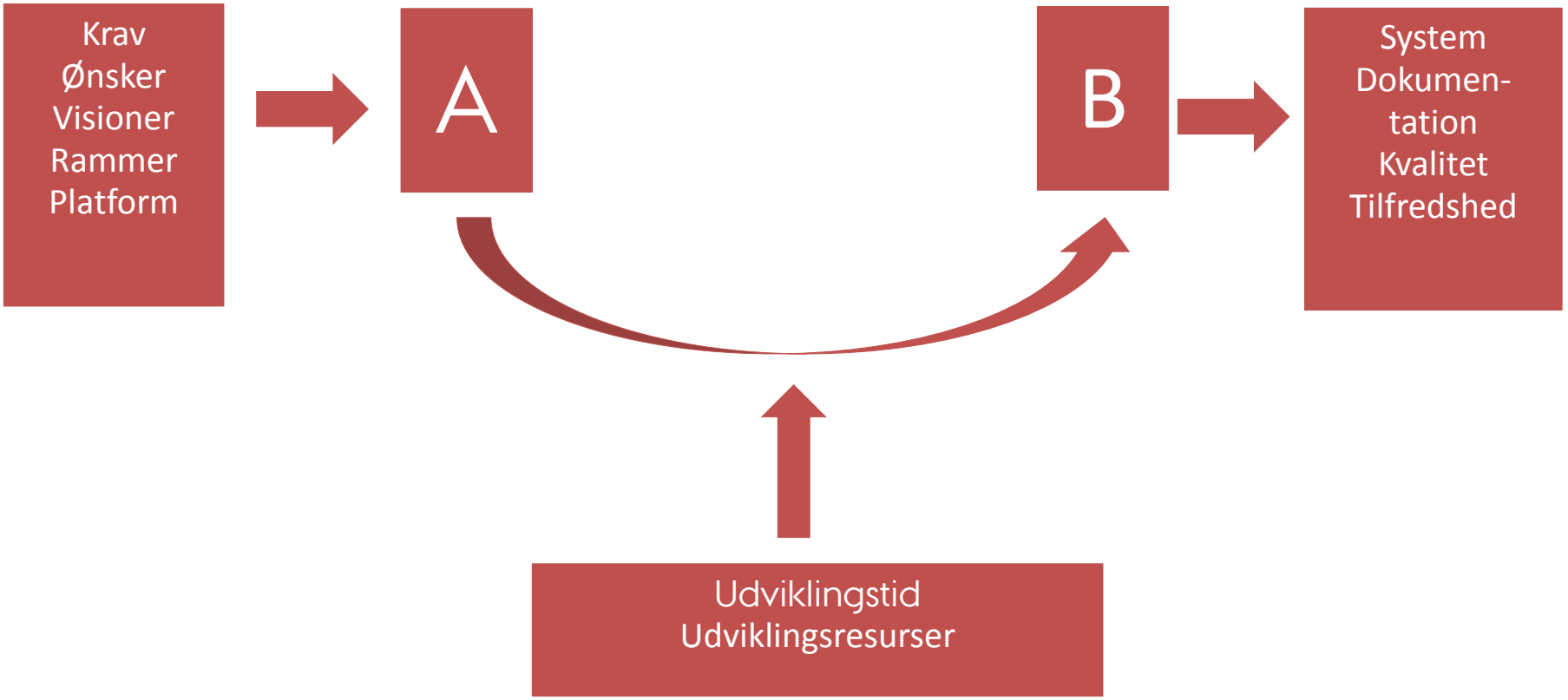
System Domain Analysis and Domain Models

Introduction to Systems Engineering
I2ISE

Introduction

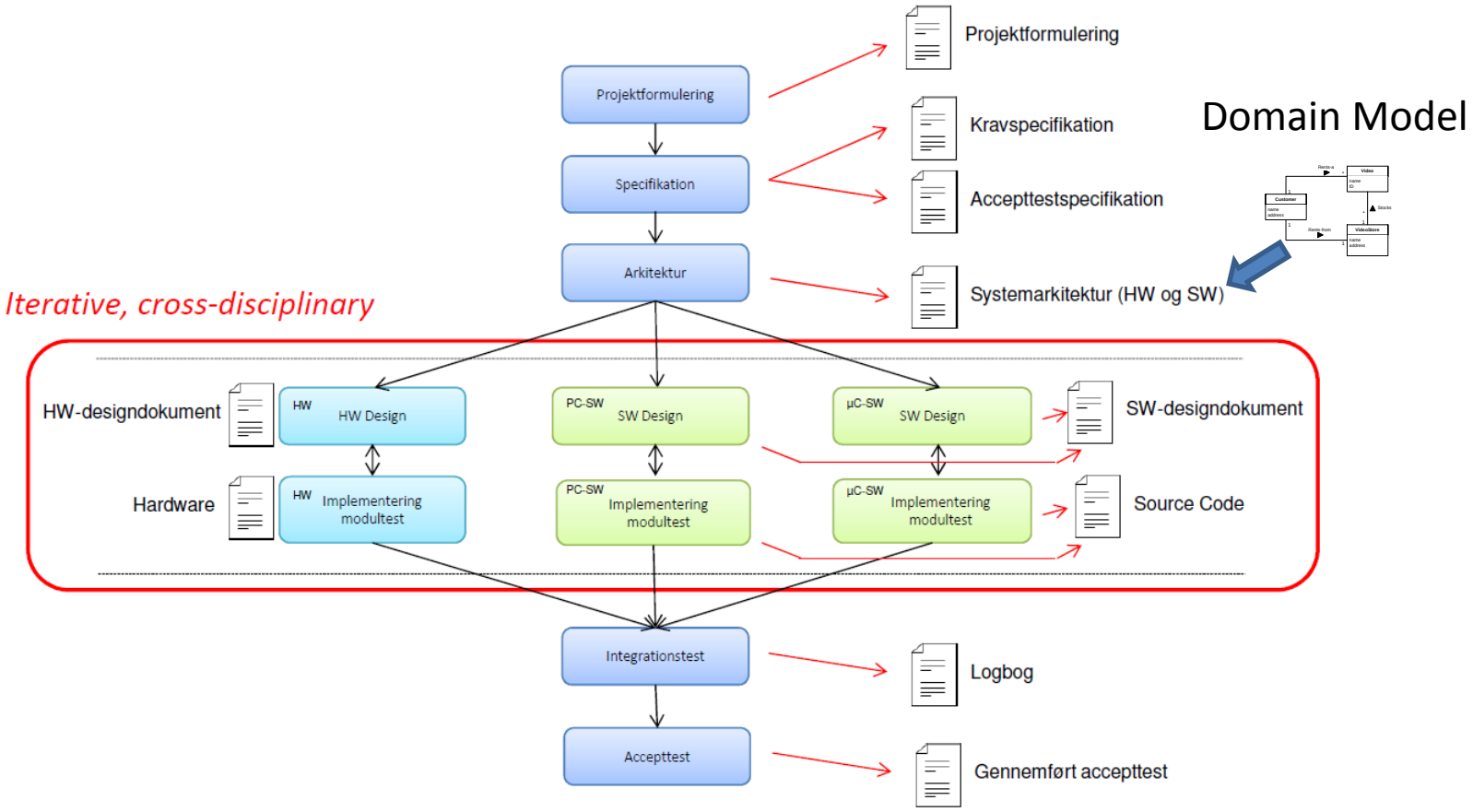
- What is *System Domain Analysis*?
- What is a *Domain Model?*
- Why create the Domain Model?
- How to create a Domain Model?

Systems engineering – skematisk set



The DM's place in the artefacts

The ASE Process

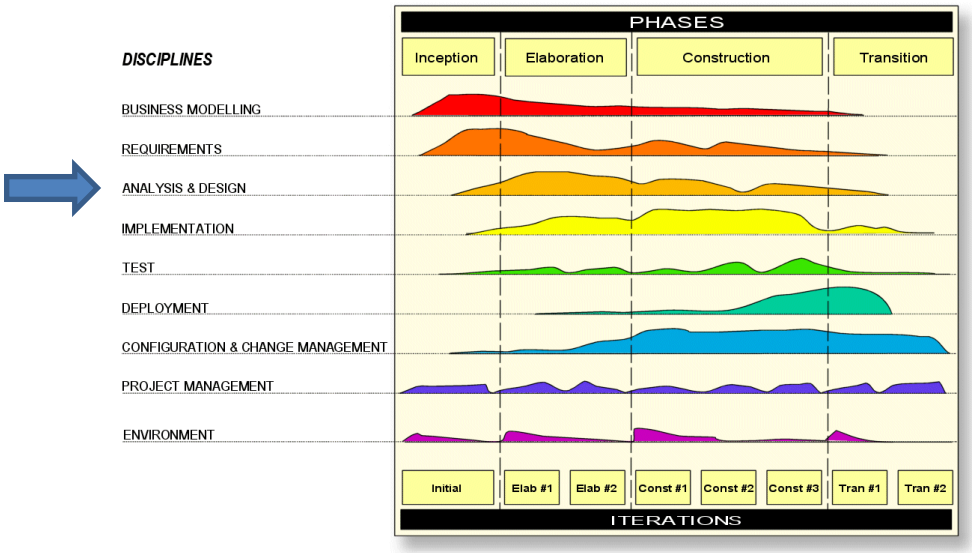


What is System Domain Analysis?

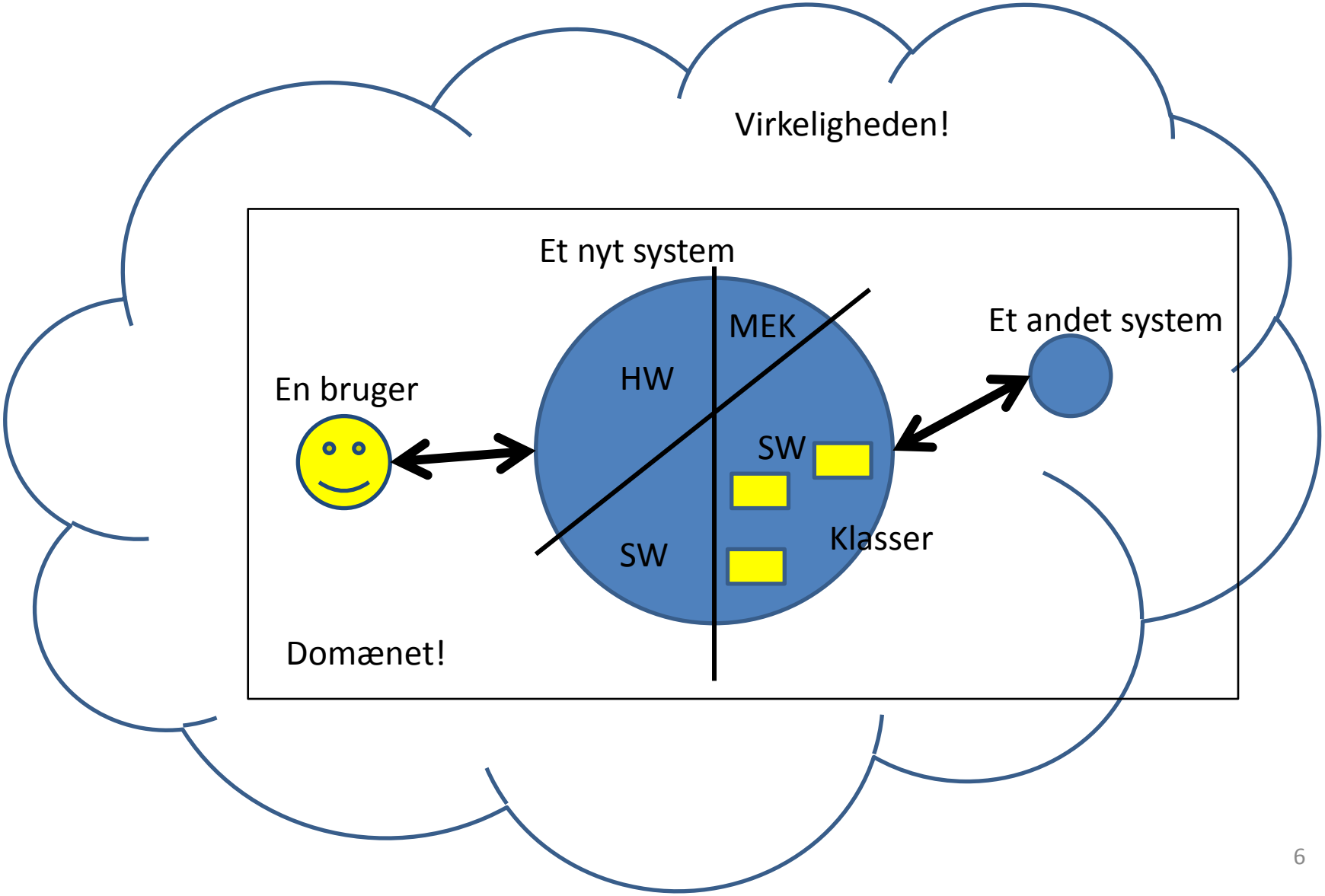
- *System Domain Analysis* (SDA) is an activity to analyse the system domain in order to find domain-specific concepts

- SDA is conducted between requirements and implementation (design)

- Prime artefact of SDA:
  The *Domain Model*



Virkeligheden og systemet



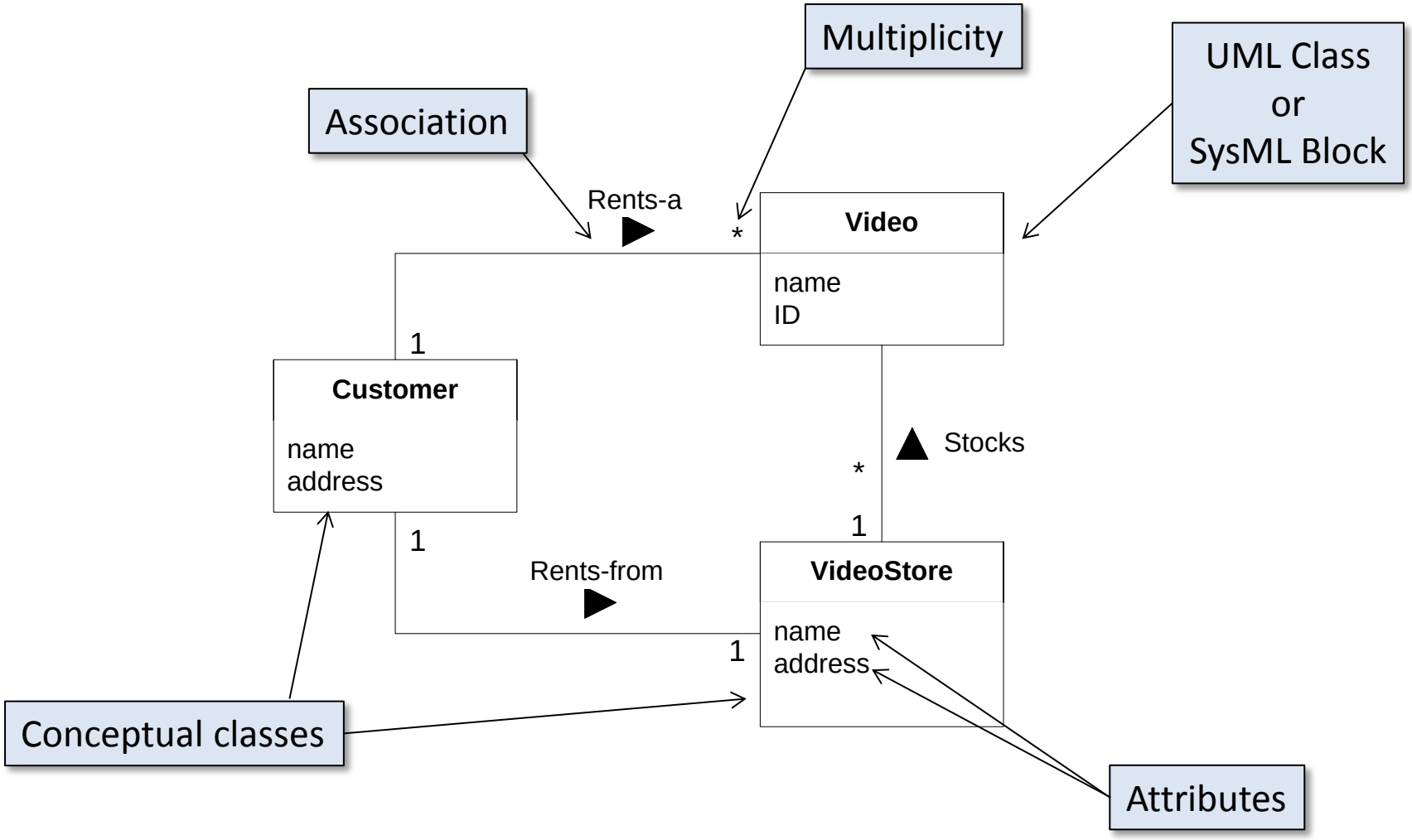
What is a Domain Model (and what is it not)?

- The Domain Model is an illustration of ”noteworthy concepts” in the system domain.
  - The concepts, their associations, multiplicity and attributes
  - Not responsibilities and operations!

- The Domain Model shows real-world concepts, *not* SW or HW entities
  - ”**`Bus`**”, ”**`Payment`**”, ”**`ATM`**” — **OK** – noteworthy concepts
  - ”**`SalesDatabase`**”, ”**`string`**” — **FAIL** – *software* entities

- The DM is a visual dictionary drawn as a UML class diagram
  - Everybody agree upon the names in the model
  - A ”new guy” can quickly pick up on terminology

Domain Model: Example

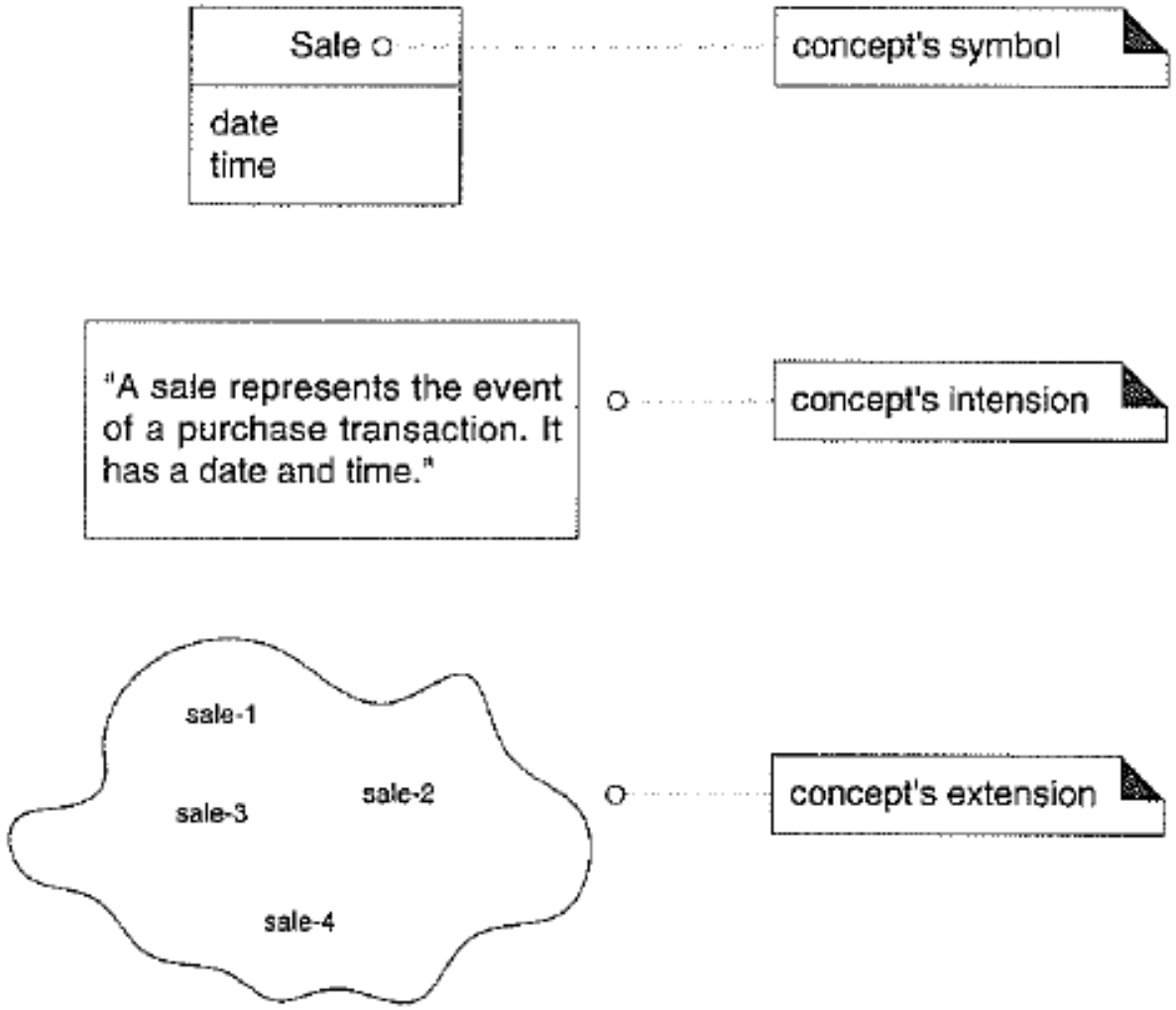


What is a conceptual class

- A conceptual class as shown on a Domain Model is an *idea*, a *thing* or an *object*

- Can also be defined by the class' *symbol*, *intension* and *extension*
  - Symbol: The word(s) or images representing the conceptual class
  - Intension: The definition of the conceptual class
  - Extension: The set of examples to which the conceptual class applies

Symbol, intension, extension

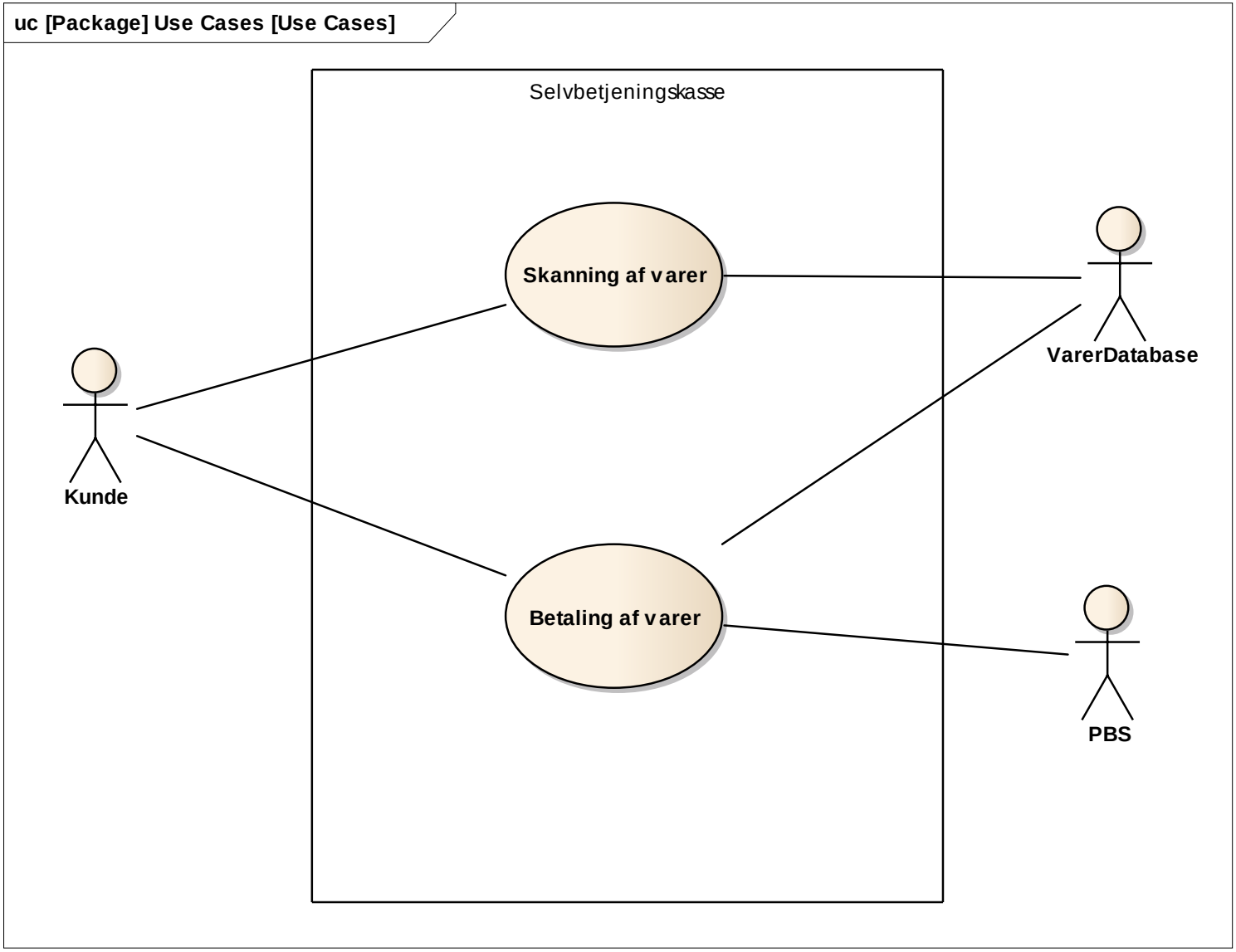


Why create a Domain Model?

- Doing SDA and creating the Domain Model helps to identify key concepts and things to investigate
- The Domain Model aids the very hard step from requirements to design
  - The first step from ”what” to ”how”
- The Domain Model lowers the ”representational gap” between domain and implementation

Selvbetjeningskasse

Selvbetjeningskasse – Use Cases



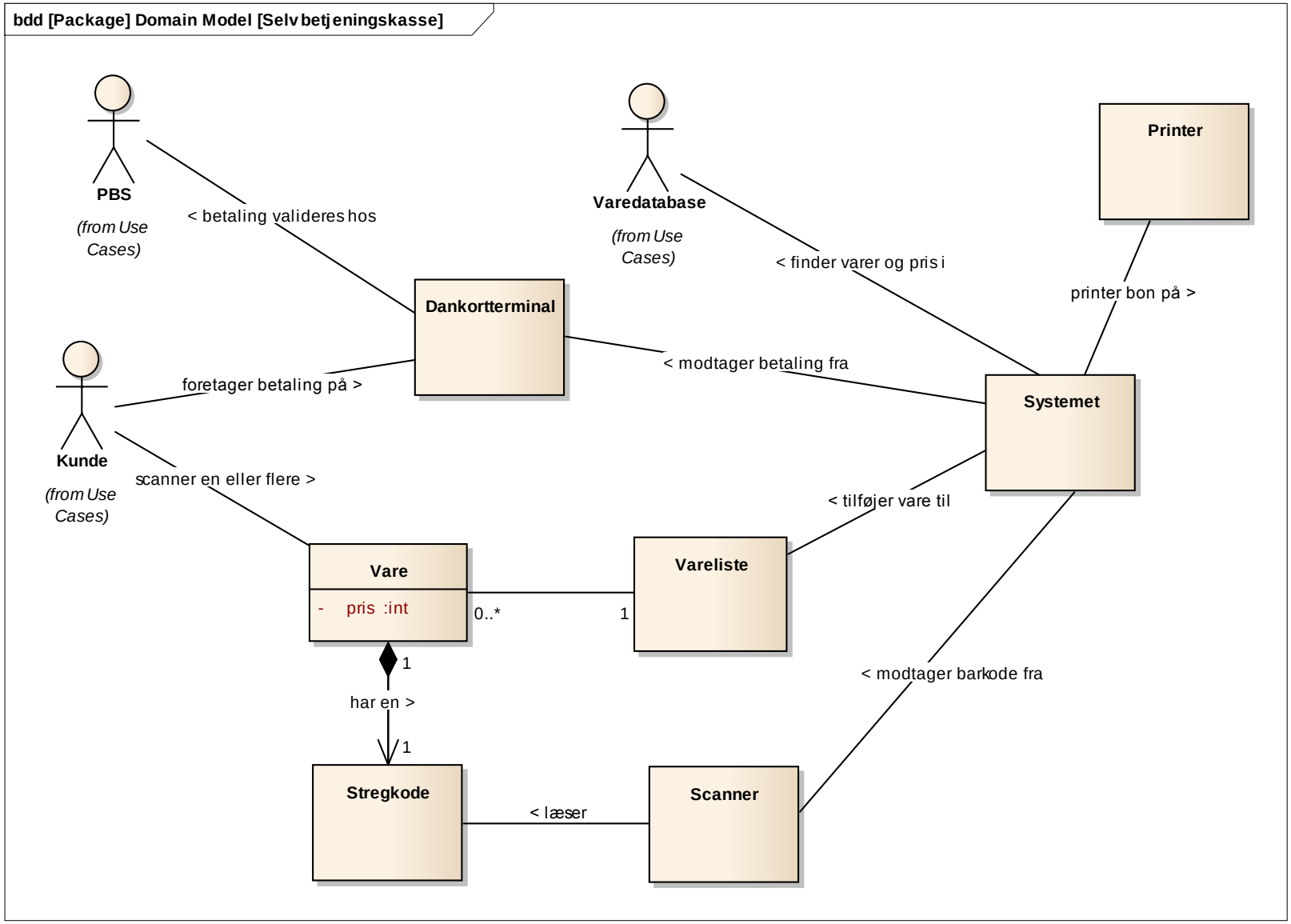
Scanning af Vare (Hovedscenarie)

1. Selvbetjeningskassen anmoder kunden om at skanne vare
2. Kunden placerer vare foran skanner
3. Systemet skanner varens stregkode
4. Systemet finder varens pris i varedatabasen
5. Vare med pris tilføjes til en vareliste
6. Kunden lægger vare i pose på bordet ved siden af skanner
7. Punkterne 1-6 gentages indtil alle varer er skannet
8. Kunden vælger afslut

Betaling af Vare (Hovedscenarie)

1. Kunden vælger betal med dankort på beløbet
2. Kunden indsætter kort i dankortterminalen
3. System viser det totale beløb og anmoder om pinkode
4. Kunden indtaster pinkode
5. Kort og pinkode valideres mod PBS
6. Printer udskriver bon med vareliste

Domain model



How to create Domain Models

- Creating a Domain Model is easy – creating a *useful* one is hard!

- Three steps to follow:

Step 1: Find the *conceptual classes* in the domain

Step 2: Draw the classes in a UML class diagram (or SysML Block Diagram)

Step 3: Identify associations and attributes between conceptual classes

Creating a Domain Model, step 1: Find the conceptual classes

- The *conceptual classes* in a Domain Model represent concepts which are meaningful in the problem domain
  - Examples: `Gate`, `Flower`, `Dog`, `Customer`, `Sale`, `Payment`, `Transaction`, `Pressure`, `FlightDescription`, …
- The conceptual classes in the Domain Model are *bounded* by the set of requirements (UCs) for the current iteration
  - Do not include stuff that are not related to the requirements for the current iteration!
- …that said, it is better to over-specify the Domain Model than to under-specify it.

Finding conceptual classes: Nouns

- From a textual description of requirements, one may also scan the text for *nouns* or *noun phrases* to find candidates
  - Again, not all nouns are good conceptual classes

- From the UC descriptions…
  - Identify meaningful conceptual classes
  - Indetify nouns that are *not* meaningful conceptual classes

UC Rent Video: Main success scenario

1. Customer arrives at checkout counter with video
2. Cashier starts a new rental
3. Cashier scans member card's magnetic strip
4. Cashier scans video's bar code
5. System registers rental of video to Customer in ledger
6. Cashier requests due amount from Customer
7. Customer pays due amount
8. Cashier hands video to Customer

Finding conceptual classes: The category list

- A *category list* is a (long) list of time-proven categories in which conceptual classes are often ”hidden”.

- Proceeding through the category list will yield conceptual classes

- …but be careful: Not all categories are relevant to all problem domains!

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*

The category list: 10 minute exercise

- Consider the ”UC Rent Video” scenario.
  Derive at least 5 conceptual classes by analysing nouns.

- For each identified conceptual class, note to which category it belongs. (pg. 134-135)

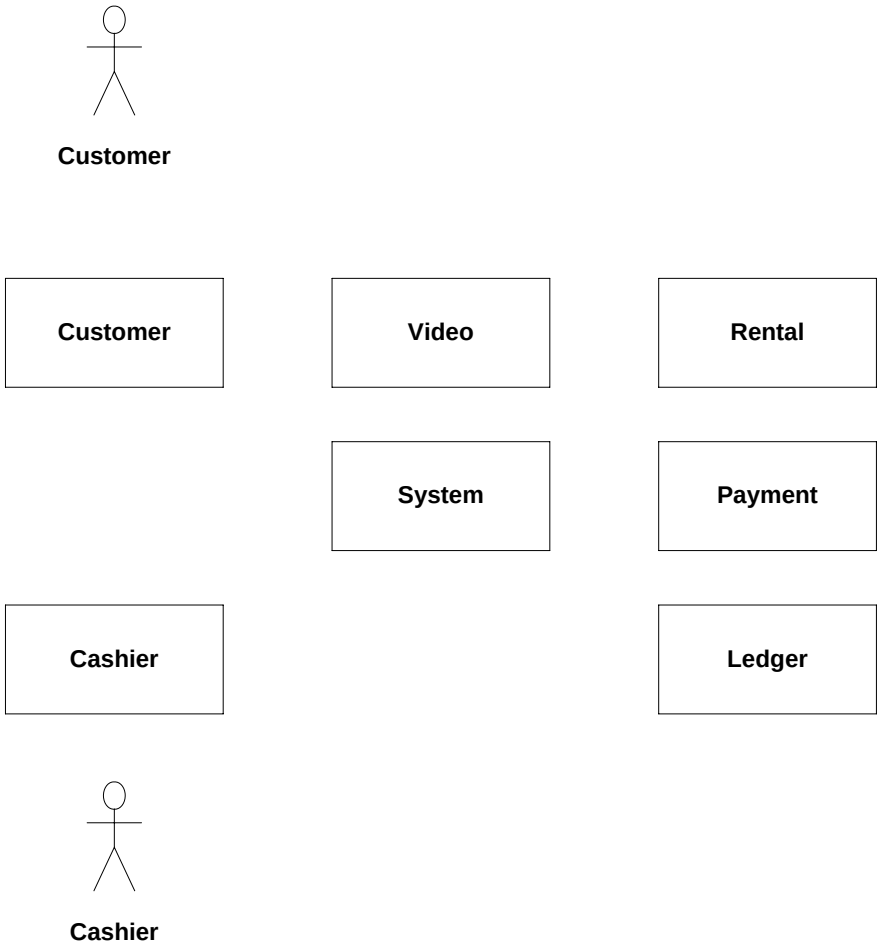
UC Rent Video: Main success scenario

1. Customer arrives at checkout counter with video
2. Cashier starts a new rental
3. Cashier scans member card’s magnetic strip
4. Cashier scans video’s bar code
5. System registers rental of video to Customer in ledger
6. Cashier requests due amount from Customer
7. Customer pays due amount
8. Cashier hands video to Customer

Nouns and categories

Noun	Category
Customer	Role, actor
Cashier	Role, actor
Rental	Transaction
Member card	???
Barcode	???
Magnetic strip	???
System	Place of transaction/service
Video	Product related to transaction
Payment	Transaction
Amount Due	???
Ledger	Recorder of transaction

First class diagram for Domain Model



How to create Domain Models

- Creating a domain model is easy – creating a *useful* one is hard!

- Three steps to follow:

Step 1: Find the *conceptual classes* in the domain

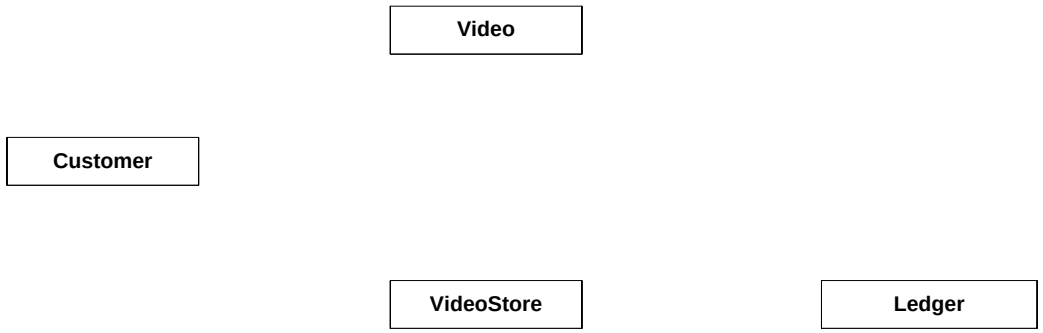
Step 2: Draw the classes in a UML class diagram (or SysML Block Diagram)

Step 3: Identify associations and attributes between conceptual classes

Creating a Domain Model, step 2: Draw the conceptual classes

- The conceptual classes identified in Step 1 can now be drawn in a UML class diagram or SysML Block Diagram.

- CASE tool or whiteboard, your choice…



Drawing – practical advice

- Use whiteboard or sticky notes or paper & pencil or a CASE program
- Work together

- Take a photo, save the sticky notes or paper, take a hard copy, or enter it in the CASE program –AFTER– you have finished your creative session

- Eventually we expect to see it in the System Architecture document in a readable form ☺

- There is no System Domain Model compiler!

How to create Domain Models

- Creating a domain model is easy – creating a *useful* one is hard!

- Three steps to follow:

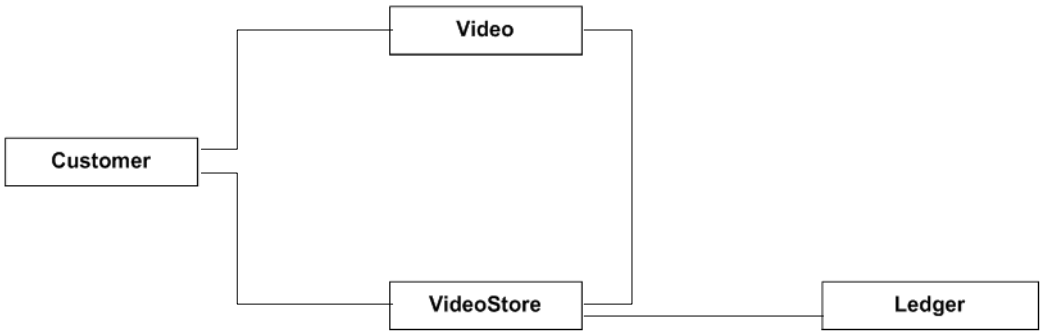
Step 1: Find the *conceptual classes* in the domain

Step 2: Draw the classes in a UML class diagram (or SysML Block Diagram)

Step 3: Identify associations and attributes between conceptual classes

Creating a Domain Model, step 3: Identify associations and attributes

- The conceptual classes identified in Step 1 and drawn in Step 2 can now be associated with each other



- …but how are associations *identified* and *named*? Some guidelines

Identifying associations between conceptual classes

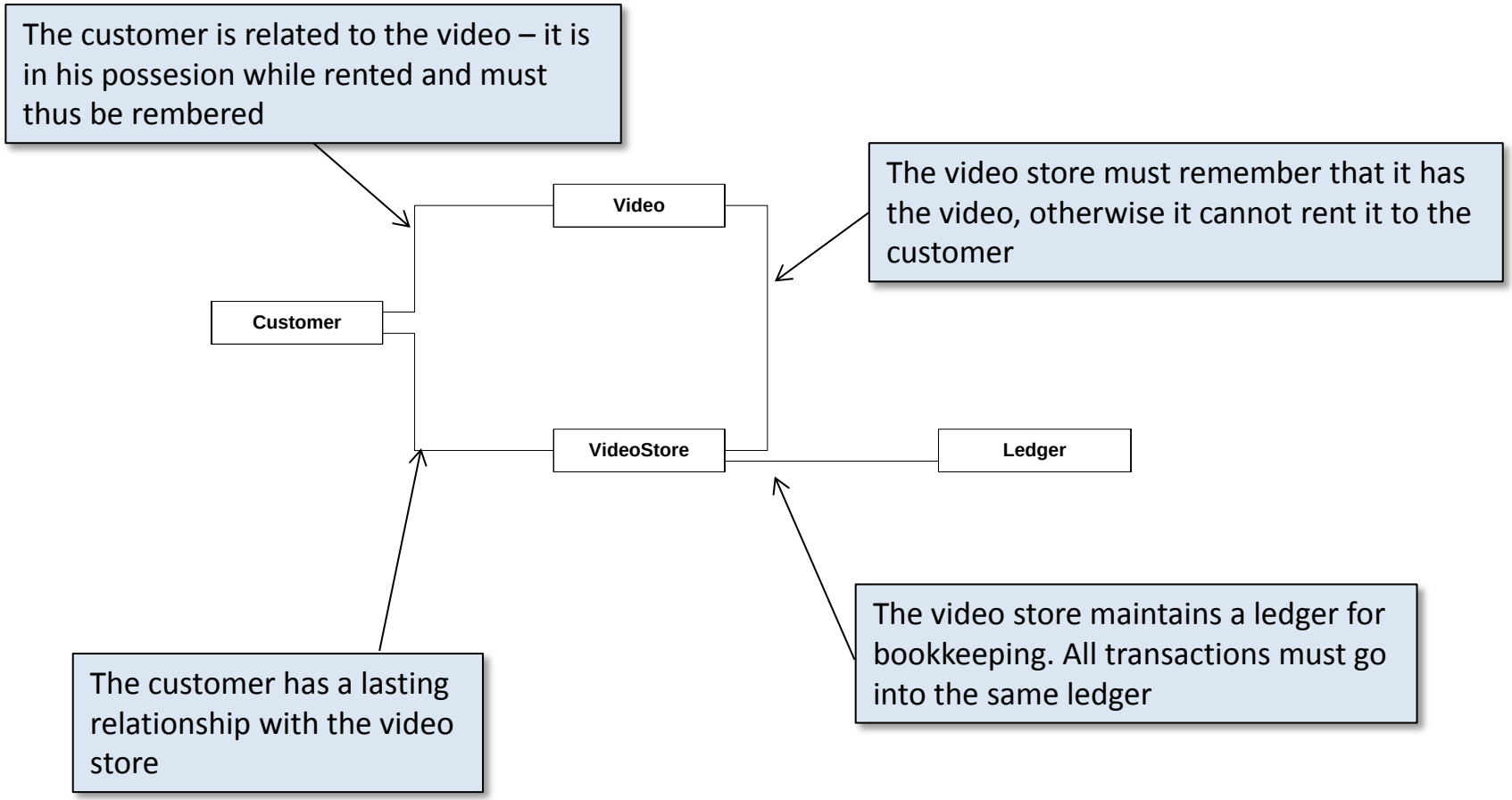
- With *n* classes you can have ~n^2 associations – which ones are important?

- Put short: ”***The ones you need to remember***”
  - Or: ”Associations for which knowledge needs to be preserved over some duration of time”

- Use the *Common Associations List*
  - Compendium page 149-150

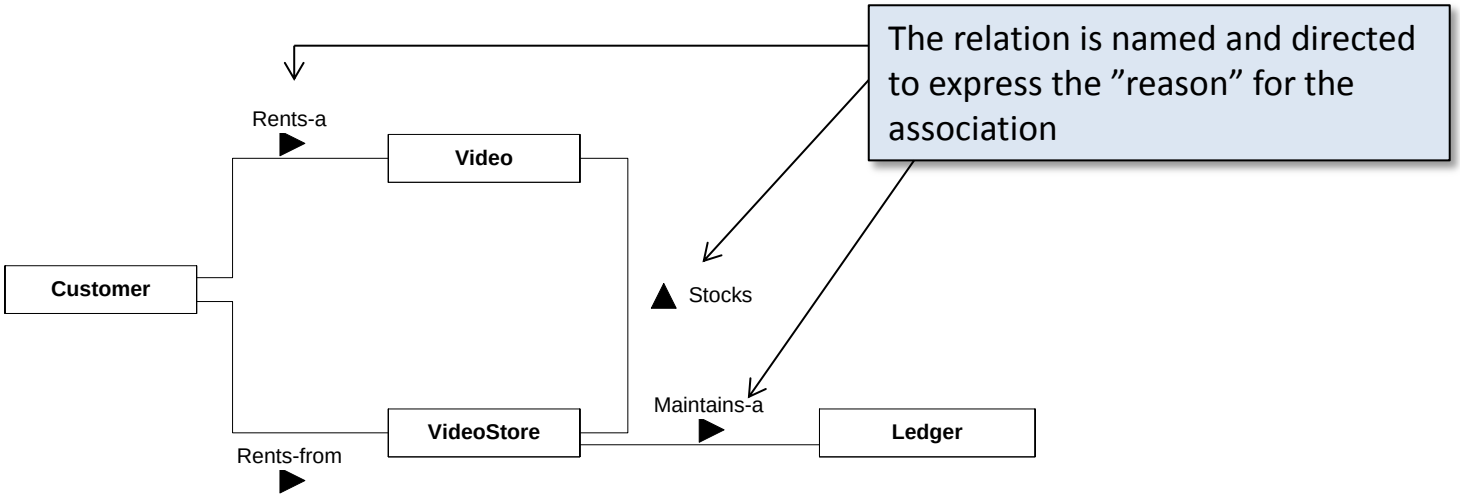
- Note: Associations in the DM *do not* imply association in HW or SW – they *only* imply associations on a conceptual level.

Domain Model: Conceptual classes and associations



Associations: Naming

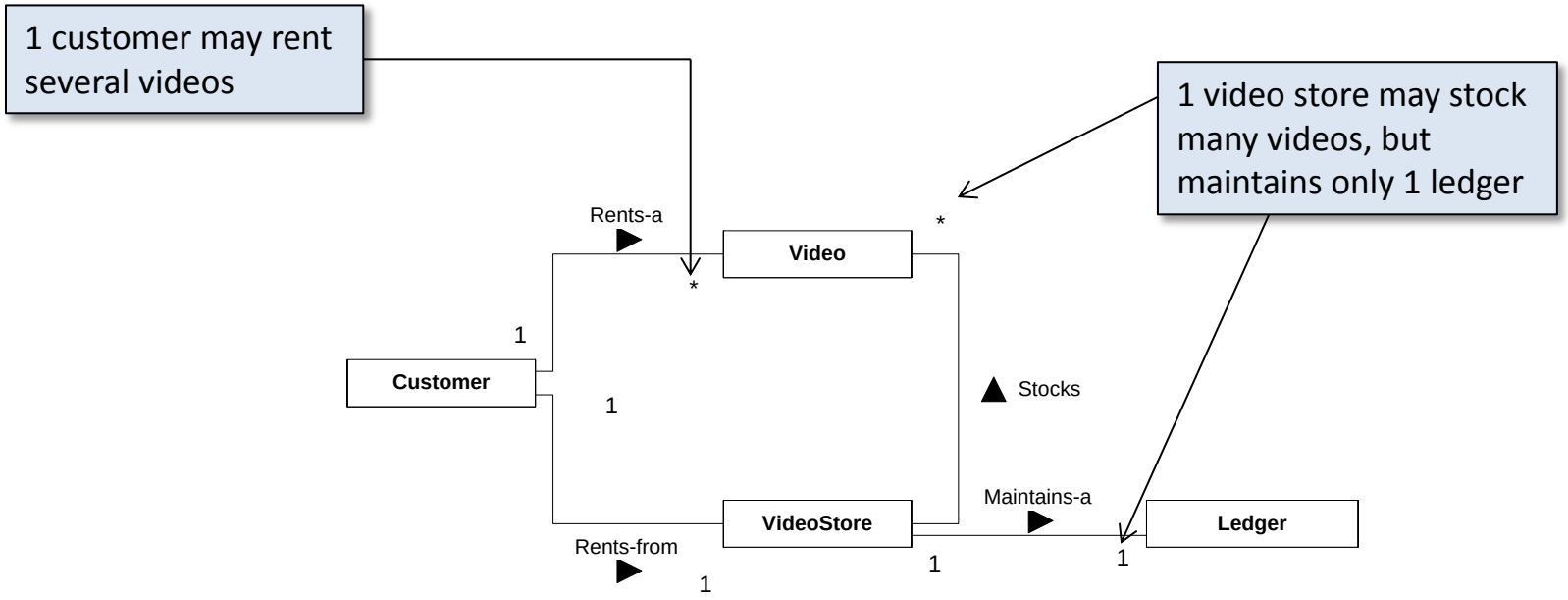
- Associations can be *named* and *directed* to further enhance the meaning and expression power of the DM



- Again: The arrow is only there to aid the understanding of the model – it implies *nothing* in terms of HW or SW association

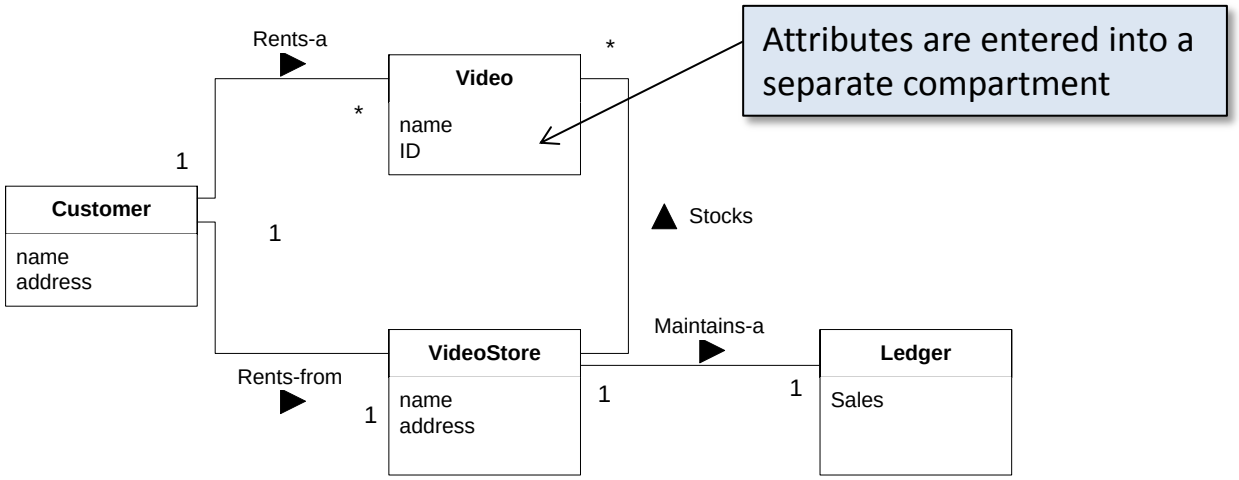
Associations: Multiplicity

- Associations can be assigned multiplicity



Identifying attributes

- The conceptual classes may also have attributes
- Attribute: A *logical data value* of the class needed to satisfy the currently investigated requirements

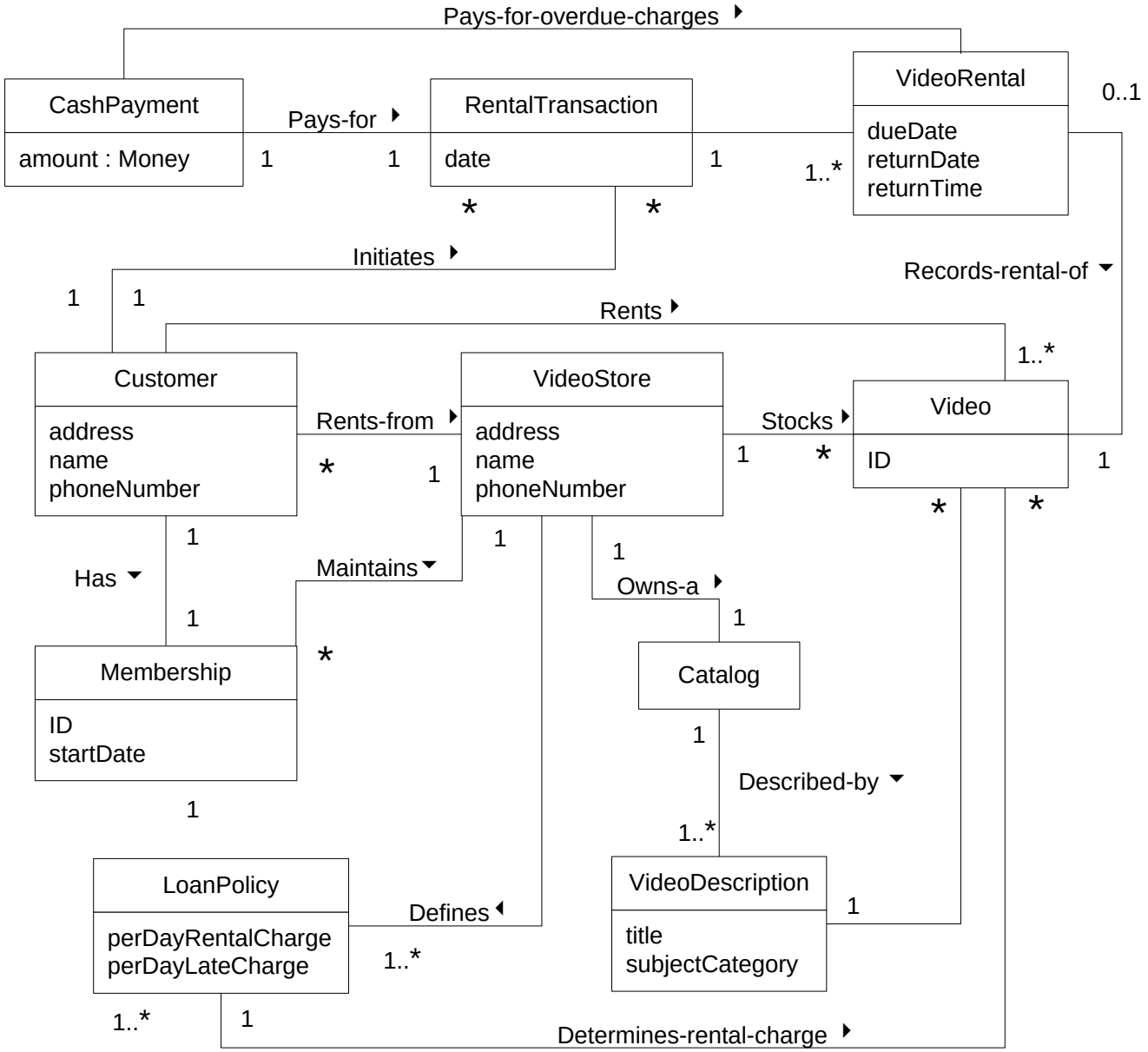


Attribute or class?

- *”Is it an attribute of a class or a class in itself?”* ‘Tis the question

- Guidelines
  - If it ”takes up space”, it’s a class
  - If it is a complex type, it’s a class
  - If it has behaviour, it’s a class
  - If it is simple and state-only, it’s an attribute

EXAMPLE: Video Rental



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

Exercise for next session: *Service station*

- Create a domain model for the service station based on the use case ”Optank Bil” (see text on BB):

1. Identify meaningful conceptual classes using the methods you have learned about
2. Create a UML Class Diagram with the conceptual classes
3. Create and name associations between classes
4. Set multiplicities where applicable
5. Add attributes to the classes

Exercise for next session: *Poultry Galore*

- Create a domain model for the new batching system for ”Poultry Galore” (see text on BB):

1. Identify meaningful conceptual classes using the methods you have learned about
2. Create a UML Class Diagram with the conceptual classes
3. Create and name associations between classes
4. Set multiplicities where applicable
5. Add attributes to the classes