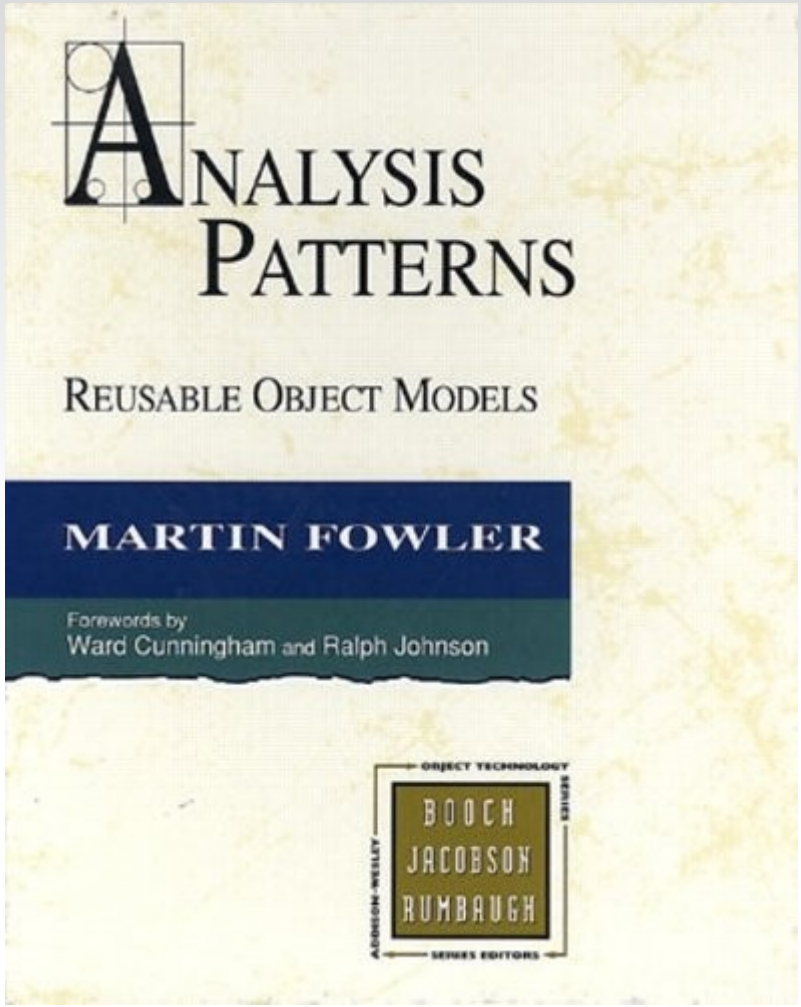
PA103 - Object-oriented Methods for Design of Information Systems

Analysis Patterns

© Radek Ošlejšek
Fakulta informatiky MU
oslejsek@fi.muni.cz

Literature

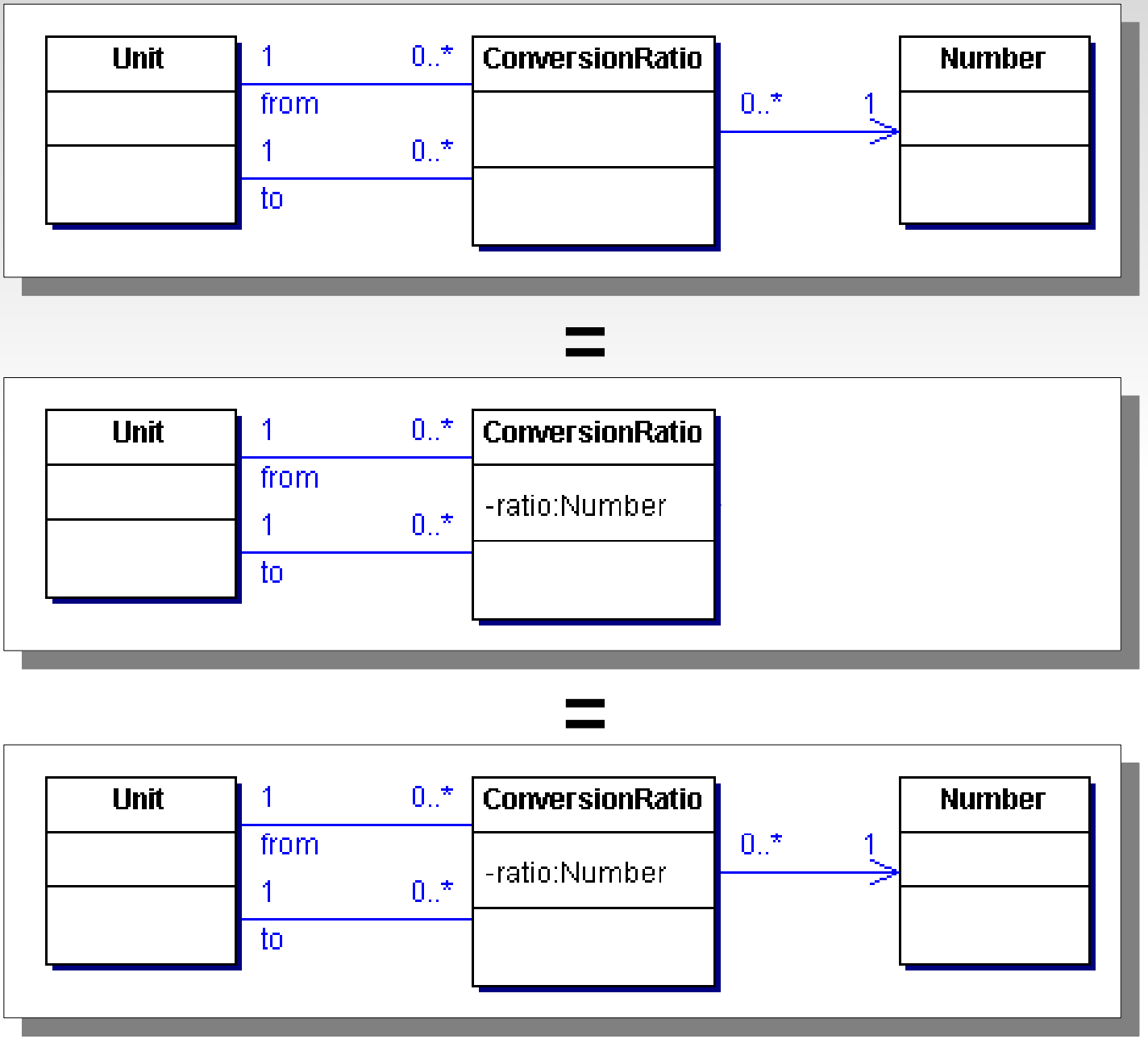


- Analysis Patterns: Reusable Object Models
  - Author: M. Fowler
  - Publisher: Addison-Wesley Professional
  - Copyright: 1996 (2010)
  - http://martinfowler.com
- Modeling Principle:
  - Models are not right or wrong; they are more or less useful.
  - Patterns are a staring point, not destination.

Analysis Patterns of M. Fowler

- 9 basic pattern collections covering various business domains:
  - **Accountability** – organizational structure and responsibilities
  - **Observations and Measurements** – measured values
  - **Observations for Corporate Finance** – analysis of complex financial relationships and results in companies
  - **Referring to Objects** – objects identification
  - **Inventory and Accounting** – cash flow and inventorying
  - **Planning** – schedules and protocols for repetitive plans
  - **Trading** – purchase and selling goods
  - **Derivative Contracts** – prices derived from the prices of other subjects
  - **Trading Packages** – trading for big companies
- Every collection consists of evolutionary sequence of patterns, from very basic form through combination of various aspects up to complex solutions
- 65 analysis patterns and 21 support patterns in total.

Note: Come to Realize that...



Lecture 6 / Part 1:
Accountability

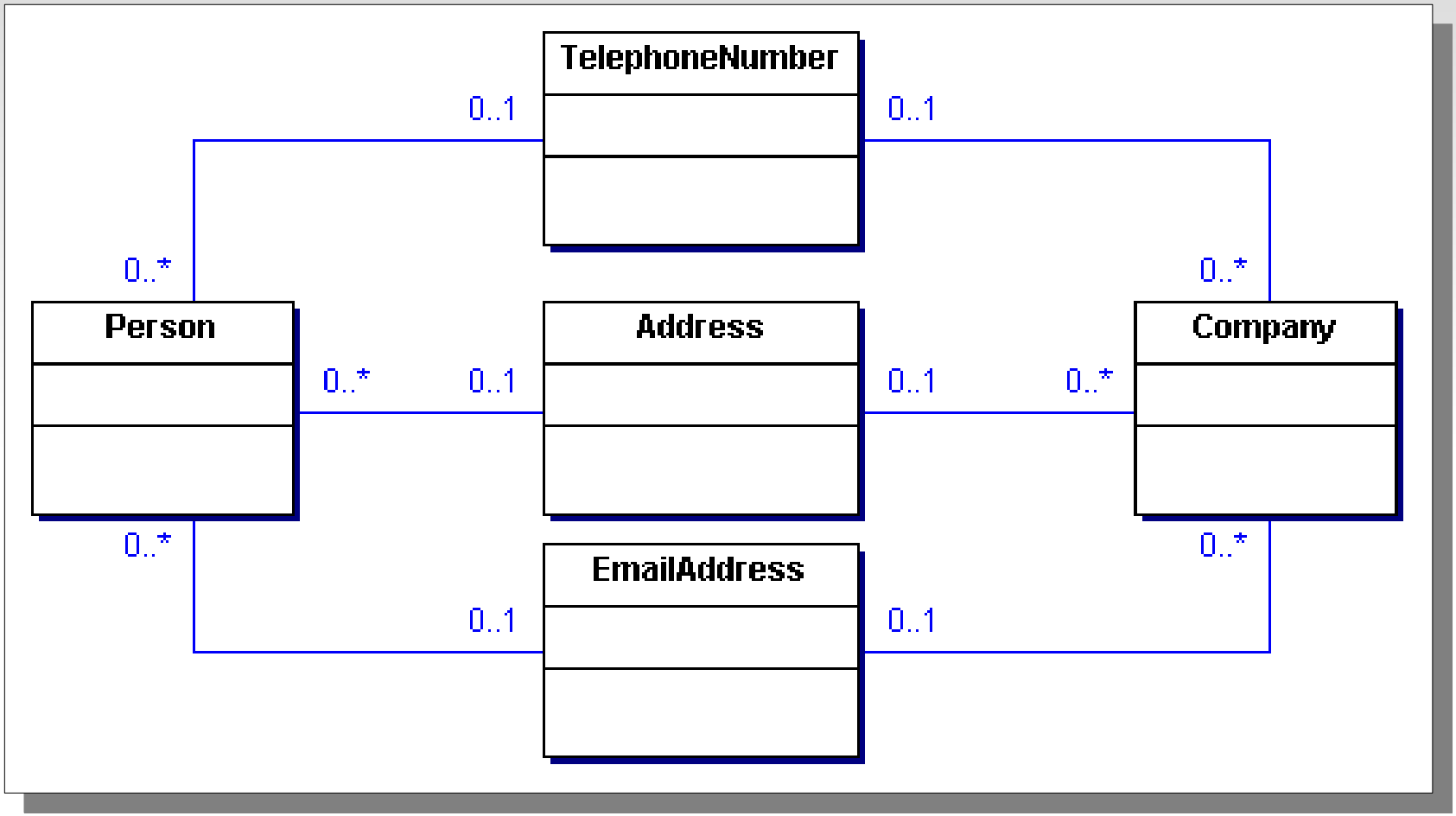
Accountability Patterns

- The concept of accountability applies when a person or organization is responsible to another.
  - Organization structure
  - Contracts
  - Employment
  - ...
- Often used in corporate information systems

Accountability: List of Patterns

- **Party** – key concept; supertype of person and organization
- **Organization Hierarchies** – simple organization hierarchy
- **Organization Structure** – complex organization structure
- **Accountability** – key concept; the combination of Party and Organization Structure
- **Accountability Knowledge Level** – rules that constrain accountabilities
- **Party Type Generalization** – hierarchy of party types
- **Hierarchic Accountability** – accountabilities for hierarchic and more complex networks of relationships
- **Operating Scopes** – responsibilities for parties
- **Post** – responsibilities delegated to a post, rather than to the person who occupies it

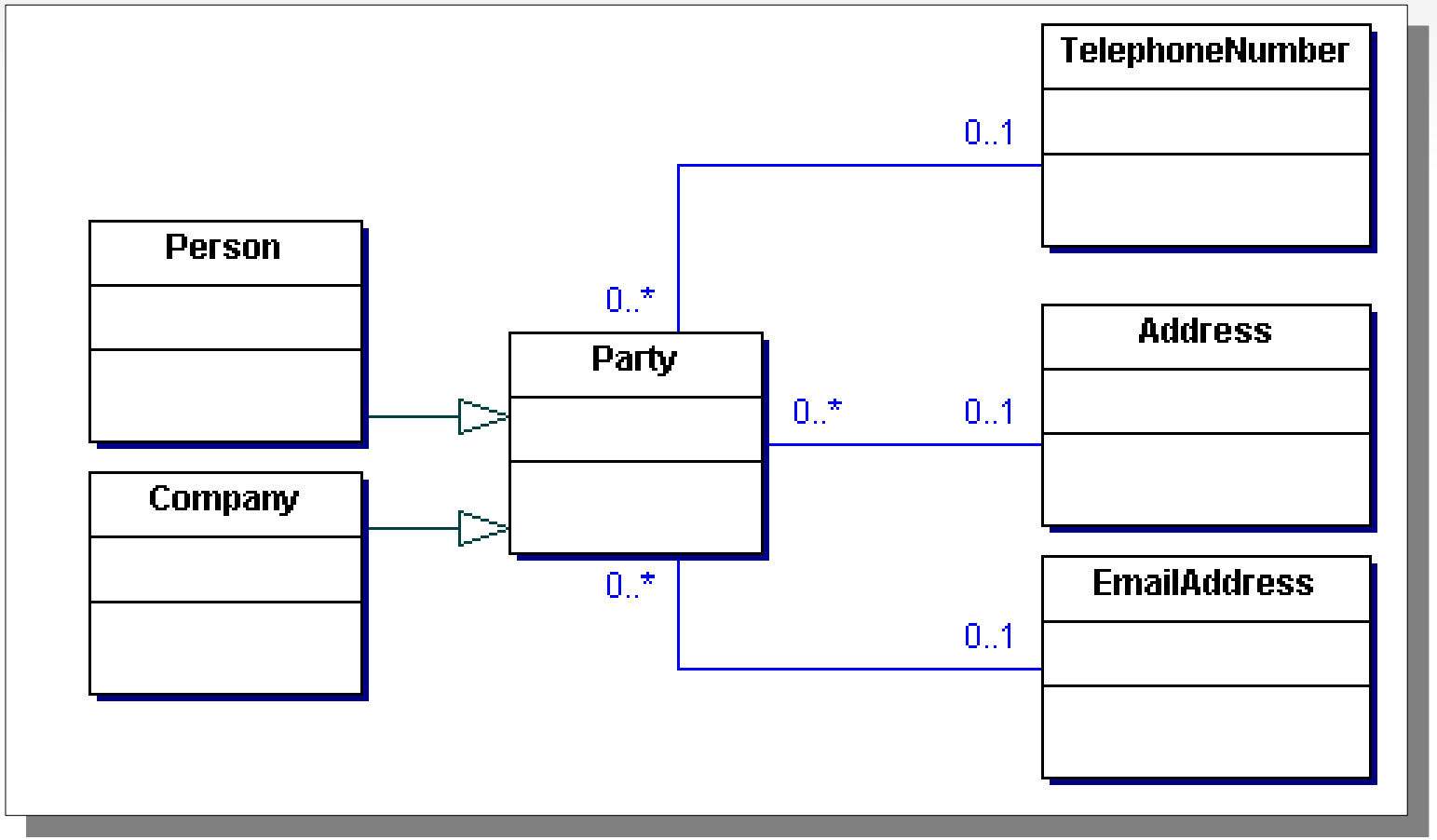
Party: Motivation



Initial model of an address book,
typical result of problem domain modeling

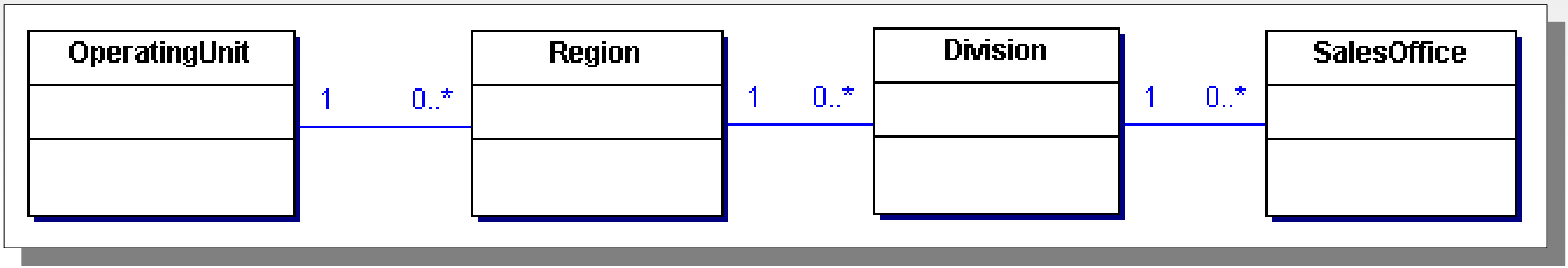
Party: Pattern

- Often the activities are performed by people, but once in awhile a company shows up => *Party* is just a unified name
- **Use *Party* when** you have people and organizations in your model and you see common behavior. However you should also consider this pattern when you don't need to distinguish between people and organizations. In this case it's useful just to define a party class and not to provide the subtypes.



Organization Hierarchies: Motivation

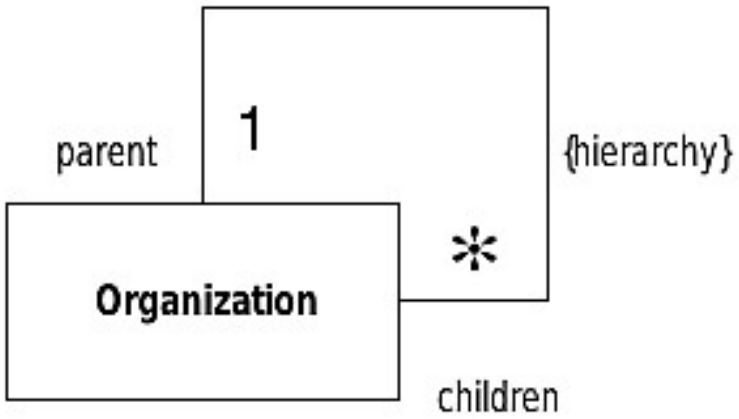
- … because companies have usually hierarchical organization structure.

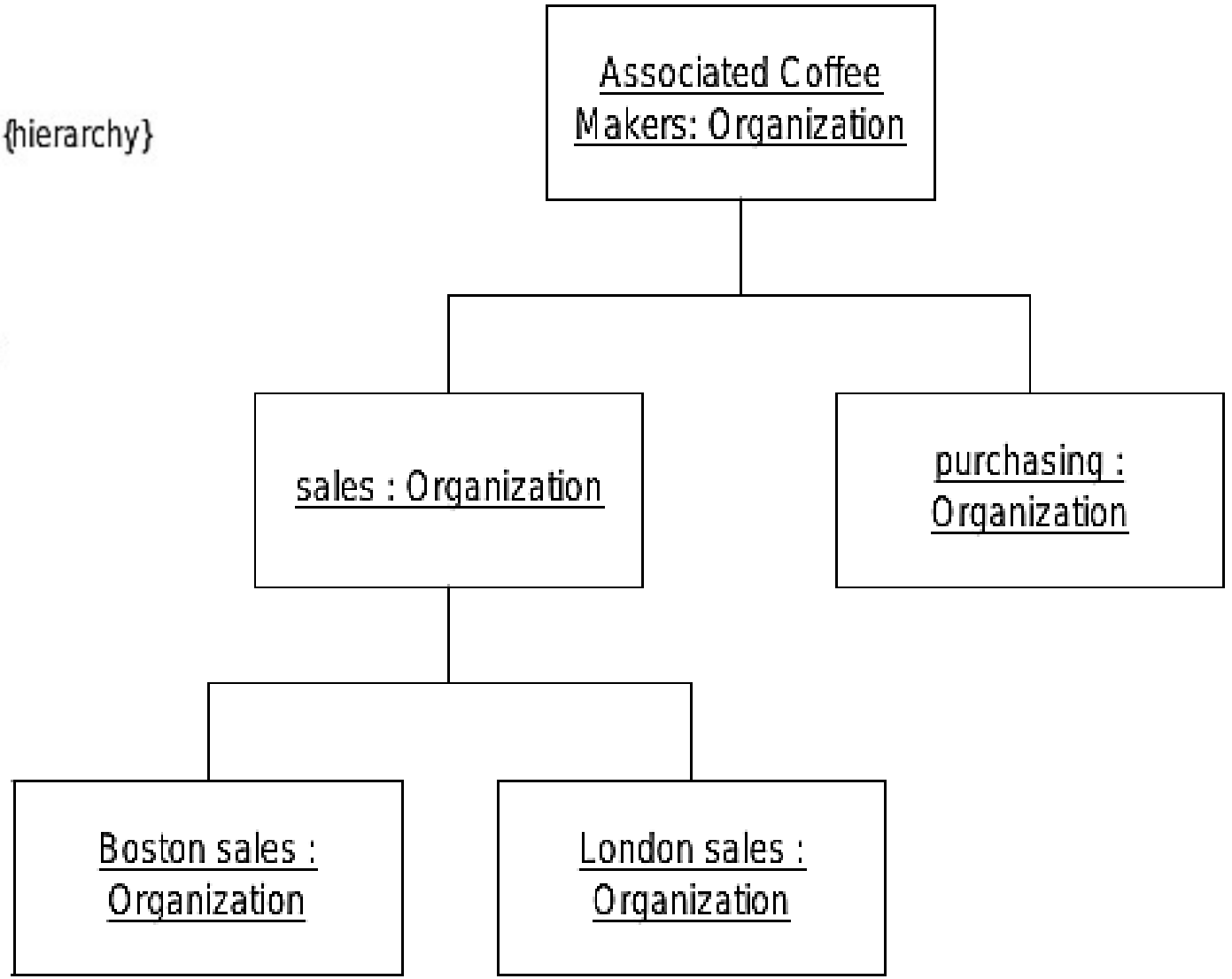


Example: organization structure of international company

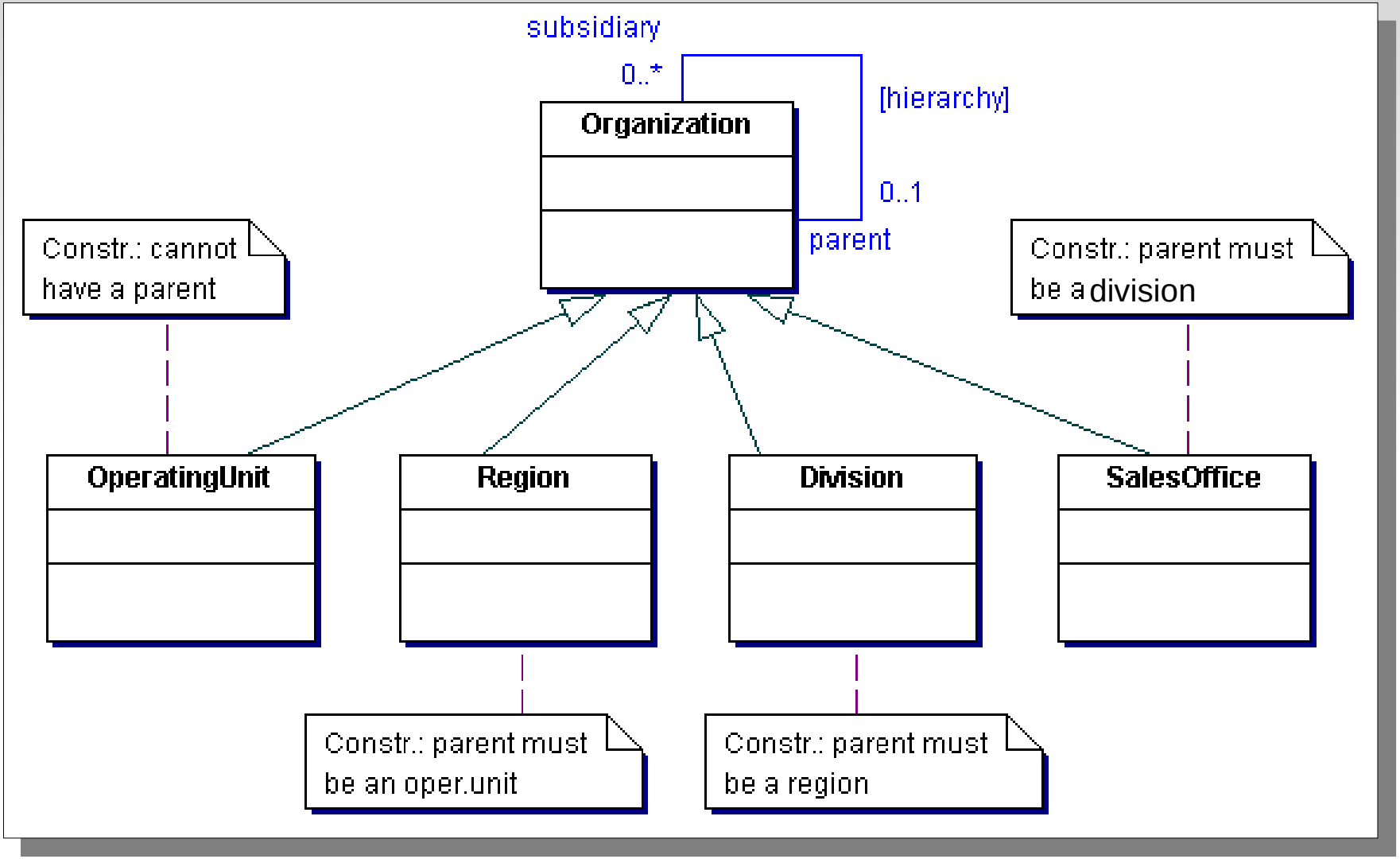
- Low flexibility: If the organization changes, e.g. Regions are taken out to provide a flatter structure, then we must alter the model.
- Solution: recursive relationships.

Organization Hierarchies: Pattern



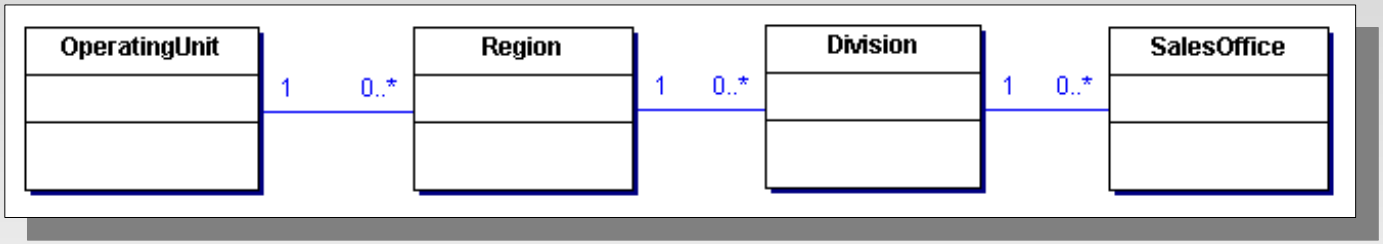


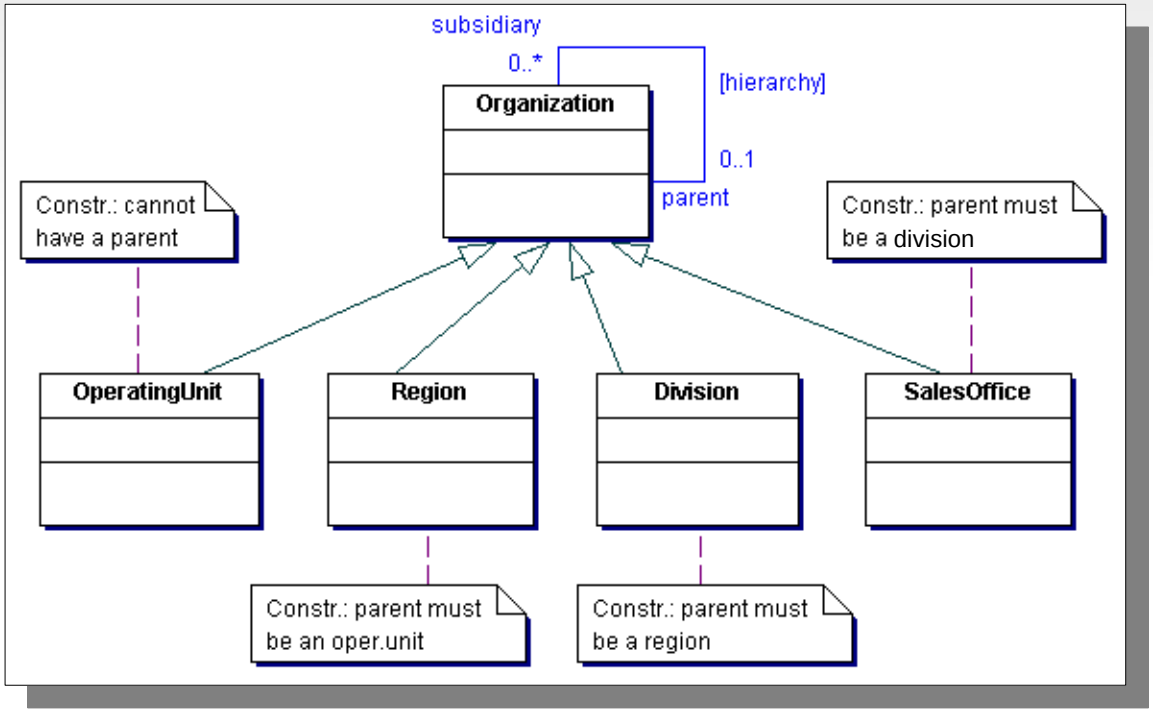
Organization Hierarchies: Pattern (cont.)

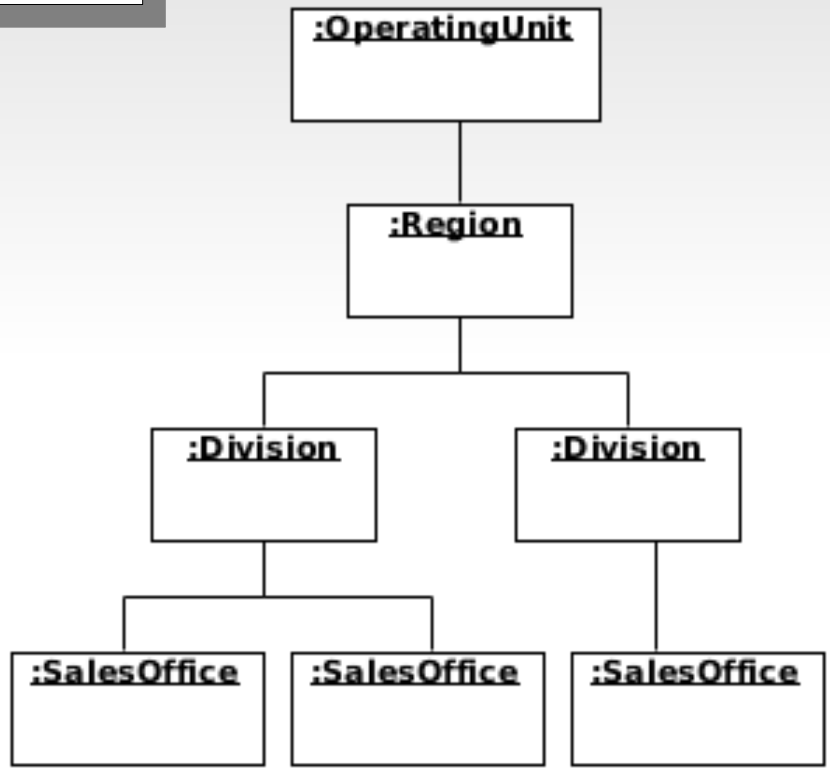


- The danger with the recursive relationship is that it allows a division to be part of a sales office => subclasses and constraints

Organization Hierarchies: Discussion

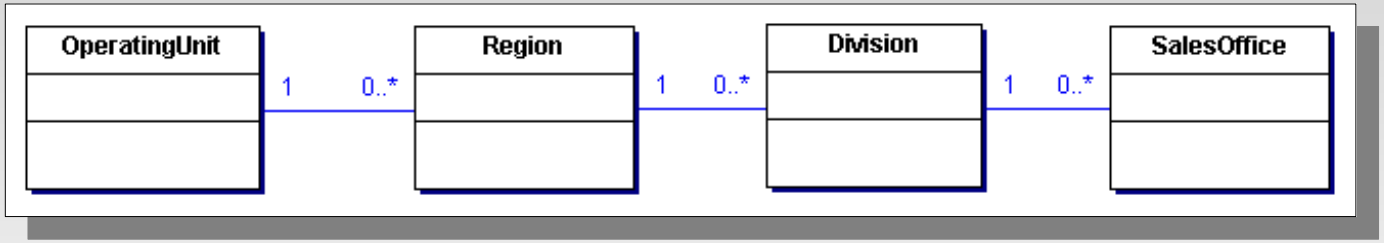


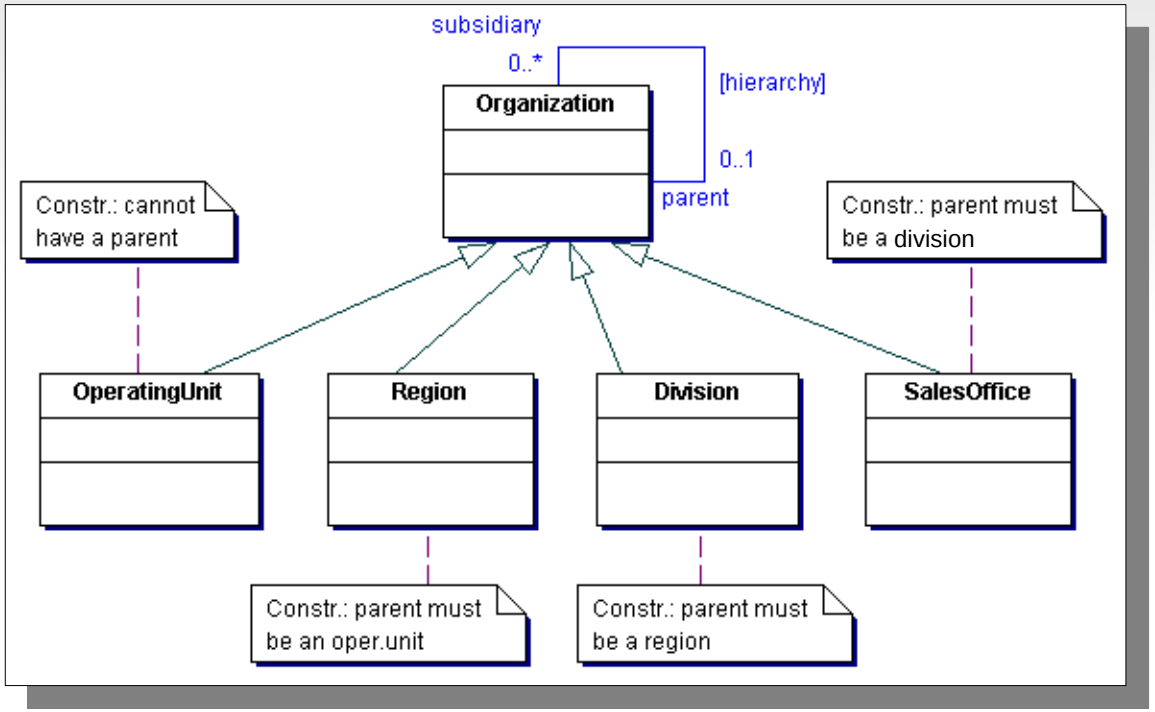




- **Q:** Having the object diagram on the right, which of the given class diagrams can instantiate this model?
- **A:** Both.

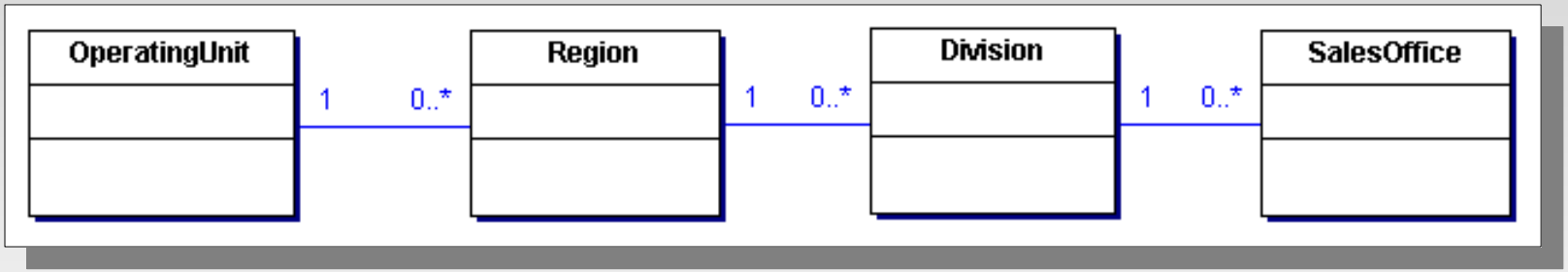
Organization Hierarchies: Discussion (cont.)



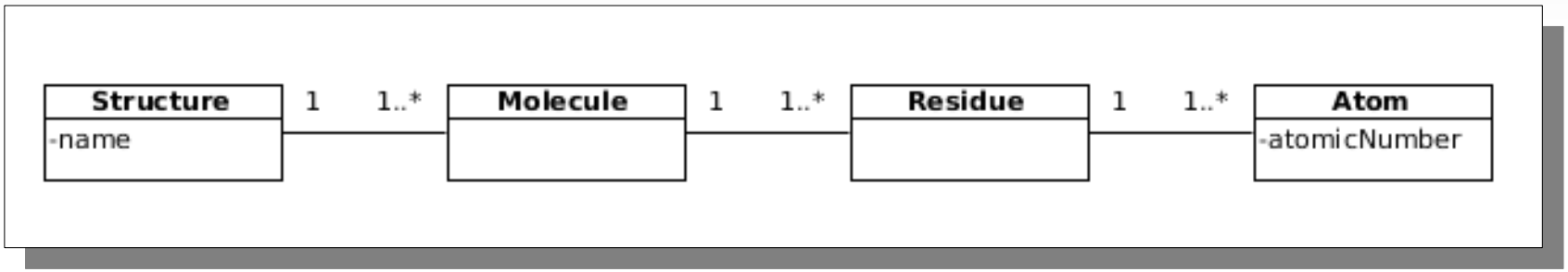


- **Q:** How easy it is to change the organization structure?
- **A:** Without pattern = changes in the model are required. With pattern = new subclass + new (OCL) rule

Where we already saw it?

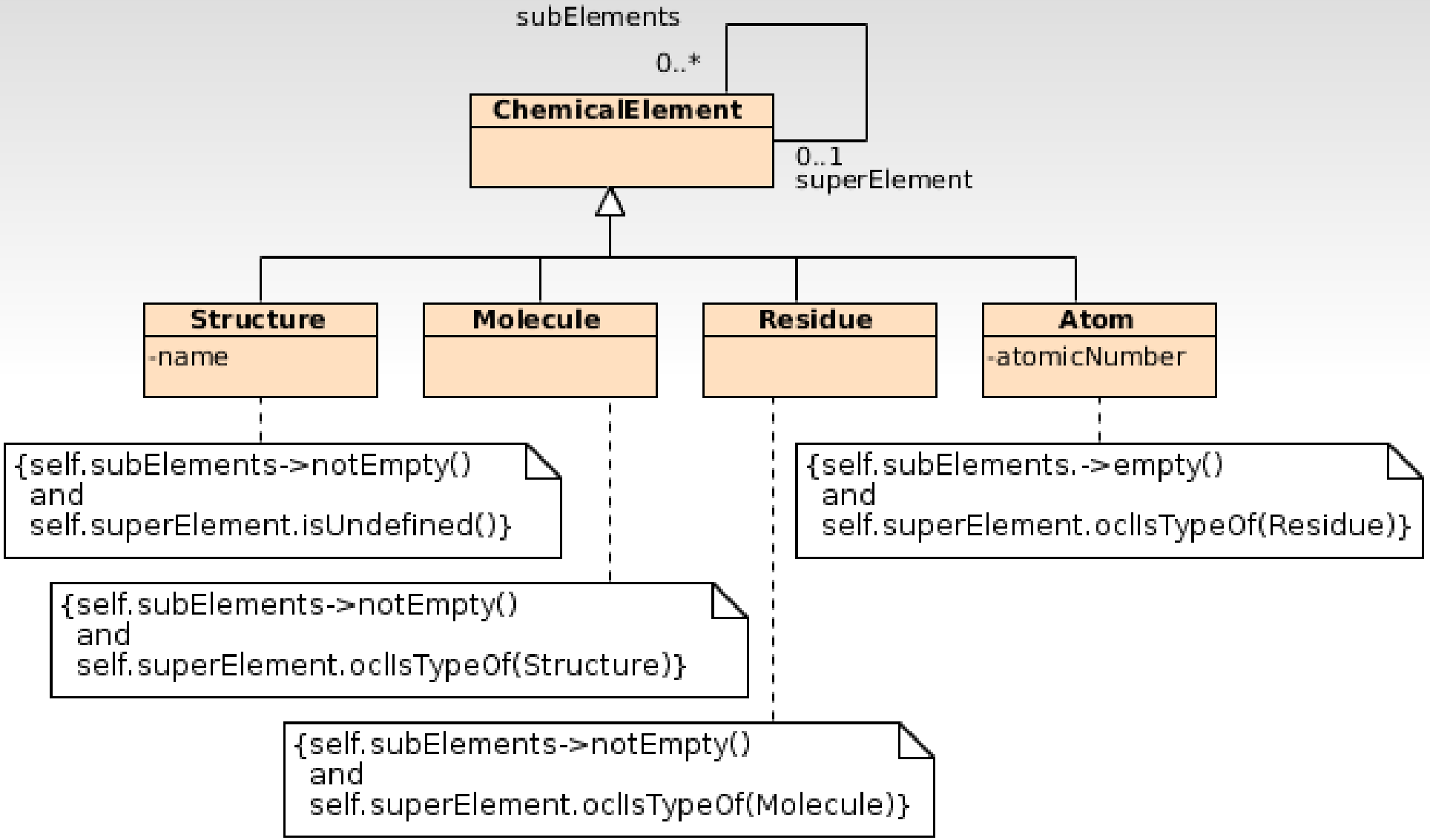


Where we already saw a similar hierarchical model?

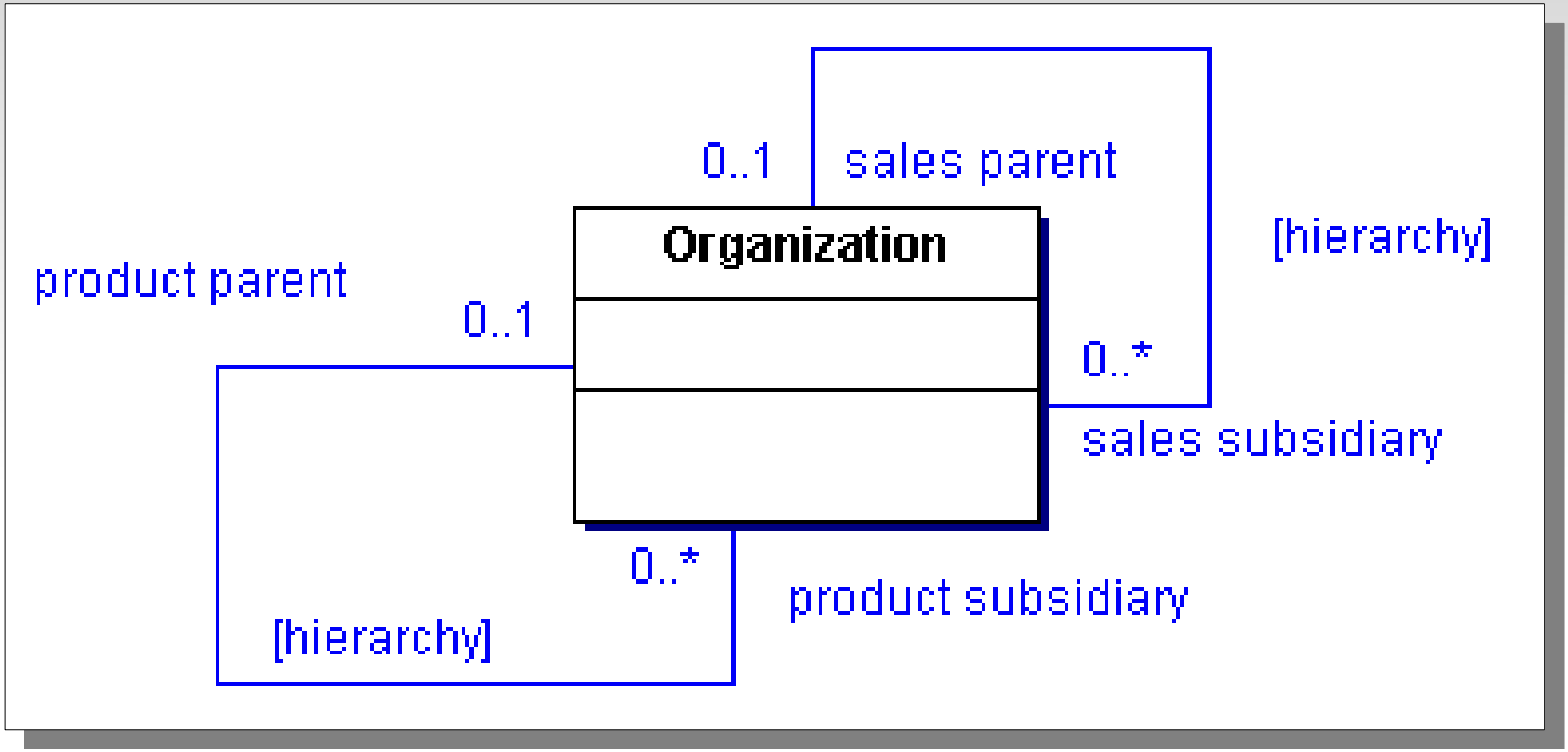


- Chemical structure!
  - Chemical structure leaded to the use of the *Composite* GoF pattern
  - => *Composite* pattern is a possible design solution enabling to distinguish between composite and leaf organizational units.

Example


subElements
0..*
ChemicalElement
0..1
superElement
Structure
-name
Molecule
Residue
Atom
-atomicNumber
{self.subElements->notEmpty()
and
self.superElement.isUndefined()}
{self.subElements->notEmpty()
and
self.superElement.oclIsTypeOf(Structure)}
{self.subElements->notEmpty()
and
self.superElement.oclIsTypeOf(Molecule)}
{self.subElements.->empty()
and
self.superElement.oclIsTypeOf(Residue)}

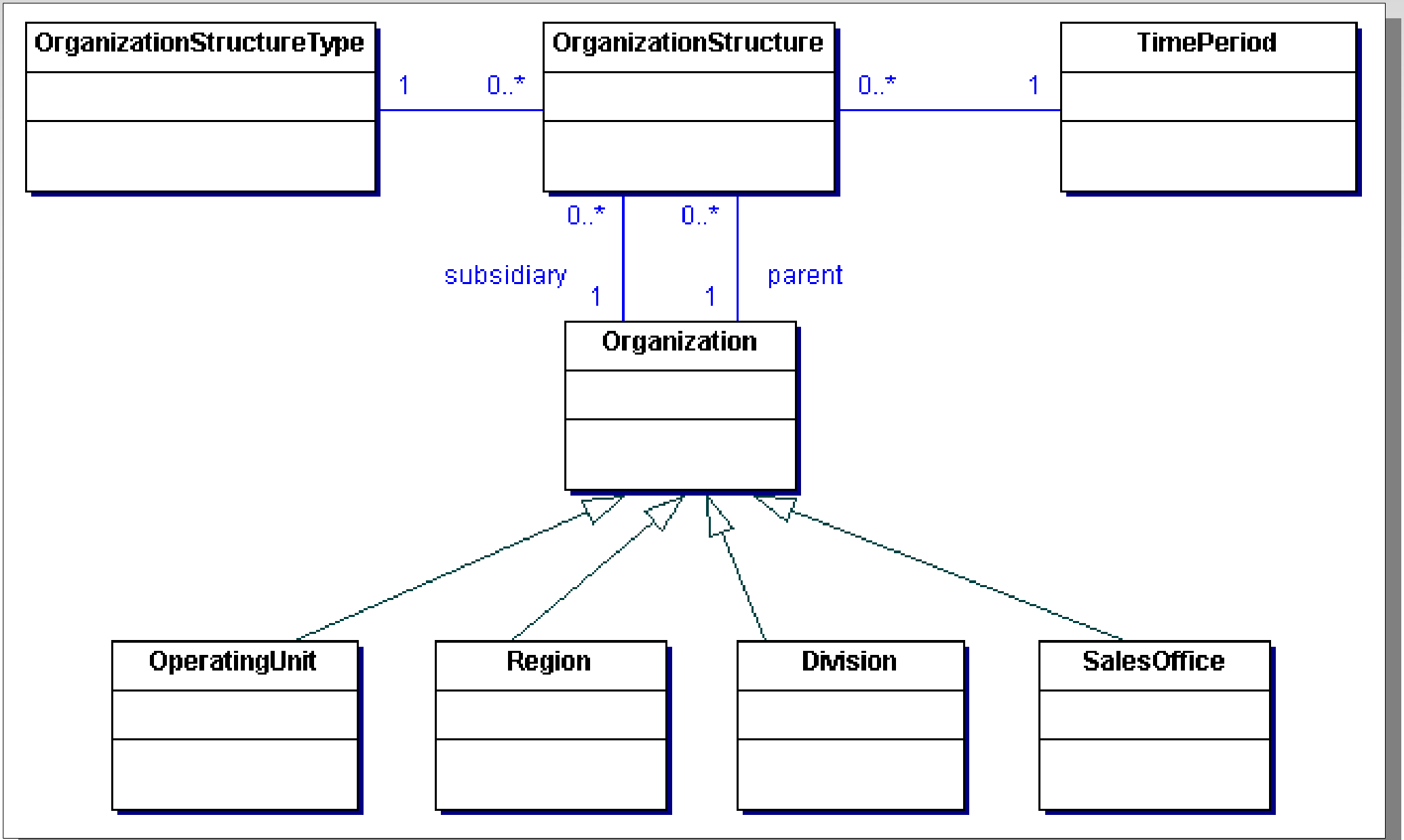
Multiple Organization Hierarchies



Example: two independent organizational hierarchies (production vs. service)

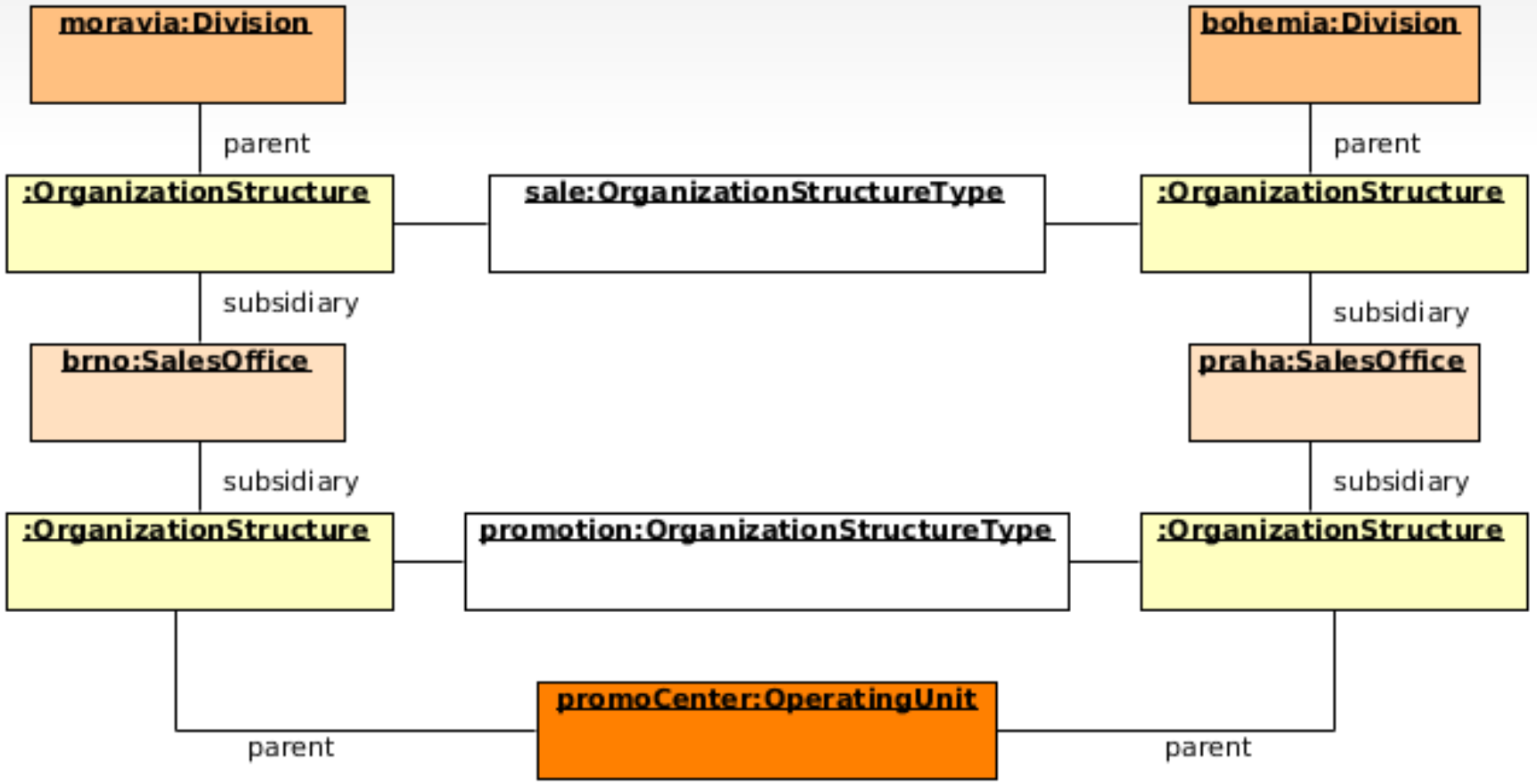
- **Note:** Subtypes of the organization are not shown.
- **Problem:** If there are many hierarchies, this will soon get out of hand.

Organization Structure: Pattern



Organization Structure: Example

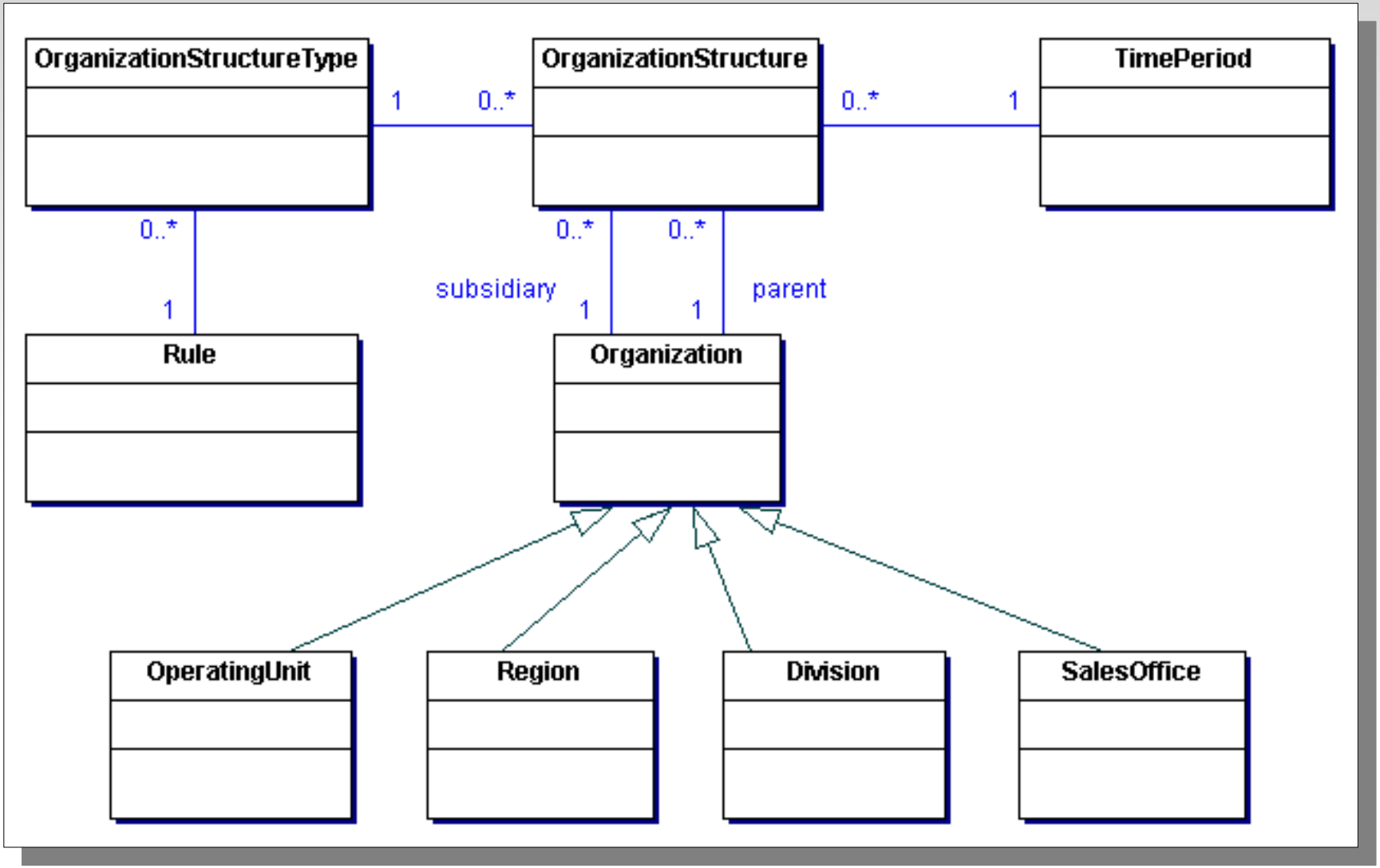
- Seller has two shops, in Brno and Prague.
- From the sales point of view, Brno sales office is subordinated to Moravia division, Prague sale office is subordinated to Bohemia division.
- From the PR point of view, both sale offices are subordinated to the common central office



Organization Structure: Properties

- Suitable for multiple organizational hierarchies.
- Adding a new hierarchy = adding a new instance of *OrganizationStructureType*
- *TimePeriod* allows us to record changes in the organization structure over time.
- Simplifying the object structure puts more emphasis on the rules.
  - e.g. „If we have an organization structure whose type is sales organization and whose child is a division, then the parent must be a region“
- Constraints to organization structure are expressed by referring to properties of the organization structure
  - => extending the system by adding a new organization structure type would require changing the rules in the organization structure.
  - Solution: Adding a *Rule* class.

Organization Structure: Rules



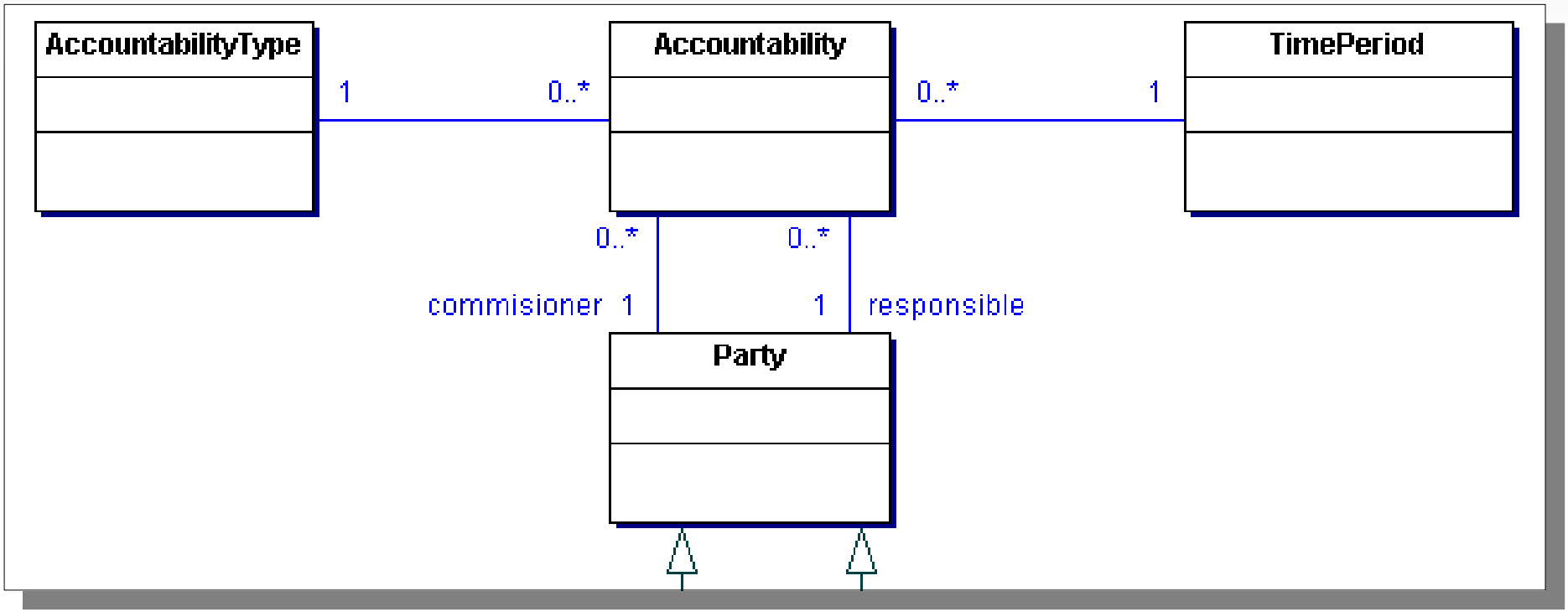
Note: Each addition of a subtype of organization would cause rule changes. Therefore, if we change the organization structure types rarely but add new subtypes of organization frequently then this model does not work well. In this case it is better to place the rules on the subtypes of the organization.

Accountability

- *Organization Structure* shows relationships between organization units. But people have often similar relationships either to organizations or other people as well.

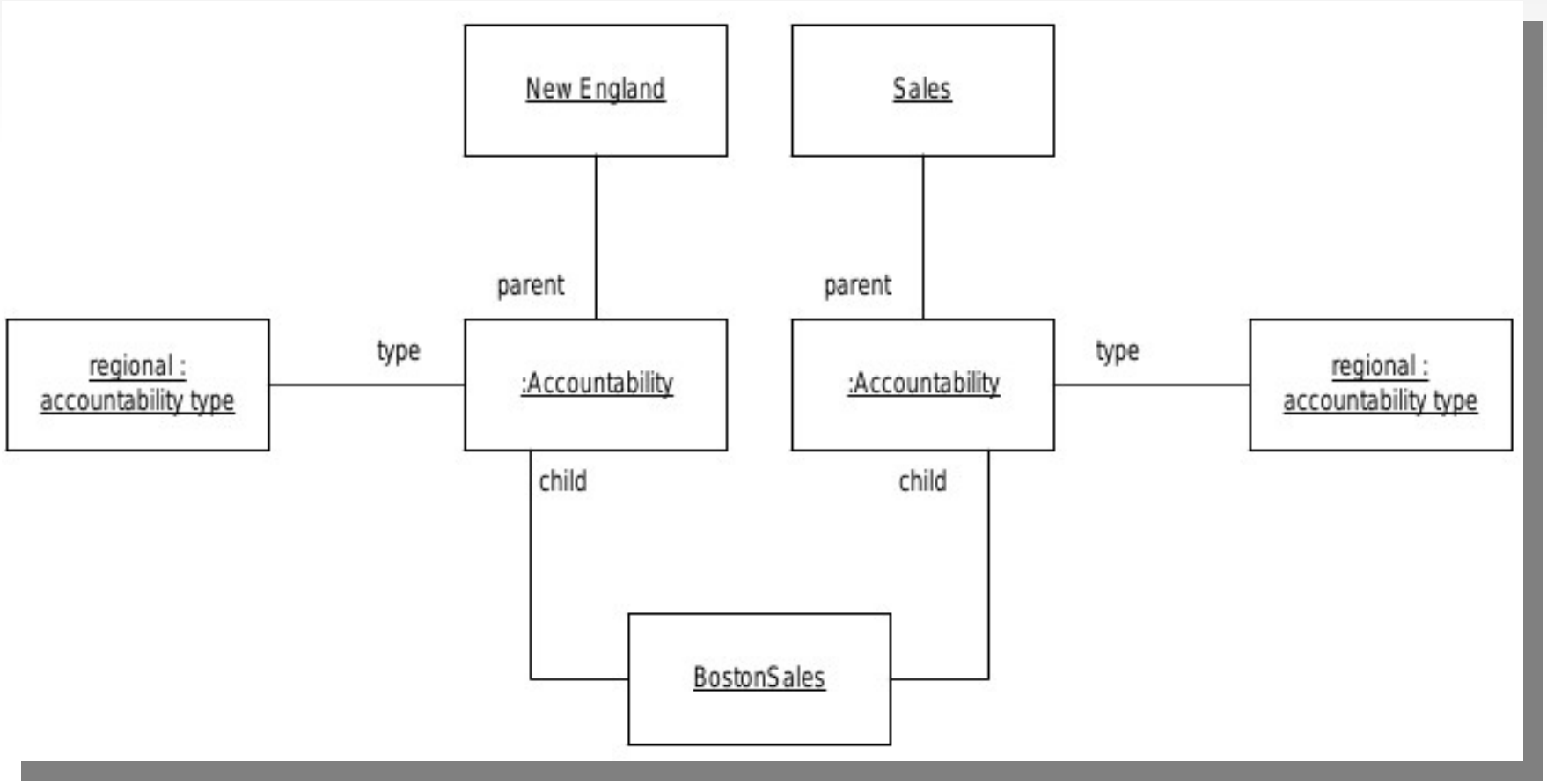
  => *Accountability* pattern combines the principles of *Organization Structure* and *Party*

- Each instance of *Accountability* represents a link between two parties, the *AccountabilityType* indicates the nature of the link. This allows you to handle any number of organizational relationships.



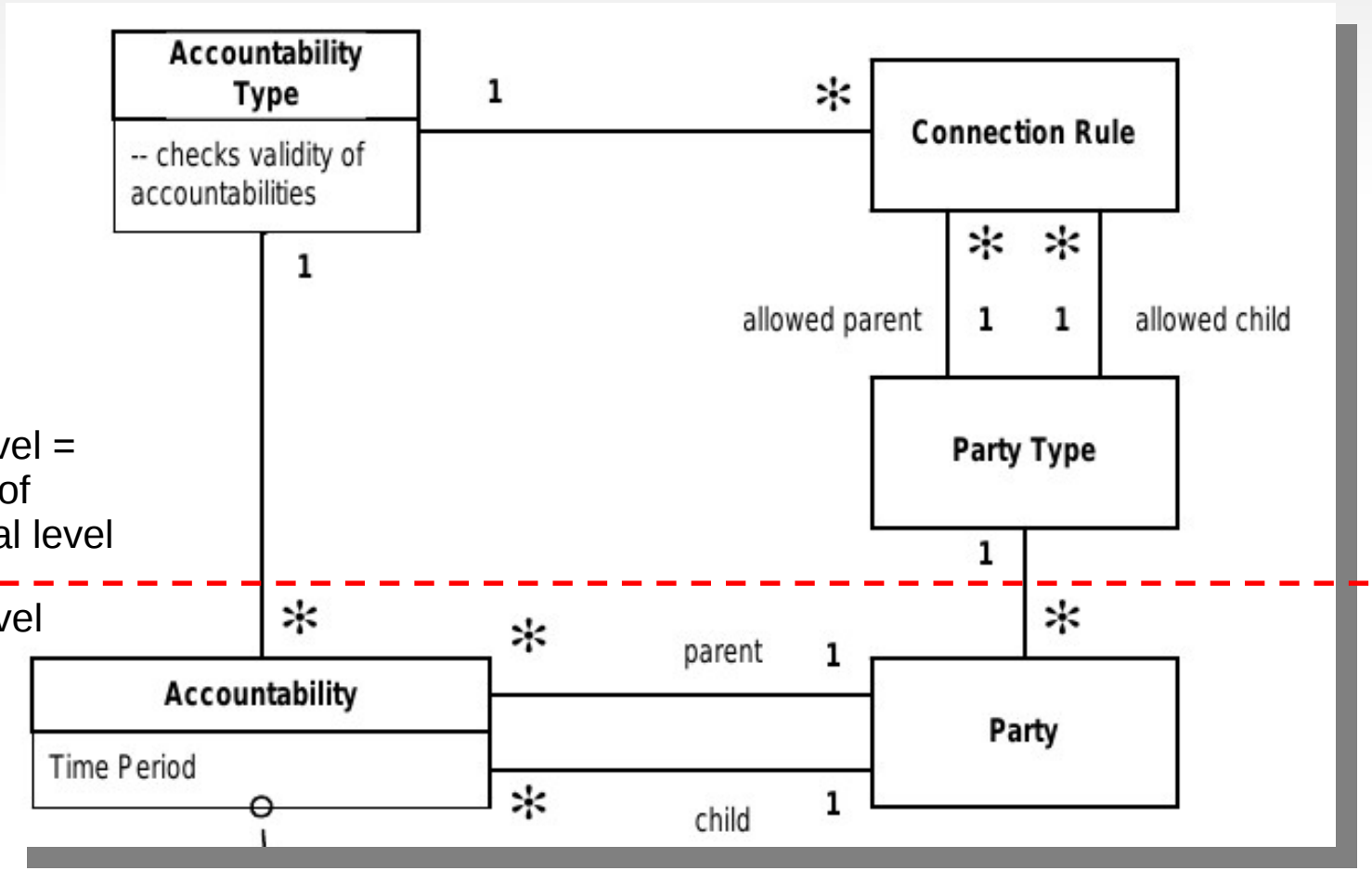
Accountability (cont.)

- Create an instance of accountability type for each kind of relationship you need, and connect together the parties with accountabilities of those types
- You can then use accountabilities to provide similar navigational behavior to what you have with *Organization Hierarchy*. The main difference is that much of this behavior is now qualified by the accountability type. So instead of saying “who are the parents of Boston Sales” you say “who are the regional parents of Boston Sales”.

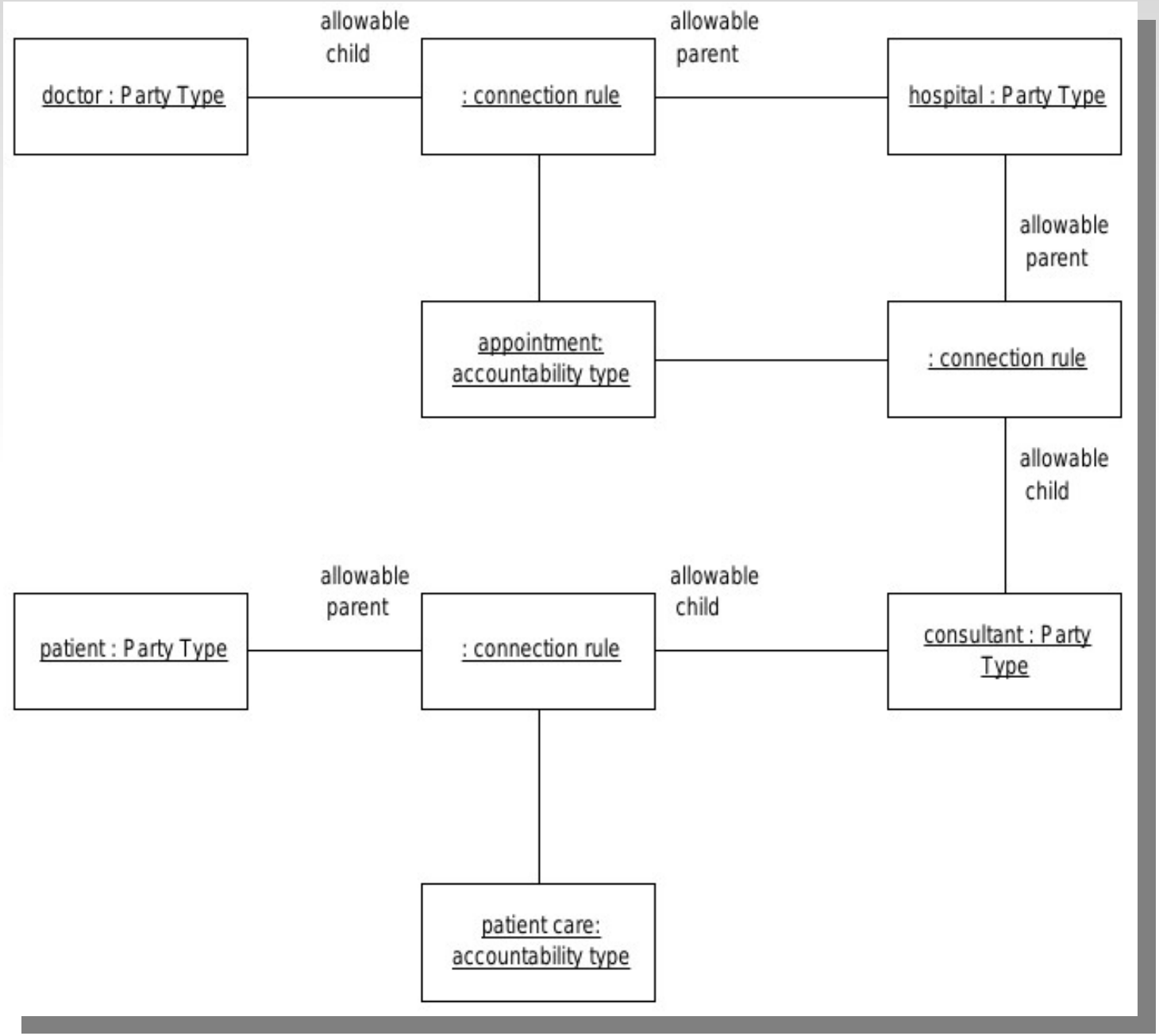


Accountability Knowledge Level

- If there are many more accountability types than there would be organization structure types => introducing knowledge level.
- Connection rules provide a simple yet very flexible form of knowledge level.
- Each accountability type contains a group of connection rules each of which defines one legal pairing of parent and child party types.



Accountability Knowledge Level (cont.)



Accountability: Further Patterns

- Party Type Generalizations
  - Adding generalization to party types makes it easier to define the knowledge level
- Hierarchic Accountability
  - Improves handling of constrains in organization structures.
- Operating Scopes
  - Define the responsibilities that are taken on when an accountability is created.
- Post
  - Another party type. Posts are used when accountabilities and scopes are defined by the post and do not change when the holder of the post changes.

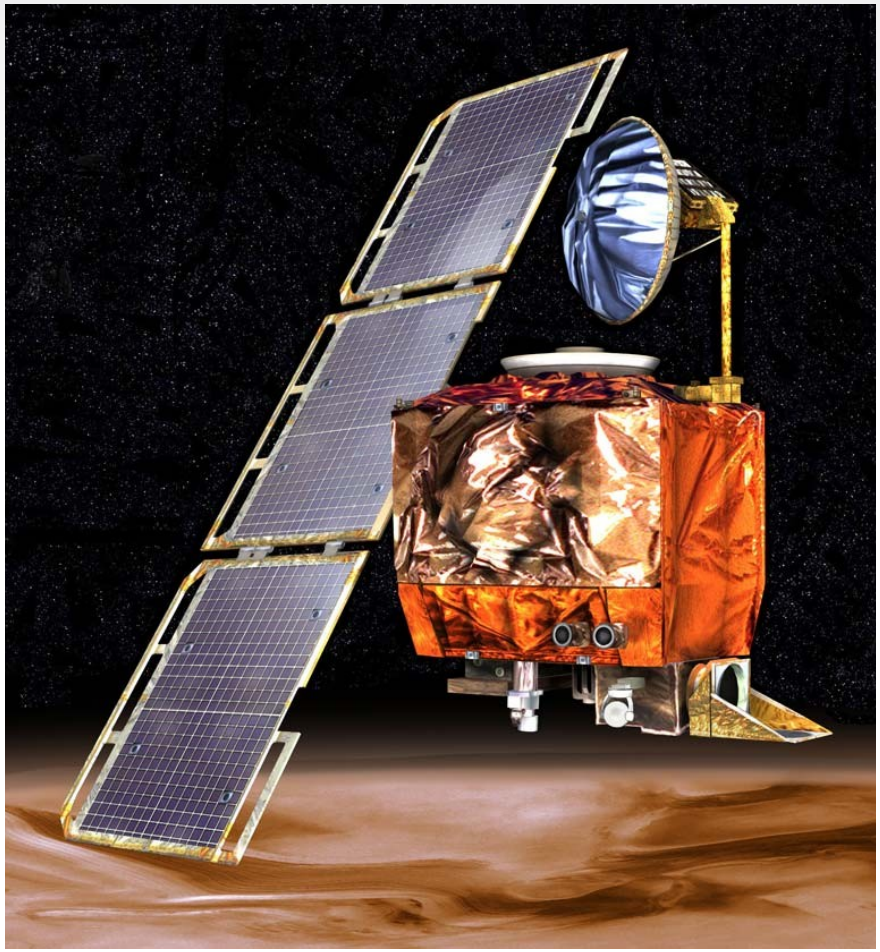
Lecture 6 / Part 2:
Observations and Measurements

Observations and Measurements

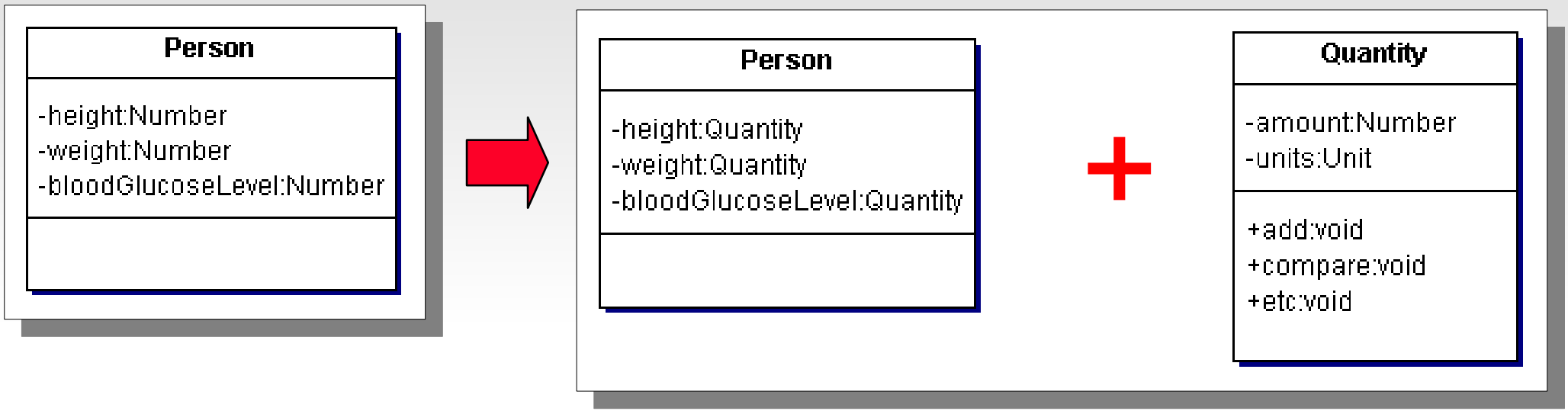
- Storage and maintenance of quantitative and qualitative data
  - Example: height, weight, blood pressure, eyesight. etc. of patients
- Sophisticated alternative to numerical attributes
- Patterns:
  - **Quantity** – values and their units
  - **Conversion Ratio** – conversion ratios between units
  - **Compound Units** – compound units :-) e.g. km/h
  - **Measurement** – measurement and recording of obtained values
  - **Observation** – general observations and their recording
  - **Subtyping Observation Concepts**
  - **Protocol** – protocols of regular measurements
  - ...

Quantity: Motivation

- The Mars Climate Orbiter was a robotic space probe launched by NASA to study the Martian climate
- On September 23, 1999, communication with the spacecraft was lost as the spacecraft went into orbital insertion, due to ground based computer software which **produced output in pound-seconds instead of the metric units of newton-seconds**.
- The cost of the mission was $327.6 million

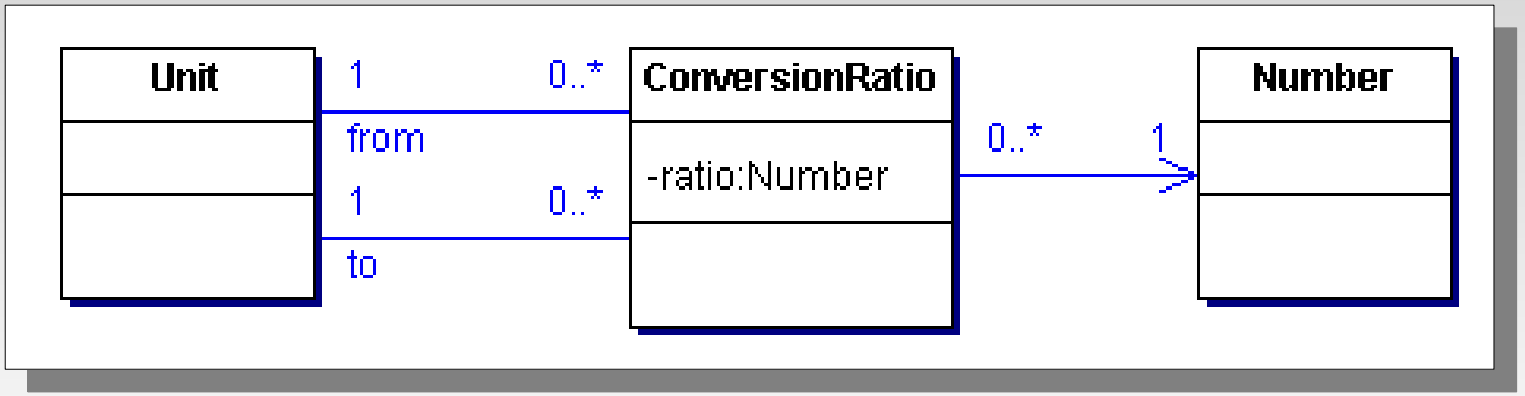


Quantity: Pattern



Question: What means the *weight* value? kg, g, ounce, something else?
Principle: By combining numbers and their units modeled as objects, we can describe how to convert quantities with conversion ration, etc.

Conversion Ratio: Pattern



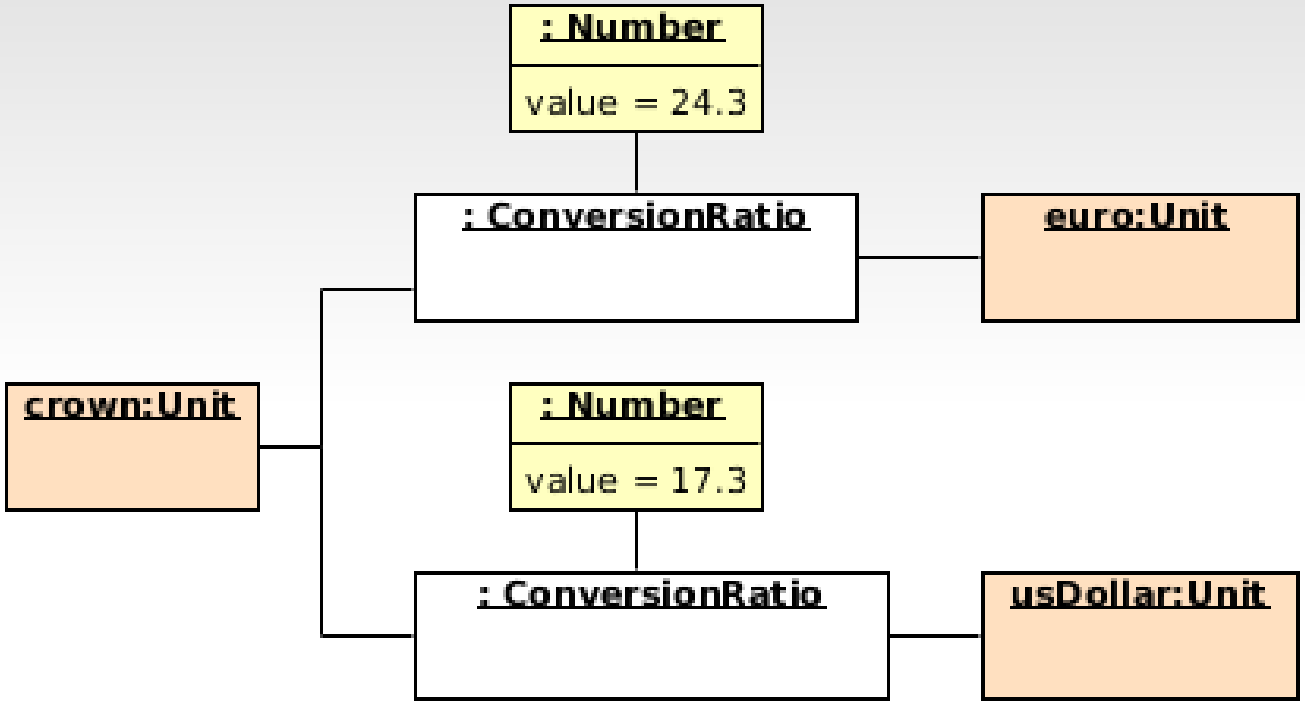
- Suitable to convert quantities with conversion ration.

- **Problem:** How to convert non-linear units, e.g. Celsius to Fahrenheit, days to months, etc.?
- **Solution:** Employ the *Individual Instance Method* pattern, e.g. replace *ConversionRatio* with *Strategy* GoF pattern.

- **Problem:** How to convert compound units, e.g. km/h to m/s?
- **Solution:** Next pattern.

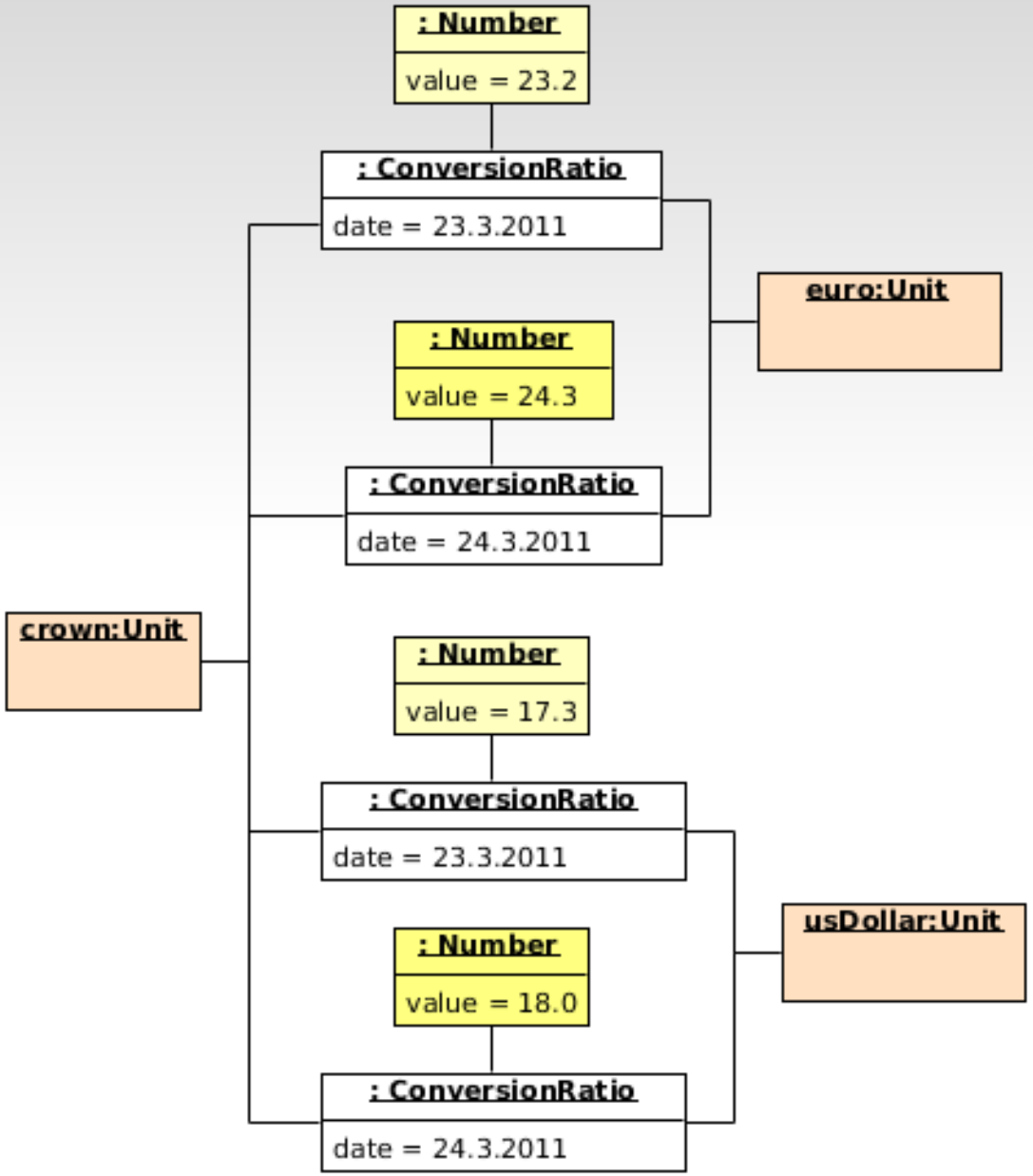
- **Task:** Create exchange table

Conversion Ratio: Exchange Table



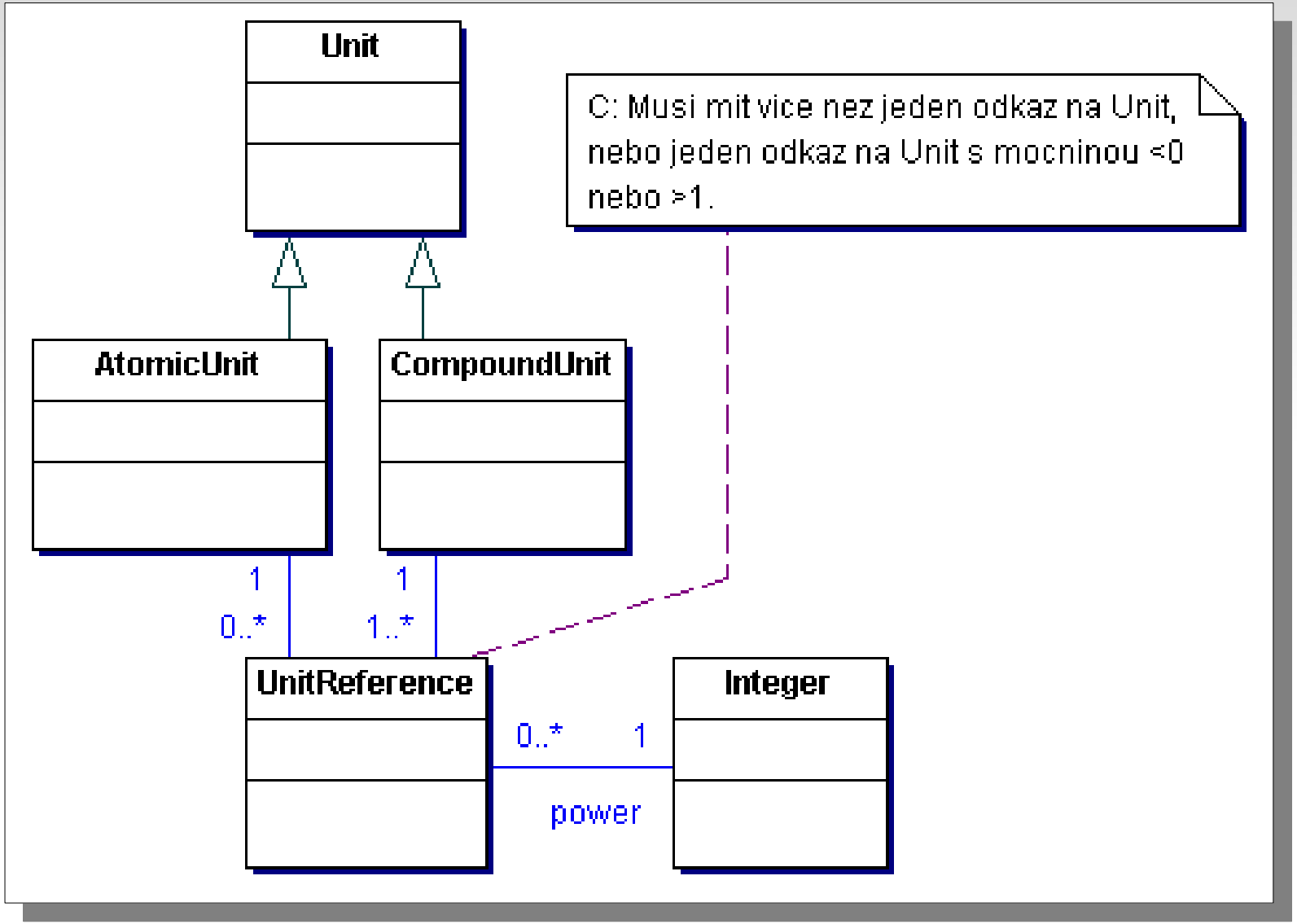
- **Task:** Create exchange table with the history of rates (design solution)

Conversion Ratio: Exchange Table with History



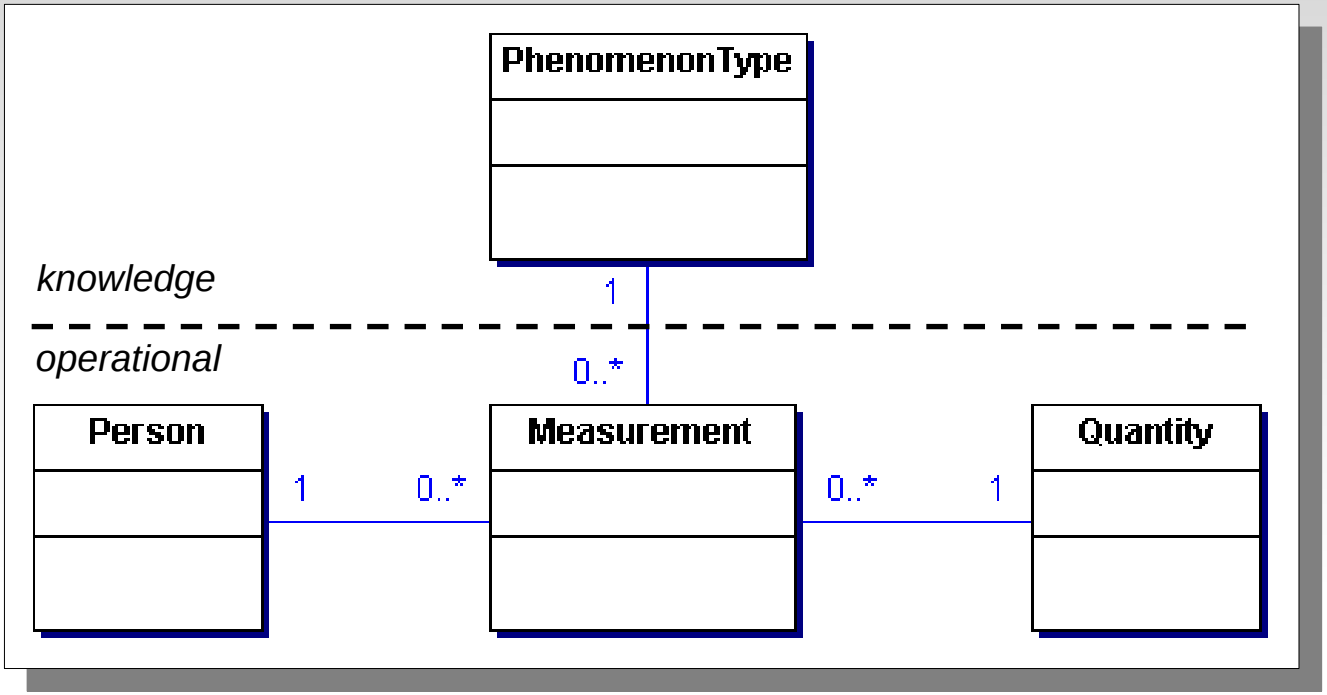
Compound Units: Pattern

Usage:
acceleration,
Km/h = km hod^{-1},
J/cm^2 = J cm^{-2}



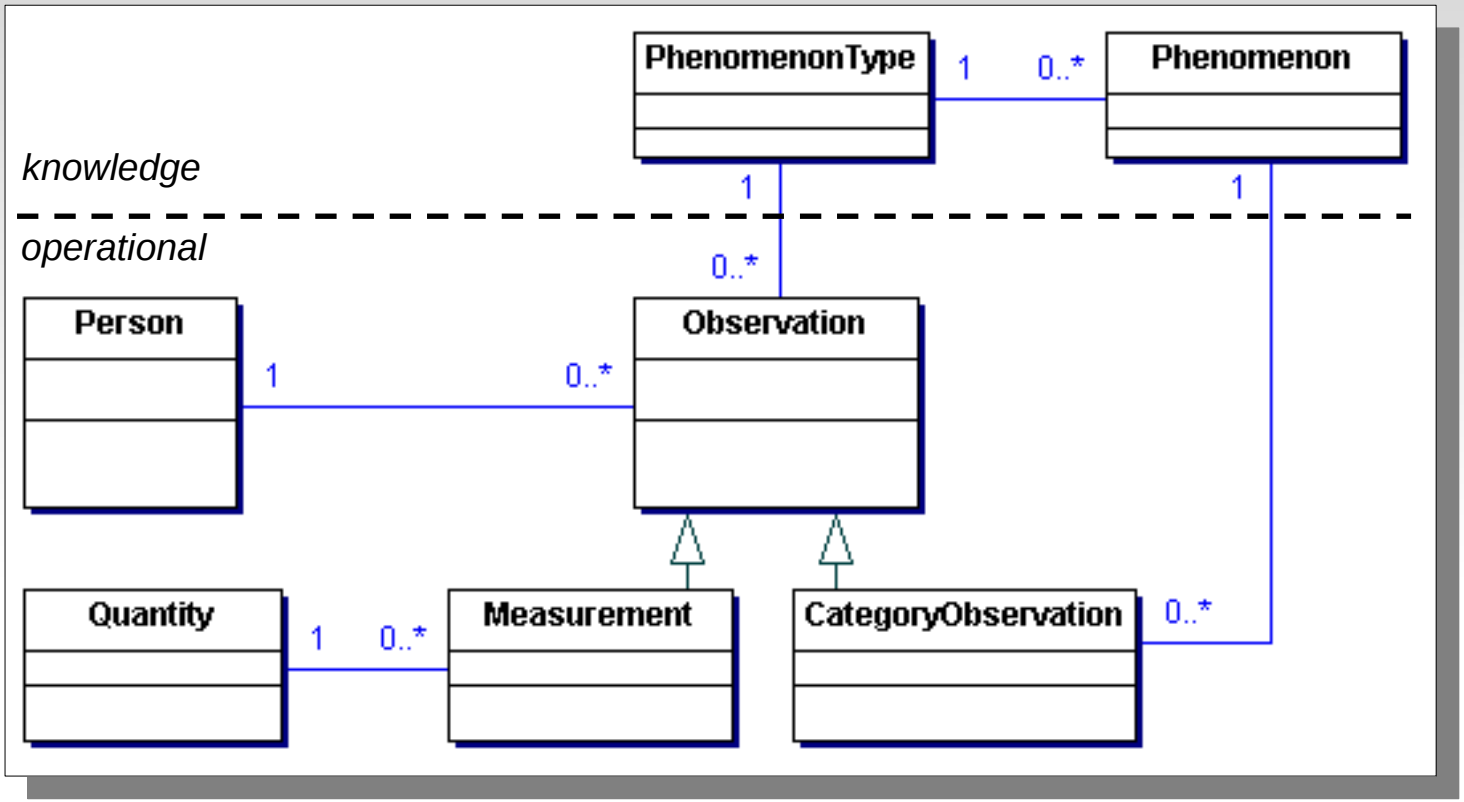
Task: speed in knots, mile/h, km/h, m/s

Measurement: Pattern



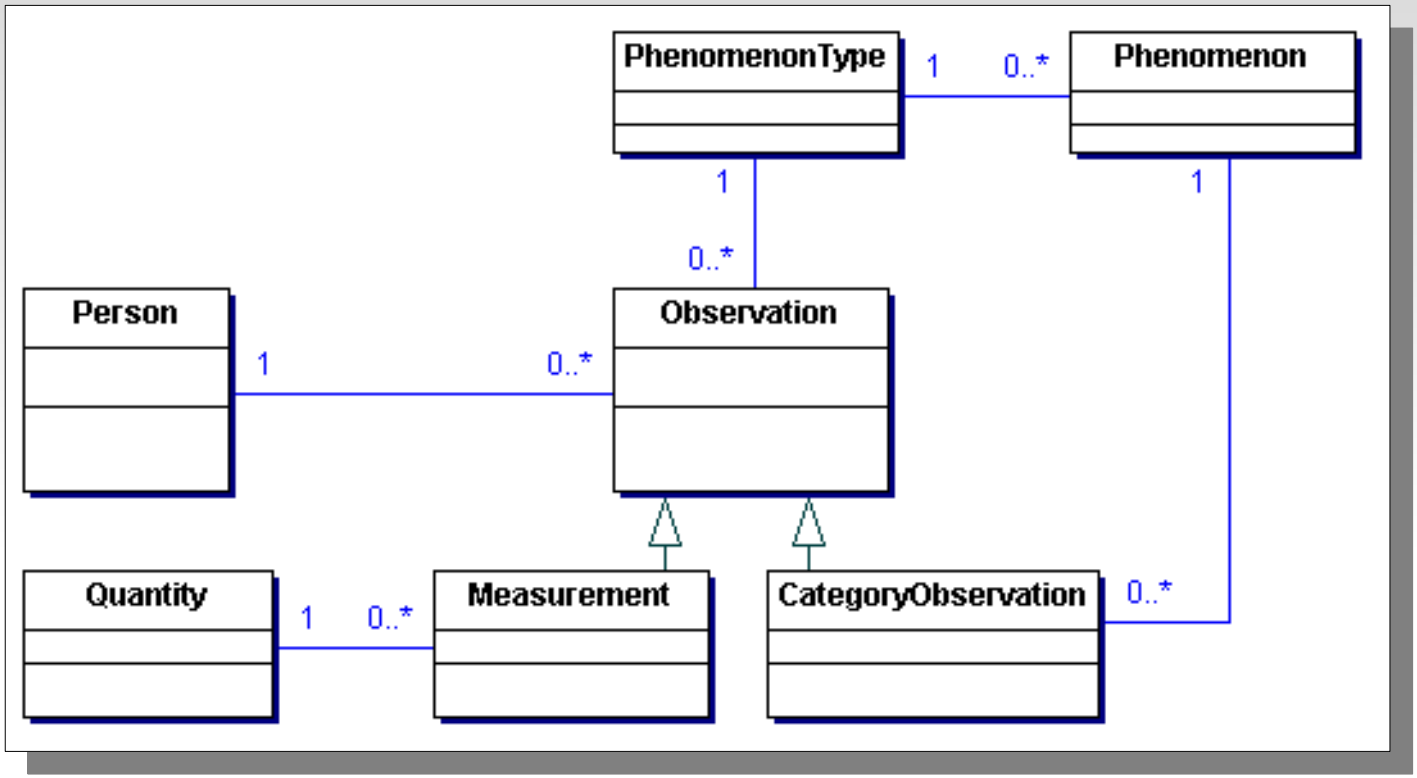
- *PhenomenonType:* things to be measured, e.g. height, weight, blood glucose level, etc. in a hospital.
- *Measurement:* concrete measured value
- *Person:* example of data assigned to the measurement, in this case a patient. Another attributes could be associated, e.g. who did the measurement, when it was done, where it was done, etc.
- The operational level has those concepts that change on a day-to-day basis. Their configuration is constrained by a knowledge level that changes much less frequently.

Observation: Pattern



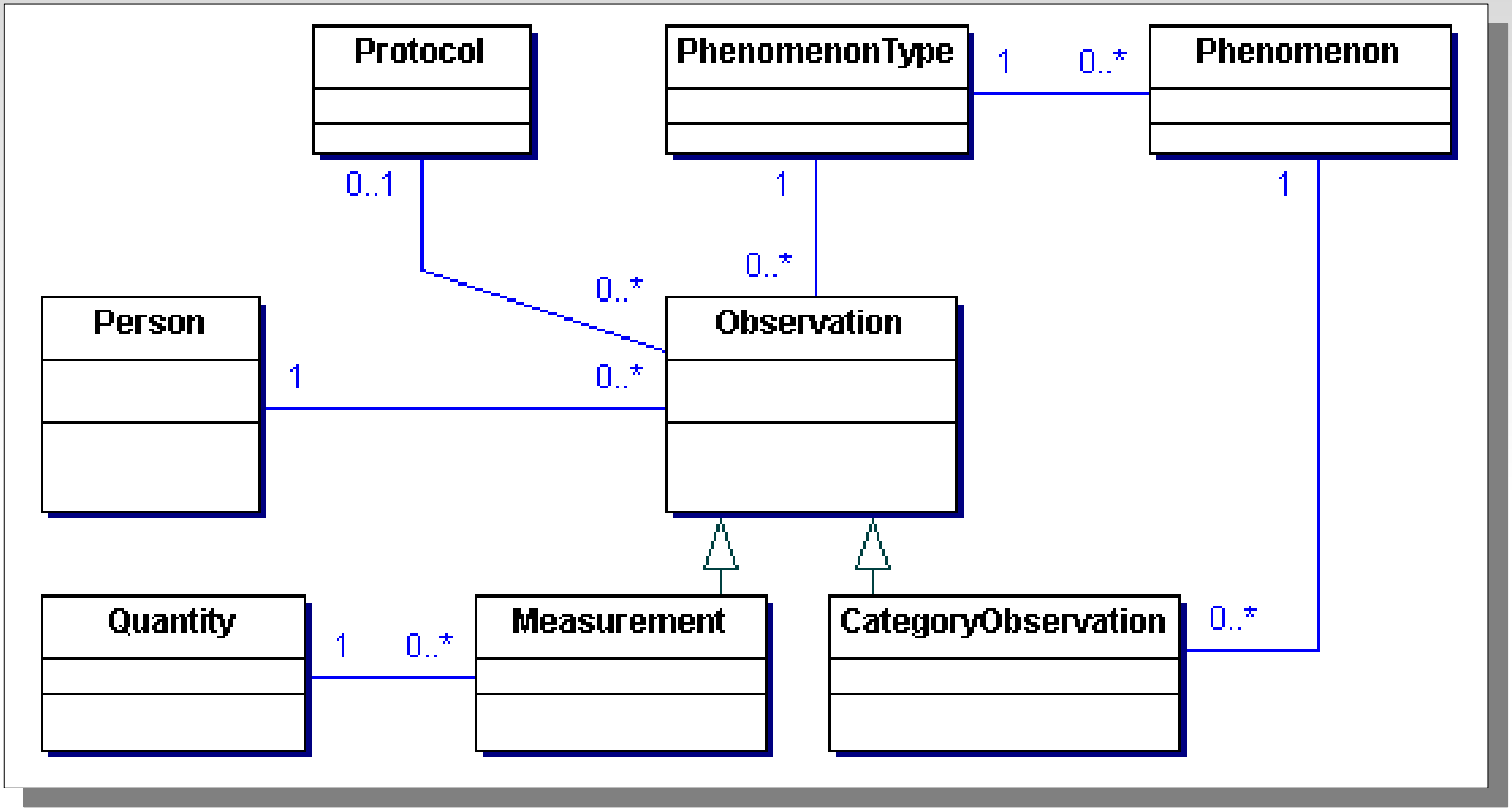
- **Q:** How to record a non-quantitative values, e.g. gender (male/female), blood group (A+,A-,0, …), etc?
- *PhenomenonType:* the same as before, e.g. blood group
- *Phenomenon:* possible values, e.g. A+, A-, 0, …
- *CategoryObservation:* Concrete observed qualitative value. It is similar to the measurement but has a general phenomenon instead of quantity.

Observation: Pattern



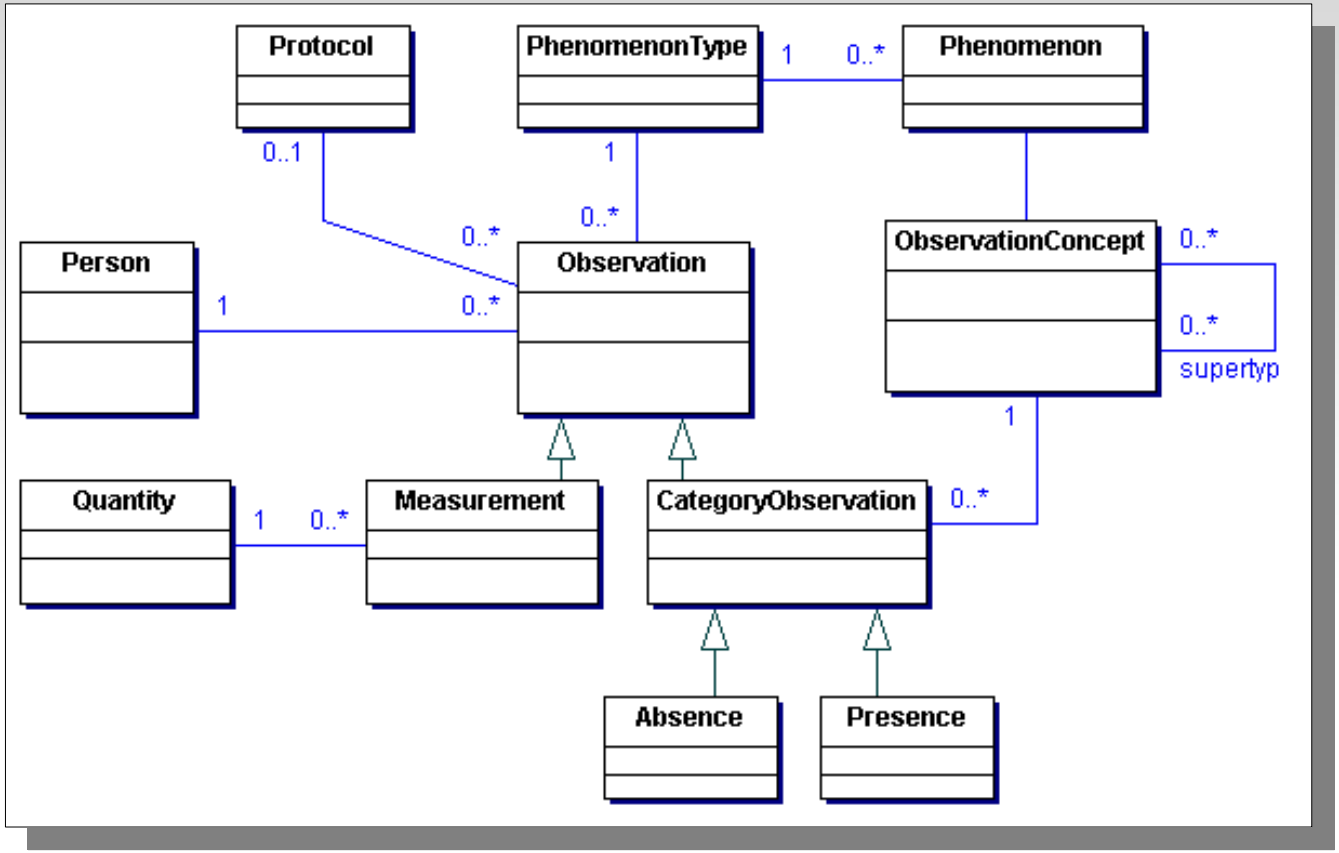
- **Task:** How to model a low oil level in a car?
- **Solution:**
  - *PhenomenonType* = *„oil level“*,
  - *Phenomenon* = *„over-full“*, *„OK“*, *„low“*.
  - *Observation* = Instance that links the car to the low phenomenon.

Protocol: Pattern



- *Protocol:* the method by which the observation were made.
- **Example:** We can measure a person's body temperature by placing a thermometer in the mouth, armpit, or rectum.

Subtyping Observation Concepts: Pattern



- If an observation is made of the presence of the subtype, then all supertypes are also considered to be present.
- If an observation is made of the absence of a subtype, then that implies neither presence nor absence of the supertype.
- **Example:** Diabetes has two subtypes: type I and type II. An observation that type I diabetes is present for J. Smith implies that diabetes is also present for J. Smith. Absence of type I diabetes does not mean that J. Smith is not diabetic.

Obs. and Measurements: Further Patterns

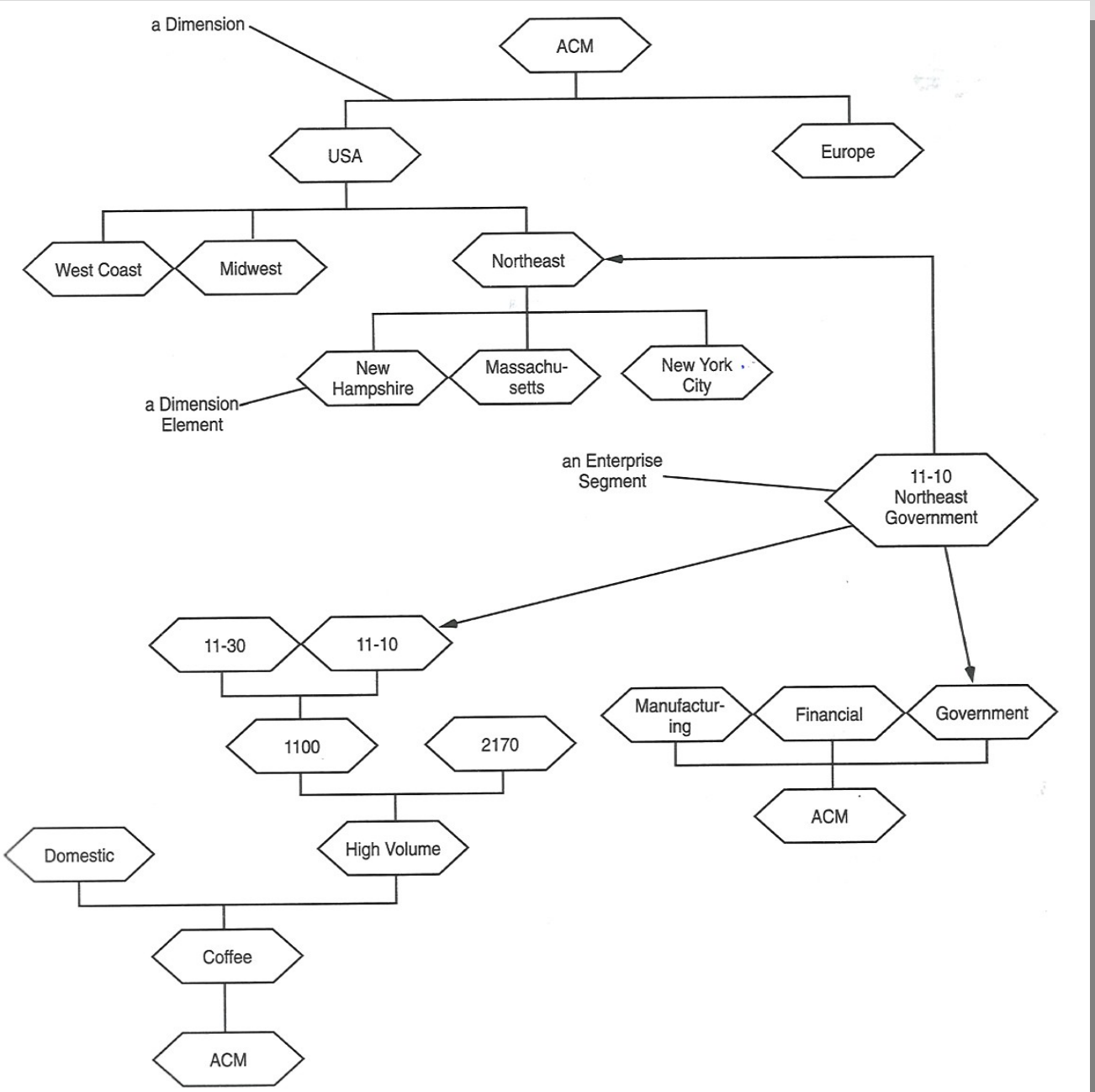
- Dual Time Record:
  - Different occurring and recording times/periods of events.
  - Ex.: *29.2.2013* doctor records that his patient J. Smith had chest pains *six months ago that lasted for a week*.
- Reject Observation
  - Observations cannot be deleted if a full audit trail is needed.
- Active Observation, Hypothesis, and Projection
  - As observations are recorded, many levels of assurance are given.
  - Subtypes of Observations: Hypothesis, Projection, Active Observations, etc.
- Associated Observation
  - Ways to record the chain of evidence behind a diagnosis.
- Process Observation
  - Behavior models for observations

Lecture 6 / Part 4:
Observations for Corporate Finance

Observations for Corporate Finance

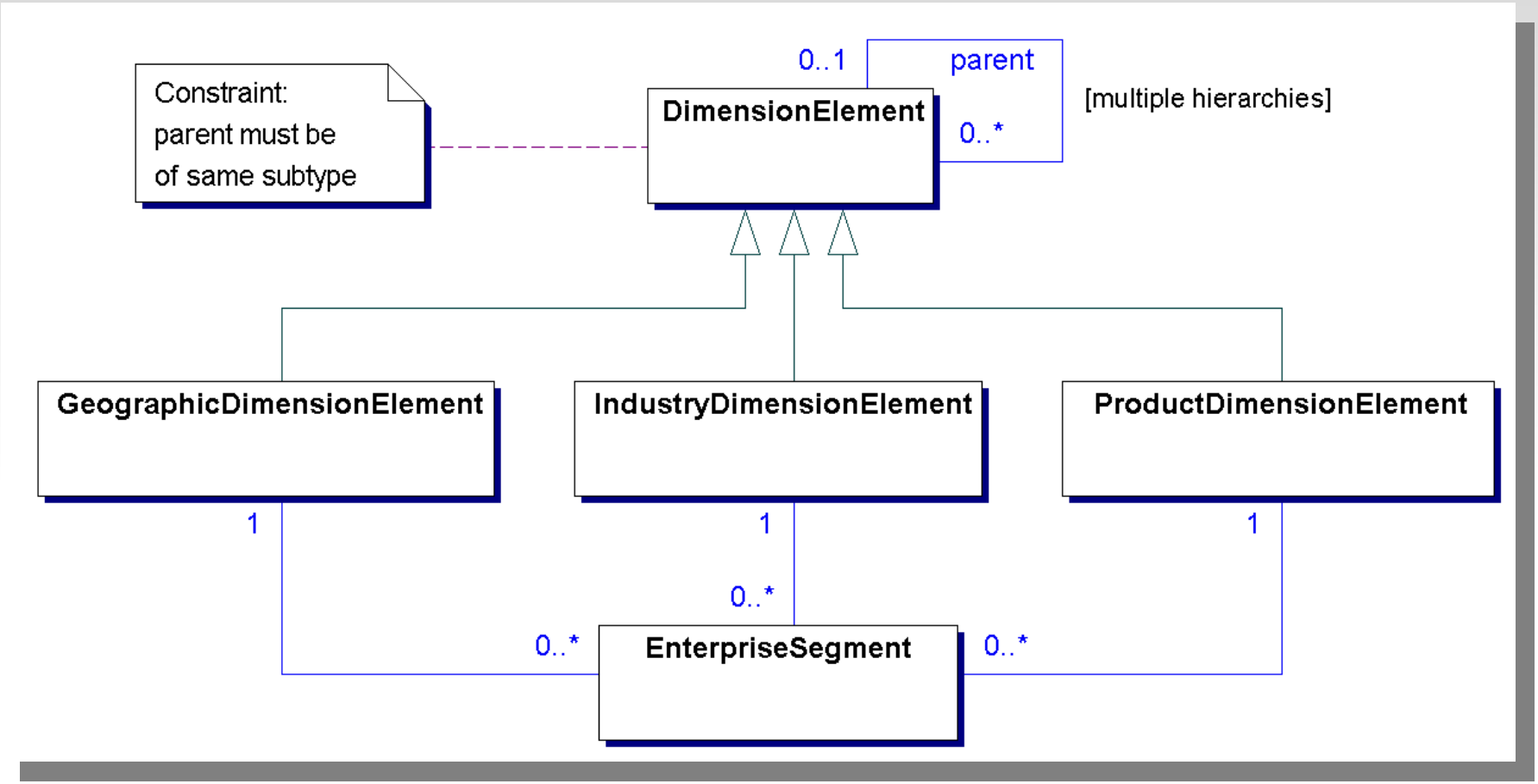
- Extension of the Observations and Measurements collection of pattern focused on (financial) balance of companies.
- Patterns:
  - **Enterprise Segment** – dividing enterprise due to dimensions, e.g. geographical location, product range, market, industry sector, etc.
  - **Measurement Protocol** – how measurements can be calculated from other measurements using formulas that are instances of model types
  - **Range** – range between two quantities + operations with ranges
  - **Phenomenon with Range**

Enterprise Segment: Principle



In some cases we look at the whole company, but in other cases we observe only part of the company, such as 10-11 espresso sales to the government in the Northeast region.

Enterprise Segment: Pattern



- Dimension = hierarchy of dimension elements, i.e. the criteria for dividing enterprise
- *DimensionElement:* Node in the dimension hierarchy, e.g. West Coast area
- *GeographicDimensionElement*, …: root nodes of dimension hierarchies.

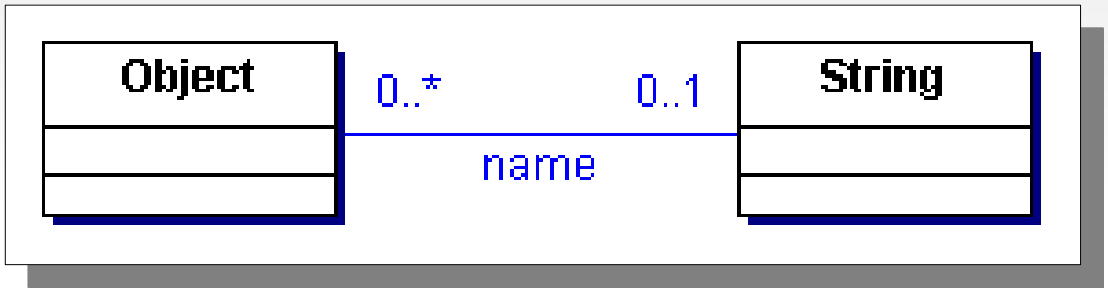
Lecture 6 / Part 5:
Referring to Objects

Referring to Objects

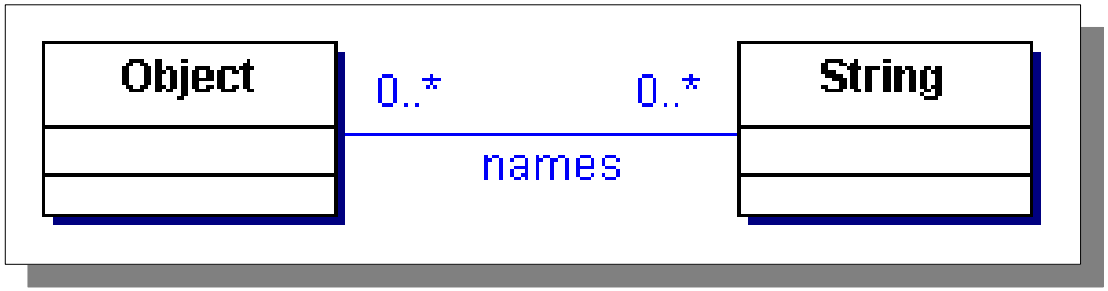
- *Conceptual* thinking about object identity, i.e. references to objects that humans use.
- Patterns:
  - **Name** – objects identification by names
  - **Identification Scheme** – brings context to objects identification
  - **Object Merge** –
  - **Object Equivalence** –

Name: Motivation

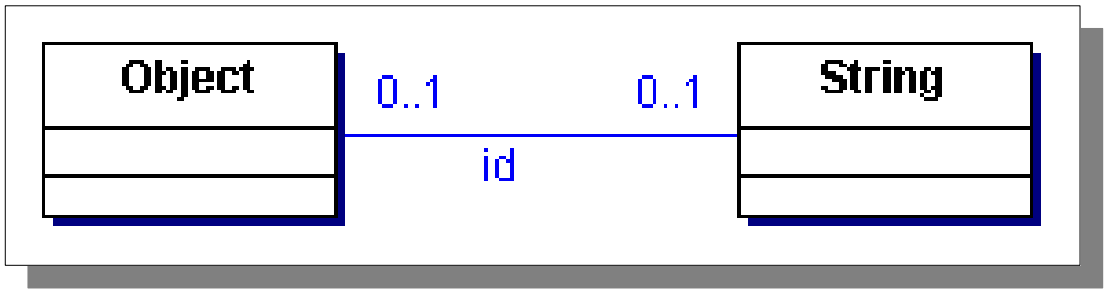
- The simplest identification for an object is a name.
- The problem is that names are not guaranteed to refer a specific object in all circumstances.
  - e.g. „Masaryk square“



Not all objects have to have a name, a string can be used as a name for many objects. Names can be structured.

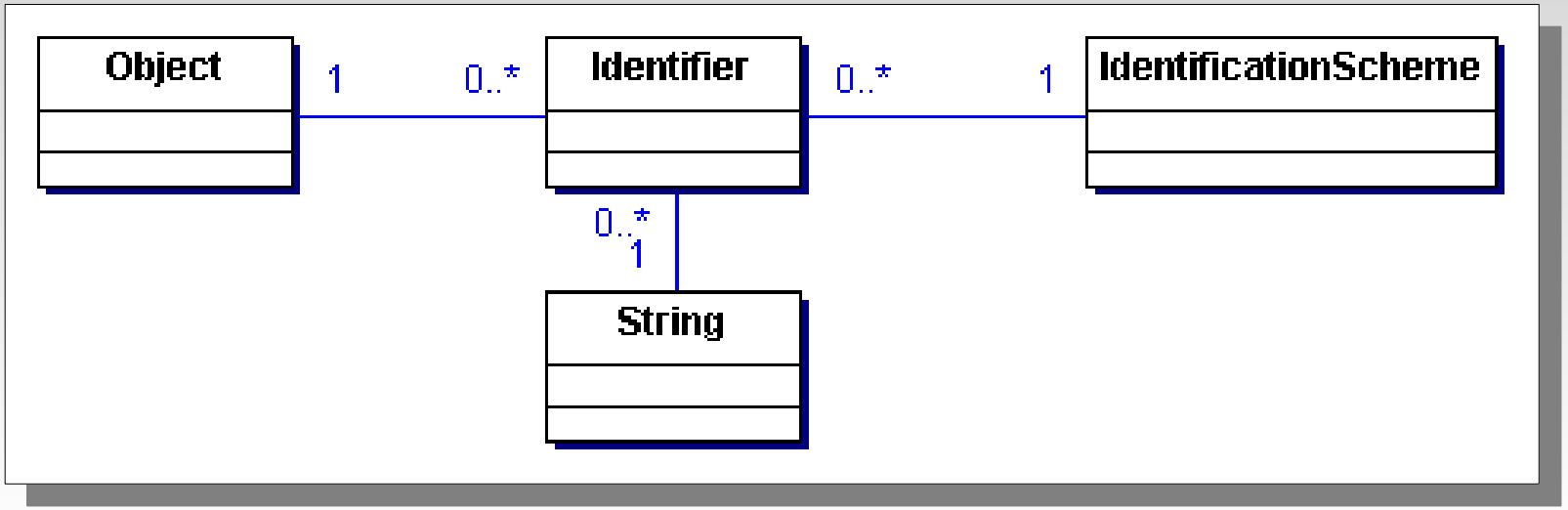


Object with aliases,
see also *Identification Scheme*



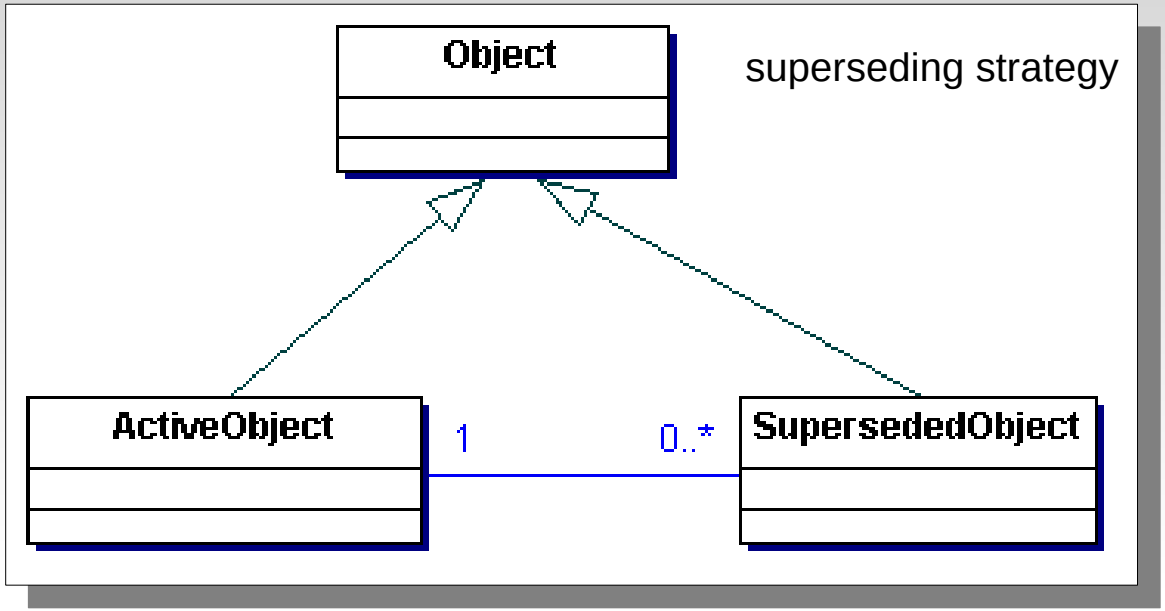
String represents a true identifier (unique ID). Not all strings identify an object, but to be a true identifier, it should identify only one. Example: bar codes.

Identification Scheme: Pattern



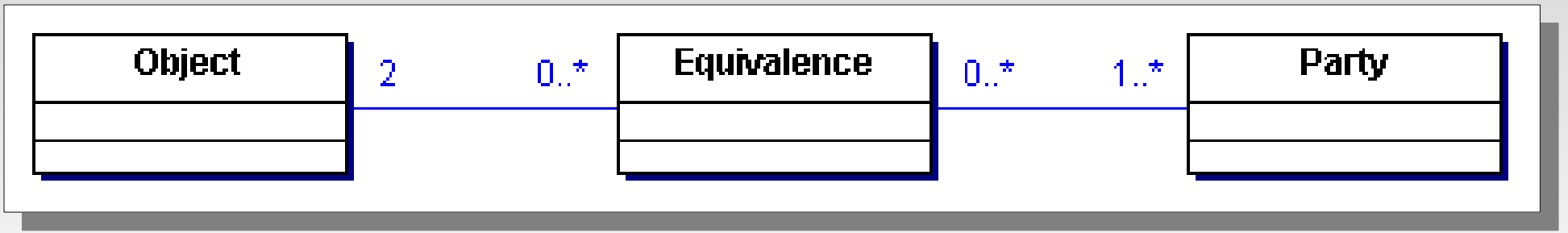
- Many identification schemes for object,
- **Examples:**
  - Identification schemes for banks: SWIFT, sort codes, CHAPS, etc.
  - Each hospital assigns a case number to a patient, departments have individual numbers.
- **Q:** Which schemes do you know to identify a person?
- **A:** RČ, name, nickname, “my colleague Peter”, etc.
  - *IdentificationScheme* = by ID, by name, by description, …
  - *Object* = concrete person
  - *String* = value, e.g. „790229/1234“

Object Merge: Pattern



- When (partially) duplicated objects come into existence in a system
  - e.g. we realize that our patient is also an out-patient at another department.
- *Copy and Replace* strategy:
  - Copy all the properties of one object over to the another and delete the copied object
  - Problem in dealing with references to deleted object.
- *Superseding* strategy:
  - Active object replaces the superseded object. Superseded object delegates all messages to the active object (references are preserved and we know the history).

Object Equivalence: Pattern



- When objects can be considered **similar (more-or-less the same, sometimes the same, etc.)**.
- Equivalence that is held by certain party.
- **Example:** Many doctors consider the diseases hepatitis G and hepatitis GBC to be the same disease, but this is not universal. If a doctor wants a list of patients suffering from hepatitis G and that doctor is a party on the equivalence, then those patients suffering from hepatitis GBC are also returned.

Questions?