From ADT through OOP to COA

PA116 – L4

INVESTMENTS IN EDUCATION DEVELOPMENT

Why? …

- To make successful decisions (in business, in life, …) we need a deeper insight into a heart of the matter
- More SW engineering point of view
- To explain some misunderstandings connected with OOA

Topics

- Principle of data abstractions, ADT
- B. Mayer: Object-oriented SW construction
- OOA – Object-oriented Analysis
- Molten objects
- Paradigm shift in physics and other sciences (except of informatics)
- Connection based perception of the physical world behavior principles
- Connection based perception of the cyber-space

Principle of data abstractions, ADT

- ADT = Abstract Data Type
- ADT—first paper: Barbra Liskov in year 1974
- To obtain the very accurate description of focused Objects, a methods possessing the following three conditions are necessary:
  - Description has to be accurate and unambiguous,
  - Description has to be complete or at least complete to this level, which is needed in each particular case of its application,
  - Description has to be not “over-specified”.

ADT Principle in detail

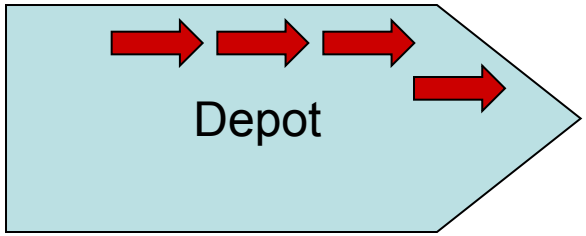
- First feature of ADT is „information hiding“.
- Each Object communicates (acts together) with its environment by a set of operations (methods), that enable to co-operate with this object.
- The way of realization of published methods is not relevant to the environment, and it remains hidden.

Specification of ADT

- Types
- Functions
- Axioms
- Conditions

Specificaton of ADT (1)

- Types
  - G
  - STACK [G]

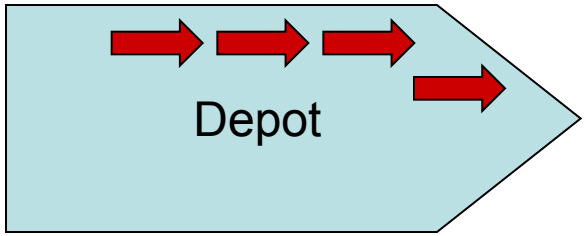


- Functions
  - put: STACK[G] × G → STACK[G]
  - remove: STACK[G] → STACK[G]
  - item: STACK[G] → G
  - empty: STACK[G] → BOOL
  - new: _ → STACK[G]
- Axioms
- Conditions

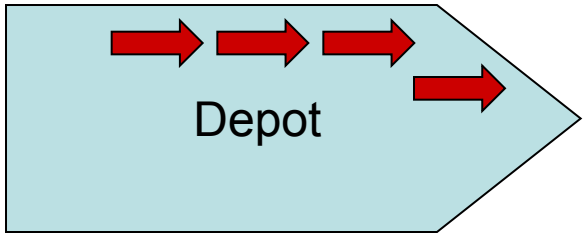
Specification of ADT (2)

- Types
  - G
  - STACK [G]
- Functions
- Axioms

  For any x::G, s::STACK[G]
  - A1. item(put(s,x))=x
  - A2. remove(put(s,x))=s
  - A3. empty(new)
  - A4. **not** empty(put(s,x))
- Conditions



Specification of ADT (3)



- Types
  - G
  - STACK [G]
- Functions
- Axioms
- Conditions
  - s::STACK[G]
  - remove(s) **requires not** empty(s)
  - item(s) **requires not** empty(s)

B. Mayer: Object-oriented SW construction (monograph)

- Cluster model of SW application life-cycle replaces traditional waterfall models
- Waterfall is about „all or nothing“ according to a scheme:

  FEASIBILITY STUDY – REQUIREMENTS ANALYSIS – SPECIFICATION – GLOBAL DESIGN – DETAIL DESIGN – IMPLEMENTATION – VALIDATION and VERIFICATION – DISTRIBUTION.

- Within cluster model all system is decomposed into clusters to enable well balancing between:
  - **sequential ordering of activities, where this ordering is necessary,** and
  - **parallel execution of activities, where this parallelism is possible**.

OOP and cluster model (1)

- Basic building block in object oriented approach is ***class***.
  (not a ***cluster*** !)
- Cluster is collection of in a way interconnected classes or clusters.
- Typical clusters are:
  - *Parsing cluster* which will provide user's text inputs analysis,
  - *Graphical cluster* which will provide graphical manipulations, or
  - *Communication cluster*.
- Clusters are not language constructs. They are means of management and organization of development and implementation.

OOP and cluster model (2)

- Inappropriate decomposition into clusters can slow down the development project, however it usually doesn't result in a dysfunction of the developed system.
- A critical factor of successful functioning of developed system is well done selection of ***classes*** (i.e. selection of proper data abstractions).
- The decomposition into clusters is a key to efficient project process, only.

Class

- A ***Class*** is abstract data type (ADT), which is partially or completely implemented;
- "partially" covers "not at all", too.
- ADT is a mathematics notion.
- Its implementation is a computer version of ADT.
- *Effective Class* – fully implemented
- *Deferred Class* – partially implemented or not implemented at all

Process of creating of an effective class:

- E1: Create a specification of ADT
- E2: Chose a relevant or advantageous representation
  - E.g. for STACK it could be couple (array, count)
- E3: Map ADT(from E1) into representation (from E2) so that axioms of ADT (from E1) will be accomplished and so did conditions of ADT (from E1)
- Note: Deferred classes provide „a track“ of analysis within the implemented system.

Example: STACK[G]

- E3 must contain mechanisms of put, remove, item, empty a new representation. For put it looks, e.g.:
- put(x,s) **is**
  --Push x onto stack s.
  --(No check for stack overflow.)
  do
  count:=count+1
  array[count]:=x
  end
- …

Using classes in OO programing

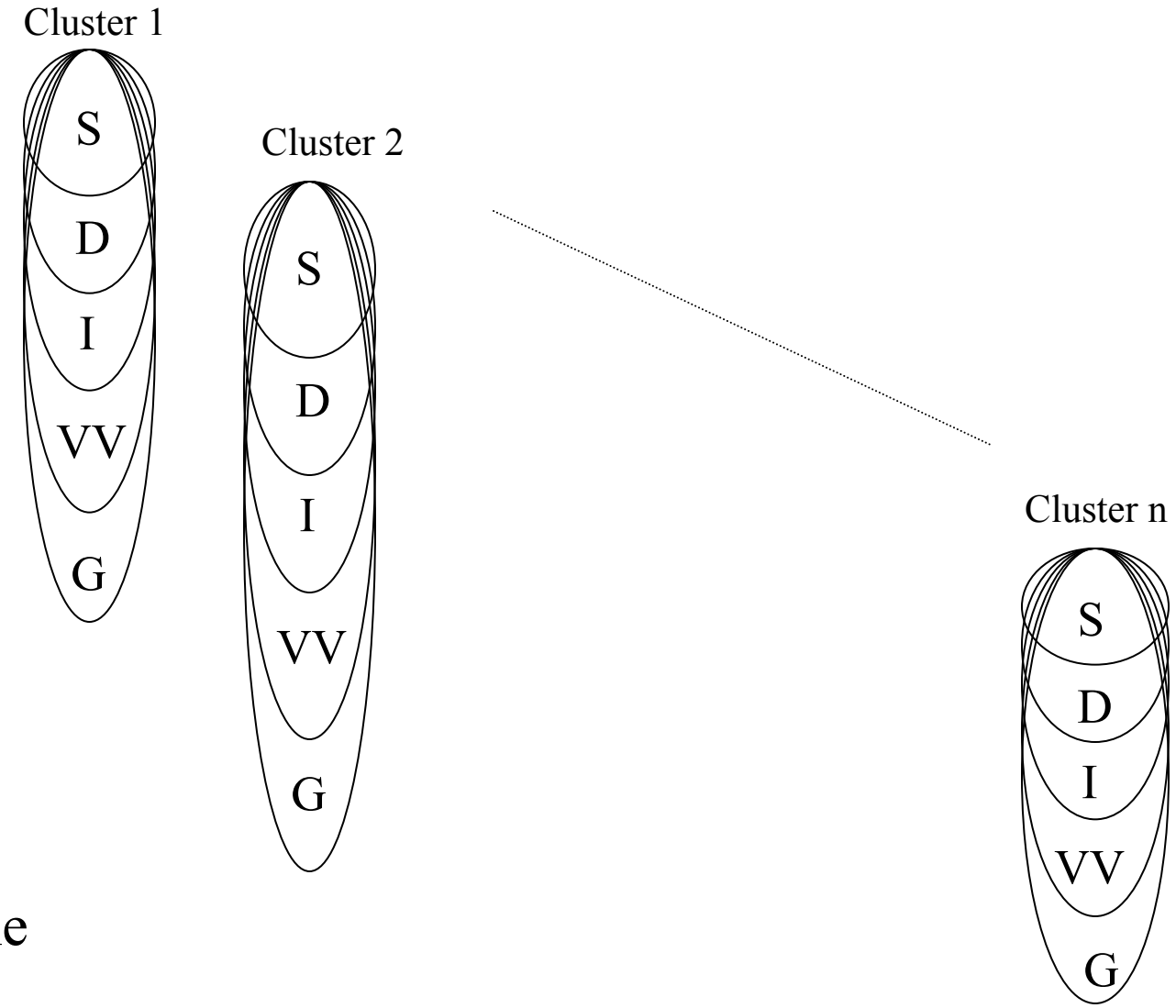
- Information hiding !!!
- Public part
  - Specification of ADT (E1)
- Hidden part
  - Selection of representation (E2)
  - Implementation of particular functions in alignment with axioms and conditions (E3)
- It is a very useful approach !!!
- From one deferred class we can inherit various effective classes, …

Cluster model of Object Oriented SW Development

- S – Specification,
- D – Design,
- I – Implementation,
- VV – Verification+Validation,
- G - Generalization
- „stalactite growth“ of clusters

FEASIBILITY STUDY

DIVISION INTO CLUSTERS



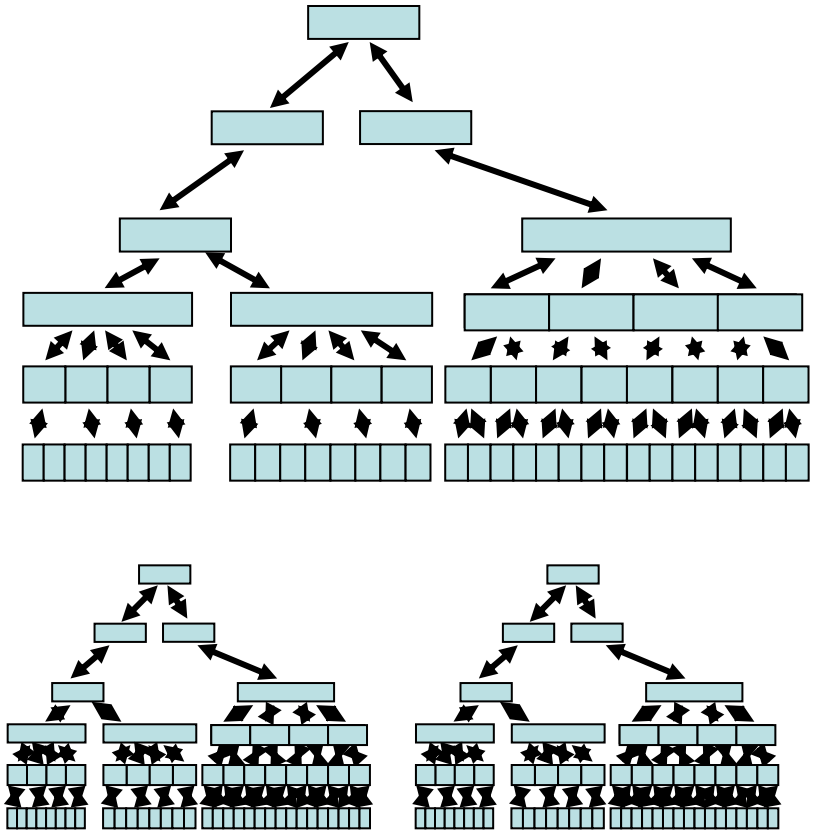
Time

Cluster model

- Each cluster has its own „mini-life-cycle“. Mini-life-cycle consists in continuous growing out of the following steps:
  SPECIFICATION – DESIGN – IMPLEMENT-ATION – VALIDATION&VERIFICATION – GENERALIZATION;
  each step grows out into its successor.
- The most important is the „seamless“ nature of cluster development: this is based on diverse level of implementation of particular classes and on the step-by-step directing from pure ADT to an effective class !

OOP = one of the best inventions

- …an intelligent use of the “Fundamental hierarchy”
- … a reverse process to “Breakdown structures”
- … in a way a simulation of the natural process of “cognition by creation”



OOA – Object-oriented Analysis

the process and basic features

- (1) Find required objects
- (2) Objects classify into classes and establish appropriate structure of those classes
- (3) Build up the problem solution by mutual connections and communications of objects/objects_classes
- ***Objects*** have their "***state***", their "***memory***" and they have a ***capability to communicate*** with their environment
- The base of success is "to feel" right objects/objects classes, to which attention has to be focused
- Crucial question: ***What to see as one object and what to see as a cluster of objects ?!?***

The Class in an OO analysis

- ***Class*** defines “the shape” of its instances
- Each ***object***, which is worth focusing attention, must be assigned to a ***class***.
- Improper ***class*** selection “today” can cause big problems “tomorrow”!
- Situation change in real world causes necessity to change the assignment of some objects to classes, or to change the class structure design.

Issues of OOP (1)

- ***Object Oriented Paradigm*** works well in the realm for which it was originally developed.
- This is **Programming**.
- **The realm of "artifacts" creating**.
- To mirror a **realm of continual changes**, improvements and developments doesn't fit to OOP very well.
- But Business Systems analysis and specification really is this realm.

Issues of OOP (2)

- The most often argument for OOP is “reusability”
  - This is like: taking any screw M6 it fits to any nut M6.
- But, does it work really?
- Does a designed class “cross the border” of its analyst-creator mind?
- Are the Objects really so solid and fixed?

“Molten” objects (1)
(according to Žemlička, Král, CUNI)

- Motivation:
- Historical map: time-sensitive map of territory of CZ in years 1000-2000
- Lot of relevant objects perished, lot of relevant objects arose, and lot of objects changed dramatically
  - E.g. former castles, now called ruins
  - (?) to what class to assign such object: to ***castle*** or to ***ruin***

“Molten” objects (2)

- Object is not conceived as entity, which has to be first explicitly assigned to chosen class and then used by means of available methods in a way
- Object is conceived as an entity, which is in a given time and a given situation determined by a set of properties.
- This set of properties can change over time or according to changed situation.

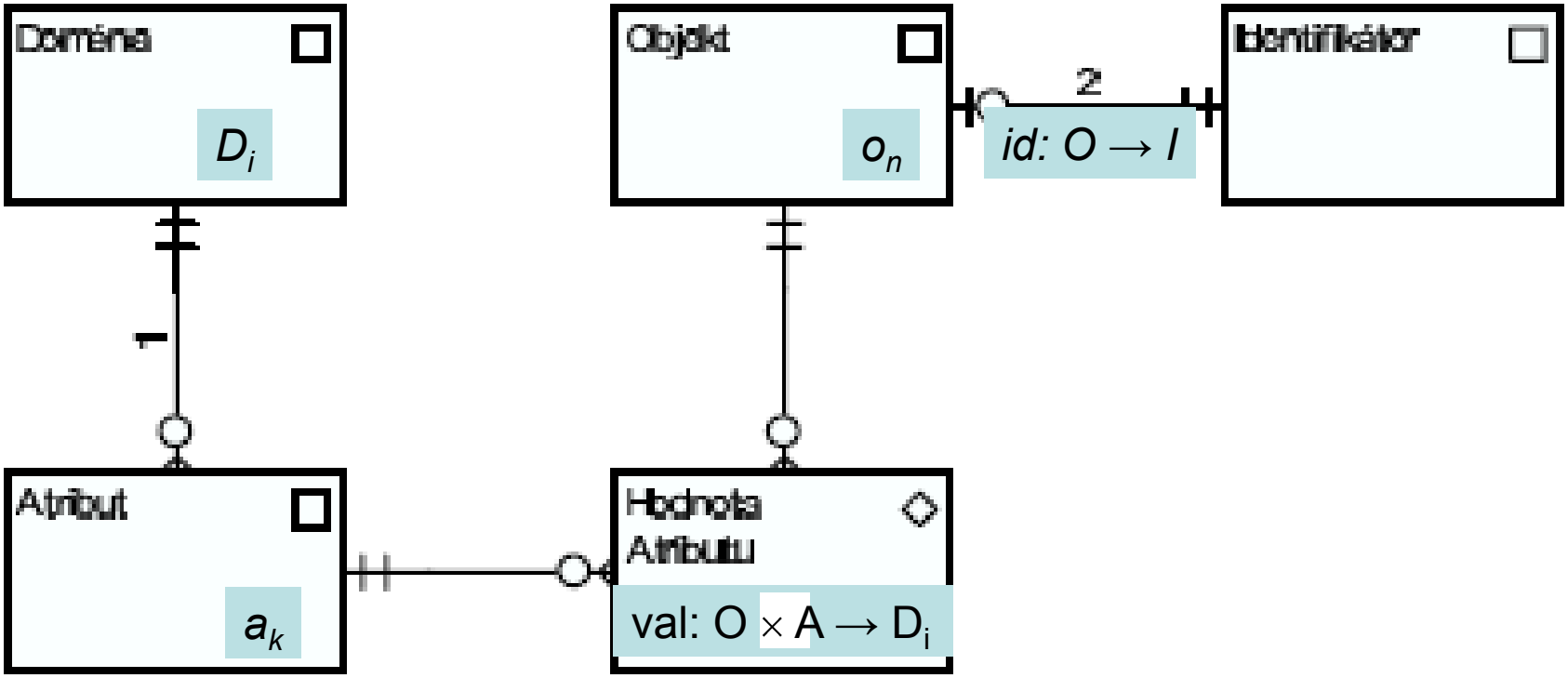
Molten objects – data scheme

- $S = (A, I, F)$
- $A = \{a_1, \ldots, a_k\}$ … finite set of attributes
- I … finite set of identifiers of object instances
- $F = \{f_1, \ldots, f_l\}$ … finite set of data-manipulation functions
- $a_i::D_i$, D_i … domain of attribute a_i;
- $\forall i(\perp \in \mathrm{D_i})$, $\perp$ … object “undefined”

Molten objects – Data Scheme Instance

- $S_{inst} = (O, id, val)$
- $O = \{o_1, \ldots, o_n\}$ … finite set of objects
- $id: O \rightarrow I$ … injection
- val: $O \times A \rightarrow D_i$, where D_i is determined by $a_i \in A$.
- Object $o \in O$ can have in different data scheme instances different attributes
- The others, that is to say, take the value $\perp$.

Data model of the Molten Objects data scheme



Object as a set of relationships

- The Object can be according this scheme conceived as a set of relationships.
- The Object is uniquely determined by this set of relationships.
- Obj^o … set of relationships, which determines the object $o \in O$.
- Obj^o contains just one instance of connection “2” and arbitrary number of instances of associative sort *Attribute Value*, not determined in advance.
- Construction of the set Obj^o is a matter of empirical cognition, i.e. it depends on our perception of the object *o* in a given time moment and in a given situation – simply it depends on (*w,t*).
- Construction of such a set can be seen as an abstract procedure called “*objecting”.*

… development in other sciences ?

Paradigm shift in physics and other sciences

- The World according to Newton and Descartes:
  - "Phenomenons can be reduced to properties of solid material elements."
  - "Behavior of arbitrary complex system could be analyzed exclusively from the properties of its parts/components."
- World today (F. Capra, …)
  - "Web of events, in which alternations and/or overleaps and/or combinations of connections of different kind occur, determines tissue of the whole."
  - In biology: from the concept function to the concept organization.

Fritjof Capra: The Web of the life

- ,,*A great shock of the science of 20th century was, that a system cannot be understood by using analysis of this system only.*
- *Properties of parts are not attributes of these parts, only, but attributes of those parts in a given context.*
- *Thus these properties could be understood only within a greater whole . . .*

Fritjof Capra: The Web of the life

- *A system thinking is contextual one in opposite to the analytic thinking.*
- *Analysis means to take something separately from the other things/issues in order to understand.*
- *System thinking means to place* this something *to the context of broader whole* (in order to unerstand).“

Basic principles of system thinking (F. Capra)

- 1. shift from parts to the whole – i.e. we start to accept that behavior of the whole cannot be understood by analyzing of properties of its parts (there exist some emerging properties)
- 2. ability to focus attention to different levels of the system
- 3. understanding that parts doesn't exist – in fact this what we call "part" in point 1. is nothing more than ordering in non-separable web of connections; thus **shift from parts to the whole could be considered as shift from objects to connections**

Informatics today:

- OOP: well done manipulating with complex relationships
- OOA: a mistake in perception of the world
- Early nineties:
  - “Who is not object oriented, is excluded from decent society !”
  - Example of CASE tool LBMS Systems Engineering
- Current Informatics didn’t experienced the turn, which other science disciplines experienced recently !

Back to our historic and philosophic perspective!

Connections based perception of real world behavior principles

- History of cognitive processes research
  - Semantics networks
  - Connectionist model
- History of data modeling
  - Classical ERD
  - UML diagram

Semantics network

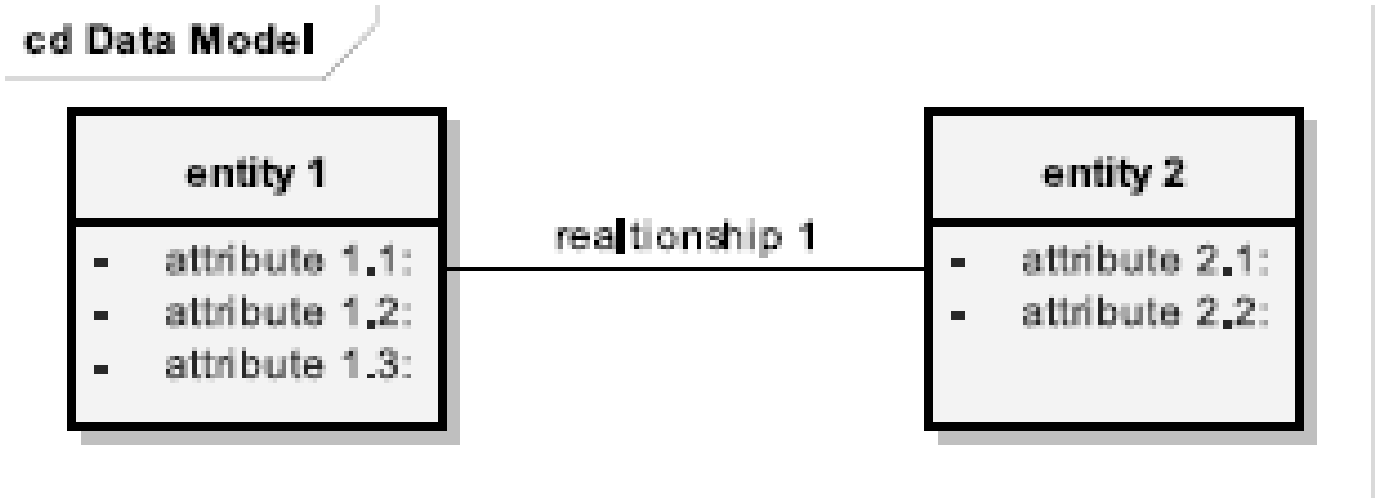
- Semantics network is formed from mutually connected items, which are called *nodes;* nodes represent concepts.
- Nodes are connected by *labeled connections*.
- Labels assign notion or semantics to these connections
- Connections enable to organize concepts into more complex structures.

Connectionist model

- Network is constituted from elements similar to neurons.
- Such “neuron” itself doesn’t represent a concept or other kind of information.
- Thus *no particular points in network*, but *ordering or organization of their connections (synapses) represents knowledge*.

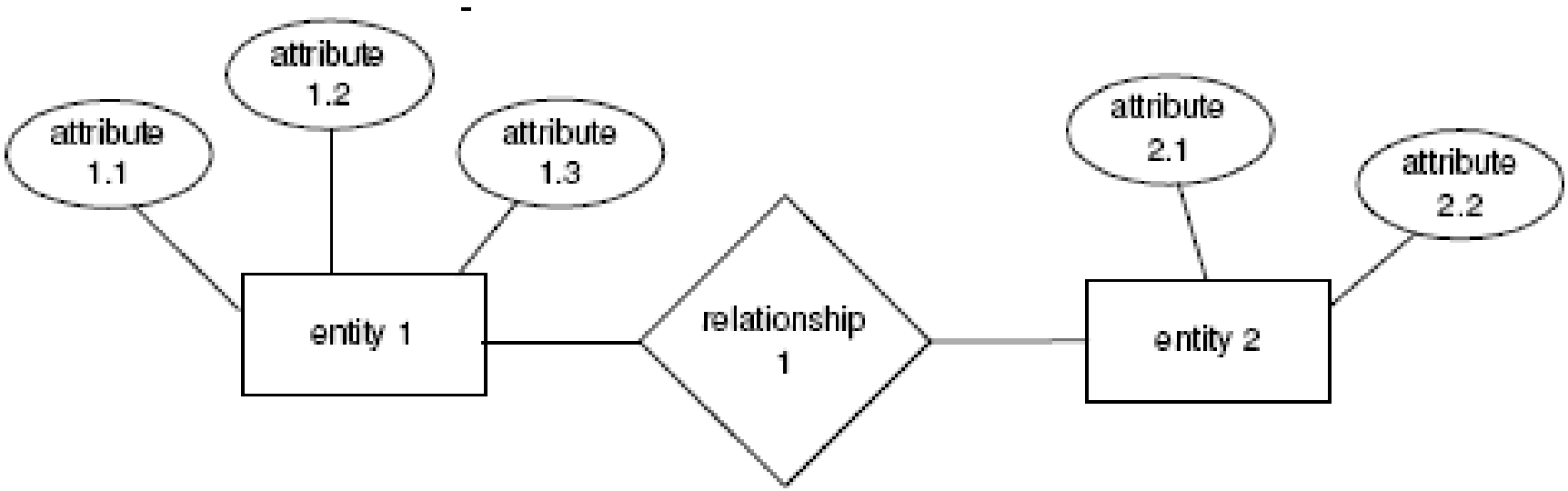
UML diagram

- This is a point we came to during evolution:



- What was the starting point?

Classical ERD



- Attribut and entity are in equal relationship; their visualization (entity - rectangle, attribut - ellipse) gives to each of them its own autonomy
- Organization of nodes and edges !

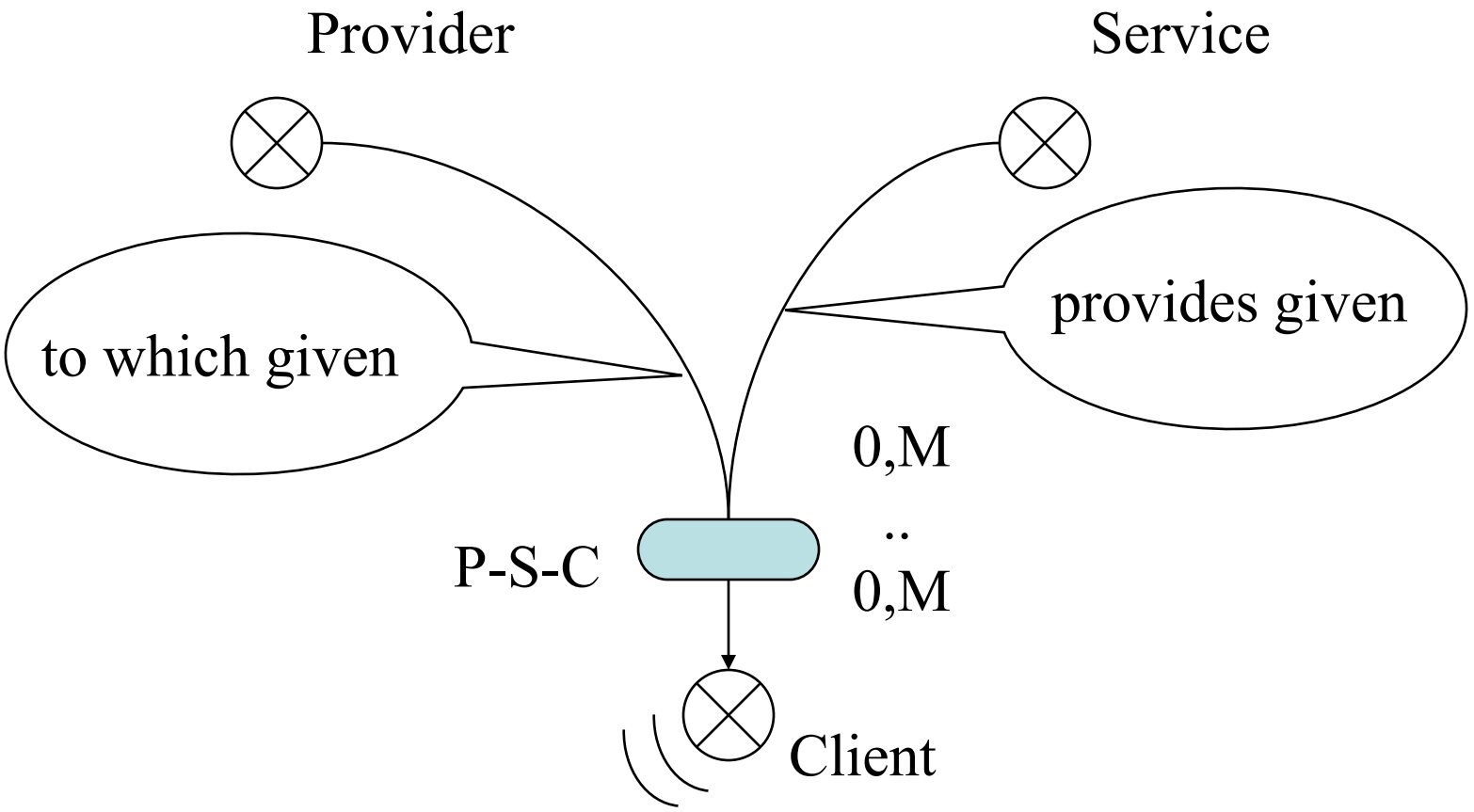
COA

Connection Oriented Approach

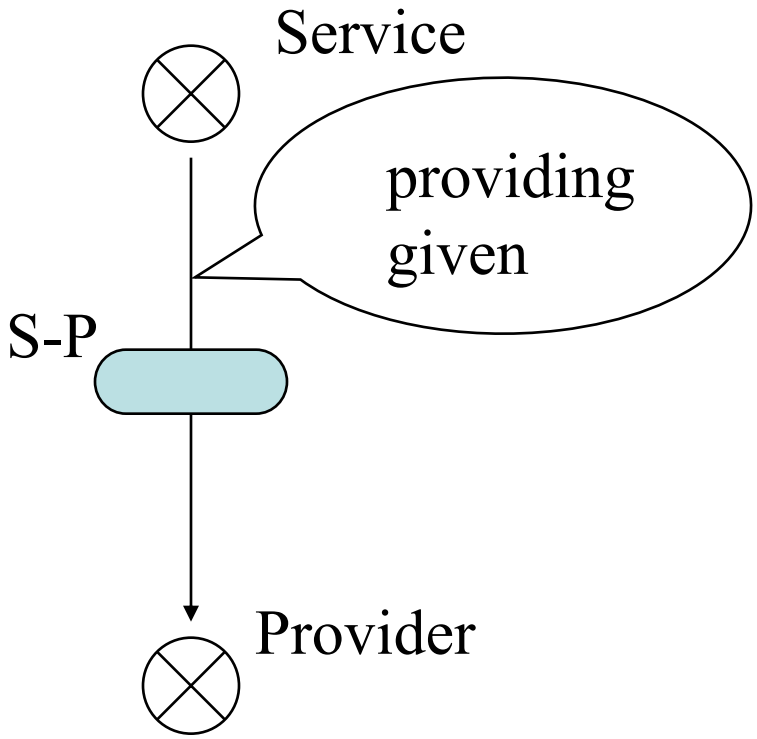
Connection oriented approach

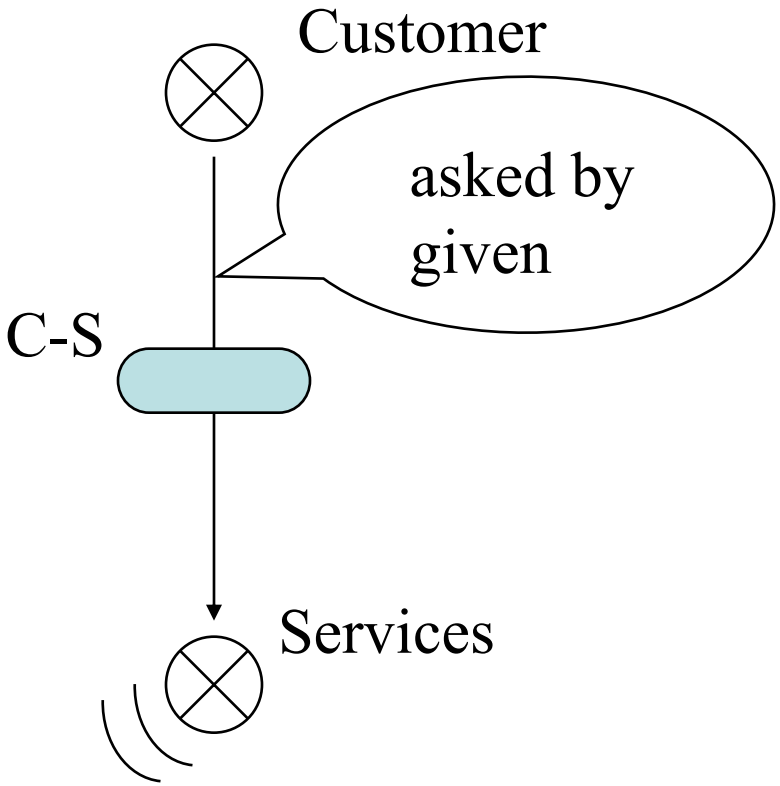
- The basics is: we store instances of relationships not instances of previously determined complexes in a form of tables (from the beginning fixed)
- **Principle of connection based perception of models**
  Seeing a model of anything in a form of graph and thinking on this model we focus primarily on relationships (graph edges) not on objects (graph nodes).
- Let's compare this with HIT method !!!

Attention focusing in HIT method:



Connections, connections, connections !!!





Service Systems:

- Co-creation, usefulness, C-P relationship, C-T relationship, P-T relationship, context relationship
- What can be owned?
  - Objects or Relationships?
- Is SS* about ownership or about usefulness?
- What is better to obtain usefulness?
  - Objects or Relationships=Connections ?