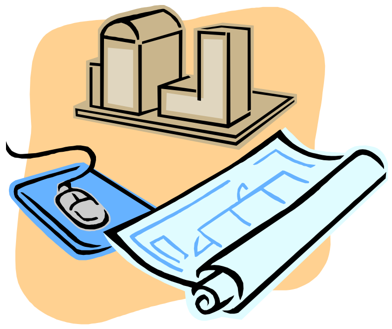
ISE

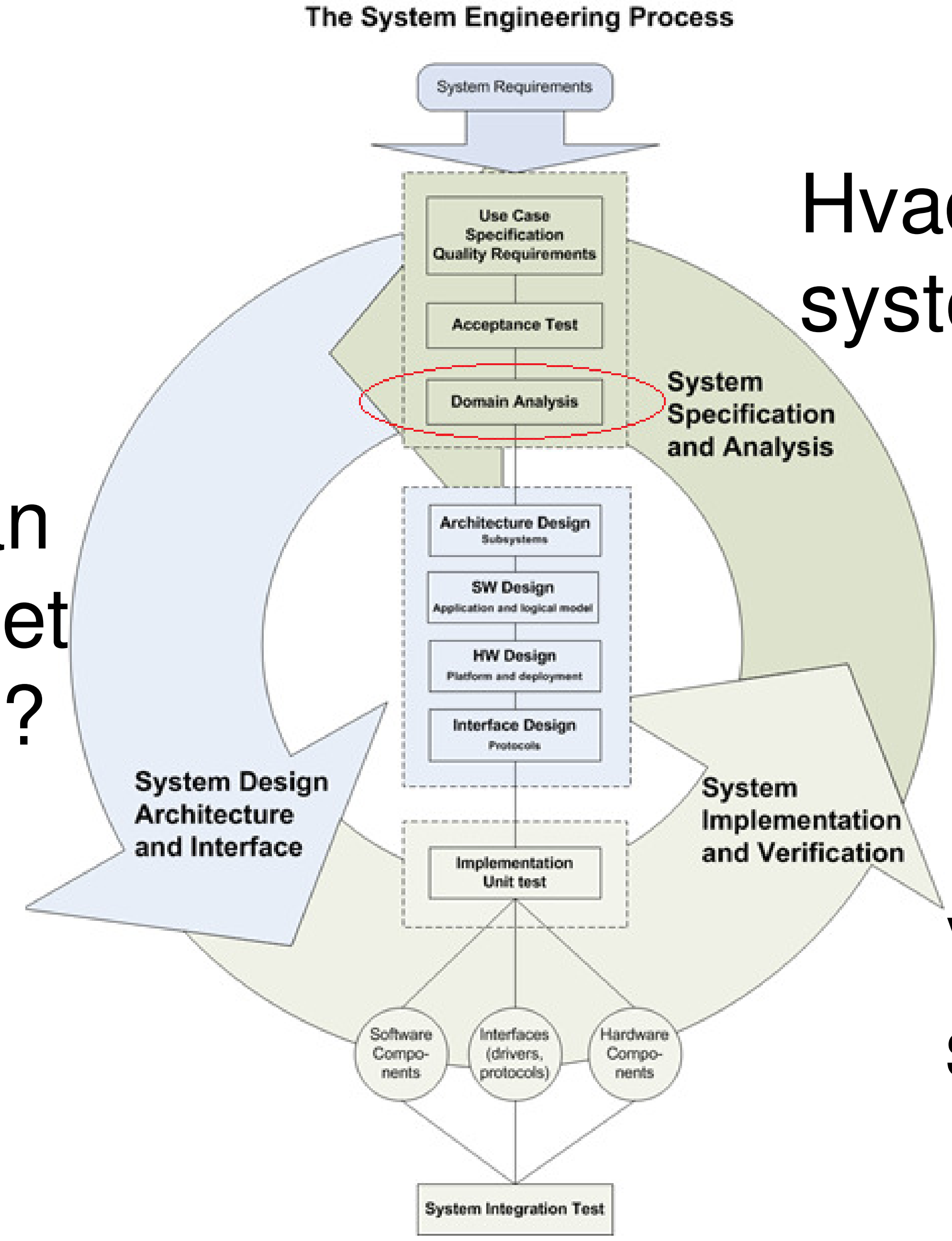
Domain Models



INGENIØRHØJSKOLEN I ÅRHUS

Lessons and topics

- ***System Specification, Test and Quality***
  - Specification
  - Use Cases
  - System Test
  - Quality Management
- ***SysML Diagrams***
  - SysML structure diagrams
  - SysML behavior diagrams
  - SysML State diagrams
- ***Process and Project***
  - Development Processes
  - Project Management
- ***System Design Architecture and Interfaces***
  - System Domain Analysis
  - System Application Model
  - System Design and Architecture
  - HW/SW Design
  - Interfaces


The System Engineering Process
System Requirements
Use Case Specification Quality Requirements
Acceptance Test
Domain Analysis
System Specification and Analysis
Architecture Design
Subsystems
SW Design
Application and logical model
HW Design
Platform and deployment
Interface Design
Protocols
System Design Architecture and Interface
Implementation Unit test
System Implementation and Verification
Software Compo-nents
Interfaces (drivers, protocols)
Hardware Compo-nents
System Integration Test

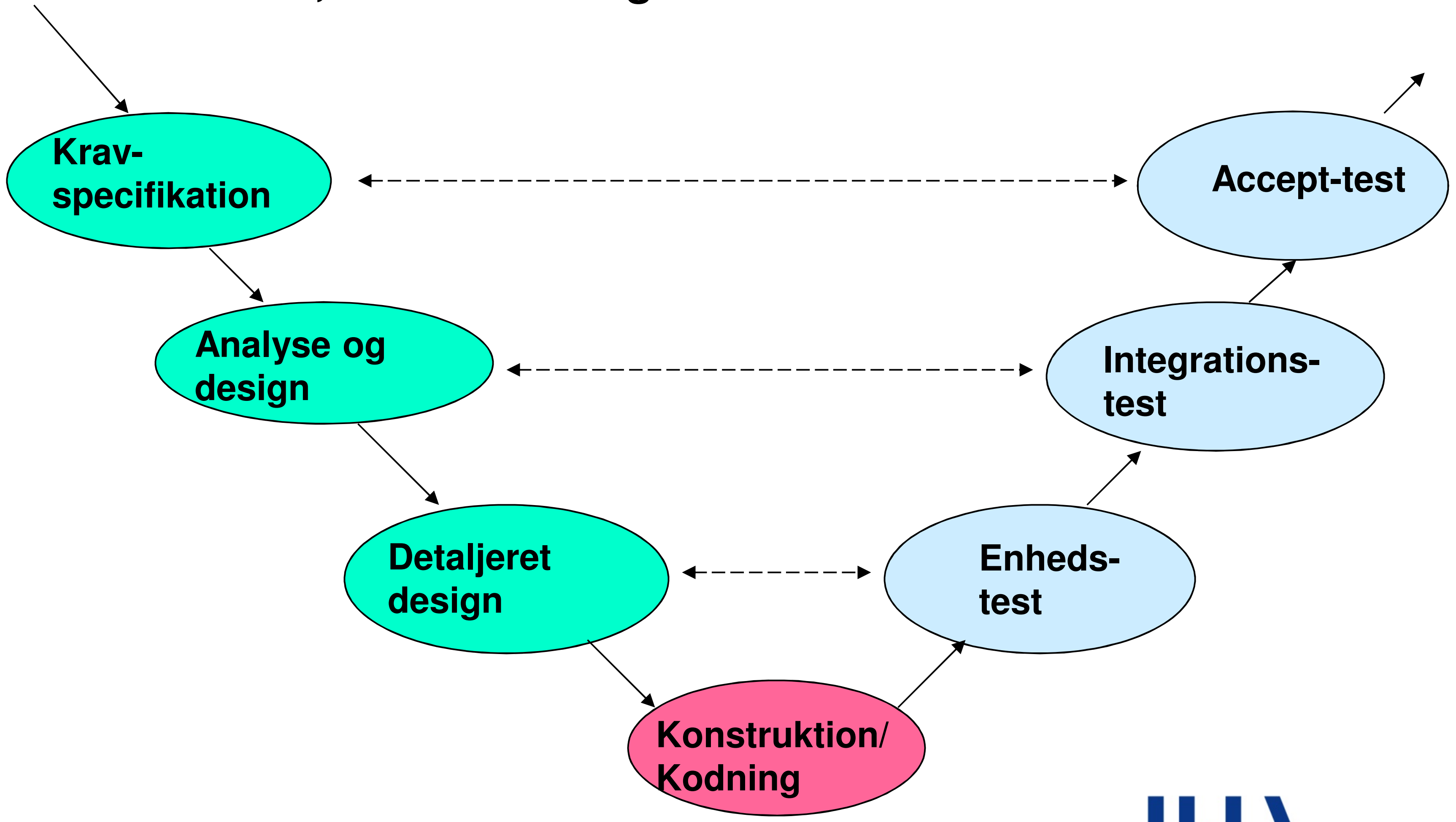
Hvad skal systemet gøre?

Hvordan systemet gør det?

Virker systemet?

V-model og systemudvikling

Udviklingsforløbet kan inddeles i faser på mange forskellige måder, men vil ofte ligne dette:



INGENIØRHØJSKOLEN I ÅRHUS

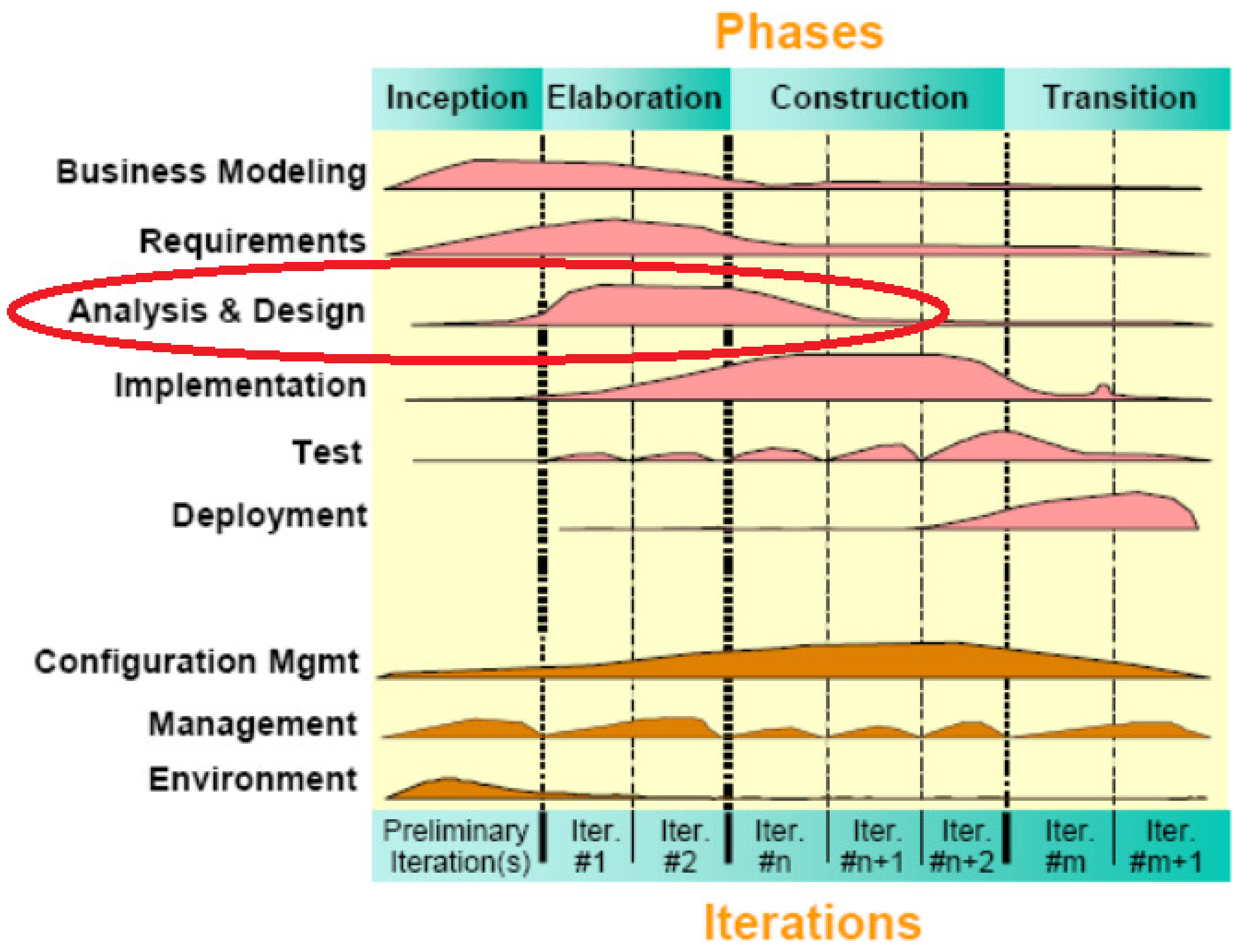
Domain Models

- Classic method for Object Oriented Analysis and Design for software development

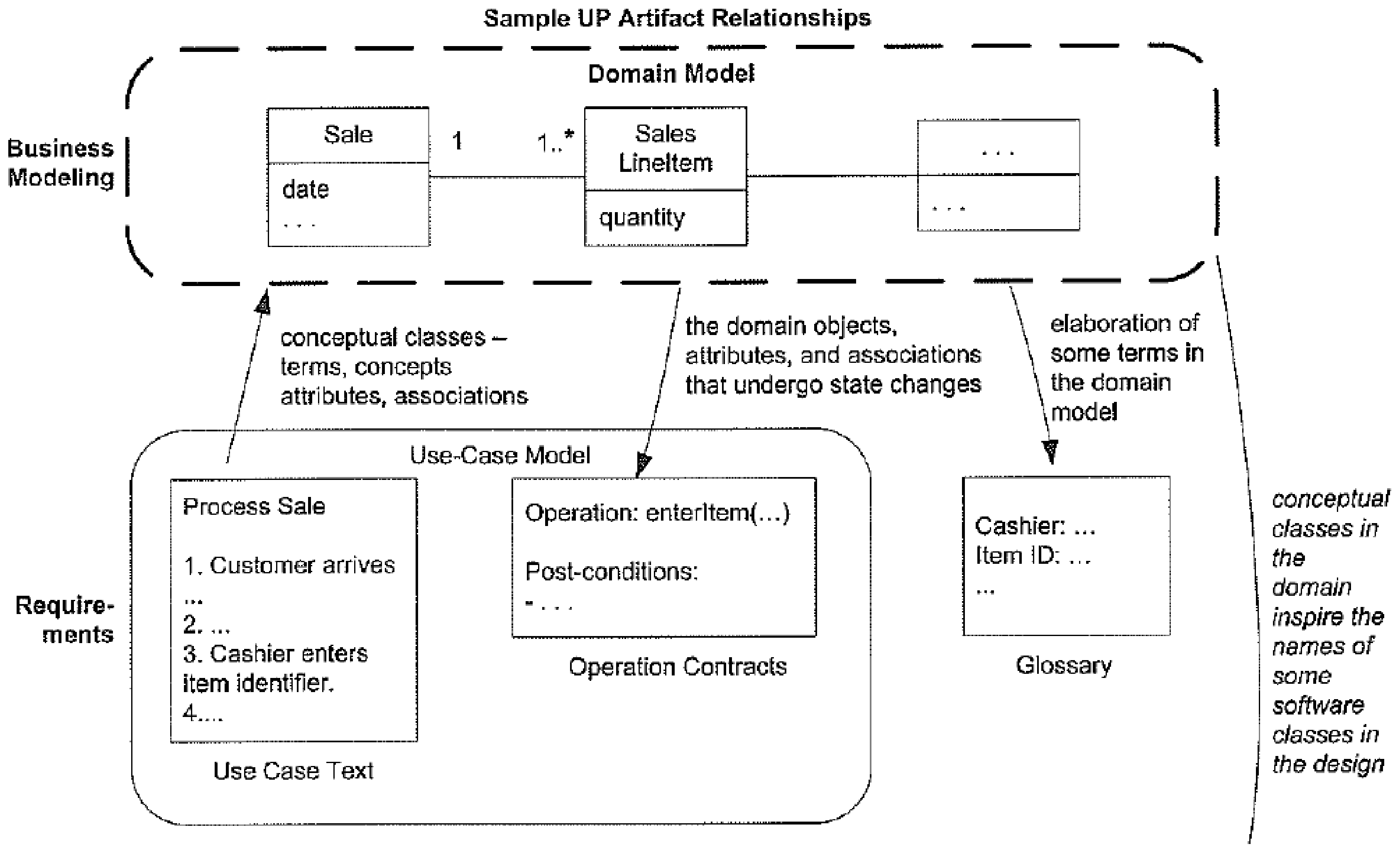
- The goal is to create a structural overview of the problem domain with focus on identifying objects (SysML = blocks)
- Same approach can be used for systems
  - SysML – block definition diagrams – ***system domain***

- Main objectives are:
  - Identify conceptual classes related to the current iteration
  - Create an initial domain model
  - Model appropriate attributes and associations

Unified Process – Elaboration Analysis & Design



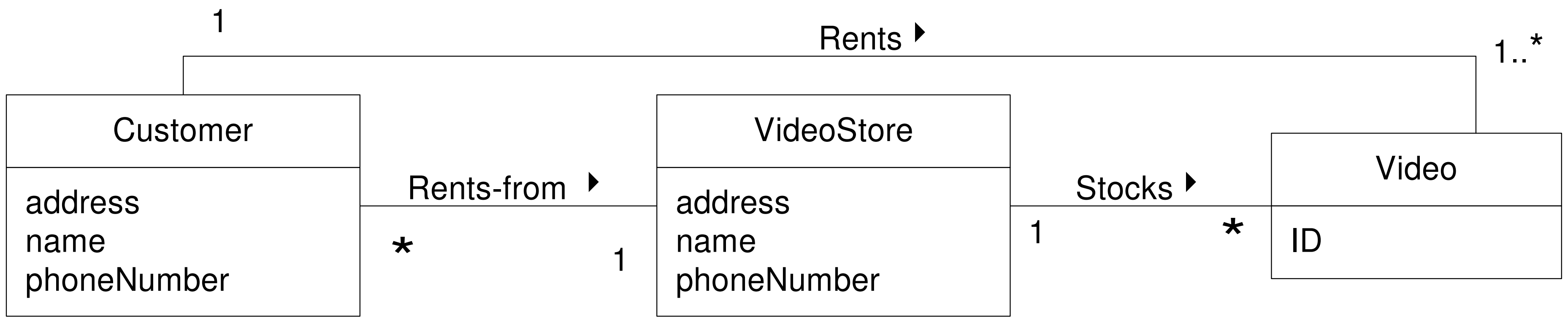
Domain Model and Requirements



Sample UP Artifact Relationships
Business Modeling
Domain Model
Sale
date
. . .
1
1..*
Sales LineItem
quantity
. . .
. . .
conceptual classes – terms, concepts attributes, associations
the domain objects, attributes, and associations that undergo state changes
elaboration of some terms in the domain model
Requirements
Use-Case Model
Process Sale
1. Customer arrives ...
2. ...
3. Cashier enters item identifier.
4....
Use Case Text
Operation: enterItem(…)
Post-conditions:
- . . .
Operation Contracts
Cashier: ...
Item ID: ...
...
Glossary
conceptual classes in the domain inspire the names of some software classes in the design

INGENIØRHØJSKOLEN I ÅRHUS

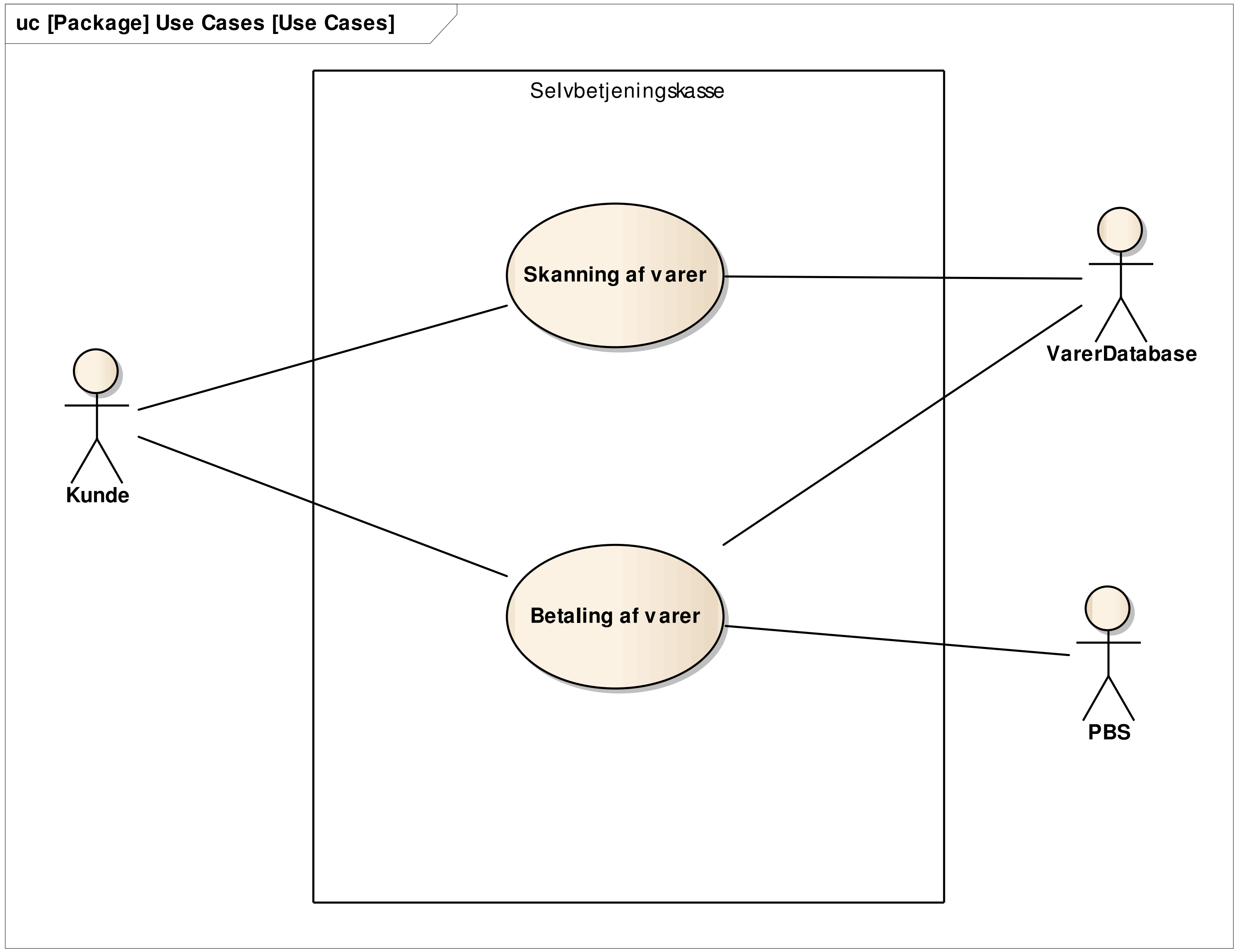
EXAMPLE: Partial Domain Model



Selvbetjeningskasse

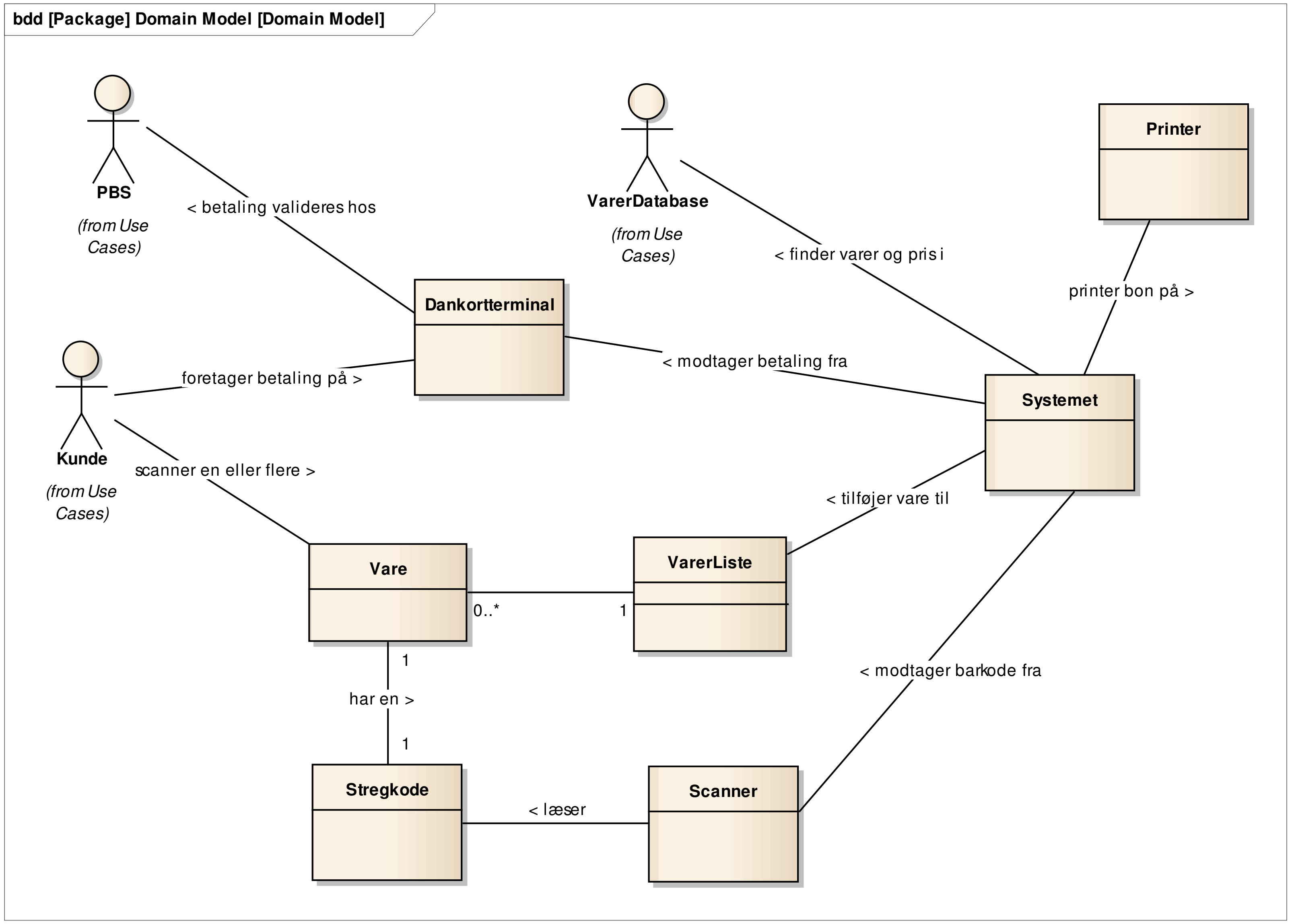
INGENIØRHØJSKOLEN I ÅRHUS

Selvbetjeningskasse – Use Cases



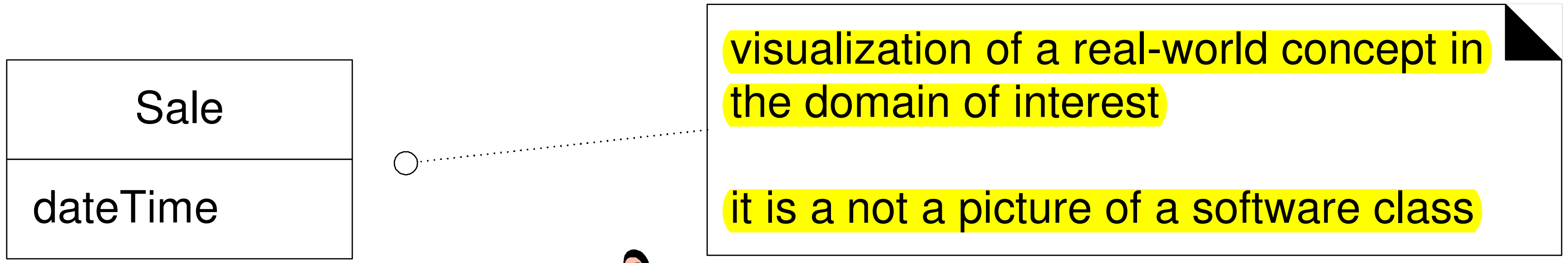
INGENIØRHØJSKOLEN I ÅRHUS

Domain model



A Domain Model

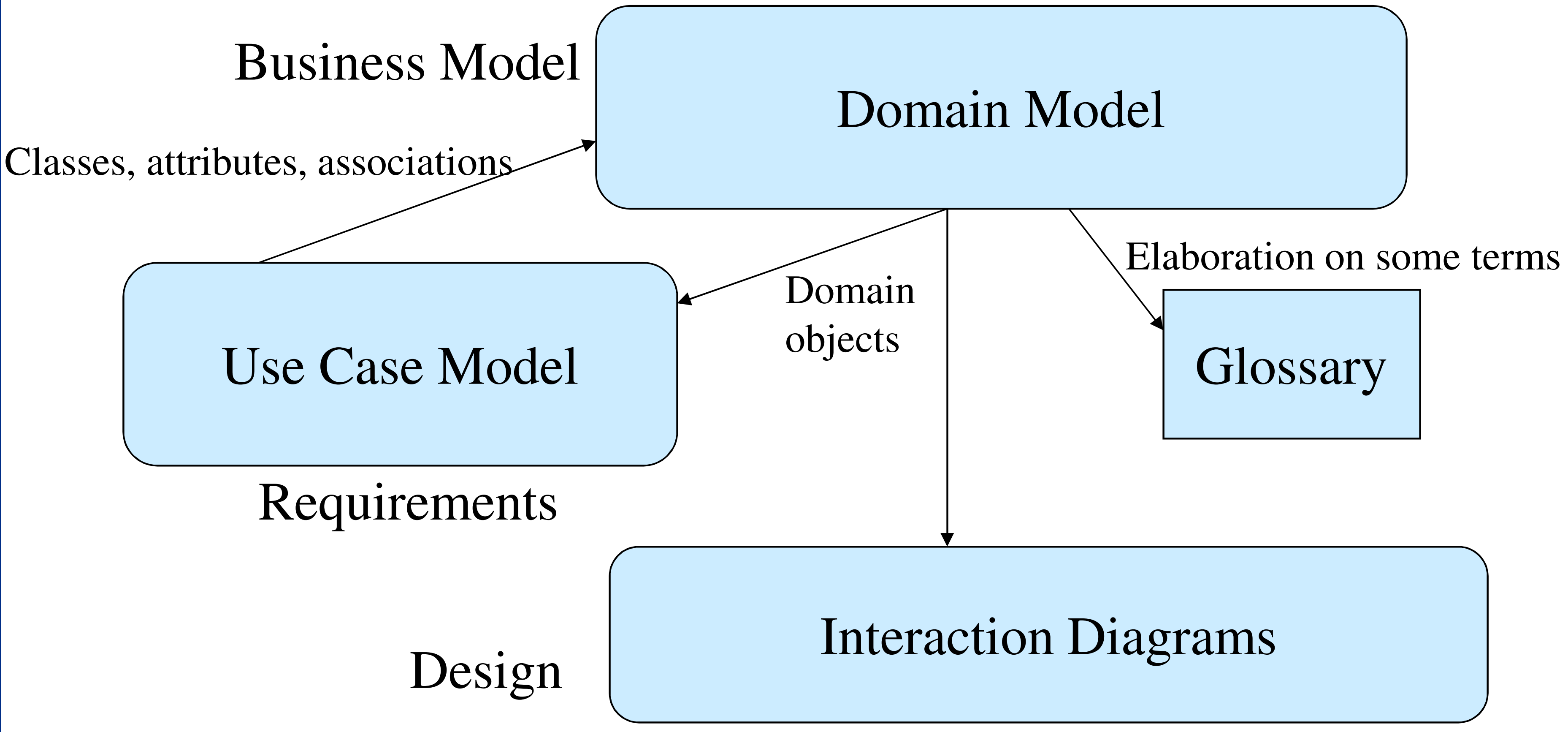
- illustrates meaningful conceptual classes in a problem domain.
- is a representation of real-world concepts, not software or hardware components.
- is NOT a set of diagrams describing software classes, or software objects and their responsibilities.



visualization of a real-world concept in the domain of interest

it is a not a picture of a software class

Domain Model Relationships



A Domain Model is the most important OO artifact

- Its development entails identifying a rich set of conceptual classes, and is at the heart of object oriented analysis.
- It is a visual representation of the decomposition of a domain into individual conceptual classes or objects.
- It is a visual dictionary of noteworthy abstractions.

Domain Model UML Notation

- Illustrated using a set of class diagrams for which no operations are defined.

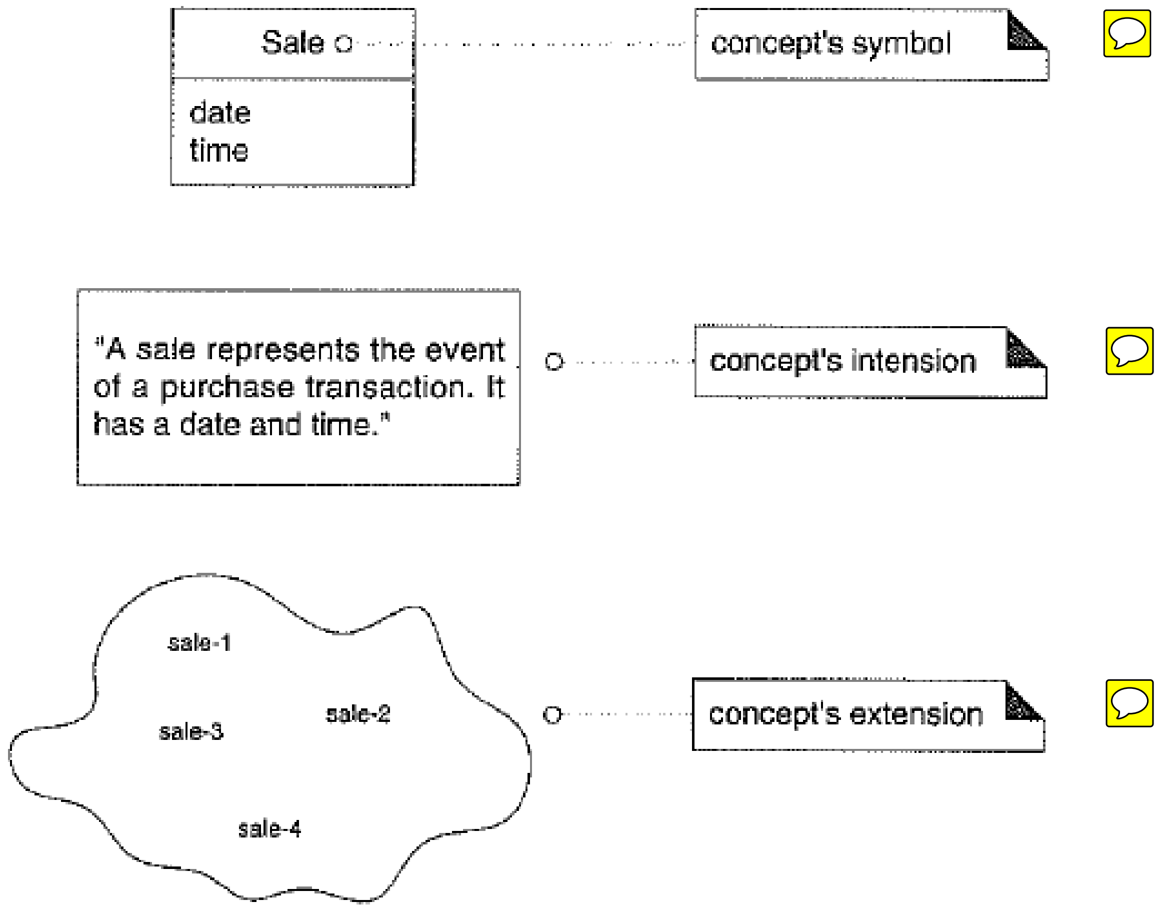
It may contain:

- Domain Objects or Conceptual Classes
- Associations between conceptual classes
- Attributes of conceptual classes

Think of Conceptual Classes in terms of:

- Symbols – words or images
- Intensions – its definition
- Extensions – the set of examples to which it applies
- Symbols and Intensions are the practical considerations when creating a domain model.

Symbol, intension, extension



INGENIØRHØJSKOLEN I ÅRHUS

Conceptual Class Identification:

- It is better to overspecify a domain with lots of fine-grained conceptual classes than it is to underspecify it.
- Discover classes up front rather than later.
- Unlike data modeling, it is valid to include concepts for which there are no attributes, or which have a purely behavioral role rather than an informational role.

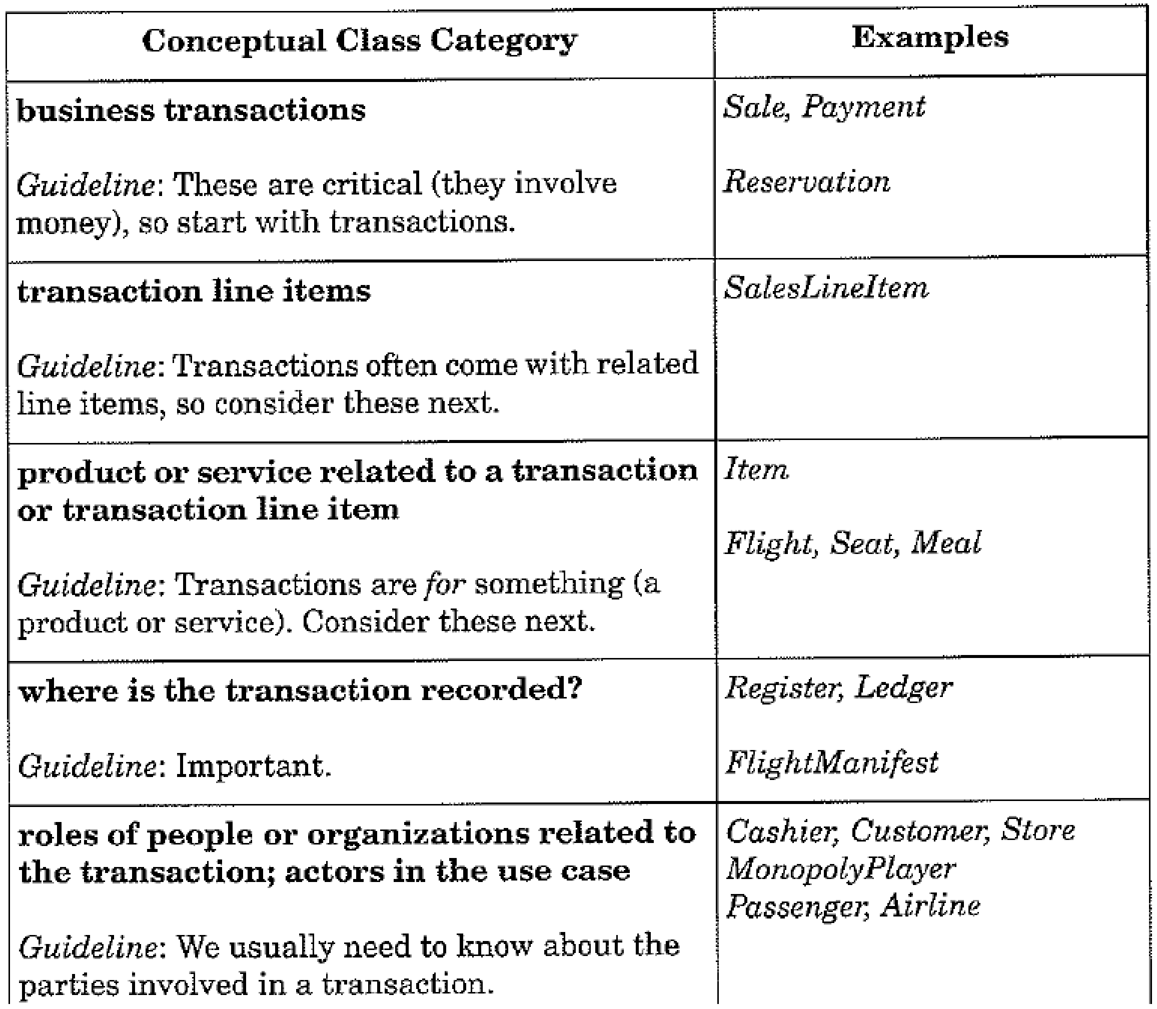
Identify Conceptual Classes by Category List:

Common Candidates for classes include:

Tangible objects, Descriptions, Roles, Places, Transactions, Containers, Systems, Abstract nouns, Rules, Organizations, Events, Processes, Written Materials, Catalogs, Records, Financial Instruments and Services

Conceptual Class Category List

- Compendium page 134 - 135



Conceptual Class Category	Examples
business transactions *Guideline*: These are critical (they involve money), so start with transactions.	*Sale, Payment* *Reservation*
transaction line items *Guideline*: Transactions often come with related line items, so consider these next.	*SalesLineItem*
product or service related to a transaction or transaction line item *Guideline*: Transactions are *for* something (a product or service). Consider these next.	*Item* *Flight, Seat, Meal*
where is the transaction recorded? *Guideline*: Important.	*Register, Ledger* *FlightManifest*
roles of people or organizations related to the transaction; actors in the use case *Guideline*: We usually need to know about the parties involved in a transaction.	*Cashier, Customer, Store* *MonopolyPlayer* *Passenger, Airline*

Identify Conceptual Classes by Noun Phrase:

- Identify Nouns and Noun Phrases in textual descriptions of the domain.
- Fully dressed Use Cases are good for this type of linguistic analysis.

It's not strictly a mechanical process:

- Words may be ambiguous
- Different phrases may represent the same concepts.

Find klasser - fra tekstuelle beskrivelser

Udsnit af systembeskrivelse

… Undervejs på *turen* kan *Tog Kontrol Centeret* ændre på *turplanen*. Disse ændringer bliver også sendt til *toget* via *radiokommunikation*.

Togets aktuelle *position* registreres når det passerer en *balise*. Baliserne er placeret langs *skinnelegemet*. Baliserne har et entydigt *nummer* og sammenhængen mellem dette nummer og balisens placering er kendt i Tog Kontrol Centeret, denne *sammenhæng* er også beskrevet i turplanen således at toget udfra balisens nummer ved hvor det er.

Når toget passerer en balise sender den sin position til Tog Kontrol Centret via radiokomunikation. Togets position består af togets nummer og *segmentets* nummer.

Kandidater til klasser

- tur
- Tog Kontrol Center
- turplan
- tog
- radiokommunikation
- position
- balise
- skinnelegeme
- nummer
- sammenhæng
- segment

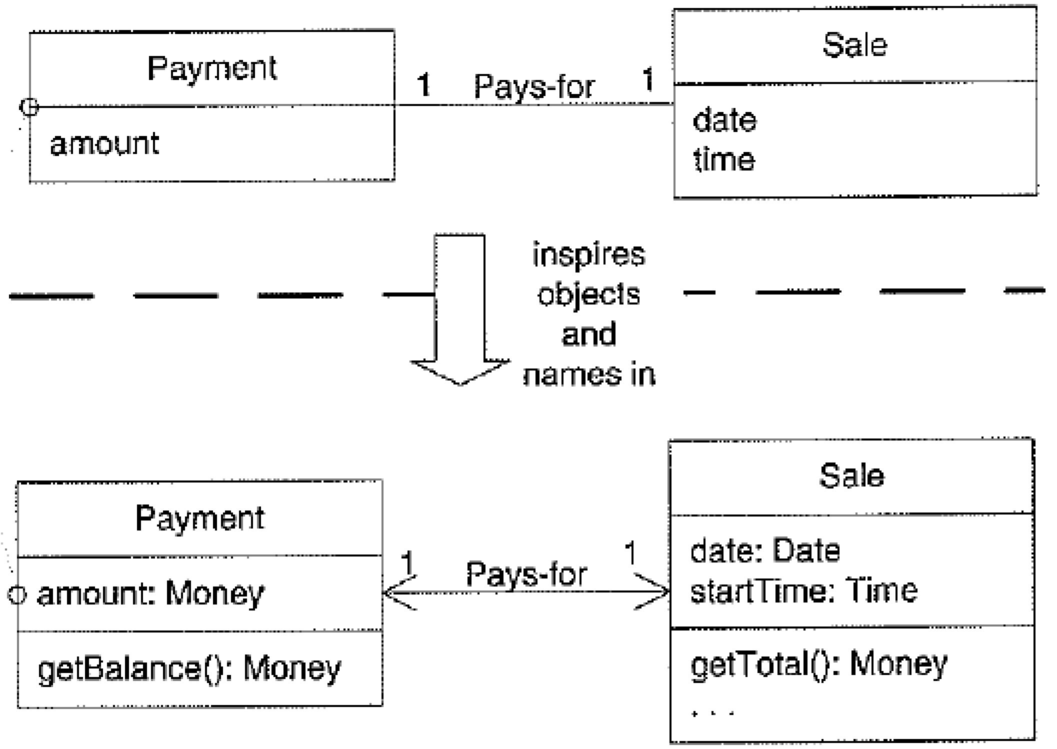
Domain vs. Application Design Model

A Payment in the Domain Model is a concept, but a Payment in the Design Model is a software class. They are not the same thing, but the former *inspired* the naming and definition of the latter.

This reduces the representational gap.

This is one of the big ideas in object technology.

UP Domain Model
Stakeholder's view of the noteworthy concepts in the domain.



UP Design Model
The object-oriented developer has taken inspiration from the real world domain in creating software classes.

Steps to create a Domain Model

1. Identify Candidate Conceptual classes
    1. Use Specification (Use Cases)
2. Draw them in a Domain Model
3. Add associations necessary to record the relationships that must be retained
4. Add attributes necessary for information to be preserved

A Common Mistake - Classes as Attributes

- Rule: If we do not think of a thing as a number or text in the real world, then it is probably a conceptual class.
- If it takes up space, then it is likely a conceptual class.

Examples:

- A Store is not an attribute of a Sale
- A Destination is not an attribute of a flight

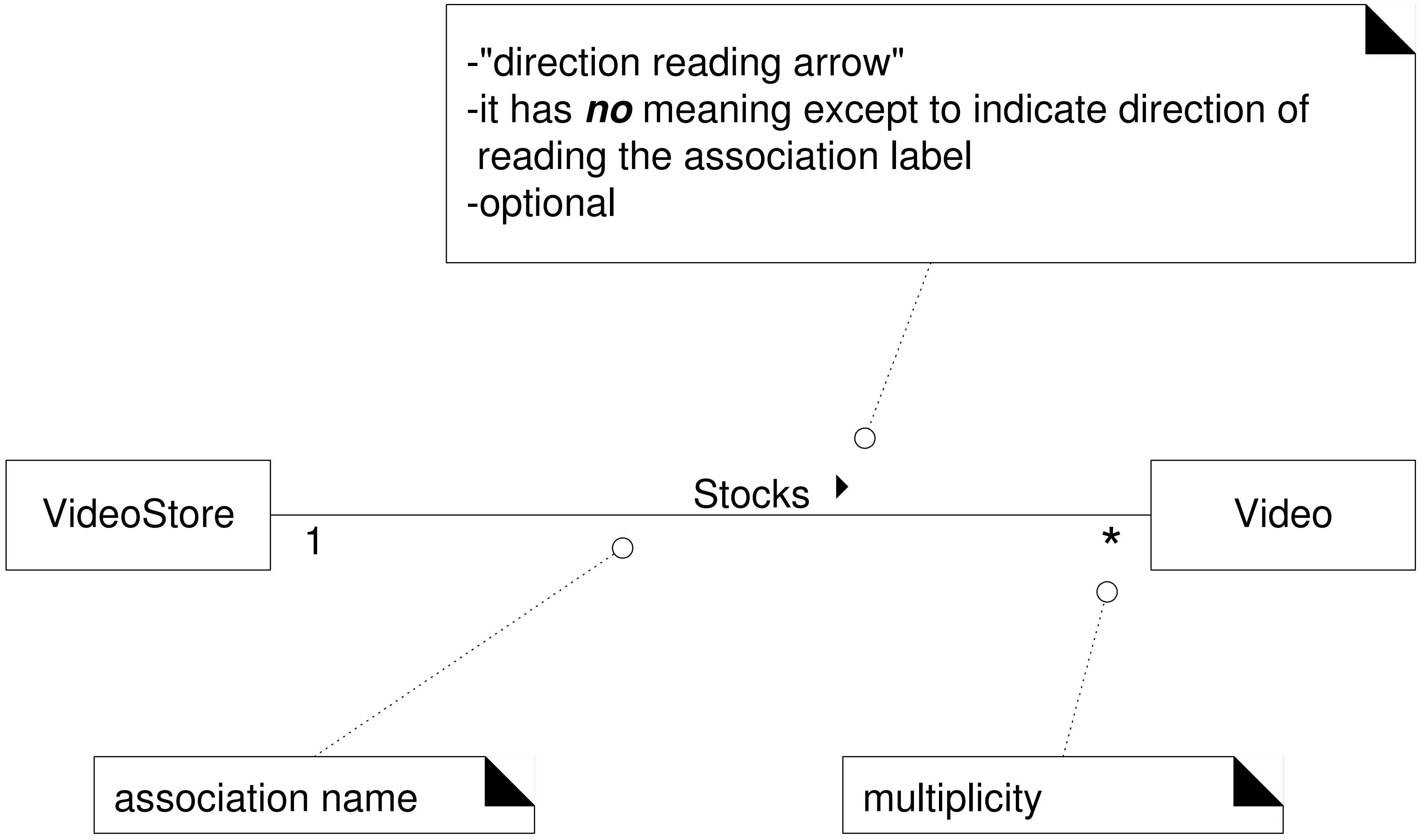
UML Notation: Multiple Perspectives

- UML/SysML describes raw diagram types, such as class and block diagrams. It does not impose a specific method or process.
- UP, the Unified Process, applies raw UML to defined methodology models.

UML can be used for 3 different perspectives:

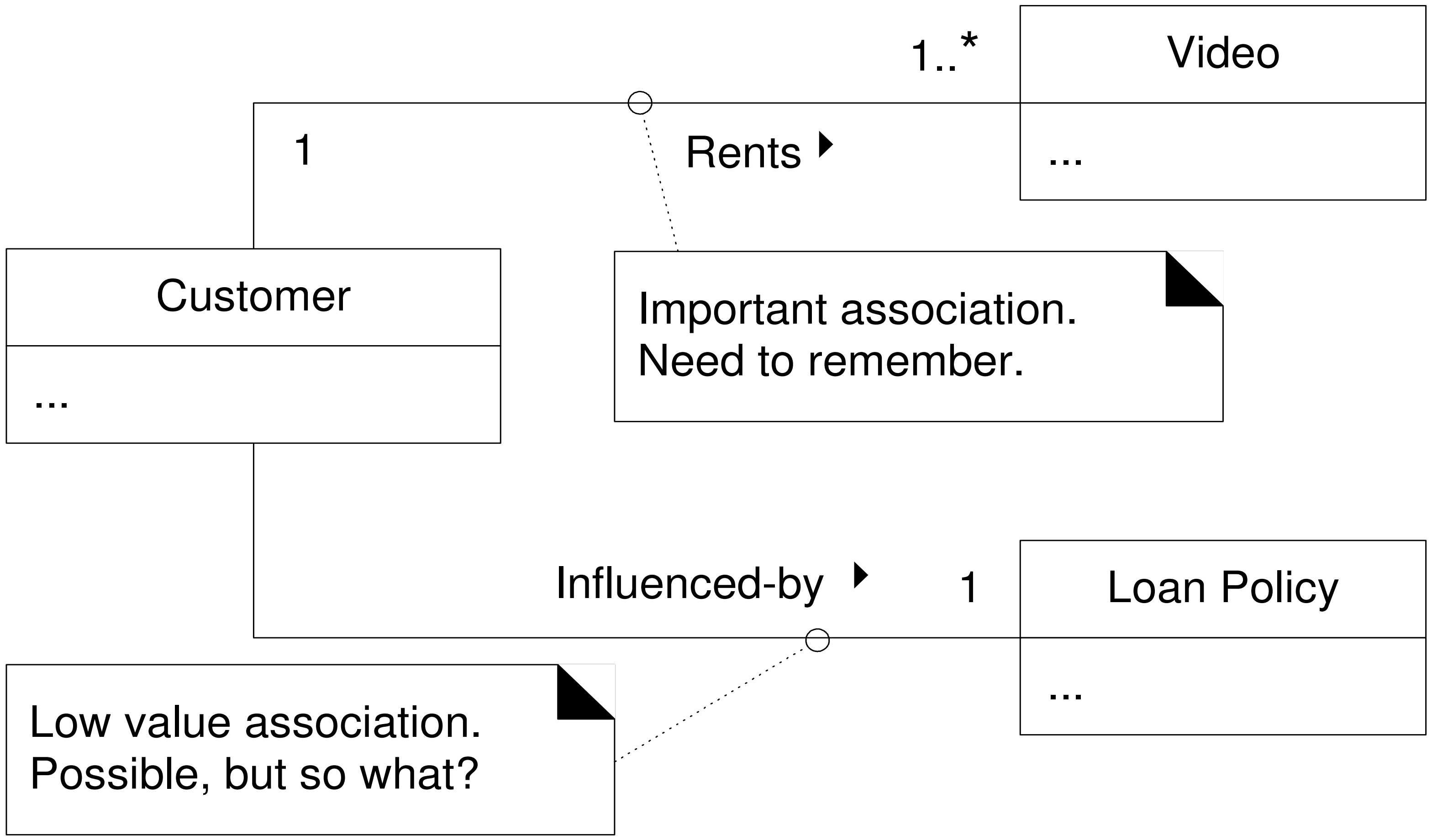
- Essential – describe the real world (+SysML)
- Specifications – software abstractions, such as components and their interfaces (+SysML)
- Implementation – specific language (Java, C#, C++)

UML: Associations


-"direction reading arrow"
-it has ***no*** meaning except to indicate direction of reading the association label
-optional
VideoStore
Stocks ▸
Video
1
*
association name
multiplicity

GUIDELINES: Associations

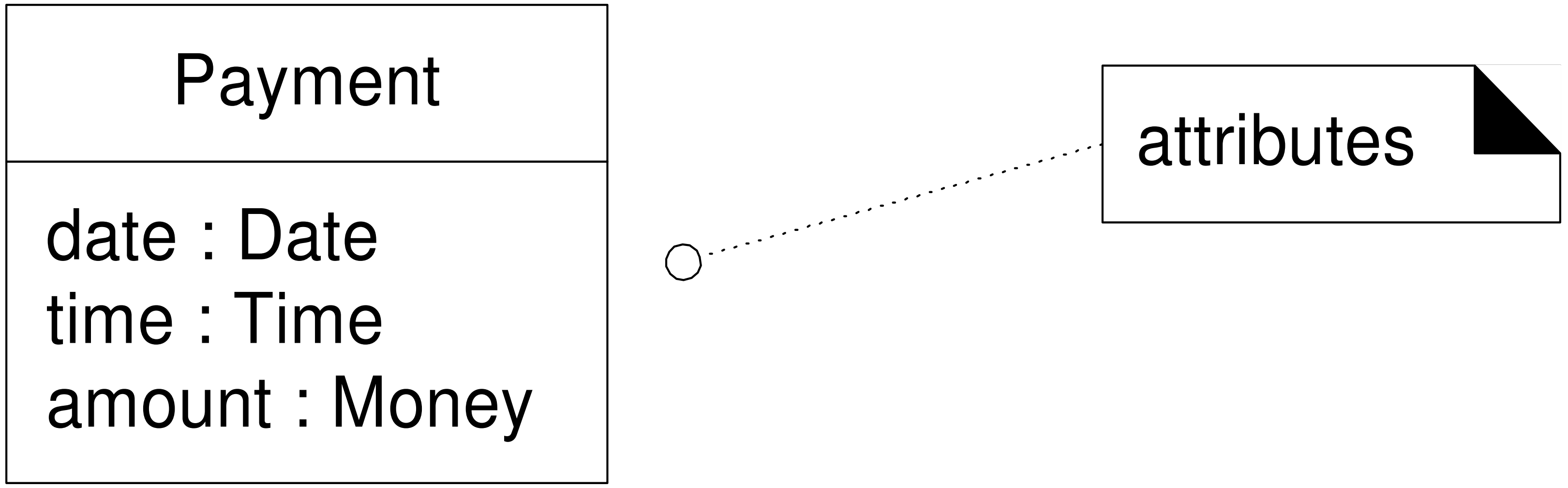
- Only add associations for *noteworthy* relationships. Literally, those for which making a "note" is worthy or business motivated.



UML and GUIDELINES: Attributes

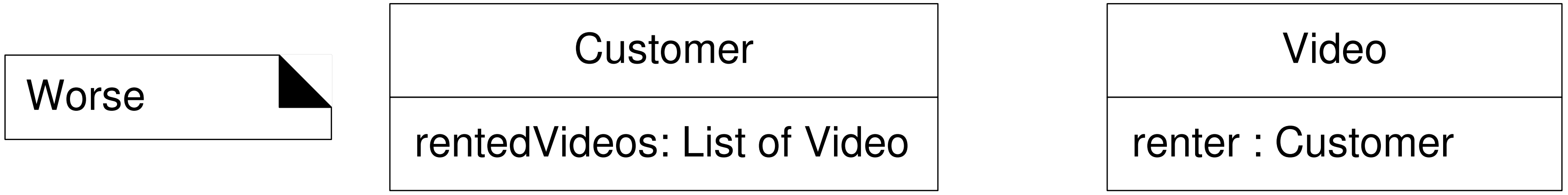
- Show only "simple" relatively primitive types as attributes.

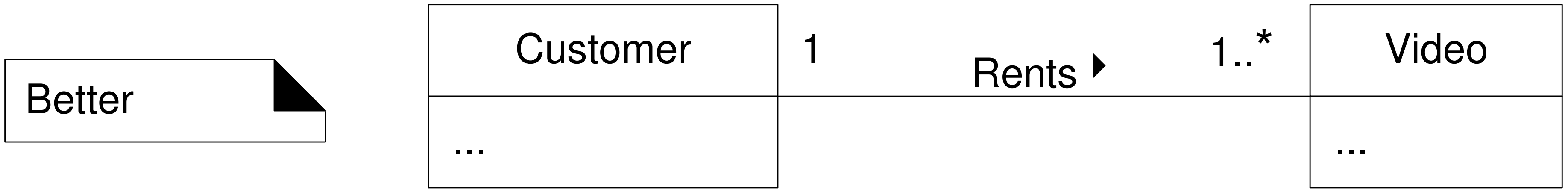
- Connections to other concepts are to be represented as associations, not attributes.



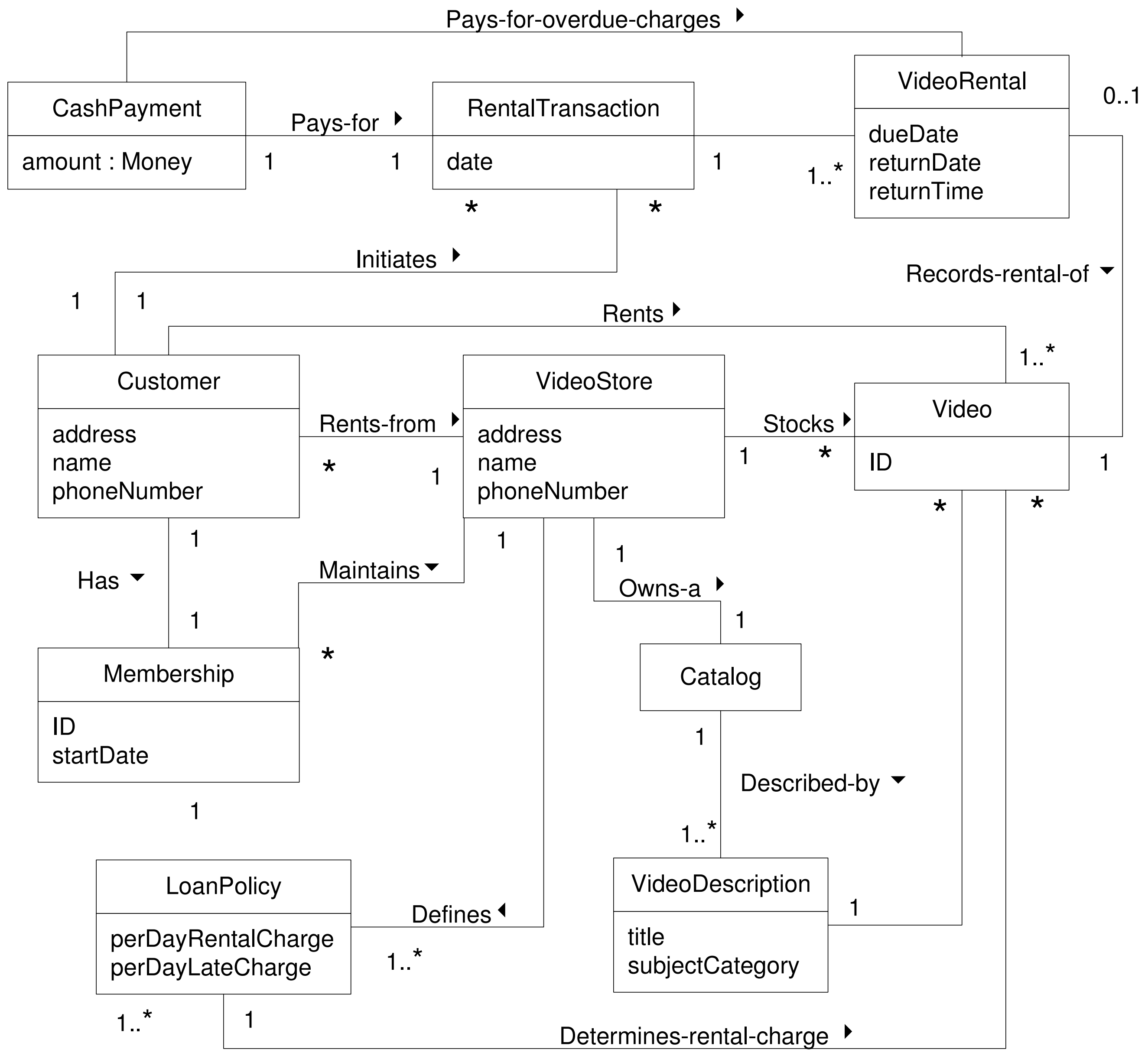
GUIDELINES: Attributes

- Why??





EXAMPLE: Domain Model



Notice how this can be viewed as a "visual dictionary." It *illustrates* concepts, words, things in a domain.

INGENIØRHØJSKOLEN I ÅRHUS