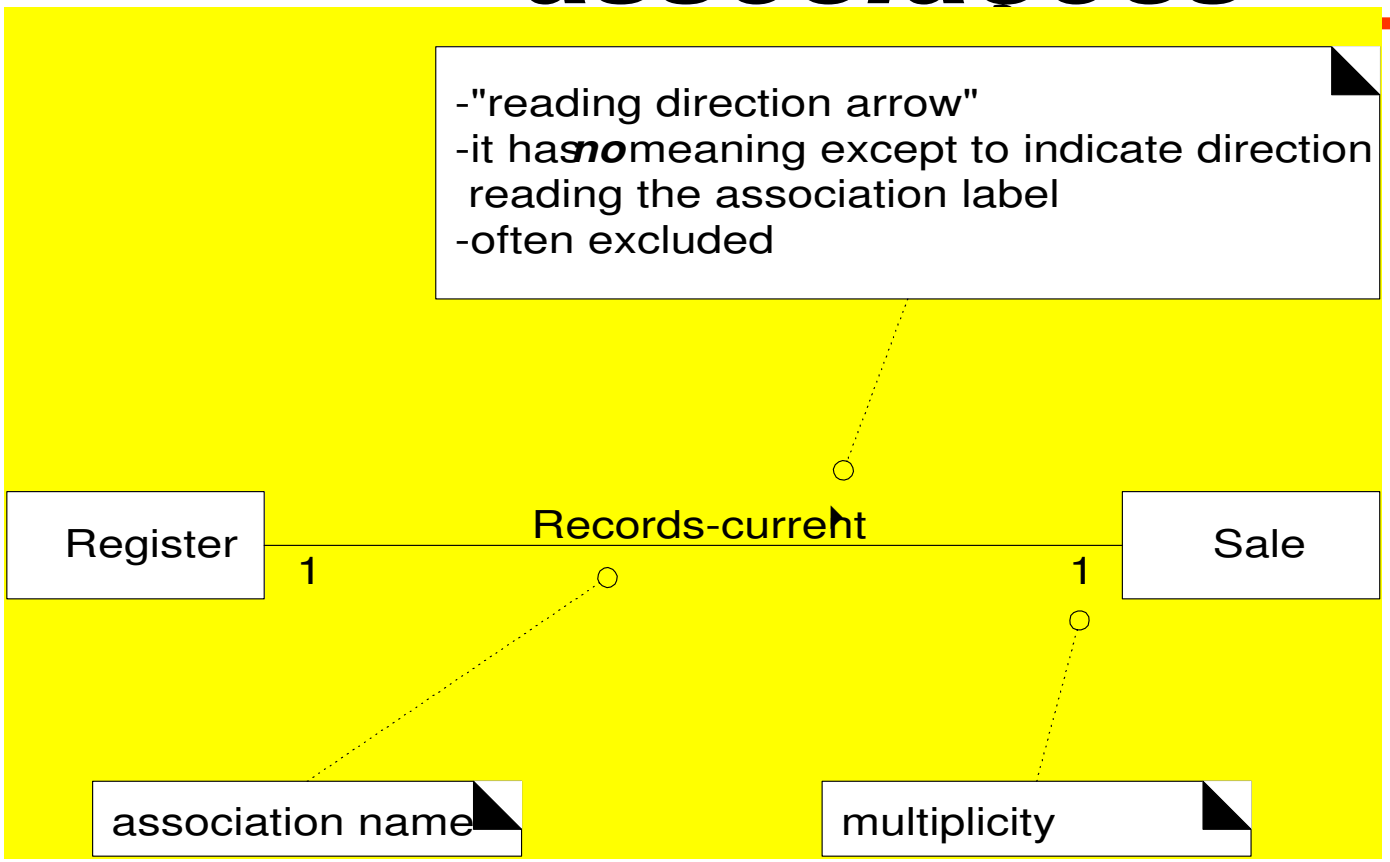
- É um relacionamento entre tipos (mais especificamente, instancias dos tipos) que indica alguma conexão significativa
- Associações importantes geralmente implicam conhecimento de um relacionamento que precisa ser preservado
  - ex. Precisamos lembrar que instancias de *SalesLineItem* estão associadas com instancia de *Sale*?
  - Precisamos registrar a relação entre uma *Sale* corrente e um *Manager*?

Associações

Inerentemente bi-direcional, abstrata



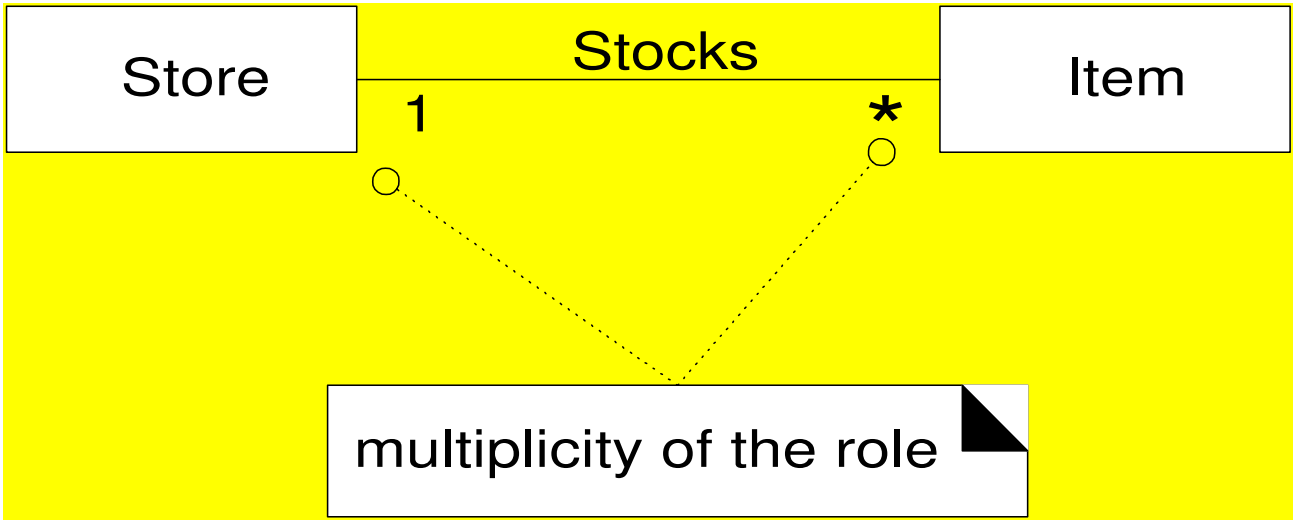
A notação UML para associações



Based on C. Larman, 2002

Expressão indicando a relação numérica entre instancias de classes

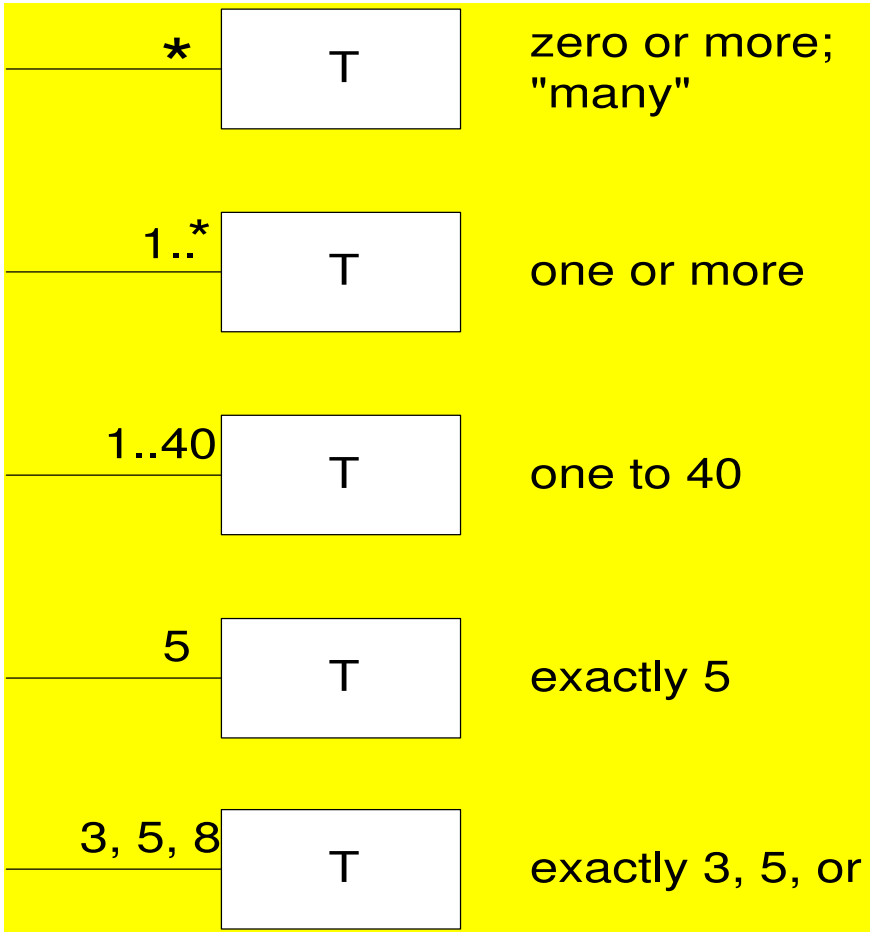
Multiplicidade em uma associação



Based on C. Larman, 2002

Ex. Uma única instancia de uma *Store* pode ser associada a **muitos** [0 ou mais] instancias de *Item*

Exemplos de expressões de multiplicidade



Based on C. Larman, 2002

Multiplicidade é dependente de contexto

Multiplicity should "1" or "0..1"?

The answer depends on our interest in using the model. Typically and practically, the muliplicity cor
domain constraint that we care about being able to check in software, if this relationship was implen
in software objects or a database. For example, a particular item may become sold or discarded, a
stocked in the store. From this viewpoint, "0..1" is logical, but ...

Do we care about that viewpoint? If this relationship was implemented in software, we would probal
that an *Item* software instance would always be related to 1 particular *Store* instance, otherwise it indicates a fau
corruption in the software elements or data.

This partial domain model does not represent software objects, but the multiplicities record constrai
value is usually related to our interest in building software or databases (that reflect our real-world c
checks. From this viewpoint, "1" may be the desired value.

Based on C. Larman, 2002

A multiplicidade

- Comunica uma restrição do domínio que será (ou poderá ser) refletida no software

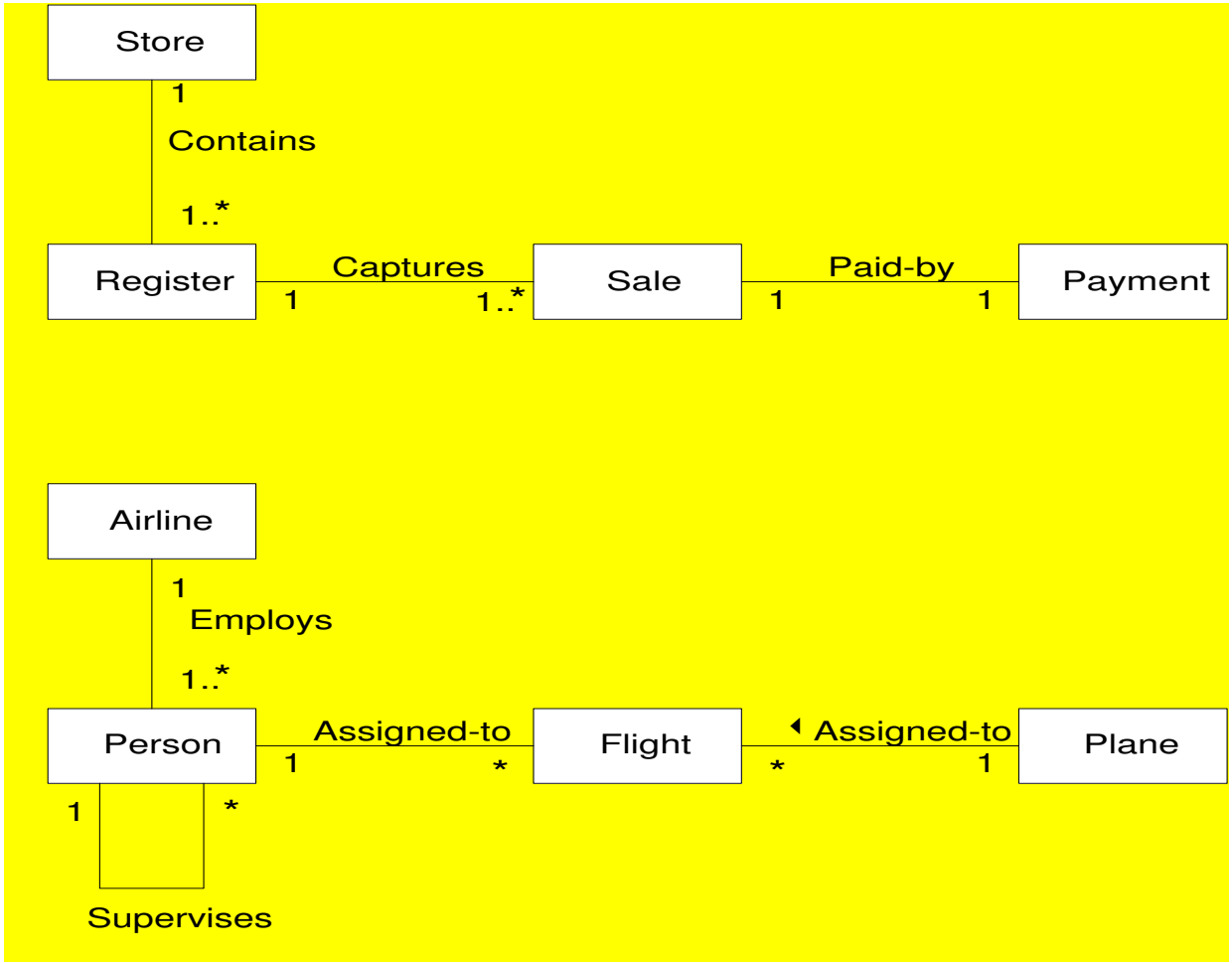
- "0..1" é lógico [um item particular pode tornar-se vendido ou descartado e não continuar estocado na loja]
- "1" pode ser desejável [multiplicidade registra restrições cujo valor prático é usualmente relacionado a nosso interesse na construção do sftw ou db]

Nomeando Associações

- Nomeie uma associação com base no formato ***TypeName-VerbPhrase-TypeName*** onde a frase verbal cria uma seqüência que é legível e significativa no contexto do modelo
  - ex. *Register* ***Captura*** *Sale*

- Elas representam classificadores de links entre instancias. – comece com letra maiúscula
  - *PaidBy* or *Paid-by*

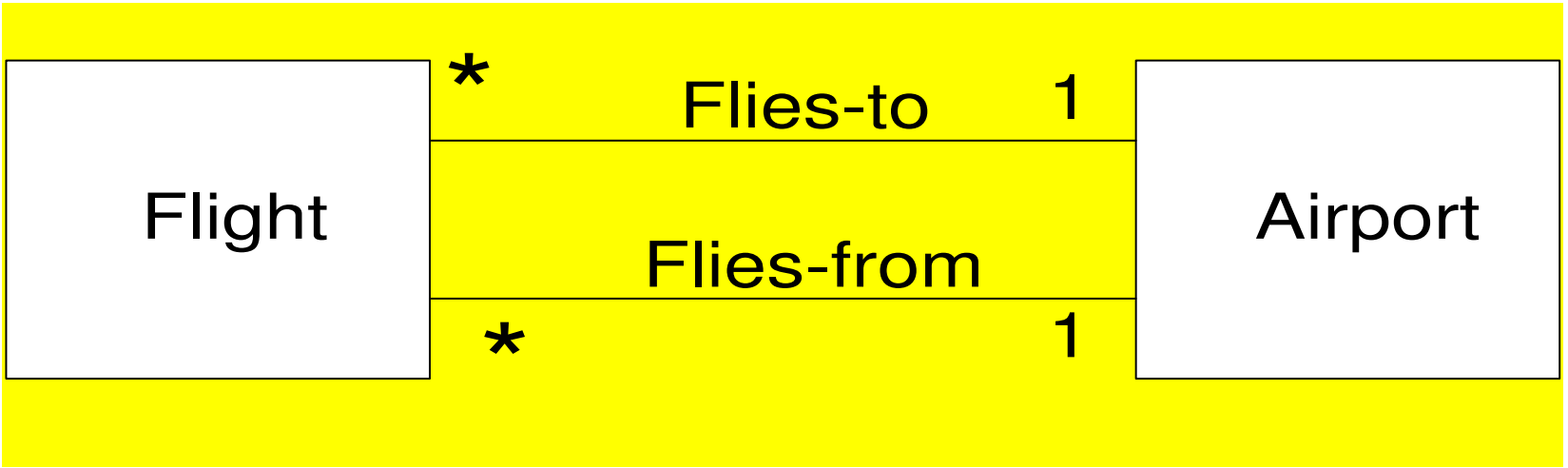
Nomes de Associações



Based on C. Larman, 2002

Associações Múltiplas

Dois tipos podem ter múltiplas associações entre eles
flying-to and flying-from são duas relações diferentes



Checklist de Categorias para encontrar Associações:

1. **A é uma parte física de B**

   Register-CashDrawer

2. **A é parte lógica de B**

   SalesLineItem-Sale

3. **A está fisicamente contida em B**

   Register-Store, Item-Shelf

4. **A está logicamente contida em B**

   ProductSpecification-Catalog,
   ProductCatalog-Store

5. A é uma descrição de B

   ProductSpecification-Item

6. A é um item de linha de uma transação ou relatório de B

   SalesLineItem-Sale

7. A é conhecido/registrado/capturado em B

   Sale-Register

8. A é membro de B

   Cashier-Store

9. A é subunidade organizacional de B

   Department-Store

10. A usa ou gerencia B

    Cashier-Register, Manager-Register, Manager-Cashier

11. A comunica-se com B

    Customer-Cashier

12. A é relacionado a uma transação de B

    Customer-Payment, Cashier-Payment

13. A é uma transação relacionada a outra transação de B

    Payment-Sale

14. A é próximo a B

    SalesLineItem-SalesLineItem

15. A é possuído por B

    Register-Store

16. A é um evento relacionado a B

    Sale-Customer

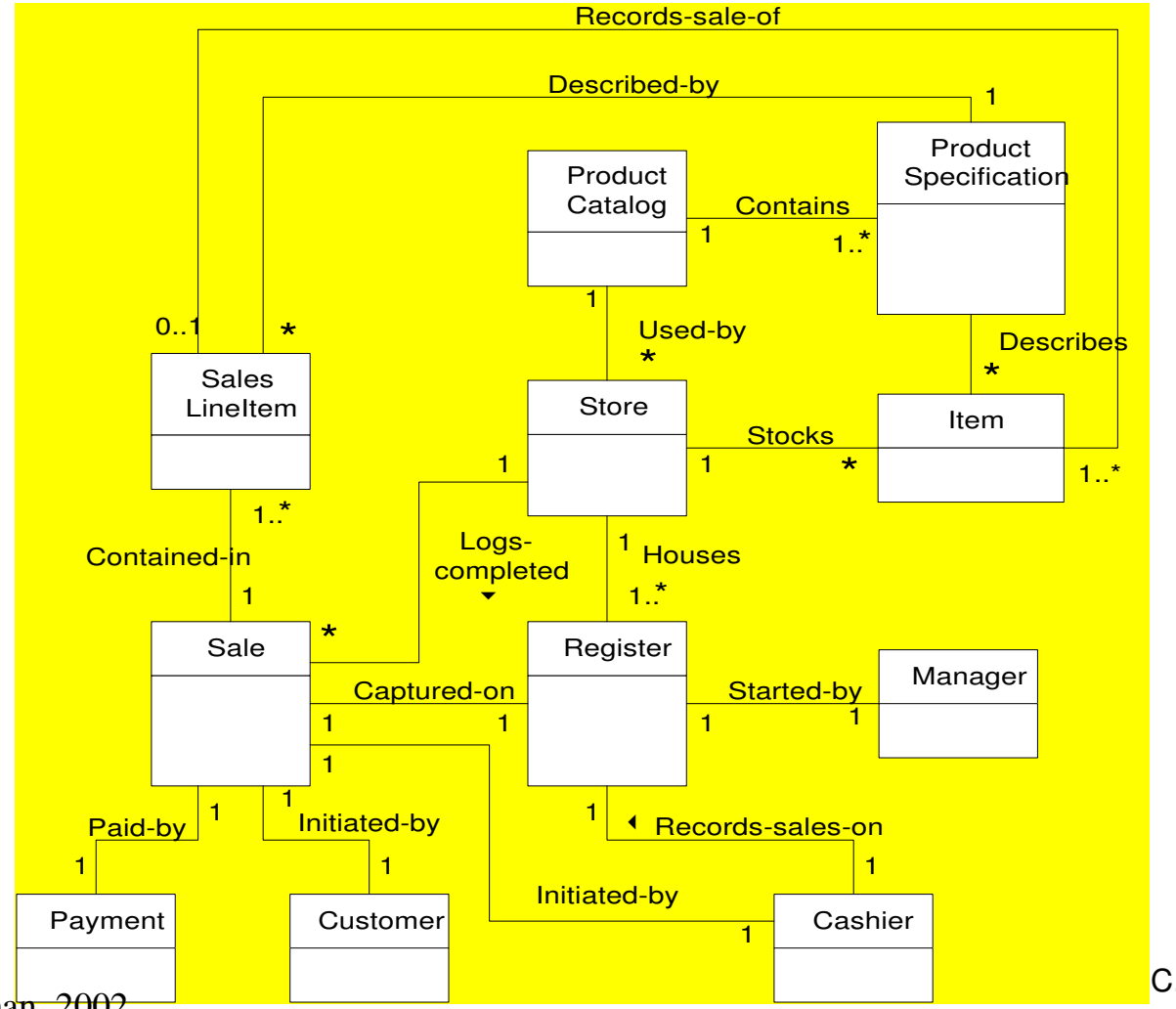
Guidelines para Associações

- Focar em categorias de alta prioridade cujo conhecimento deve ser preservado
- É mais importante identificar classes conceituais do que associações
- Associações em excesso tendem a confundir o DM em vez de clarificá-lo
- Evite mostrar associações redundantes ou deriváveis

Adicionando associações ao POS DM:

- Com base nos Casos de Uso sob consideração e na Lista de Categorias de Associações:
- *Register Records Sale*
  - Para conhecer a venda corrente, gerar um total, imprimir um recibo
- *Sale Paid-by Payment*
  - Para saber se a venda foi paga, relacionar a quantia paga com o total da venda, e imprimir um recibo
- *ProductCatalog Records ProductSpecification*
  - Para recuperar um Product Specification, dado um itemID

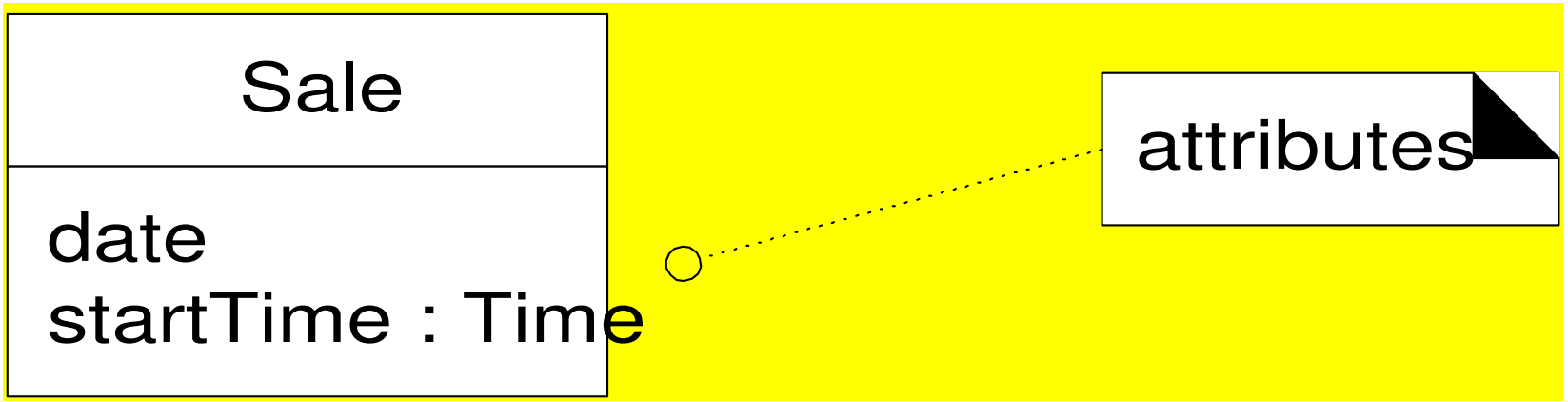
Um modelo de domínio parcial para POS



Um Atributo

- É o valor lógico de um objeto
- Incluir os seguintes atributos em um DM:
  - Aqueles para os quais os requisitos [uc] sugerem ou implicam uma necessidade de lembrar informação
  - ex. Um recibo normalmente inclui uma data, hora, e a gerência quer conhecê-los
- Intuitivamenteos tipos de atributos mais simples são tipos de dados primitivos [boolean, date, number, string, time]

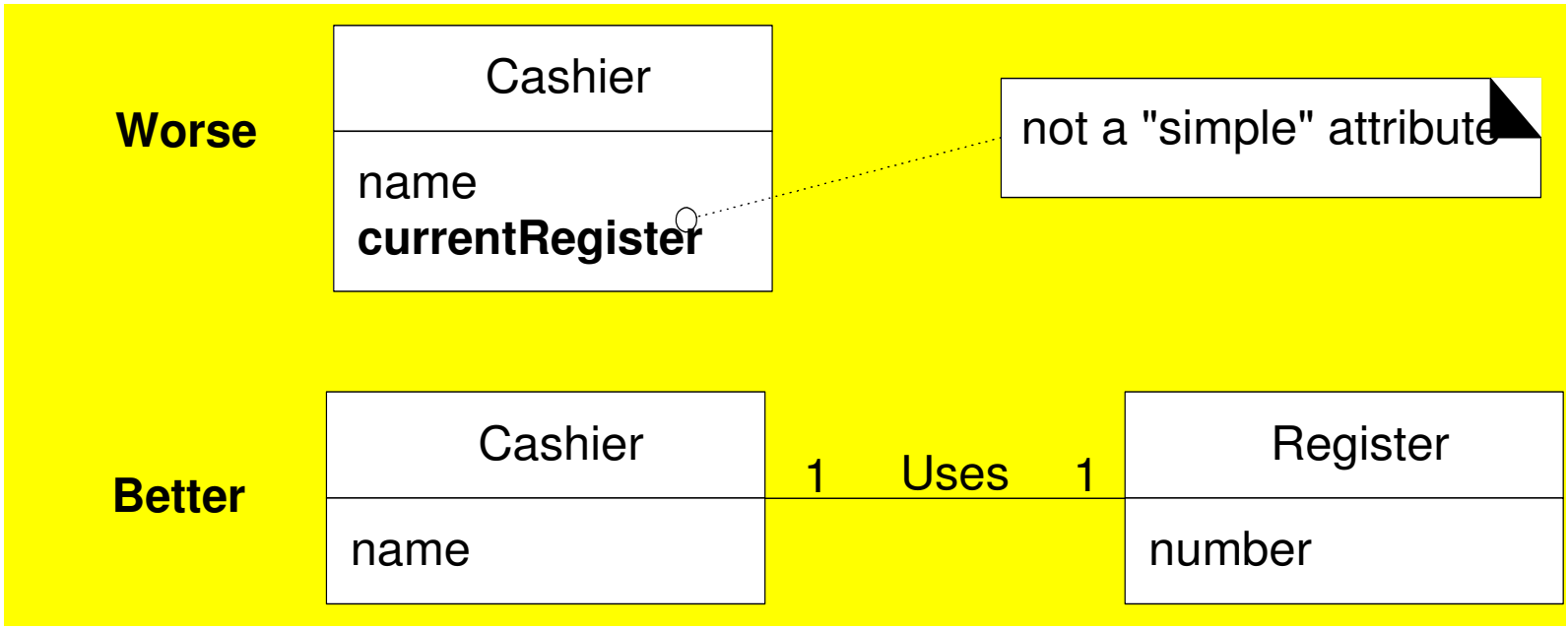
Classe e atributos



Based on C. Larman, 2002

Tipos podem opcionalmente ser mostrados

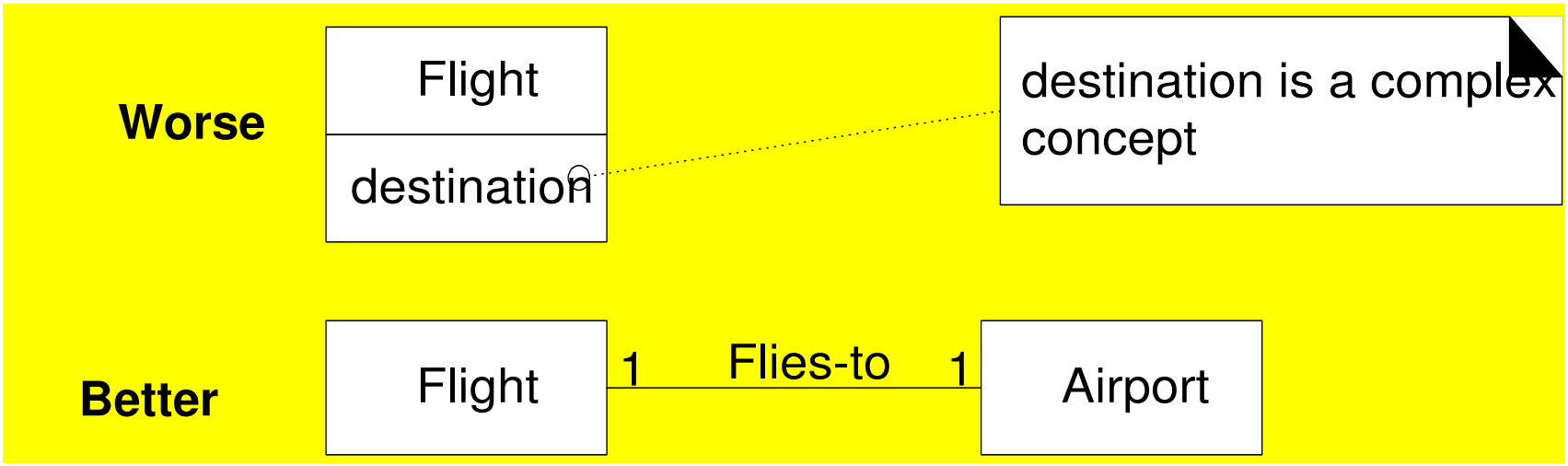
Mantenha os atributos simples



Based on C. Larman, 2002

Evite representar conceitos complexos do domínio como atributos; use associações.

Evite usar conceitos complexos do domínio como atributos

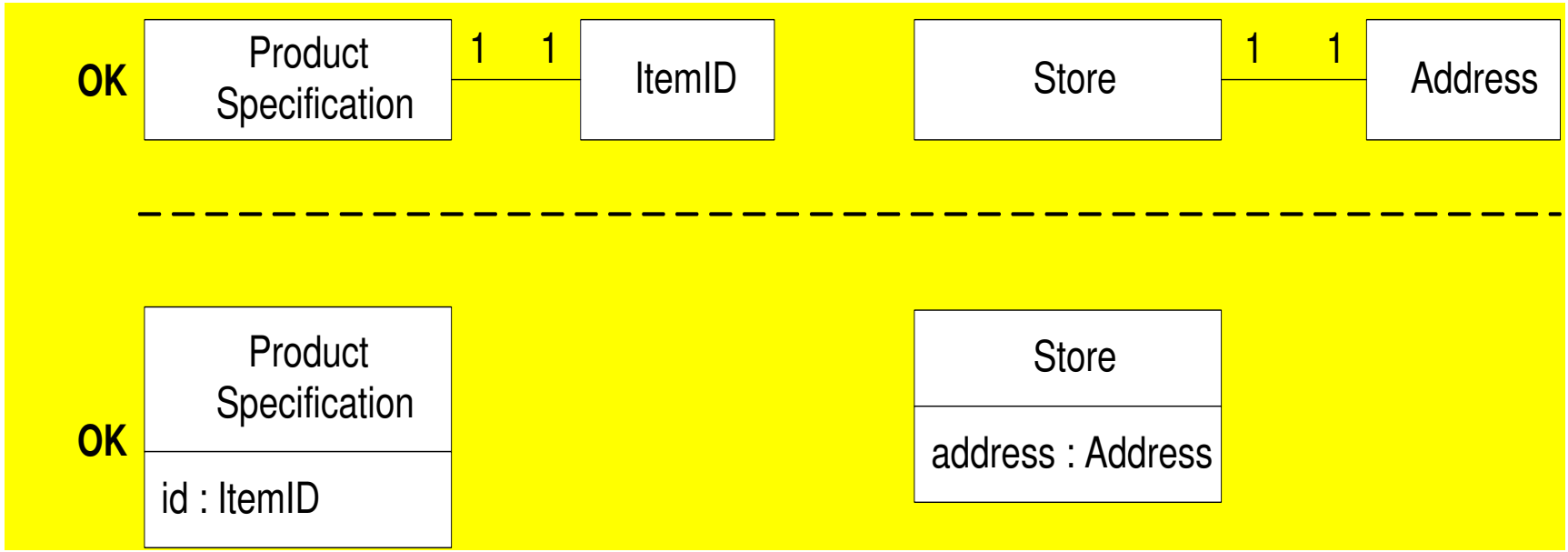


Based on C. Larman, 2002

Tipos de dados não primitivos

- Todos os tipos de dados primitivos (number, string) são tipos de dados UML [o oposto não é verdade]
- Um tipo de dado não primitivo:
  - É composto de seções separadas [phone number]
  - Há operações associadas a eles [social security number]
  - Tem outros atributos [preços promocionais devem ter datas de início e fim]
  - É uma quantidade com uma unidade [payment amount]
  - É uma abstração de um ou mais tipos [EAN European article number]

Tipos de Dados mostrados na caixa de atributos

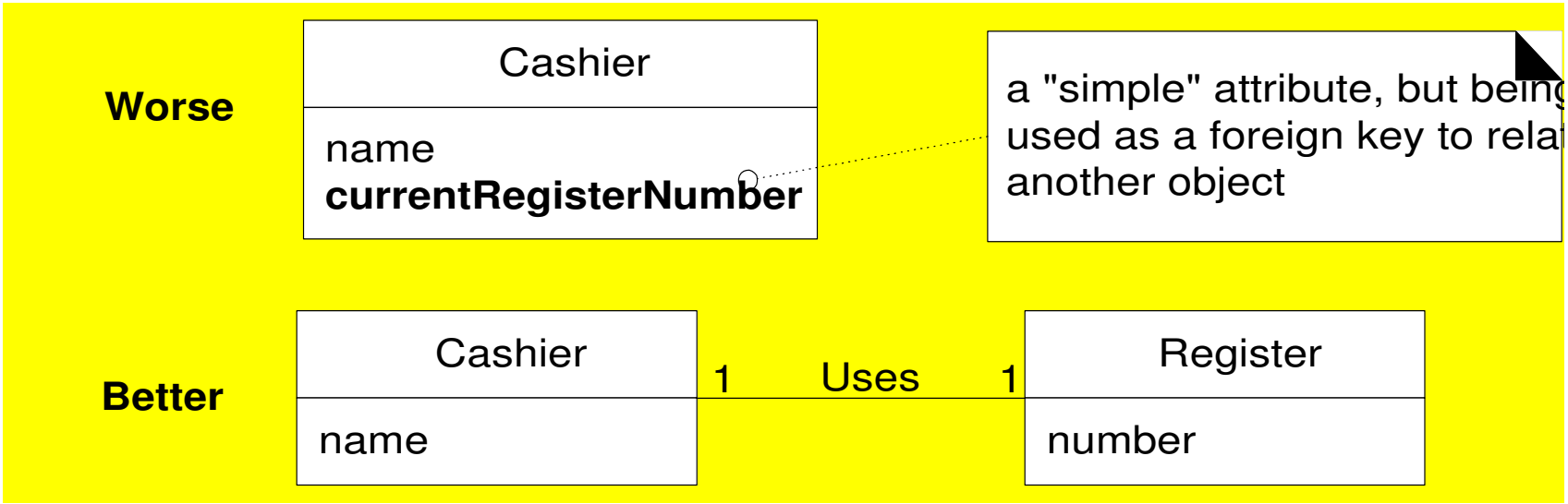


Depende de o quê se quer enfatizar no diagrama

Based on C. Larman, 2002

Não usar atributos como chaves estrangeiras

Atributos não devem ser usados para relacionar classes conceituais no DM

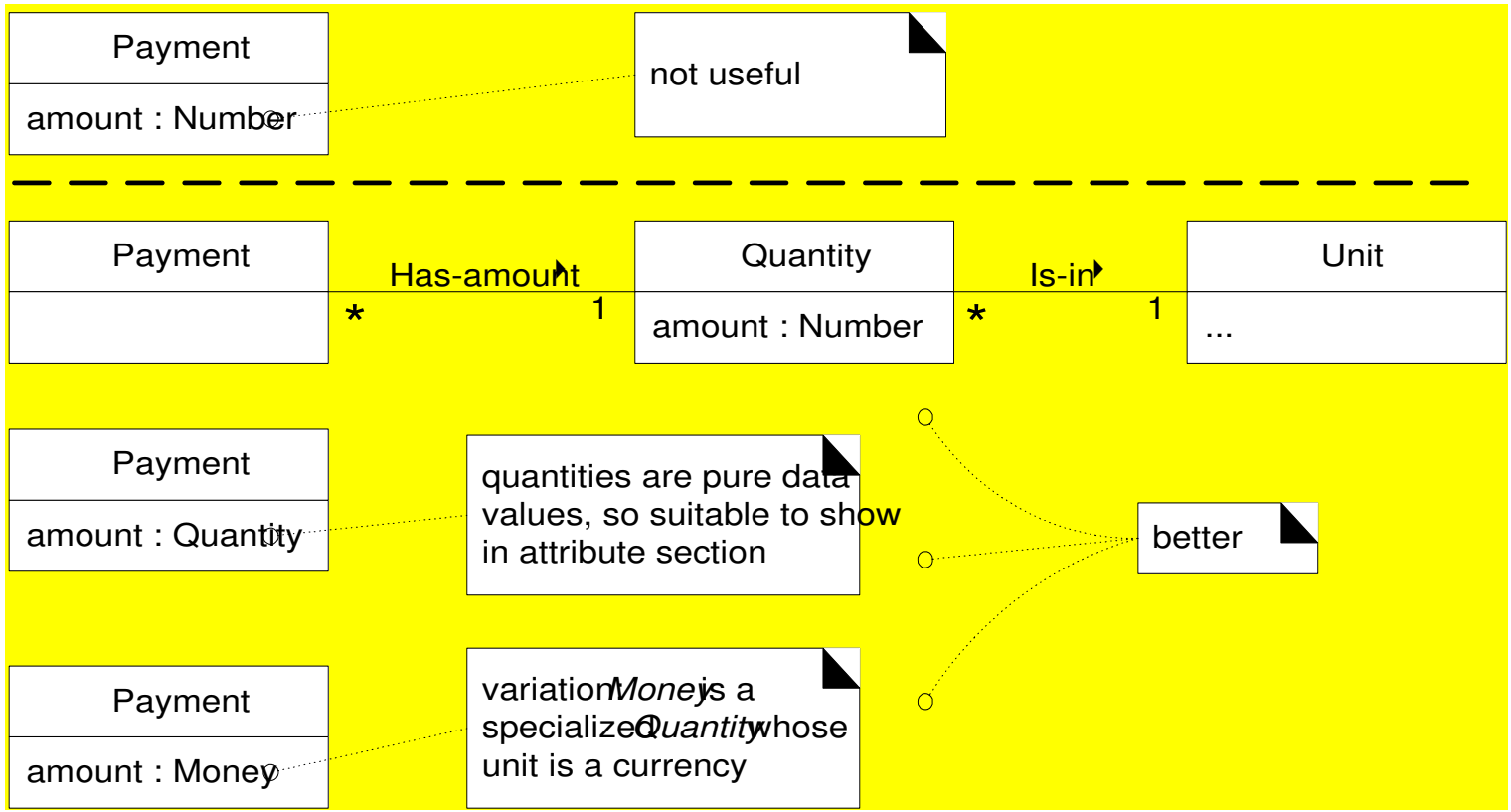


Based on C. Larman, 2002

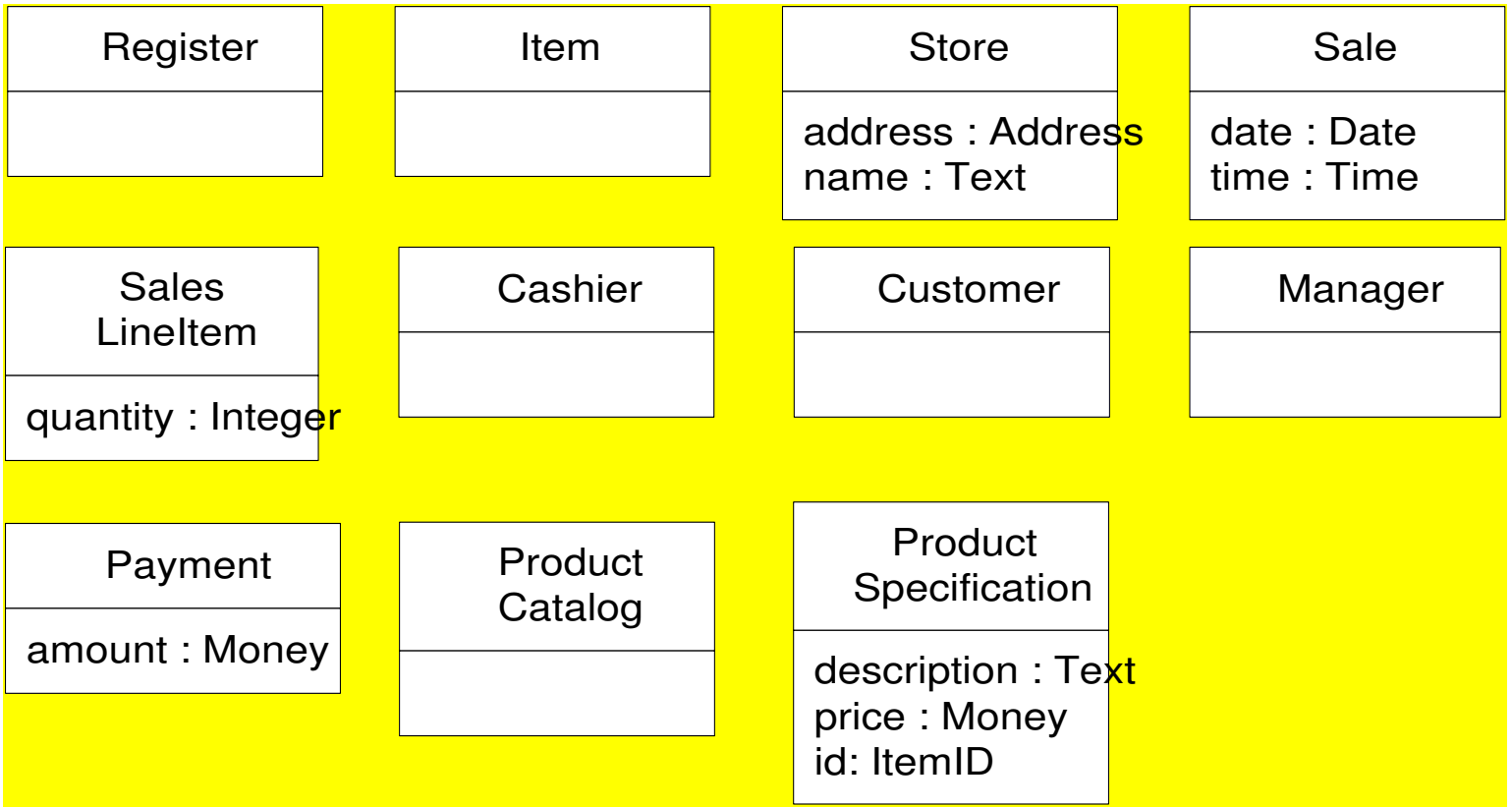
Modelando atributos de quantidades e unidades

- A maioria das quantidades numéricas não deve ser representada como números plenos.
  - Ex. preço, velocidade
- Quantidades associadas a unidades requerem conhecer a unidade e apoiar conversões
  - ex. Quantidade poderia ser uma classe conceitual distinta, com uma unidade associada
    - Ou um tipo de dado

Modelando quantidades



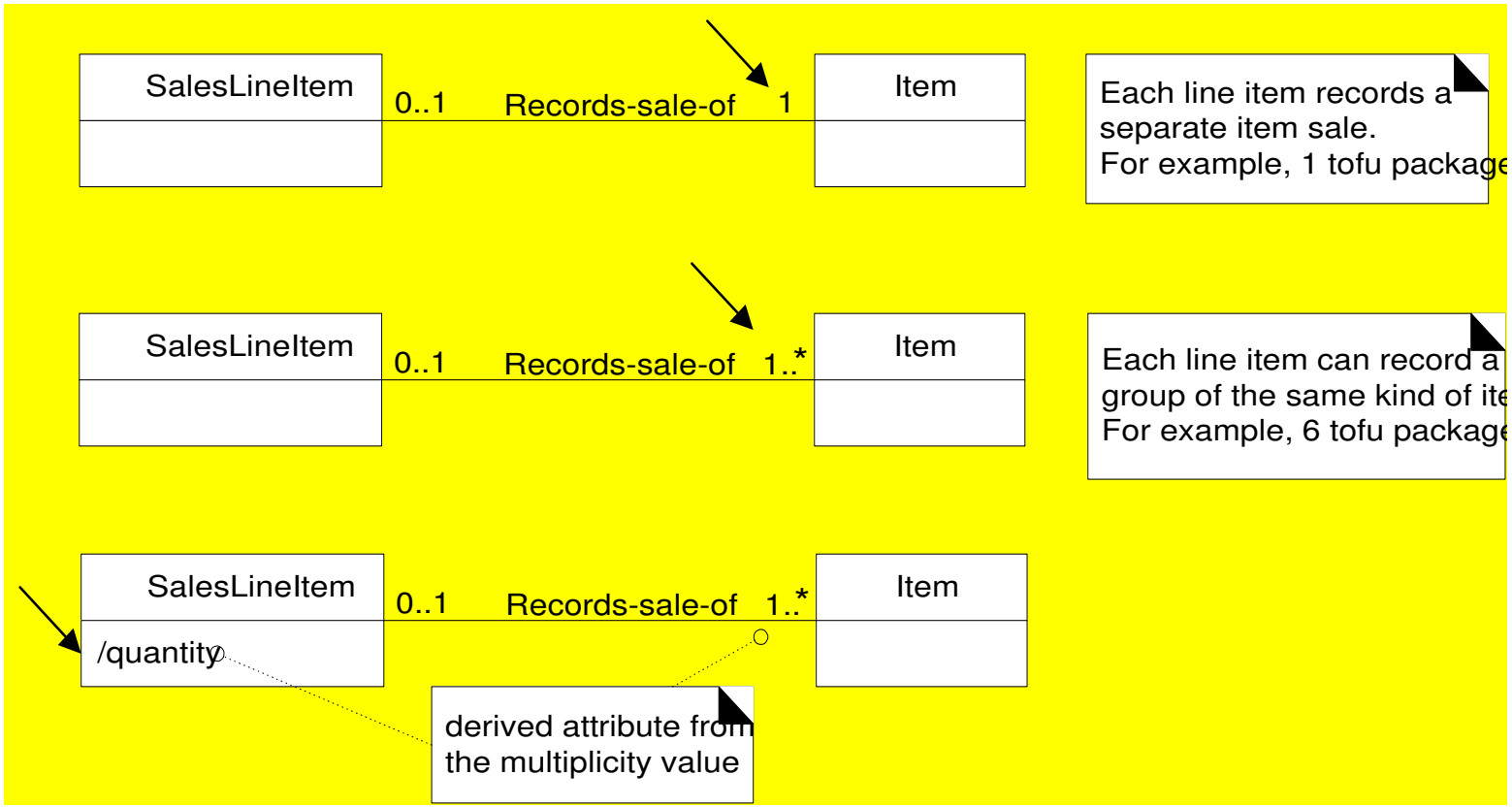
Mostrando Atributos do POS DM



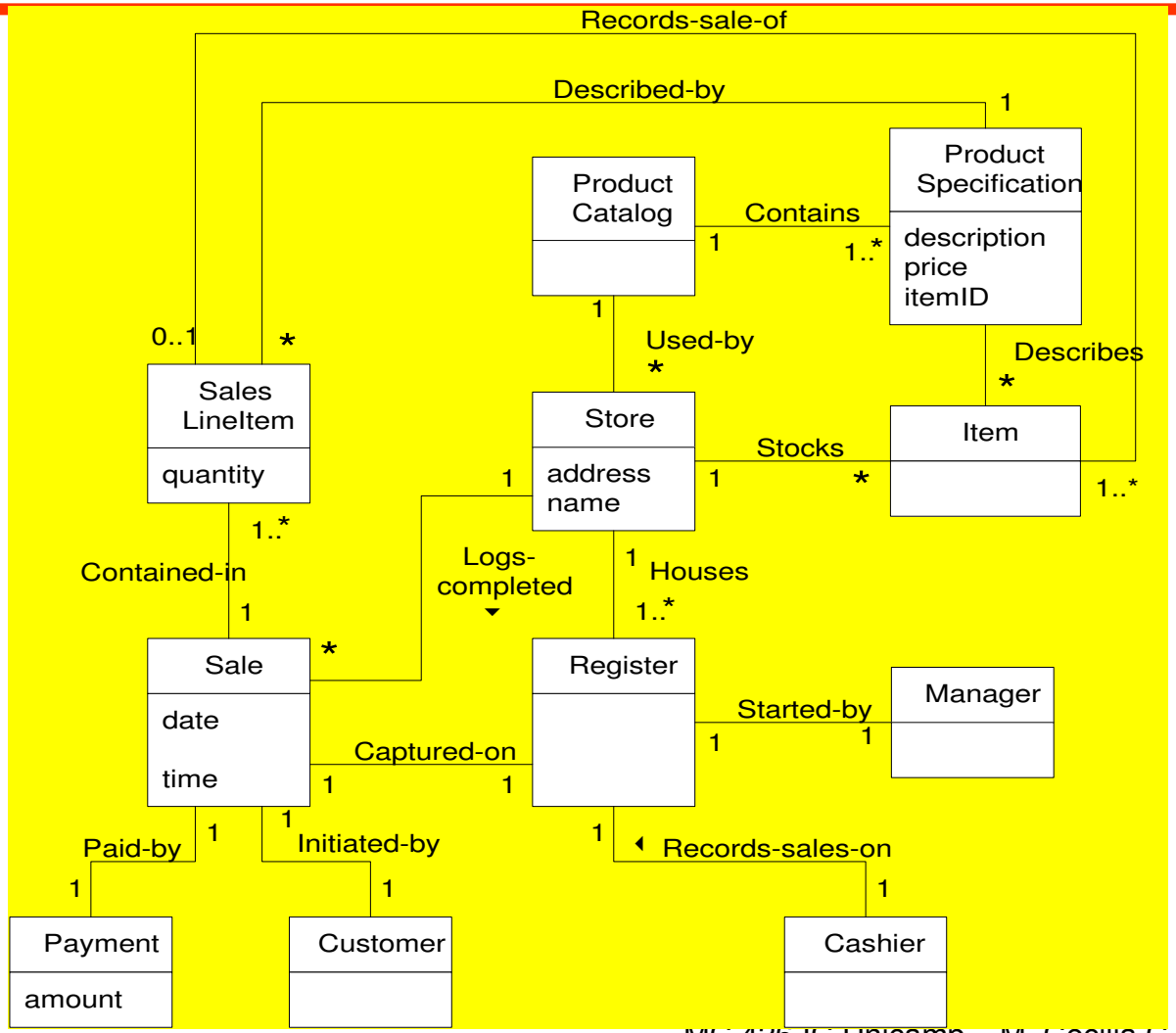
Considerando grupos de items do mesmo gênero

- É possível uma caixa receber um grupo de itens [E.g. 6 pacotes de tofu], entrar o itemID uma vez e entrar a quantidade [6]
  - consequentemente um SalesLineItem individual pode ser associado a mais de uma instancia de um item
  - quantidade pode ser caracterizada como um atributo derivado [indicado com o "/""]

Registando a quantidade de itens vendidos

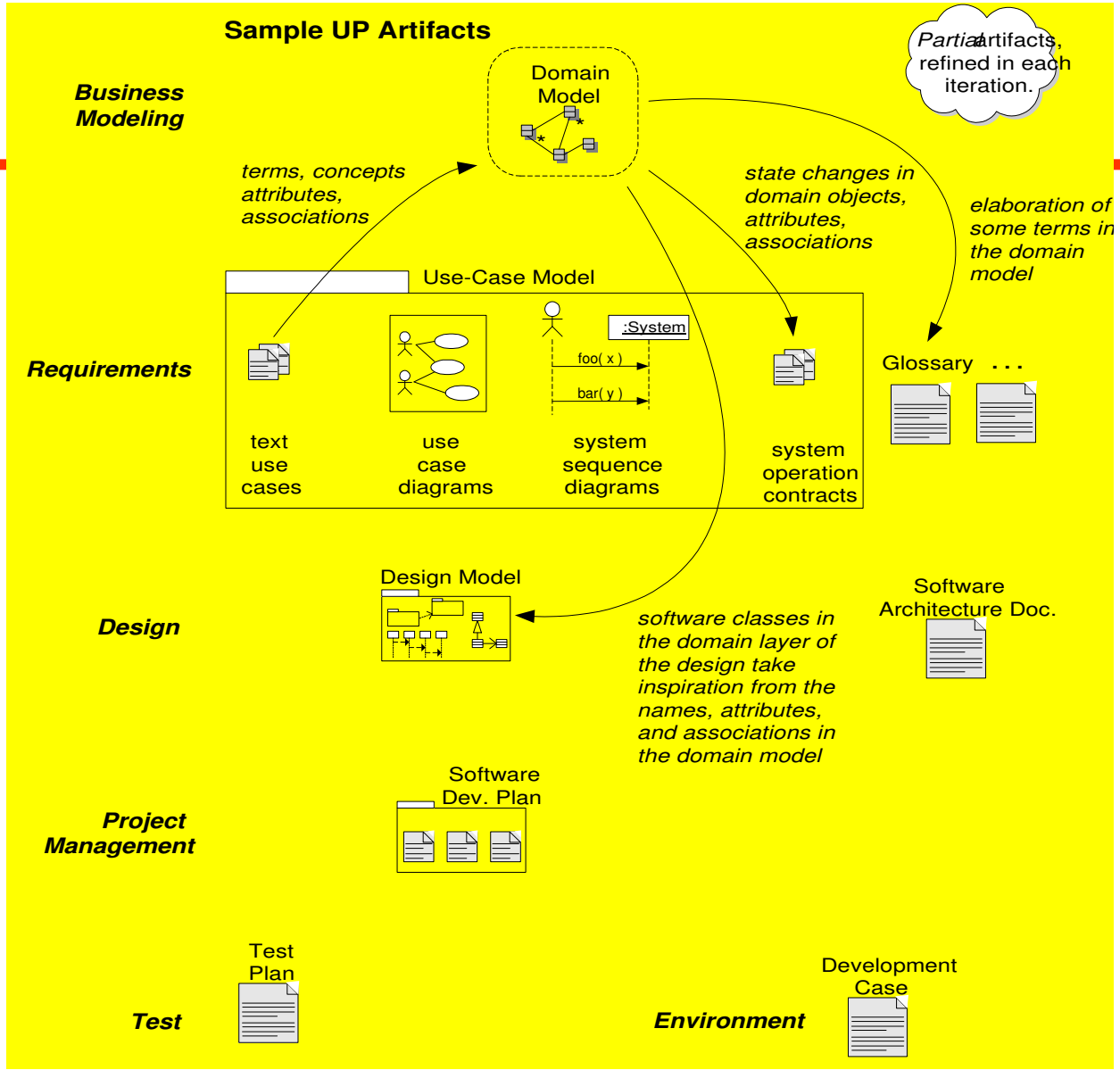


Modelo de Domínio de POS



Modelo de Domínio
Conclusão

- um DM capta as abstrações essenciais e informação requerida para entender o domínio no contexto dos requisitos correntes
- e ajuda as pessoas a entenderem os conceitos do domínio, terminologia e relações.



Based on C. Larman, 2002

Disciplina	Artefato	Incep. I1	Elab. E1..En	Const. C1..Cn	Trans. T1..T2
Business Modeling	**Domain Model**		**S**		
Requirements	Use-Case Model Vision Supplementary Specif Glossary	S S S S	R R R R		
Design	Design Model SW Architecture Doc Data Model		S S S	R R	
Implementation	Implementation Model		S	R	R
Project Manag.	SW Development Plan	S	R	R	R
Testing	Test Model		S	R	
Environment	Developmetn Case	S	R		

Referências

- Larman, C. (2002) *Applying UML and Patterns – An Introduction to Object Oriented Analysis and Design and the Unified Process*, Prentice-Hall Inc.
- Muller, P.A. (1997) *Instant UML*,Wrox Press Ltd.