Slides for
Design Methods for Reactive Systems: Yourdon, Statemate and the UML

Roel Wieringa
Department of Computer Science
University of Twente,
the Netherlands
roelw@cs.utwente.nl
www.cs.utwente.nl/∼roelw

List of Slides

Chapter 8. Entity-Relationship Diagrams

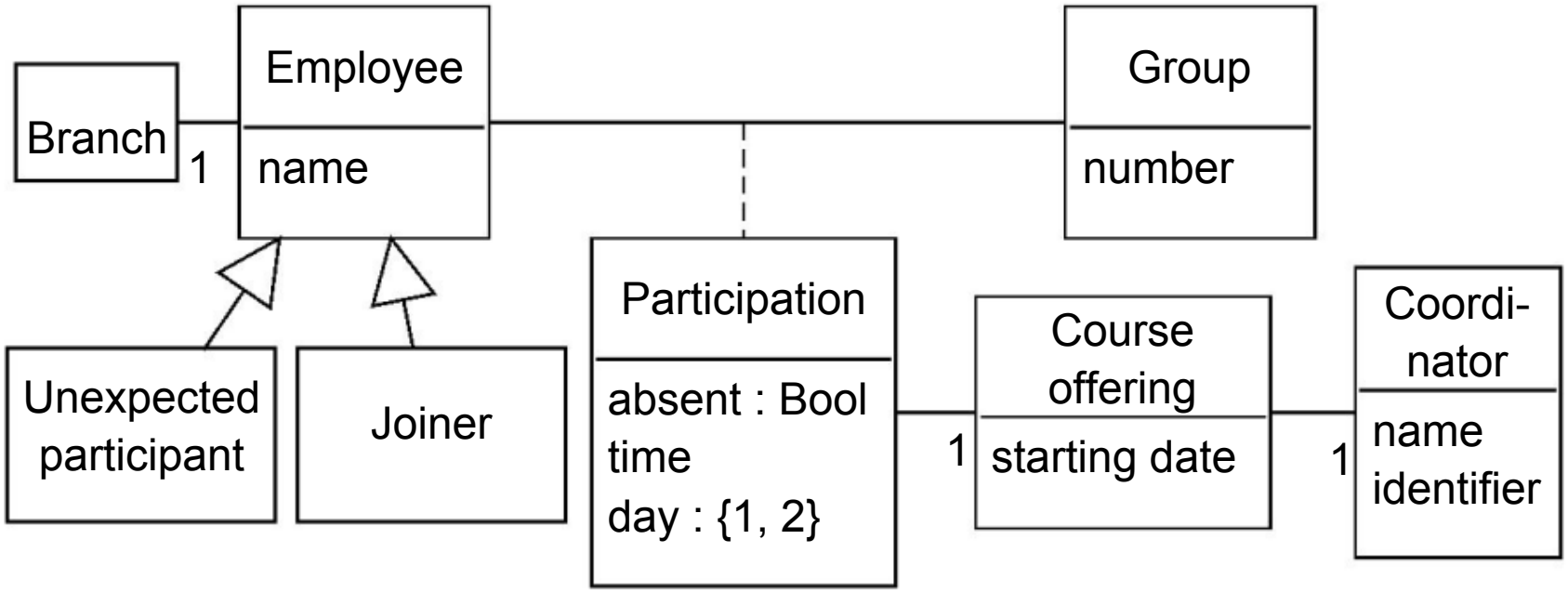
The **subject domain** of a software system is the part of the world talked *about* by the messages that cross the system interface.

To find the subject domain of a system

- Look at the messages received and sent by the system, and
- Ask what these messages are about.

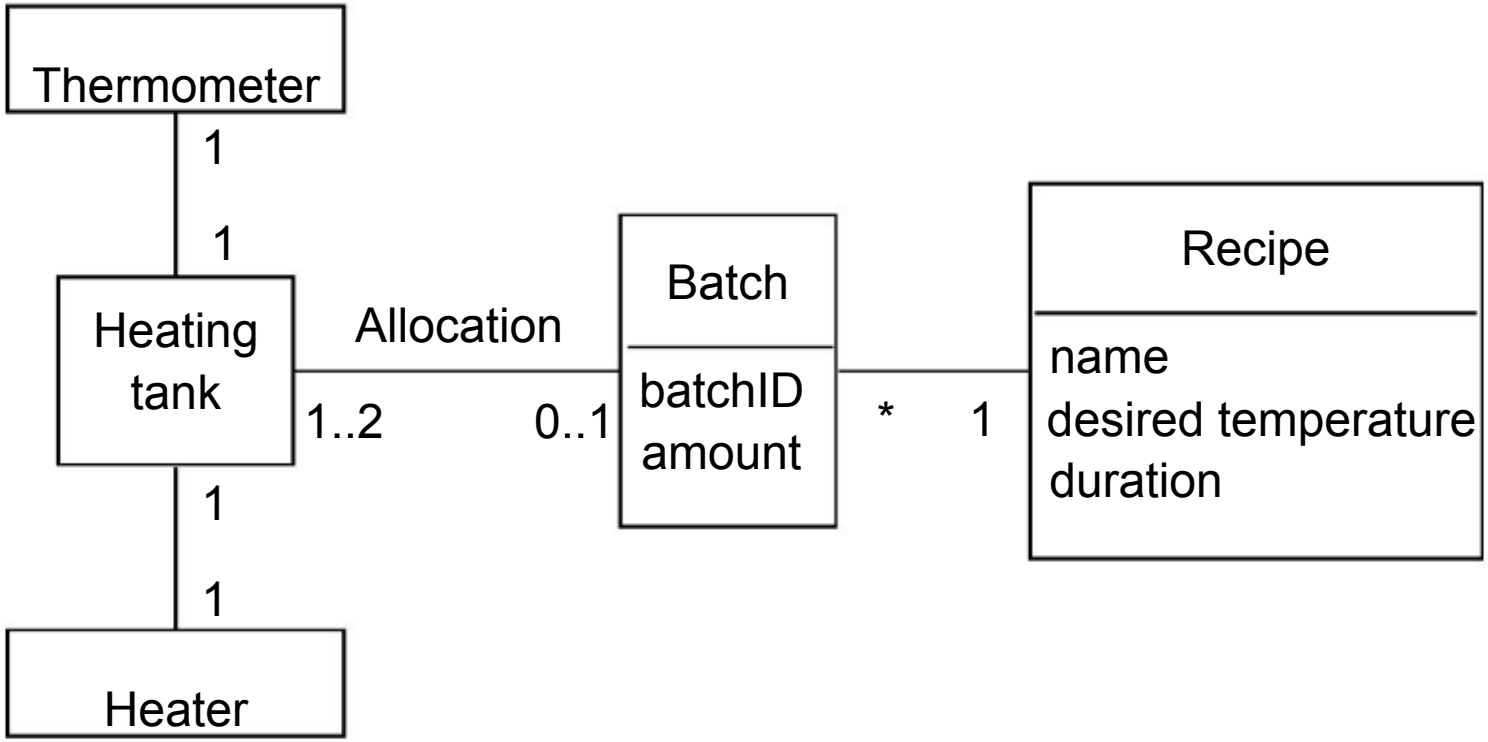
Document the meaning of these messages in a dictionary, supplemented by an ERD of the subject domain.

Example 1: Subject domain of TIS



These entities are the topic of interactions with TIS.

Example 2: Subject domain of heating controller

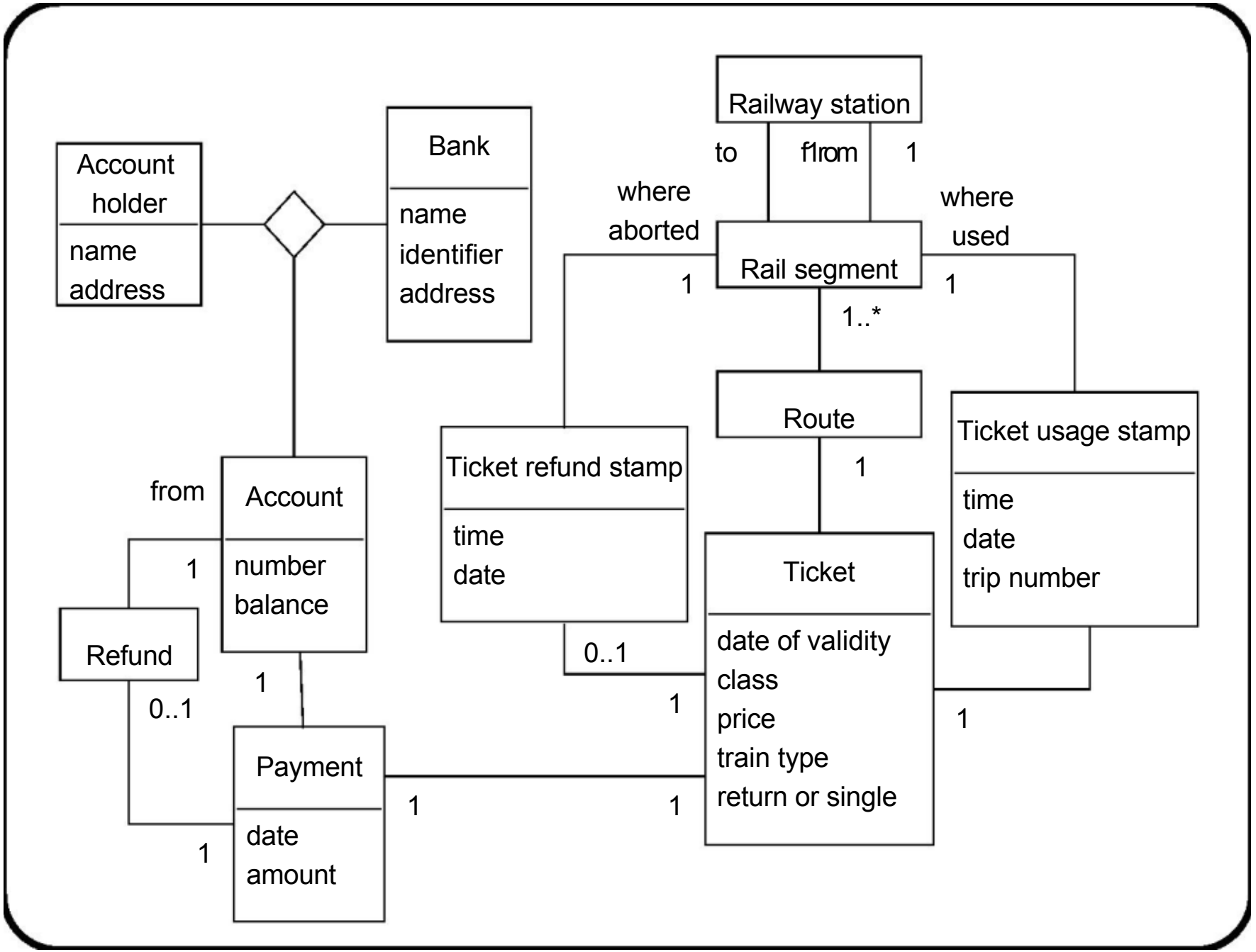


These entities are the topic of interactions with the controller.

Example 3: Subject domain of ETS

See next slide.

- These entities are the topic of interactions with the ETS.
- Some of them are lexical entities to be stored in ETS.

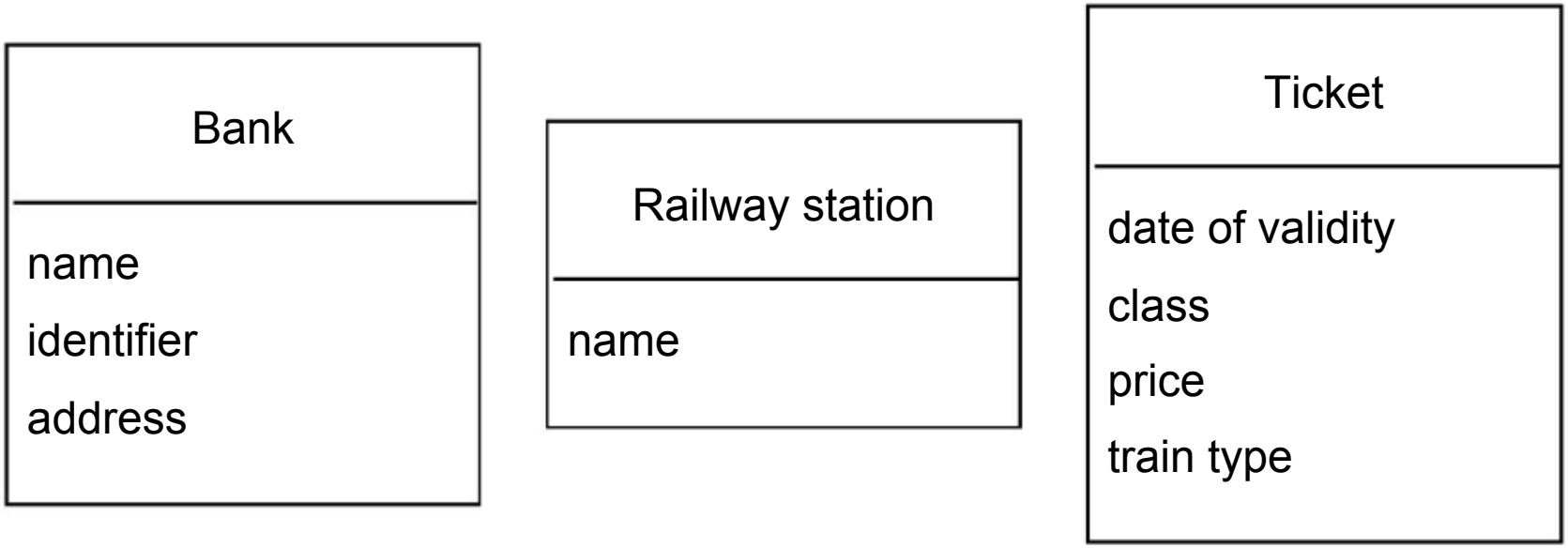

Railway station
to
from
1
where aborted
where used
Bank
name
identifier
address
Account holder
name
address
Rail segment
1
1
1..*
Route
1
Ticket refund stamp
time
date
Ticket usage stamp
time
date
trip number
from
Account
number
balance
1
Refund
0..1
1
0..1
1
Ticket
date of validity
class
price
train type
return or single
1
Payment
date
amount
1
1
1
1

Syntax and semantics of ERDs

Entity type.

- Represented by a rectangle.
- **Extension** = All *possible* instances of entity type.
- **Extent** = All *existing* instances of entity type.
- **Intension** = All properties shared by all possible instances.
- **Defining intension** = Properties used to define the entity type. This is an abstraction (simplification) of the full, informal meaning.

Attributes



- Properties of entities.
- Can be listed in a compartment directly below the type name compartment.

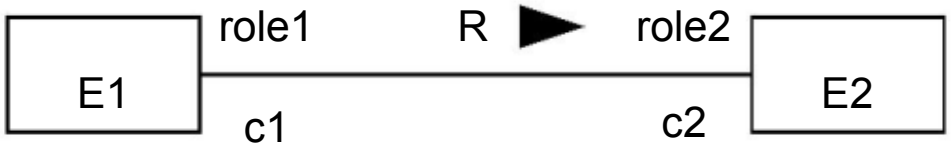
Question: Which properties of a railway station did we describe in this diagram?

Relationships

- A set of tuples of entities.
- Arity is the number of related entities. Binary, ternary, etc.
- Everything in the world is related to everything else!
- We only describe a relationship if we want to express relative

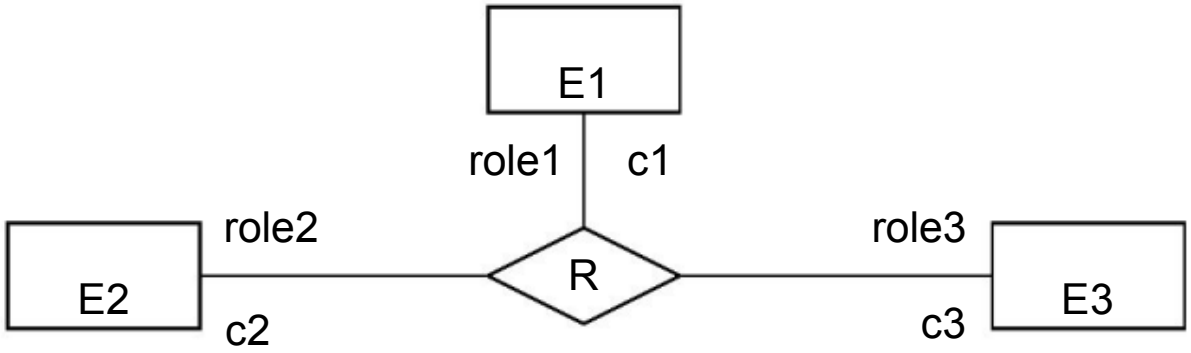
  cardinality properties: How many entities of one type can exist for each entity of another type.

Binary relationships



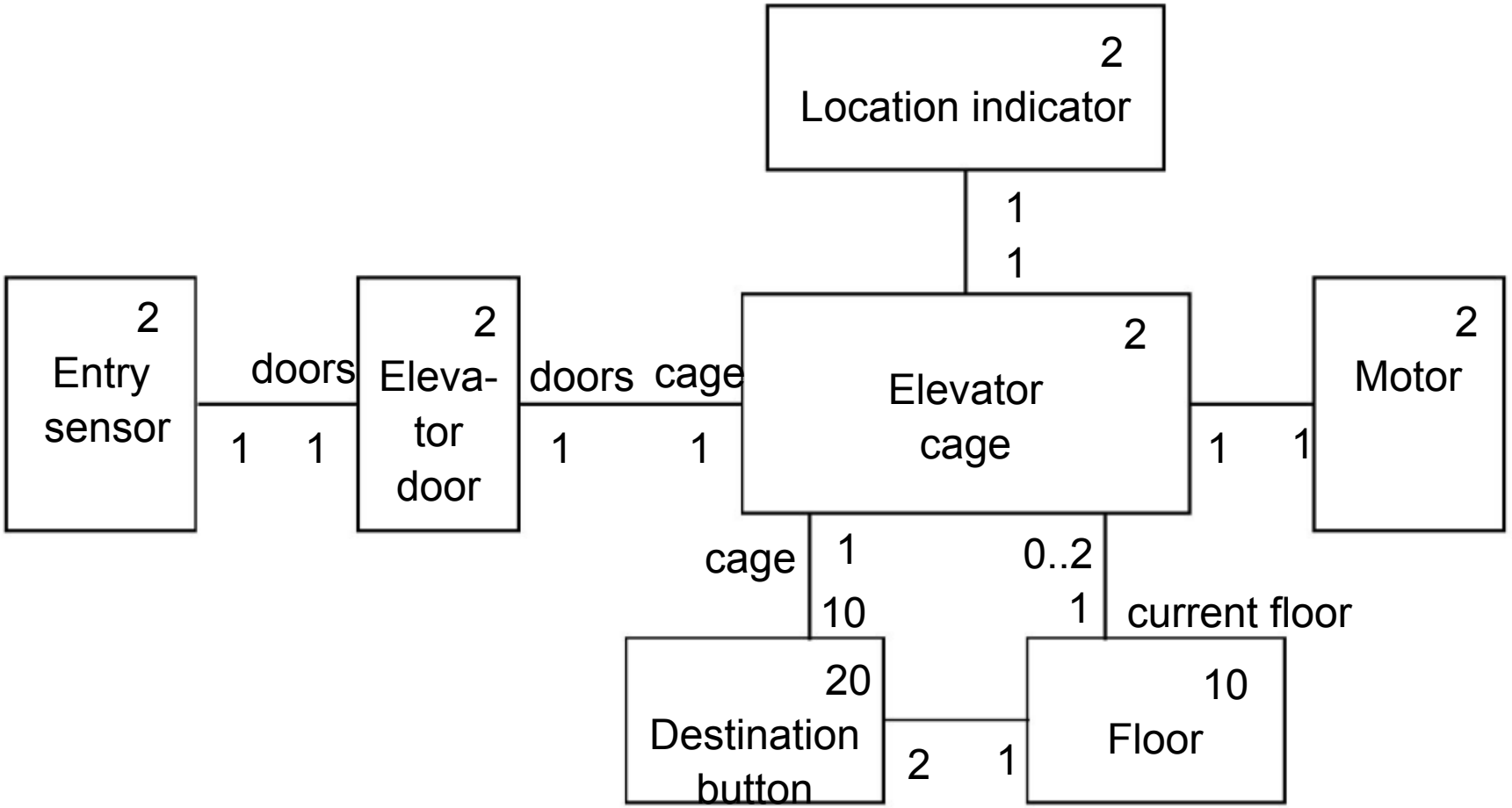
- Binary relationship represented by a line.
- Relationship name can have a read direction.
- Role names indicate the role an entity plays in a relationship.
- Relationship name and role names can all be omitted if this does not lead to ambiguity.

Relationships of higher arity



- Relationships with arity ≥ 3 represented by a diamond.
- Name must be independent from direction of reading.

Cardinality properties: Elevator example

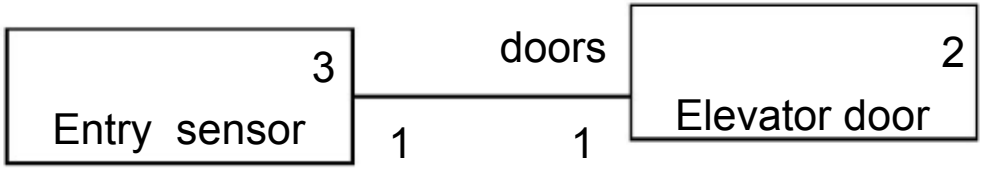


The meaning of cardinality properties

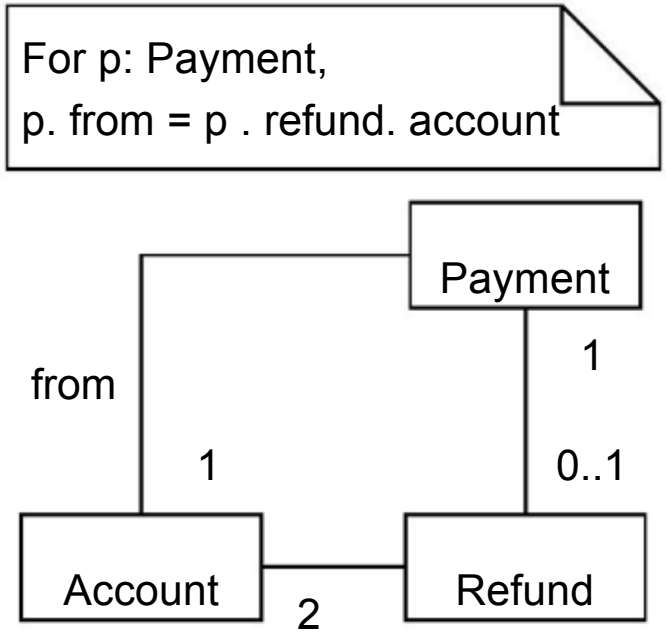
- A snapshot of the subject domain is the state of the subject domain at one point in time.
- Cardinaility properties are snapshot properties.
- An absolute cardinality property says how many instances of an entity can exist in a snapshot.
- A relative cardinality property says how many entities can exist relative to some existing entity.

To find a cardinality property, imagine looking at an *arbitrary* state of the domain and ask how many instances of some entity type can exist in that state.

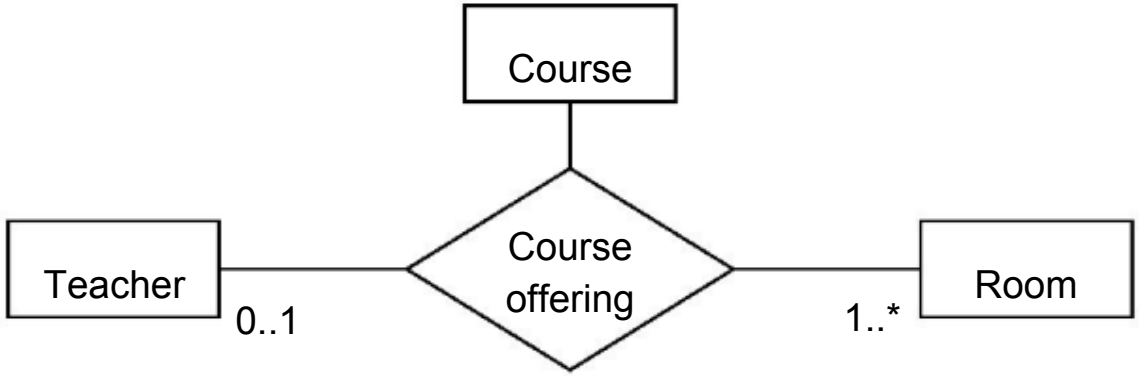
First example of inconsistent cardinality properties



Second example of inconsistent cardinality properties



Ternary relationships

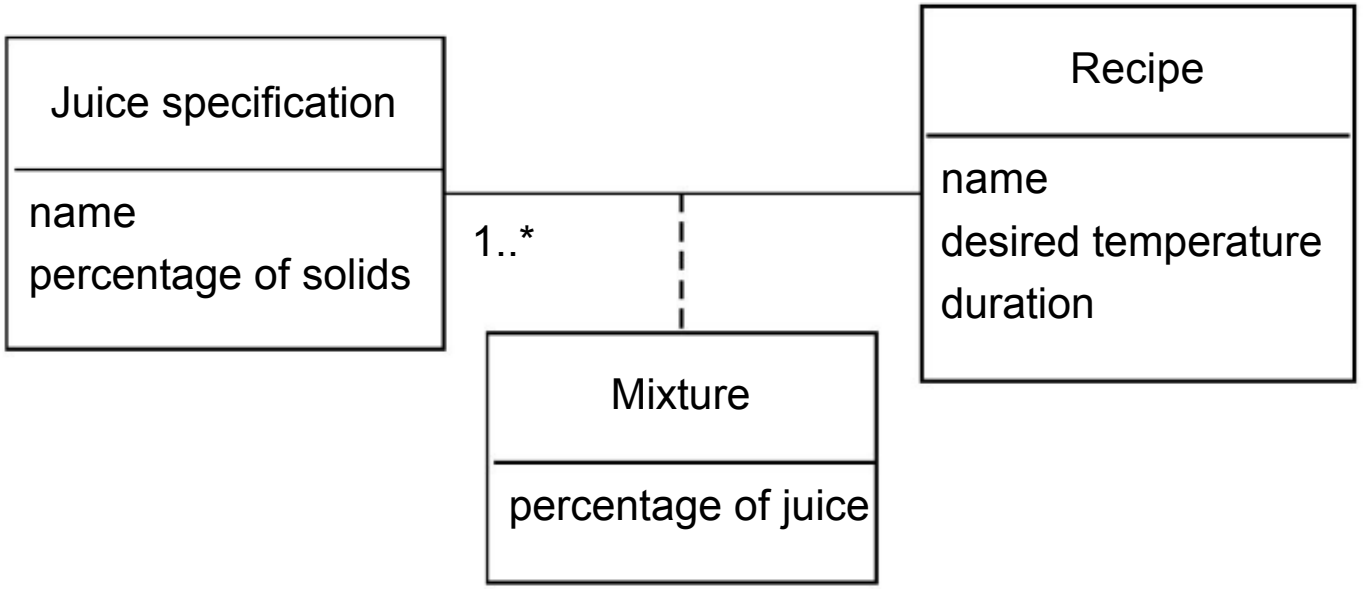


Cardinality properties of relationships with arity $/geq 3$ are hard to understand.

Questions

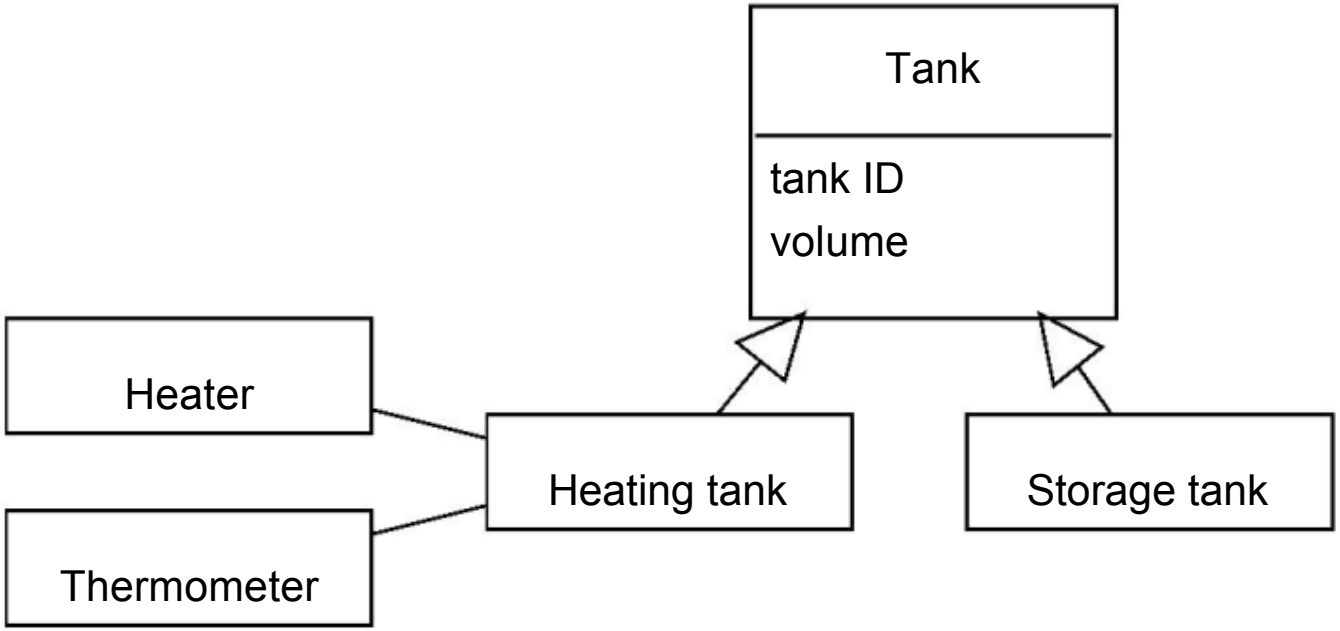
1. Can a course have more than one teacher?
2. Can a teacher give several courses?

Association entities



- An association entity is a relationship with attributes.
- Each instance is really a relationship. It is identified by a tuple of component identifiers.

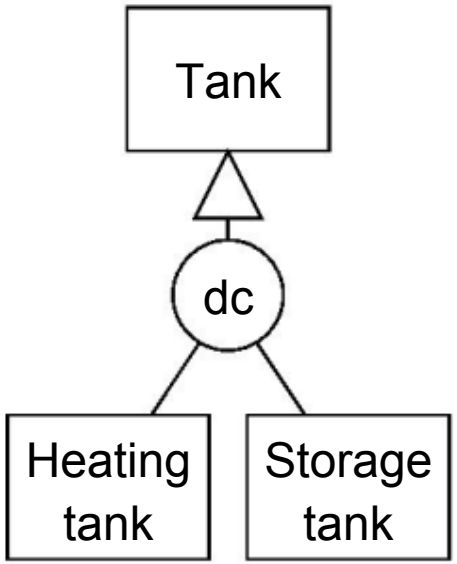
Generalization

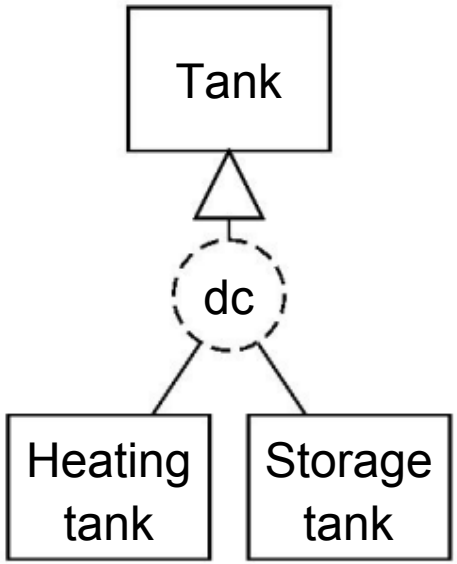


Characteristic of specialization:

- Instance of a subtype is instance of a supertype.

Dynamic versus static specialization





Static specialization: Subtype *extensions* are disjoint and cover supertype *extension.*

Dynamic specialization: Subtype *extents* are disjoint and cover supertype *extent.*

NB. If any heating tank can become a storage tank and vice versa, then all three extensions are equal! Why?

Main points

- We make an ERD of the subject domain, not of databases.

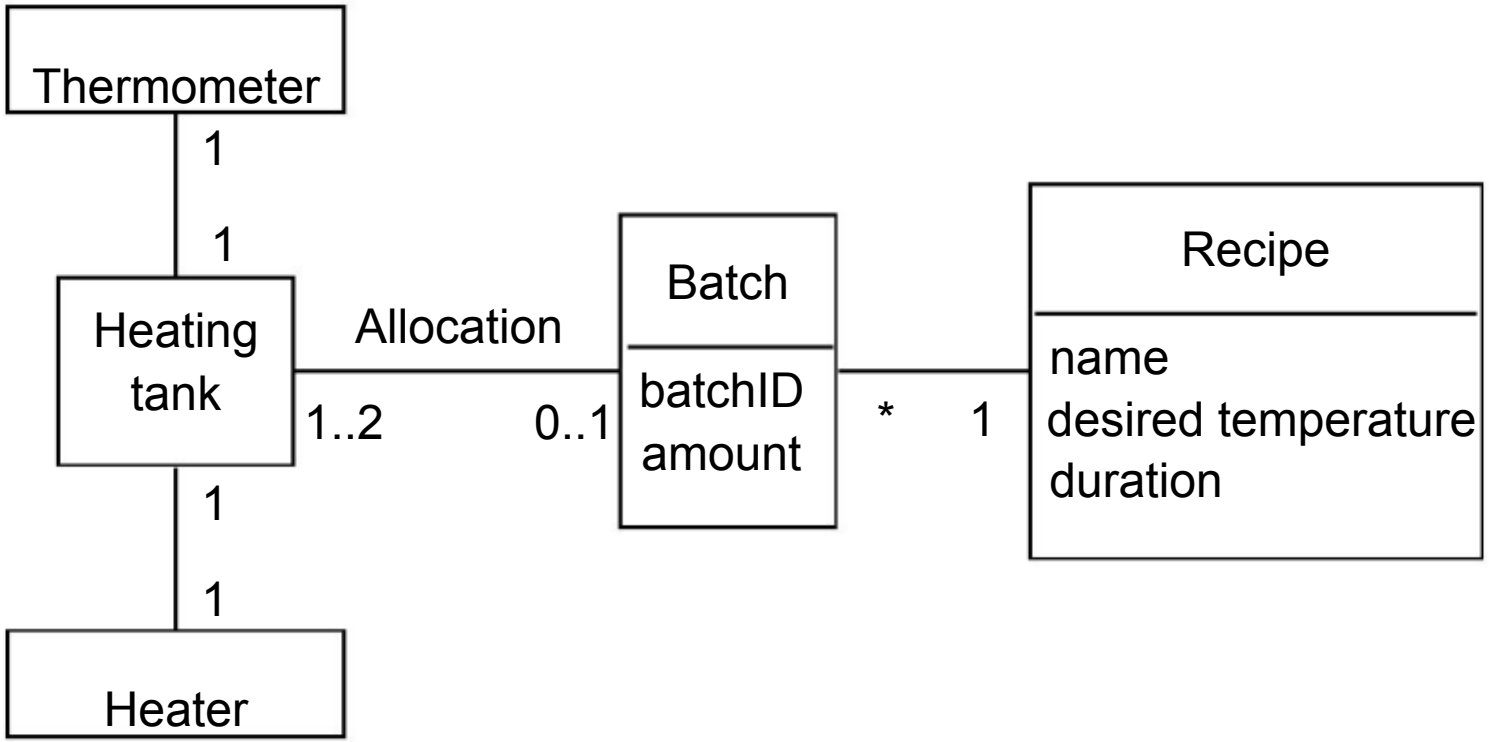
  Even virtual entities are interesting only because they are part of the subject domain.

- ERD represents identification and classification of entities; and cardinality properties.

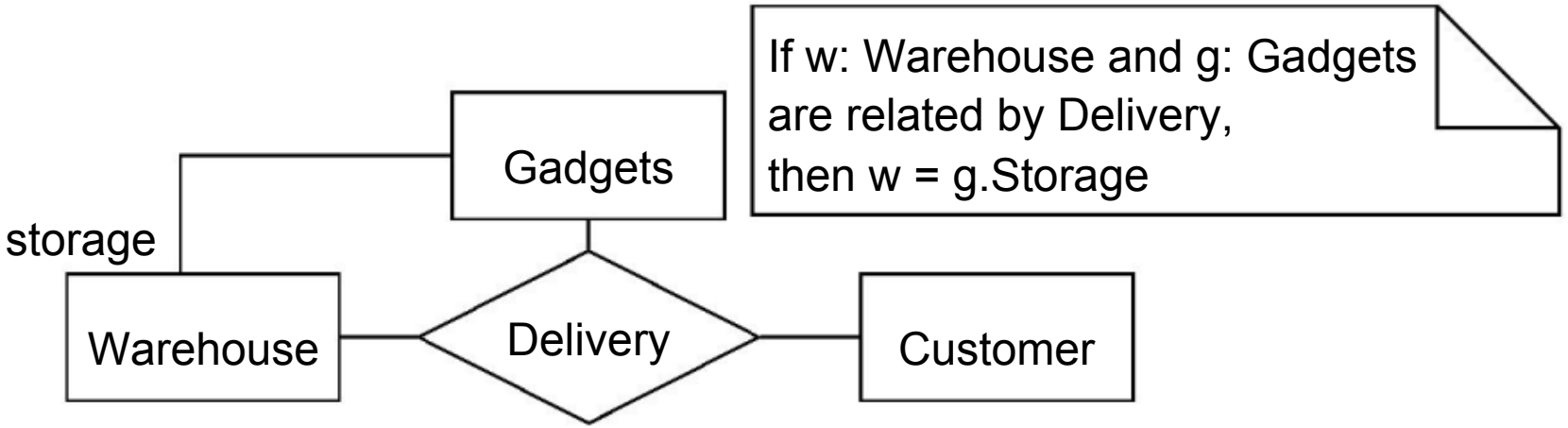
  Counting and classification are closely related. Class (= type) provides identification criterion.

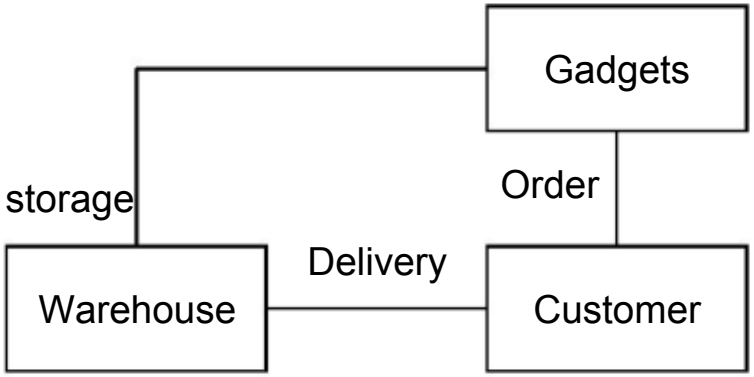
- Classification can be static or dynamic.

Questions

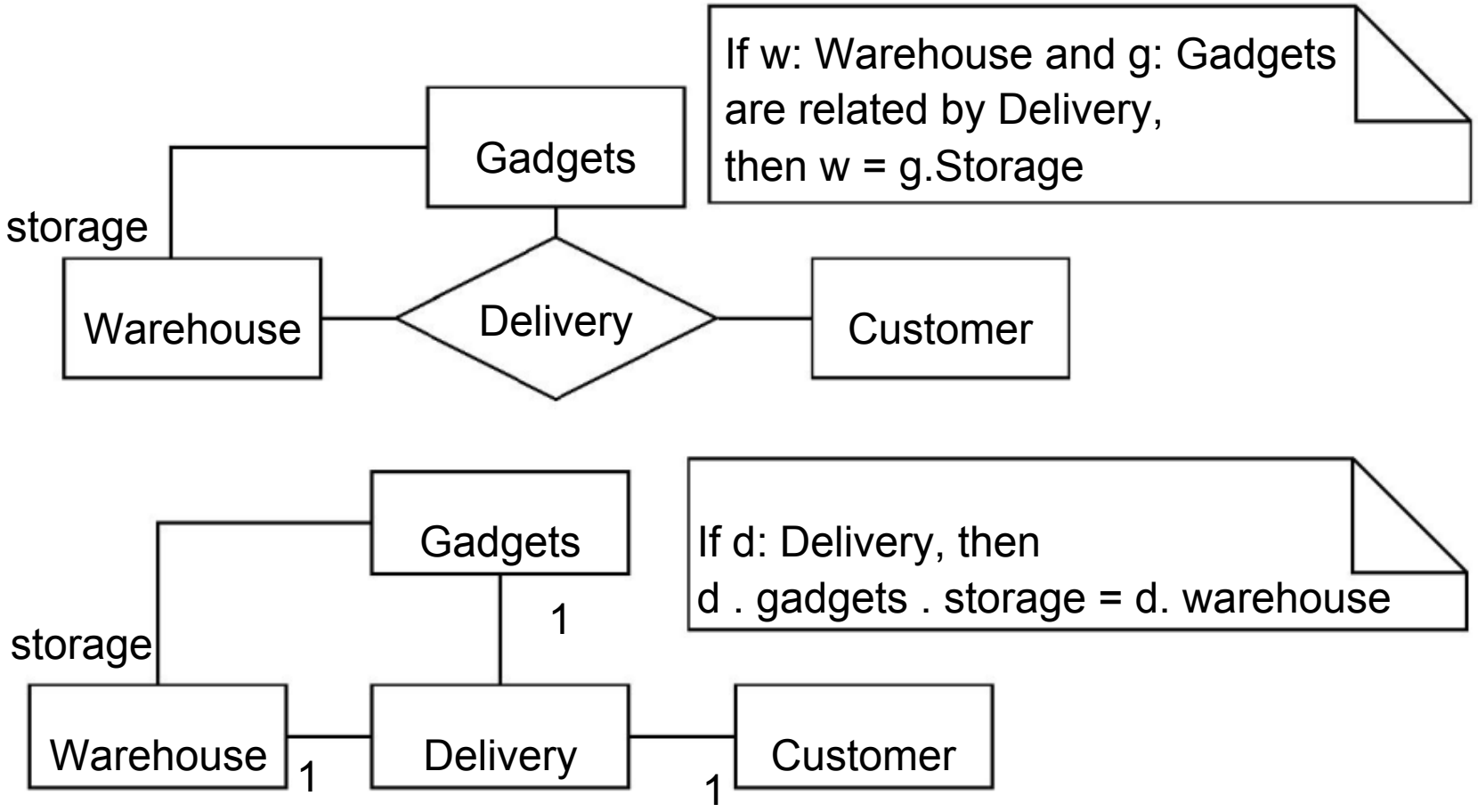


- Suppose we interpret cardinalities historically: "the number of instances related to in a lifetime". Change the cardinalities to historical ones.





- Are these diagrams equivalent? Beware of the connection trap.



- Are these diagrams equivalent?

Chapter 9. ERD Modeling Guidelines

- ERDs can be used to declare any set of entity types and their cardinality properties.
- In this course we use ERDs only to represent the structure of the subject domain.
- Guidelines to determine the boundary of the subject domain are relevant for this kind of use of ERDs only.
- All the other guidelines are relevant for all possible uses of ERDs.

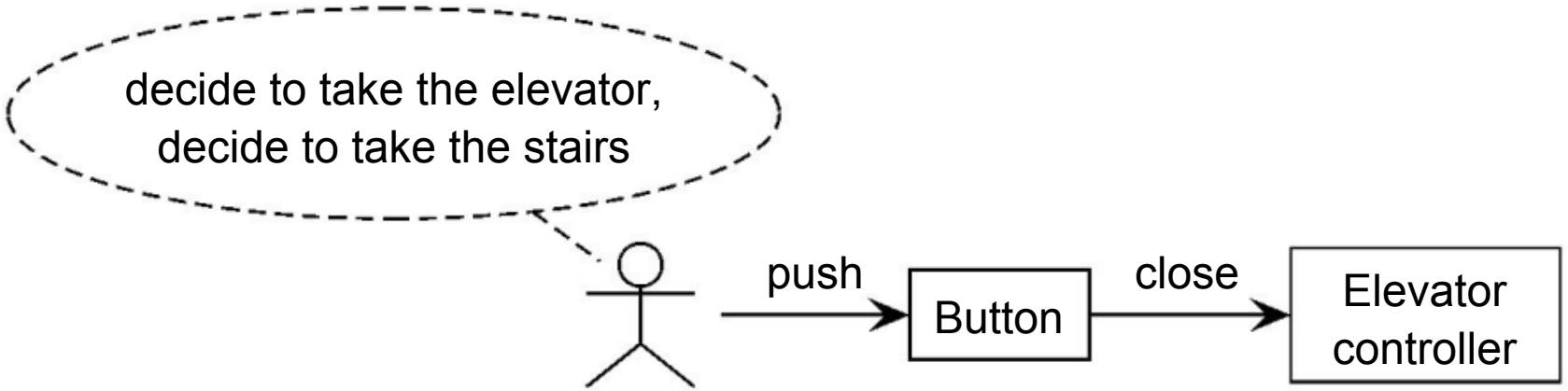
Subject domain boundary

The subject domain consists of entities and events. To find the boundary:

- √ Roughly: What entities and events do the service descriptions refer to?
- √ More precisely: What are the messages entering and leaving the system *about?*
- √ Look for
  - — Physical bodies
  - — Devices
  - — (Parts of) organizations
  - — Conceptual entities
  - — Lexical items

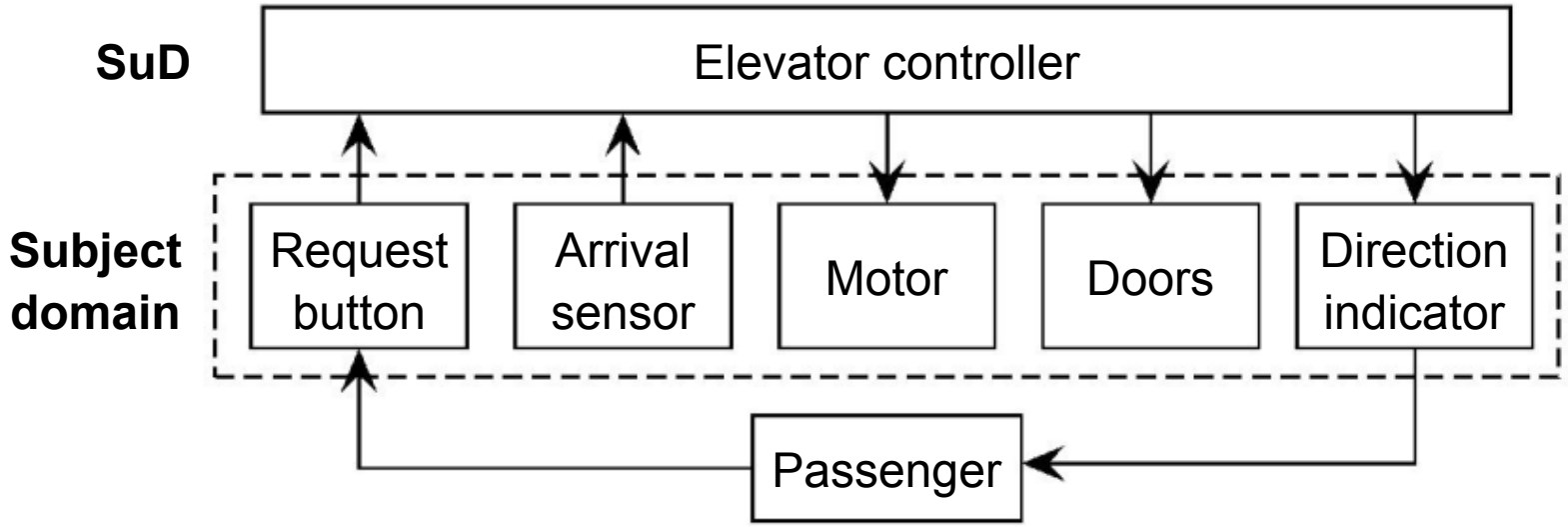
  and events in their life.
- √ Entities and events should be identifiable by the SuD.

Unobservable events



Connection between events of interest (decisions by passenger) and stimuli of controller are too weak to include passenger in subject domain.

Including unobservable entities in a diagram of external communications



- The subject domain contains devices with whom and about whom the elevator controller exchanges messages.
- However, we can include unobservable entities in a diagram

  that shows external communication channels (the context diagram).

Questions

- Which entities are in the subject domain of a railway travel planner?
  - Passenger, Station, Railway track, Trip segment, Route.
- Is a Passenger part of the subject domain of the Electronic Ticket System? Why (not)?

The answers are determined by

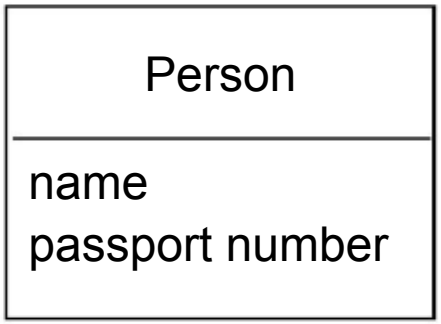
(1) desired functionality of system and

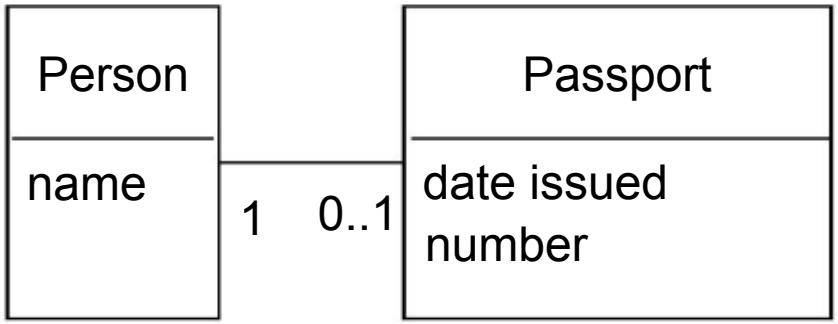
(2) choices in connection technology.

Entities versus attributes

- √ Entities are the things that the system talks about.
- √ Attributes are the things said about entities.
  - So if you want to store information about it, it is an entity.
- √ If information about it can change, it is an entity.

Example





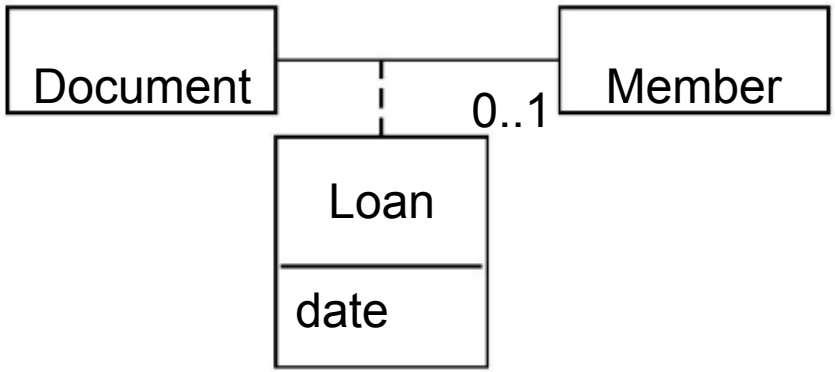
- Counting persons is not the same as counting passports.
- Persons are not created at the same time as passports.

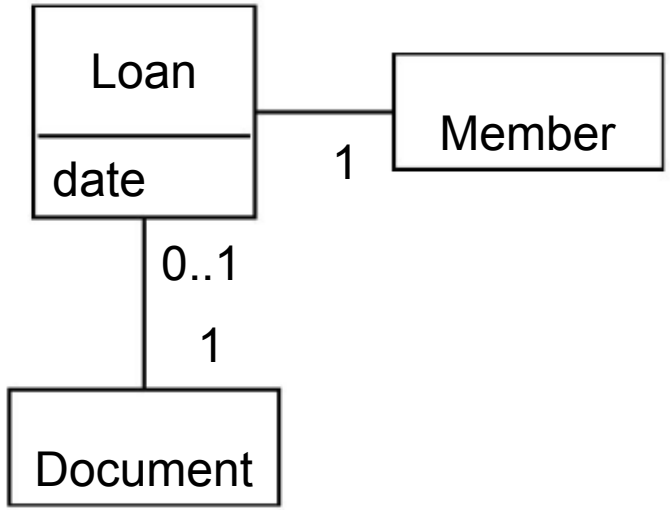
So they are different entities. But which ERD is "correct"? That depends:

- If the SuD must be able to talk about persons without passports, or
- about persons with multiple passports (change cardinality property for that),
- then we must include a separate passport entity type.

Entities versus relationships

√ Relationships are identified by their components.
So they are counted differently from the way entities are counted.



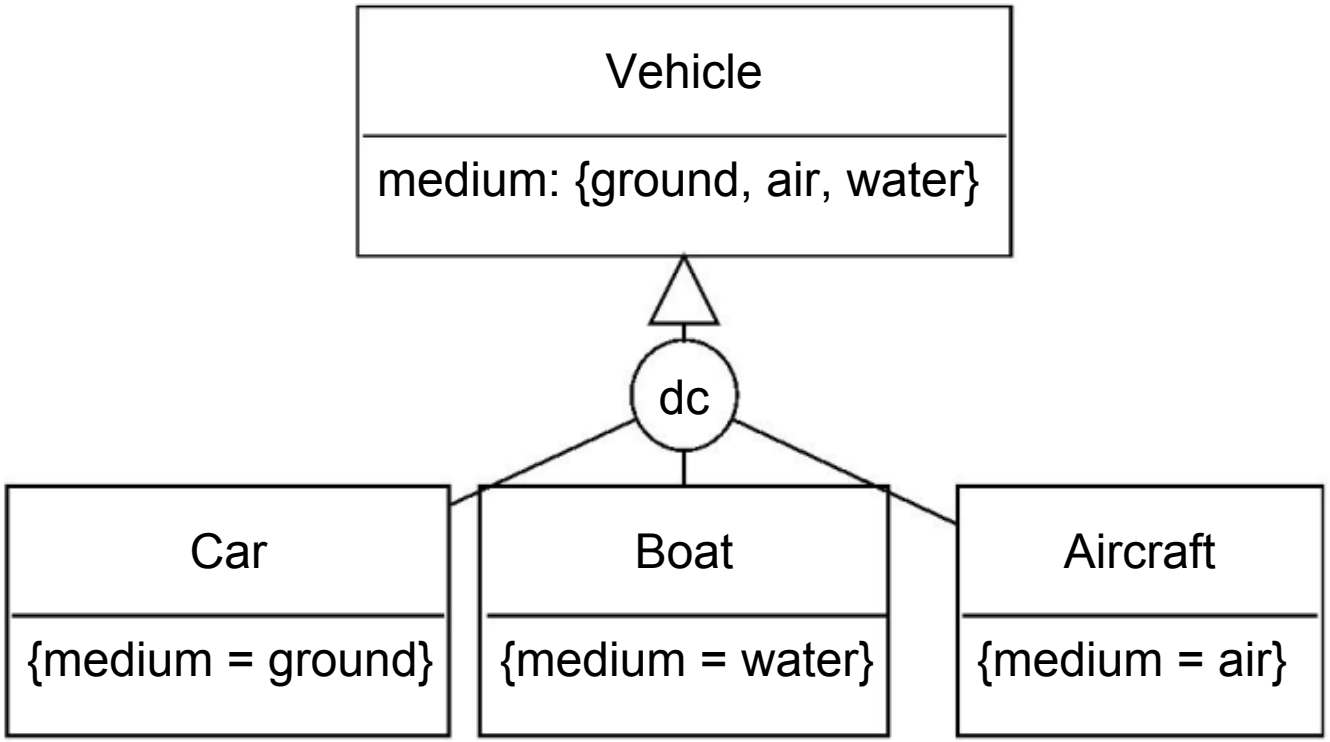


Several loans may co-exist!

Not what we intend

Taxonomic structures

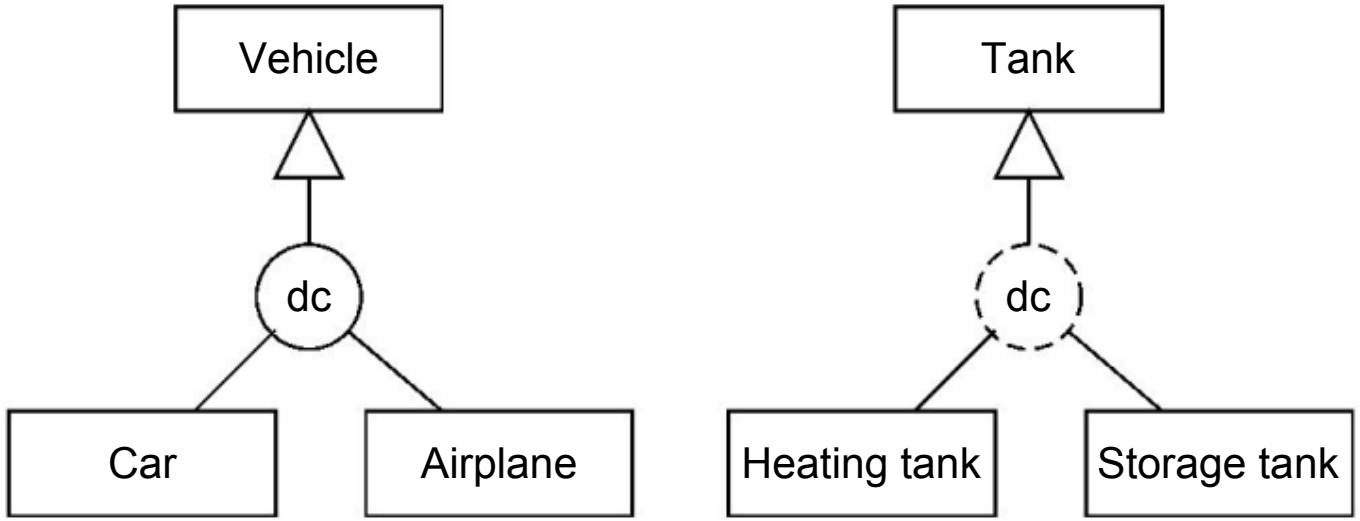
√ Use a specialization attribute.



Any subtypes missing?

Do we need these subtypes?

Static versus dynamic specialization



According to this ERD, if you are born as a car, you are always a car. But heating tanks can become storage tanks and vice versa.

- √ Creating a static subtype instance is creating a supertype instance.
- √ Creating a dynamic subtype instance may not involve creating a supertype instance.

Classification and identification

√ A type definition should provide a recognition criterion and an identification criterion.

- The recognition criterion for Car gives us the answer to this question "Is this a car?"
- The identification criterion gives us the answer to the question "How many cars do we have here?"

To find the identification criterion of type C, it is often useful to look at the creation event of an instance of C.

Questions

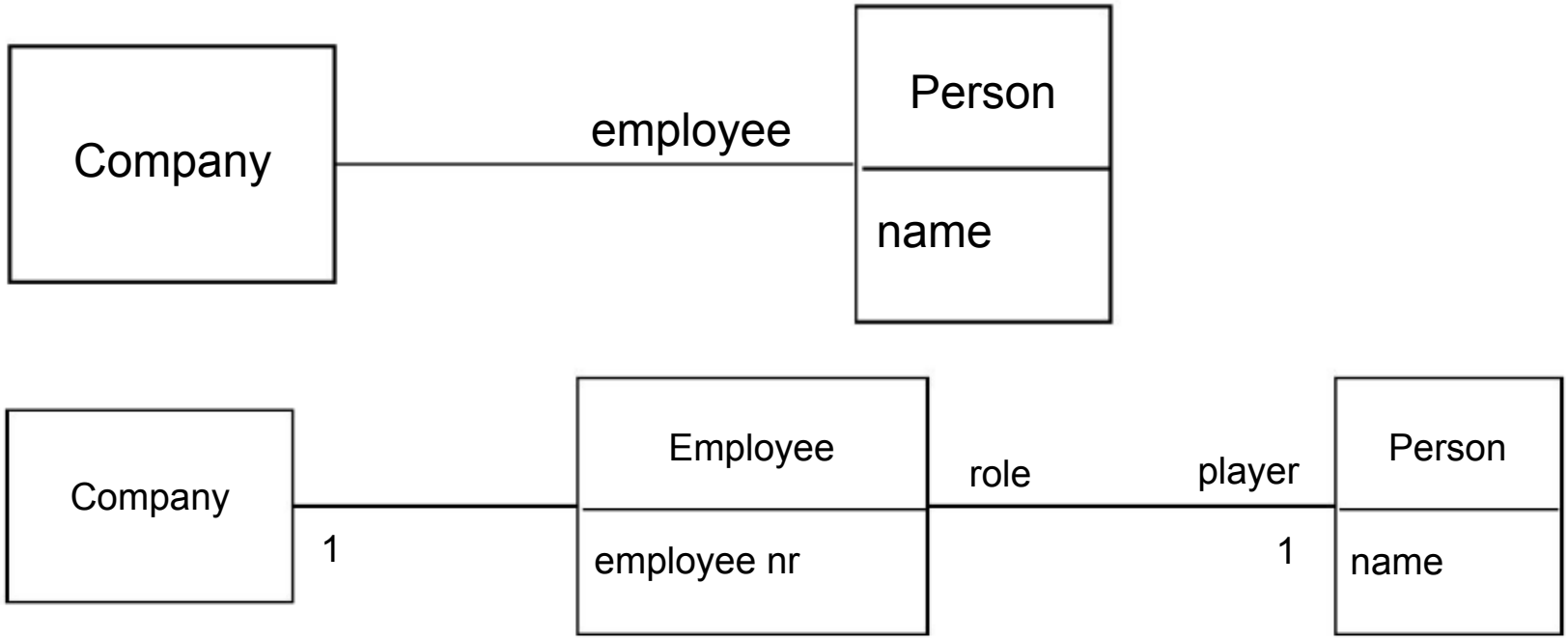
Give recognition and identification criteria for the following concepts:

- Person
- Company
- Passenger
- Elevator button
- Your PC
- The chair you are sitting on.

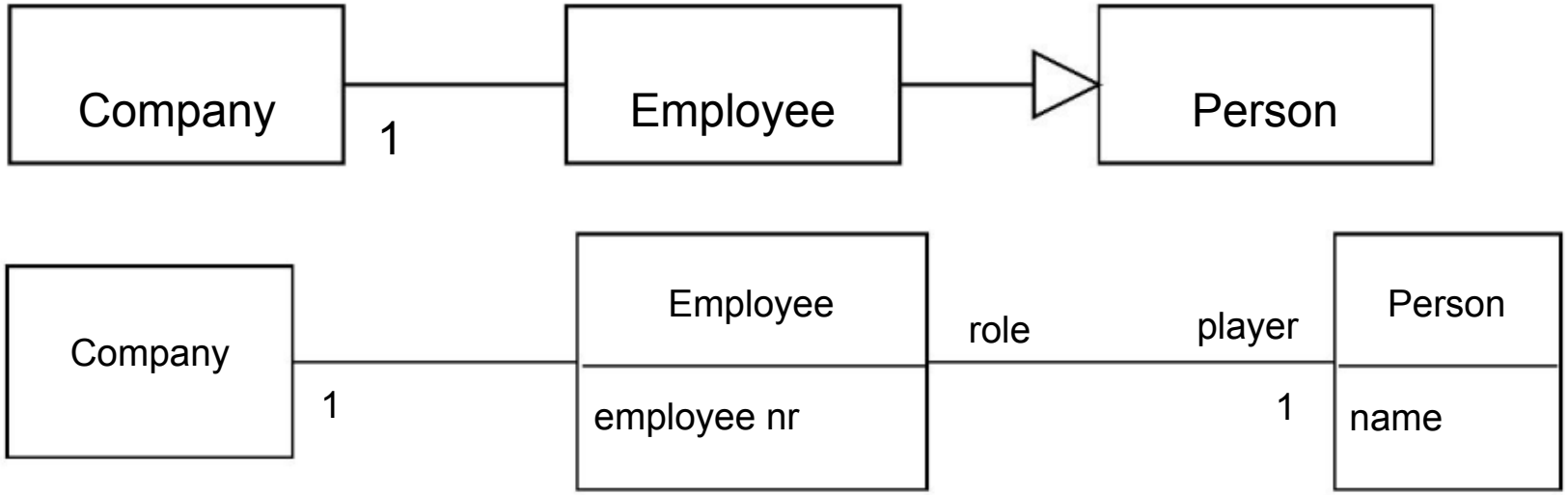
If an instance of C is in the subject domain, then the SuD is able to talk about a C, and so the SuD should be able to recognize and identify C's!

Subtypes versus roles

If an entity can play several roles of the same type, we can turn the role into an entity type.

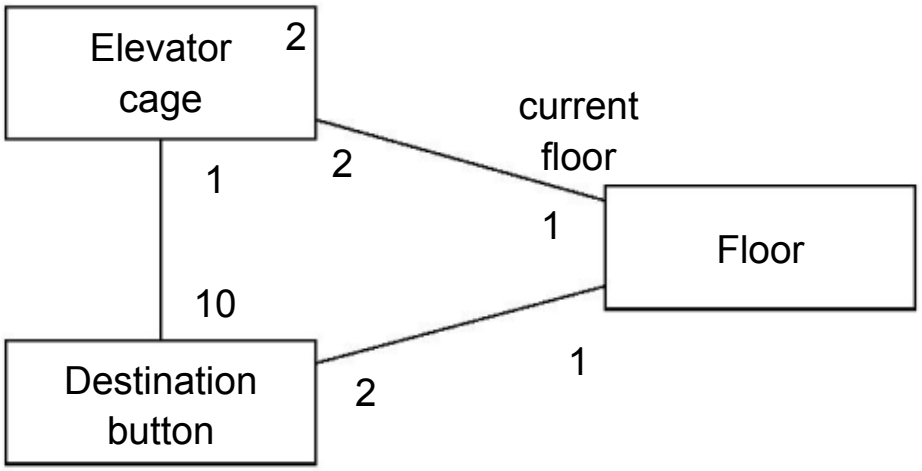


What is the difference between these two models?



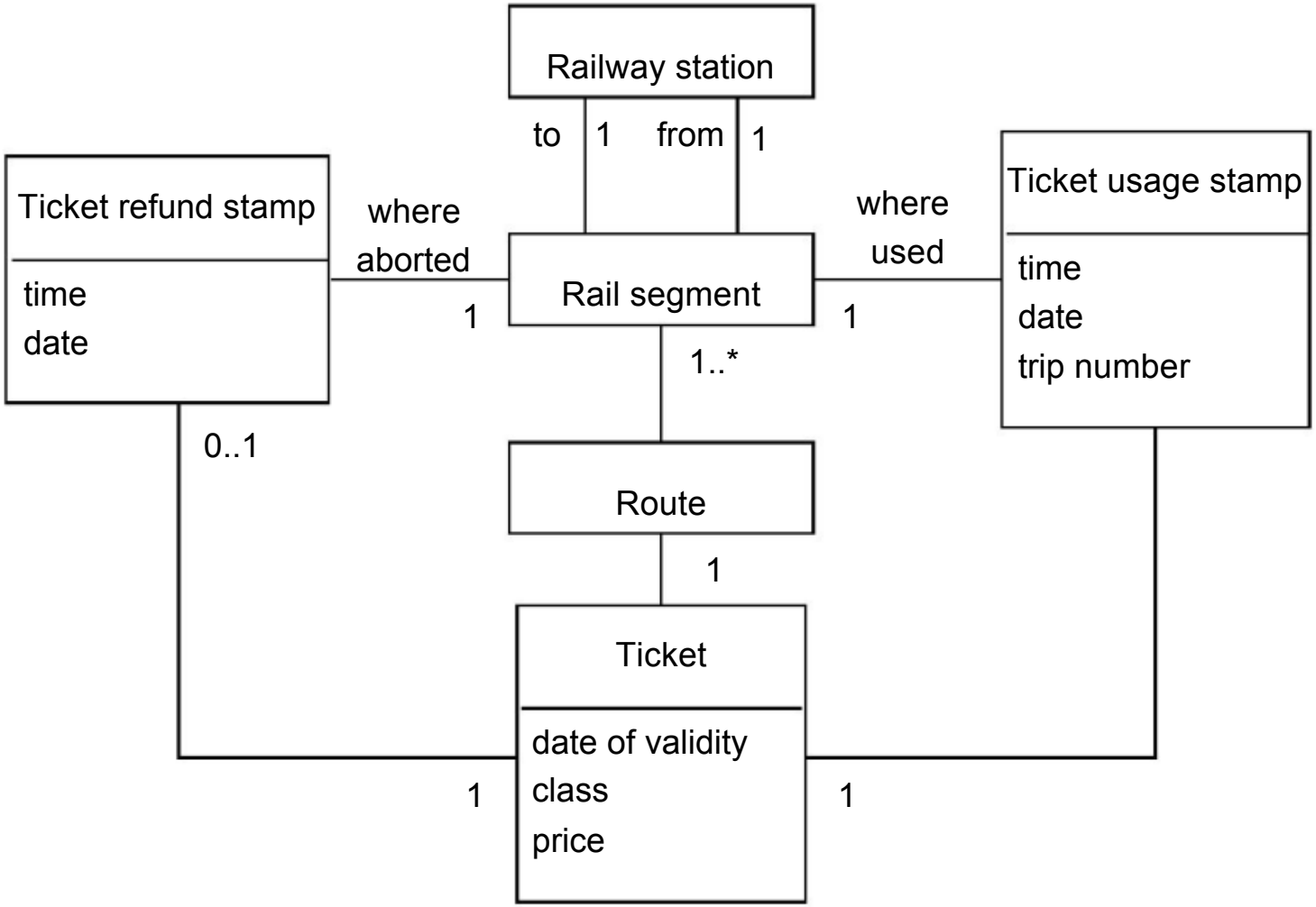
Validation of an ERD (1)

√ Check consistency of cardinality properties.



If a diagram is not consistent, it cannot represent any subject domain.

Validation of an ERD (2)



Are all loops commutative (any starting / destination node)?

Validation of an ERD (3)

√ Describe by elementary sentences and show to domain specialist.

At each moment:

- A batch of juice has exactly one recipe.
- A recipe may be applicable to any number of batches.
- At any point in time, a heating tank has at most one batch allocated to it.
- At any point in time, a batch is allocated to either 1 or 2 heating tanks.
- Each heating tank has exactly one heater.
- etcetera.

Validation of an ERD (4)

√ Describe snapshot and show to domain specialist.

Ticket	Route	Segment	Usage stamp	Refund stamp
(t1, April 2)	Return Enschede-Apeldoorn	Enschede-Apeldoorn	March 31	None
(t1, April 2)	Return Enschede-Apeldoorn	Apeldoorn-Enschede	April 1	April 1

Flaws in the model:

- Ticket cannot be used before bought!
- No usage and refund for the same segment!
- Return must be on the same day!

These are subject domain properties.

Validation of an ERD (5)

√ Check against messages entering and leaving the system.

Or against service descriptions:

- **Name:** Sell a ticket.
- **Triggering event:** Traveler requests to buy a ticket.
- **Delivered service:** Allow a traveler to buy a ticket at any time and place chosen by the traveler.

- **Name:** Show a ticket.
- **Triggering event:** Traveler requests to view a ticket.
- **Delivered service:** Display ticket attributes to the user.

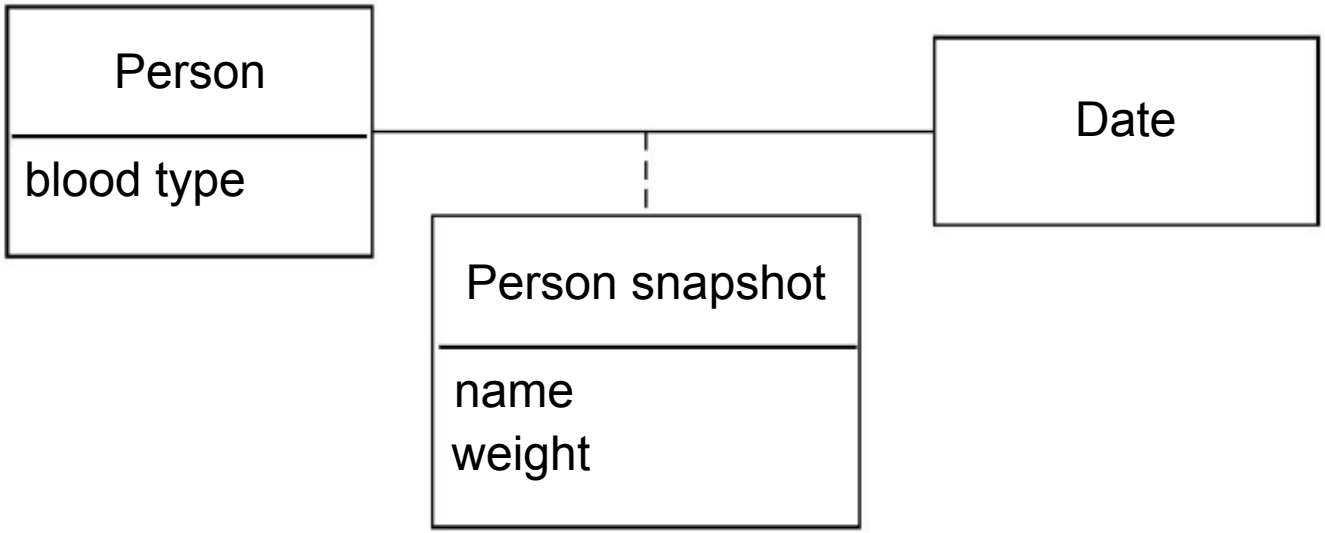
Why is "Traveler" not an entity in the subject domain?

Main points

- Subject domain bounded by the topic of the conversation with SuD.
- Entities have contingent properties, attributes do not.
- Relationships are identified by their components.
- Use specialization attribute.
- Static and dynamic specializations are distinguished by what happens at creation time.
- Classification and identification closely connected.
- Roles can be reified to entity types.
- Validate your ERD: consistency, elementary sentences, snapshots, check against interface of SuD.

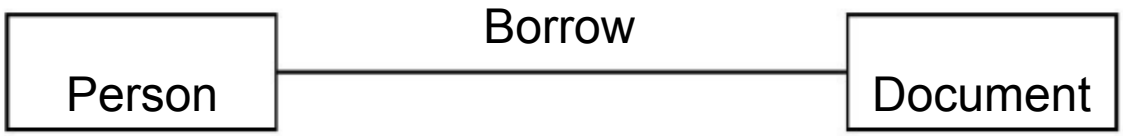
Chapter appendix (slides only): describing histories and events in an ERD

Histories

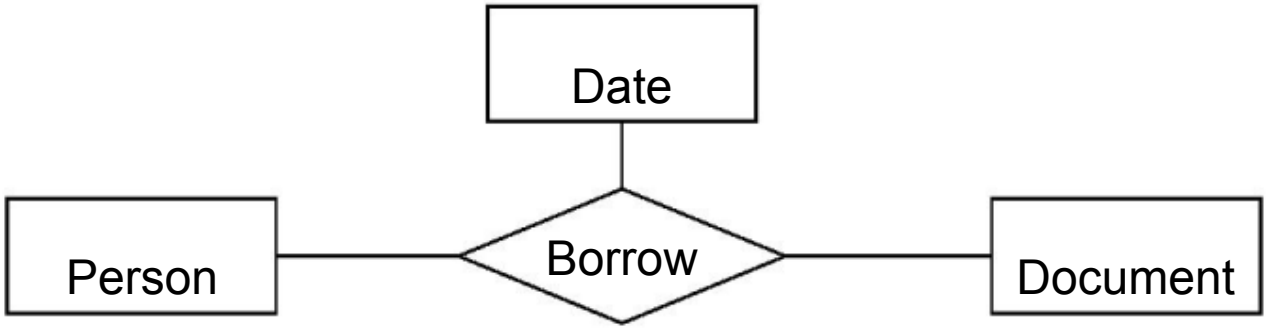


- Fixed attributes in the Person type box.
- Date-dependent attributes in an associative entity which represents the person at a certain date.

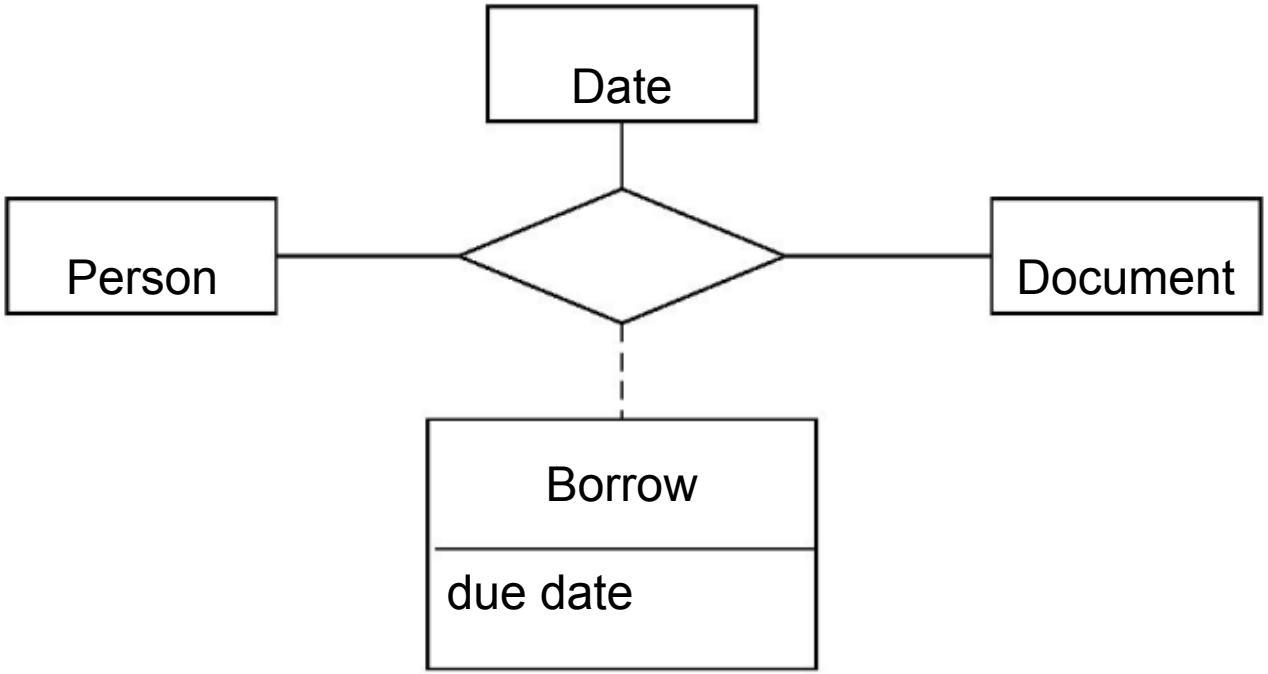
Events



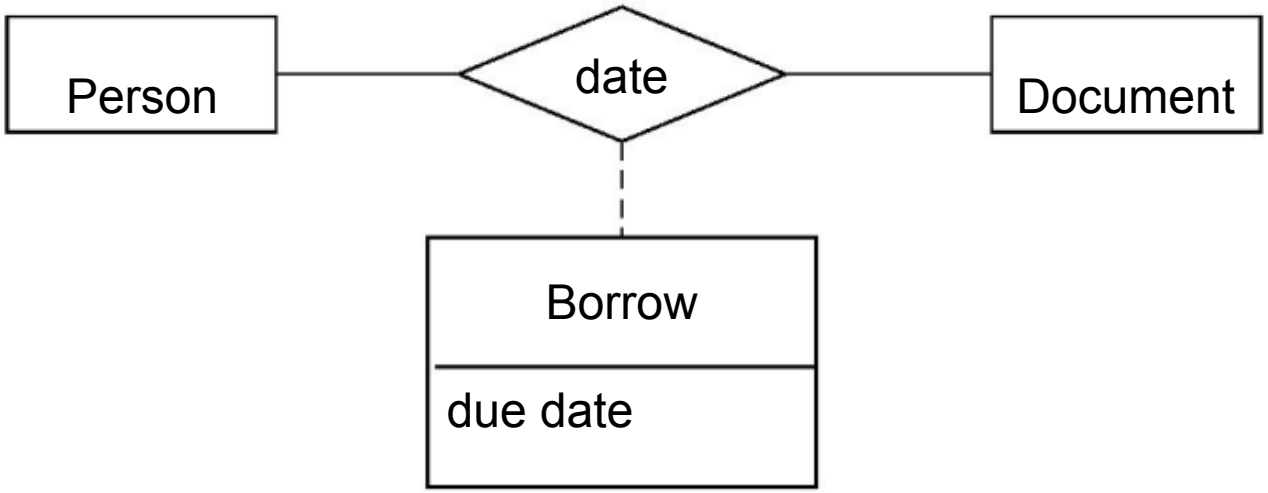
- Borrow is an event.
- An instance of Borrow exists for one moment only.



- Borrow is a dated event.
- An instance of Borrow represents the fact that at a certain date, a person borrowed a certain document.
- An instance of Borrow has identifier (p, dt, dc).
- We can include a time indication in the date too.



- Now Borrow has an attribute.



- Convention that abbreviates the diagram on the previous slide.

Chapter 10. The Dictionary

The dictionary documents the meaning of important terms:

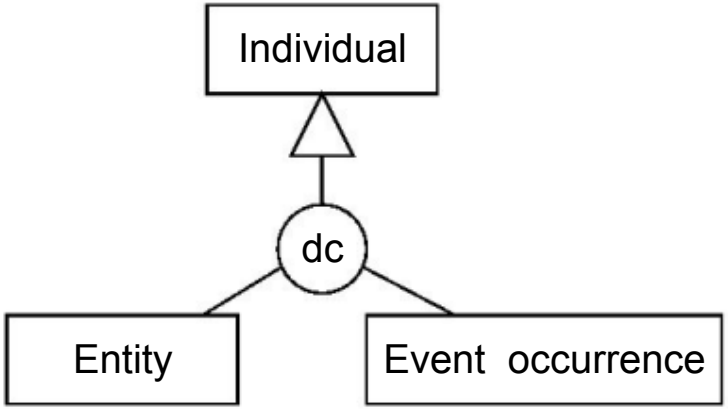
- Words used in service descriptions
- Words used in messages that cross the external SuD interface
- Domain-specific jargon
- etc.

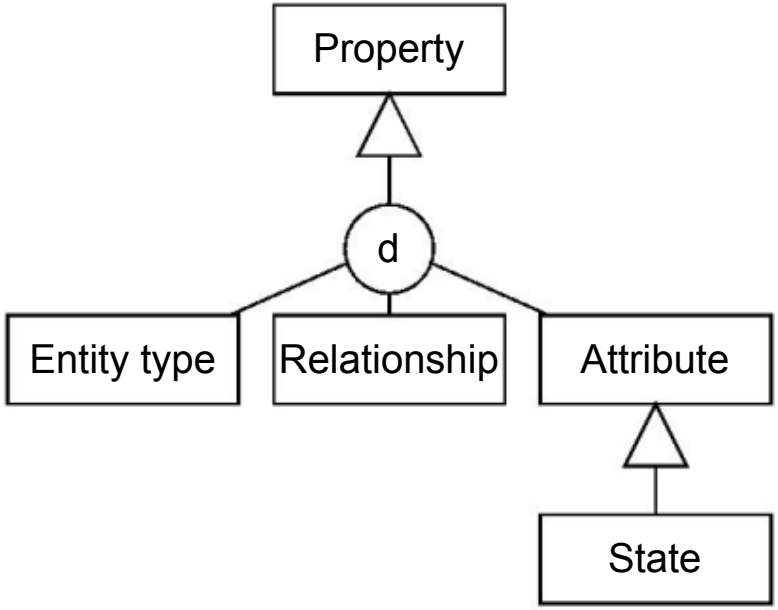
The subject domain ERD is a visual supplement to the dictionary, that adds some precision to terms that refer to subject domain entities.

In our dictionary, we use only a few syntactic categories of terms, that are motivated by our domain ontology.

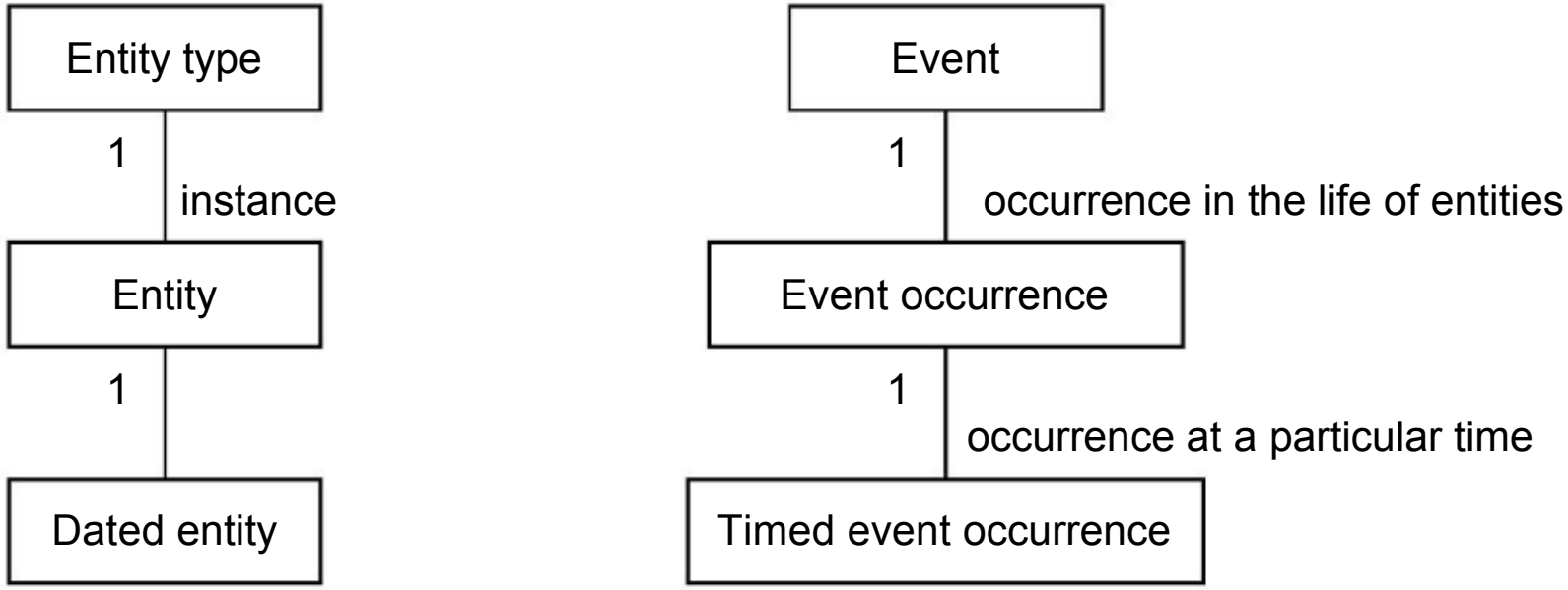
Domain ontology

- **Domain** is part of the world treated as a whole.
- **Ontology** is a metaclassification.





Individuals are entities and event occurrences



- No reincarnation of entities.
- Event occurrence can occur at many times.

  sell(coffee, 0.2 l) is an occurrence of event sell and it can occur at times 9:00, 9:02, etc.

Syntactic categories

- **Identifier.** Unique proper name.
- **Predicate name.**
  - *Entity type name.*
  - *Relationship name.*
  - *State predicate name.* Boolean property.
- **Attribute name.**
- **Event name.**

Definitions from the ETS specification

- **Railway station.** Entity type. Entity used to bound *rail segments.* May consist of platforms where passengers can enter or leave a train. Can also be a mathematical point used to bound a segment. Examples: The point where the line of rail passes the Dutch-German border; one can buy a ticket to that point. Can also be a collection of physical stations. Example, a ticket to "Berlin U-Bahn" is a ticket to any subway station in Berlin.
- **Rail segment.** Entity type. A shortest path between two Railway stations. Segments are directed and are identified by the two stations they connect. So all shortest paths through the rail network from A to B are considered the same segment, called AB; and all shortest paths from B to A are the same segment, called BA; and AB and BA are two different segments.
- **Route.** Entity type. A path through the rail network that consists of a connected series of *rail segments*, where each segment occurs at most once in the route.

Definitions from an elevator control specification

- **planned direction(c: Elevator cage).** Attribute. The preferred direction in which c will depart after closing its doors. If there are requests to be served higher and lower than the current floor of the elevator cage, then it will depart in the planned direction.
- **Round trip time.** The time in seconds for a single car trip around a building from the time the cage doors open at the main terminal until the doors reopen when the cage has returned to the main terminal floor after its trip [Barney & dos Santos 1977].
- **Atfloor(b: Request button, c: Elevator cage).** Predicate. c.current floor = b.floor.
- **continue(c: Elevator cage).** Action. Term is applicable only if c.planned direction = none.
  - If c.planned direction = up then start up(c.motor).
  - If c.planned direction = down then start down(c.motor).

  The motor of c is started in the planned direction of c.

Path expressions

Definitions may use path expressions, that refer to paths through the ERD.

See section 10.3 for syntax of path expressions.

Extensional and intensional definitions

- **Extensional definitions** list a few instances of the concept.

  Easy to give. But do not define an intension.

- **Intensional definition** lists the defining properties shared by all instance of the concept.

  Difficult to find.

Open-textured terms have no intensional definition.

√ Define these by giving a few examples, a sketch of the intent, and indicating the procedure —if any— that determines whether an instances falls under the concept.

Examples: Boat, title, document, vehicle, house, elevator, ...

When to define a term

- √ To clarify a term.
- √ To indicate that the term is open-textured.
- √ To indicate that we attach little meaning to it.
- √ The term is absolutely obvious to all stakeholders —but they attach different meanings to it.

When not to define a term

- √ To raise a cloud of obscurity. (Bad idea, but frequent practice.)
- √ Definitions can also be found in technical documentation of devices. Don't repeat these.
- √ The term is absolutely obvious to all stakeholders —and they understand the same by it.

Definition by genus and difference

Without genus	With genus
A compiler translates source code into object code.	A compiler is a program that translates source code into object code.
A catamaran has two hulls.	A catamaran is a boat with two hulls.
A heating tank heats juice.	A heating tank is a tank with a heater and thermometer in which juice can be heated.
Joiners recently joined the company.	A joiner is an employee that recently has joined the company.
A ticket represents the passenger's right to make a trip by train.	A ticket is a lexical item that represents the passenger's right to make a trip by train.

- √ The genus provides the identification criterion of the entity type.
- √ The difference provides the recognition criterion of the entity type.

Operational definitions

An operational definition of a term gives a procedure, that can be followed by anyone, to determine whether or not the term is correctly applied to a given case.

- **Heating tank.** Entity type. A tank with a heater and thermometer attached. The heater can be recognized by red, blue and black wires leading up to it, and the thermometer by its rectangular shape.

This is a operational definition by genus and difference.

- **Round trip time.** The time in seconds for a single car trip around a building from the time the cage doors open at the main terminal until the doors reopen when the cage has returned to the main terminal floor after its trip.

Operational, but not as definition by genus and difference.

Abbreviations versus correspondence rules

- **Abbreviation** reduces the meaning of a word to the meaning of other words defined in the same dictionary.
  To determine whether an individual is an instance of the defined concept, you do not need new observations but simply look up words in the dictionary.
- **Correspondence rules** relate a word to reality. The words in the definition are not defined in the same dictionary.
  To determine whether an individual is an instance of the defined concept, you must make observations.

Which definitions in the examples given are abbreviations, and which are correspondence rules?

Main points

- The dictionary contains definitions of jargon, subject domain terms, and other important tersm needed to understand the design specification.
- We can structure definitions as a taxonomic hierarchy: definition by genus and difference.
- Try to give operational definitions.
- Distinguish abbreviations (no new phenomenon described) from correspondence rules (link words to reality).