



TC184/SC5 AdHoc meeting  
Oct. 30, 2003, Washington DC

# PSLX-03

## PSLX Domain Objects

Yasuyuki Nishioka, Prof. Dr.

Hosei University,

PSLX Consortium

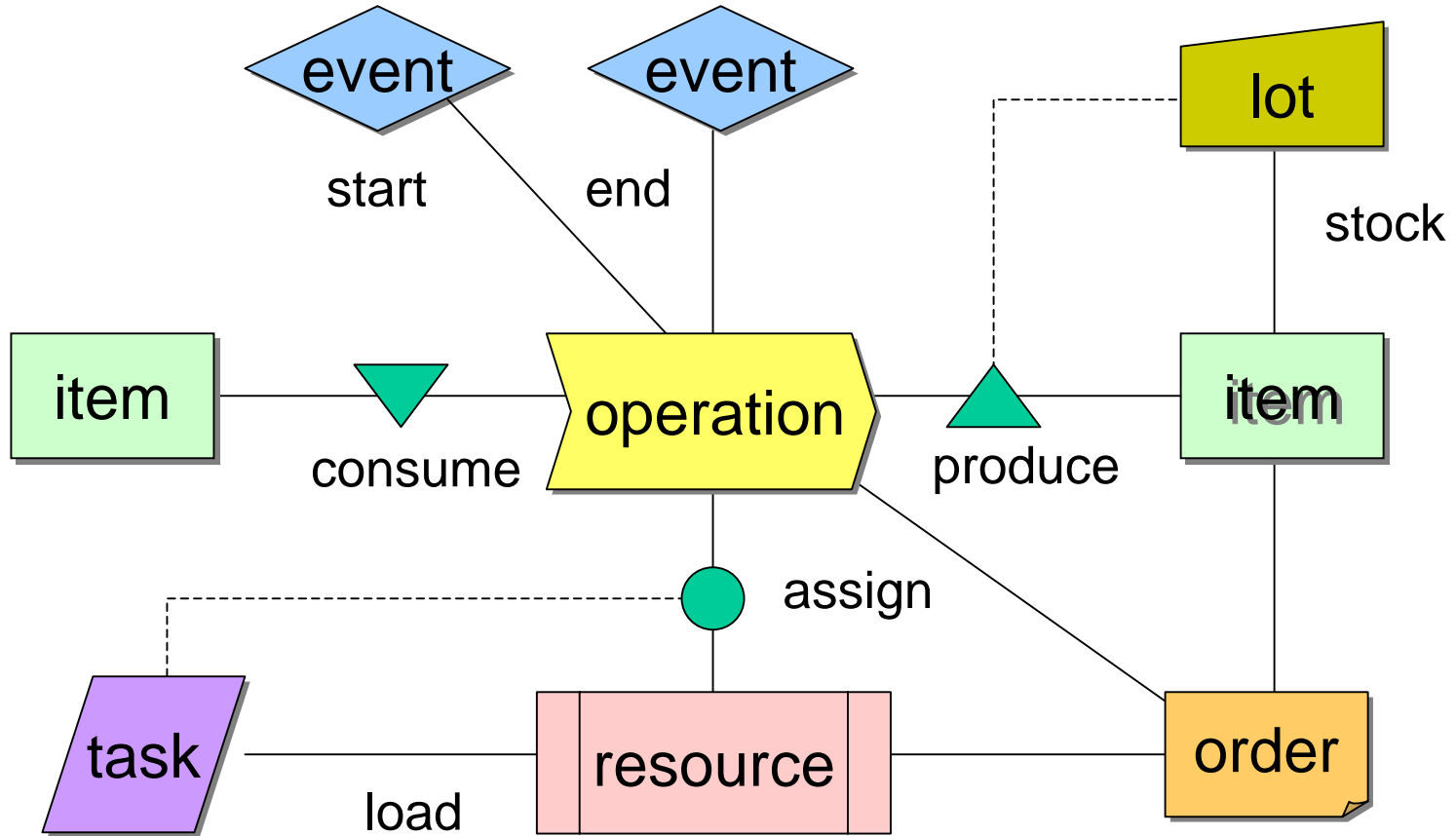
[nishioka@k.hosei.ac.jp](mailto:nishioka@k.hosei.ac.jp)



# Outline

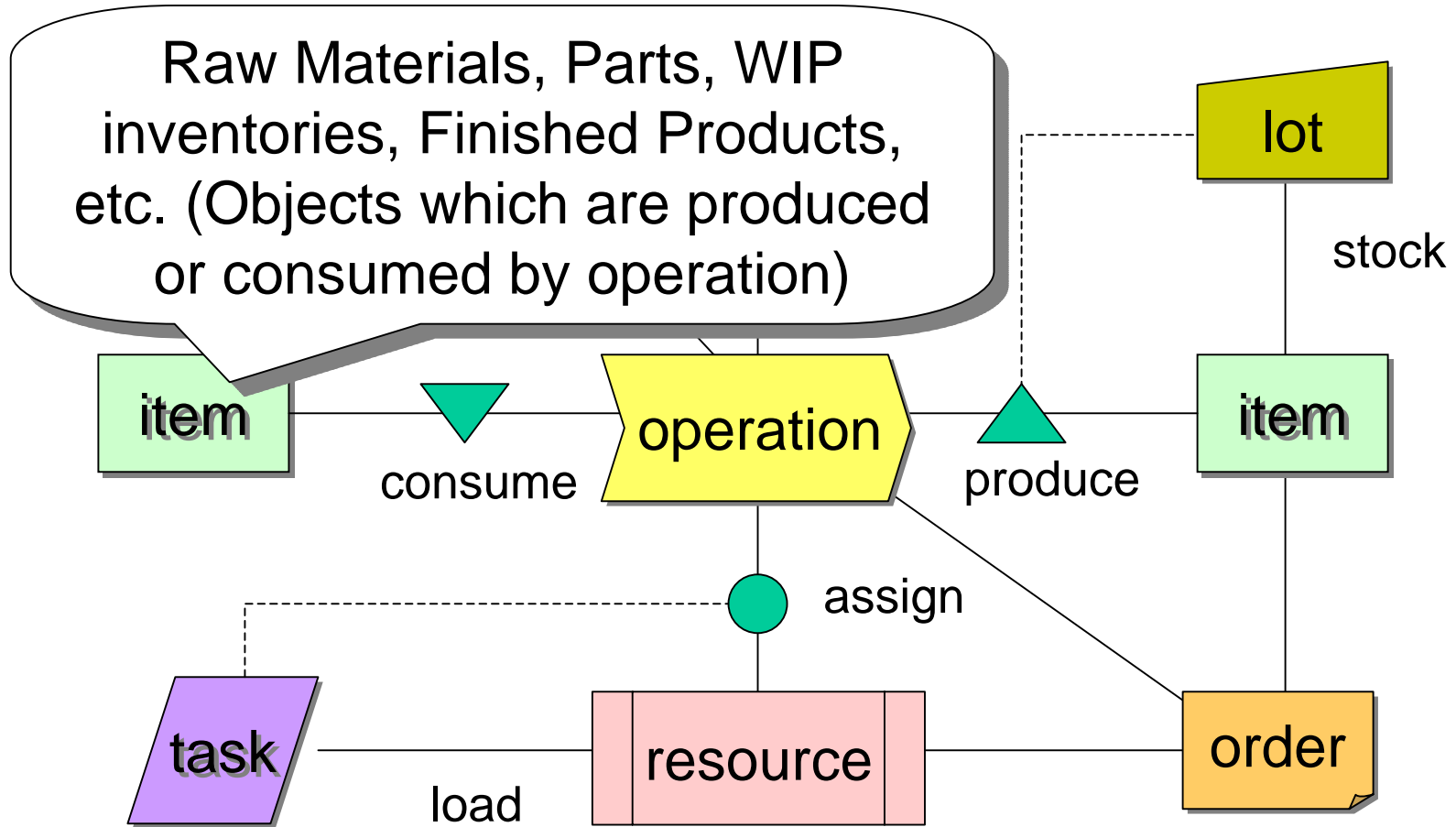
- Define core components as ontologies necessary for representing production planning and scheduling problems
- Define naming rules to create new interfaces and describe their functions in accordance with business processes
- Define translation processes from core components to particular object schema used in industrial applications

# Core components

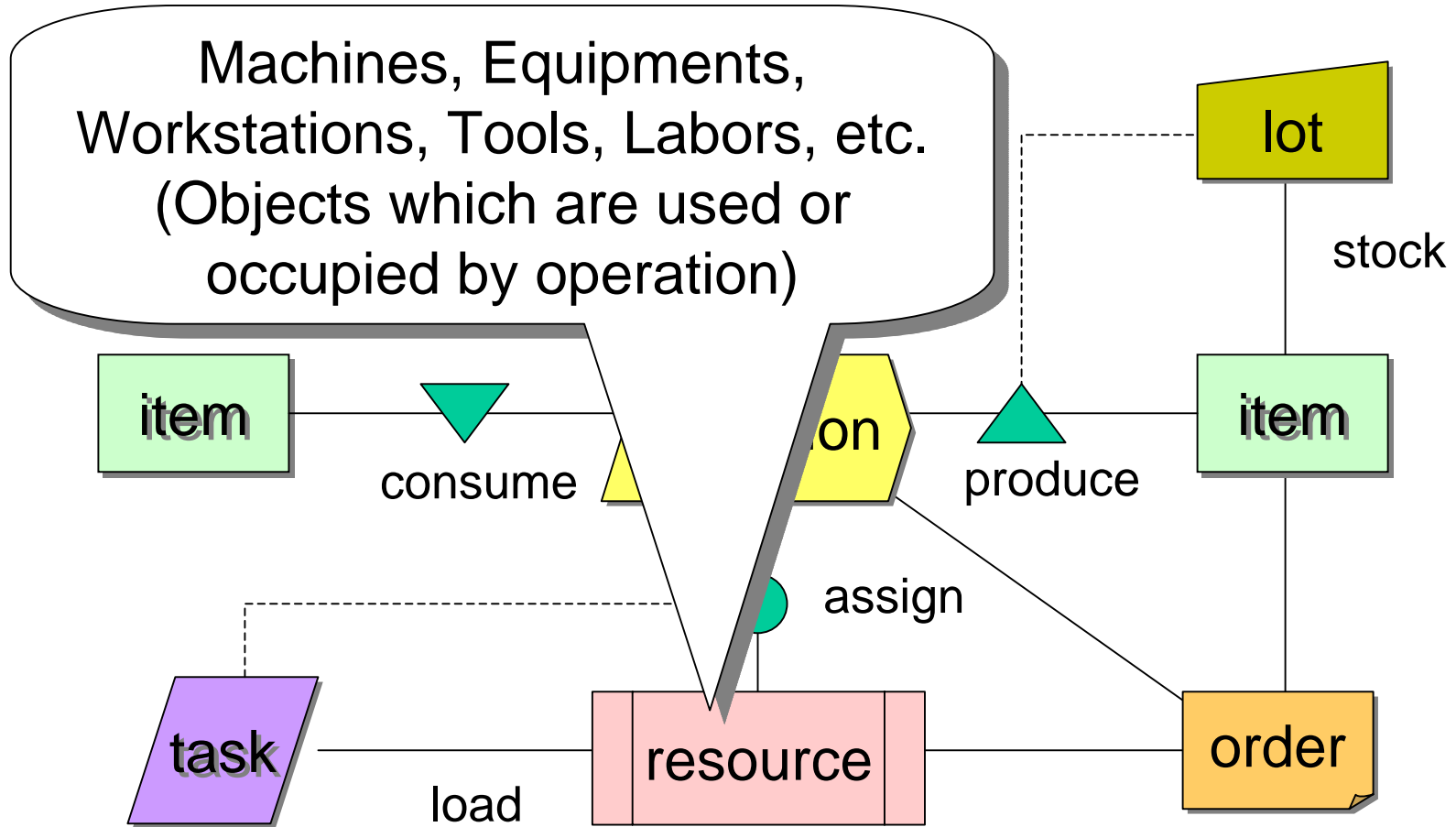




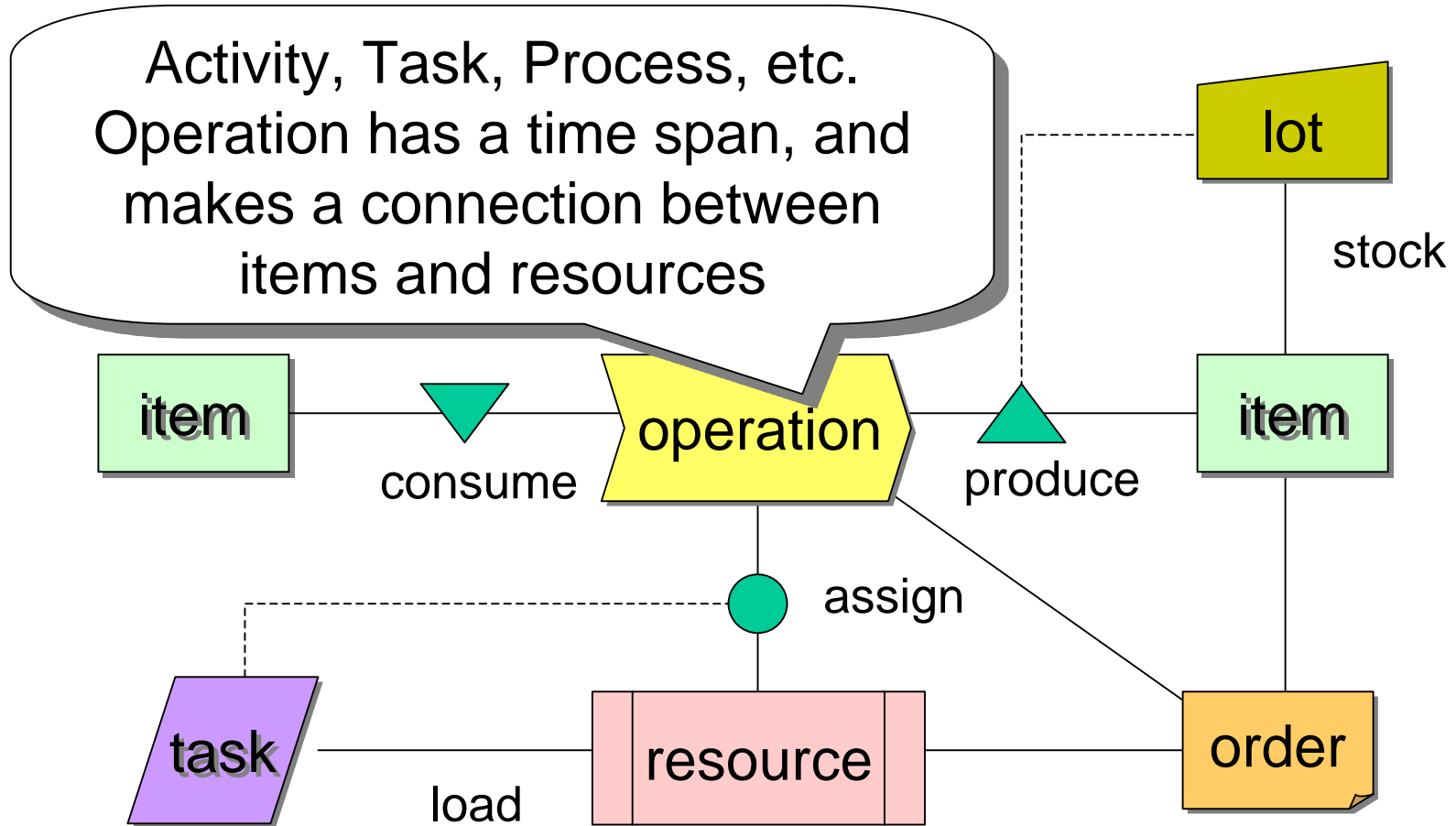
# Core components



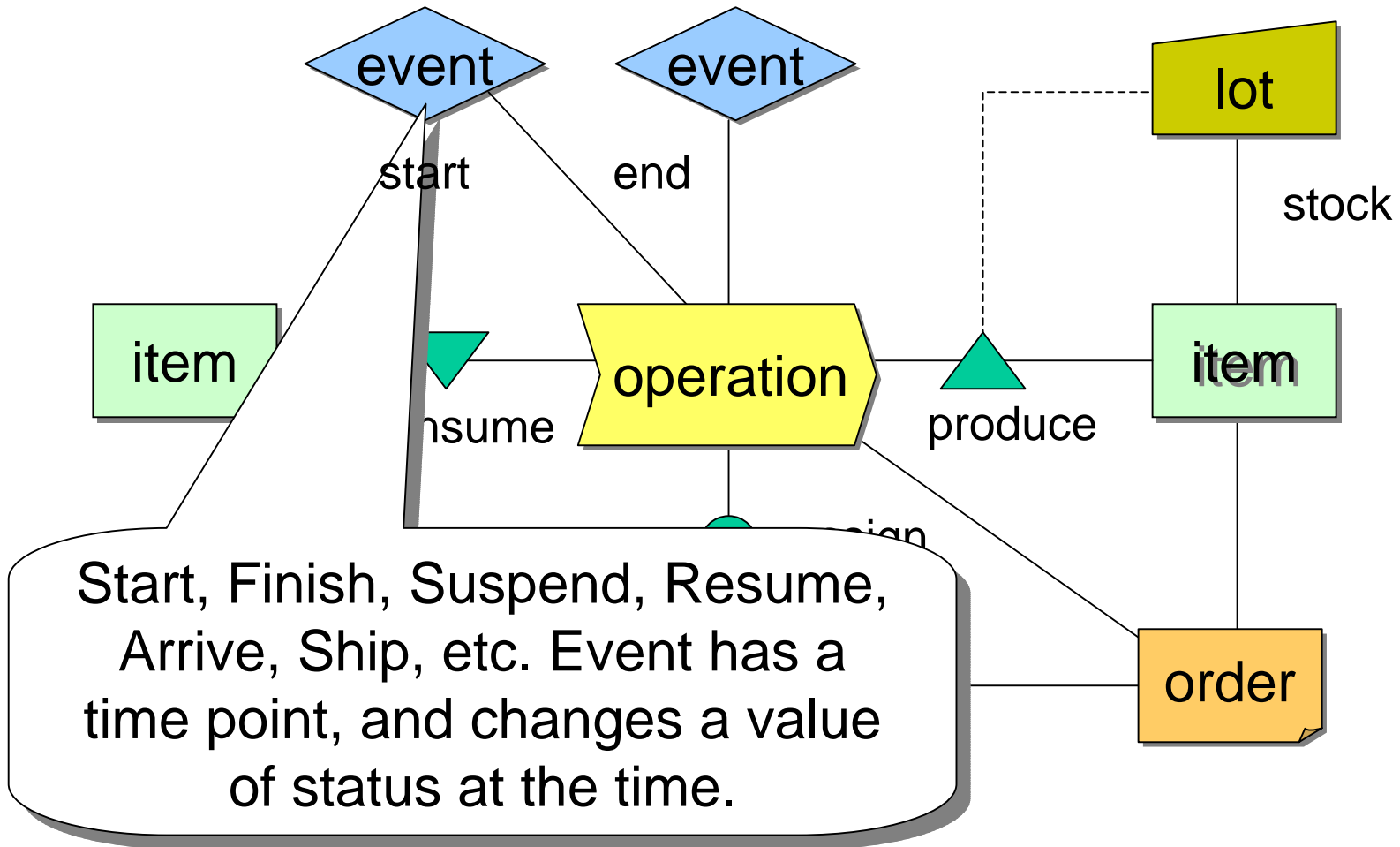
# Core components



# Core components

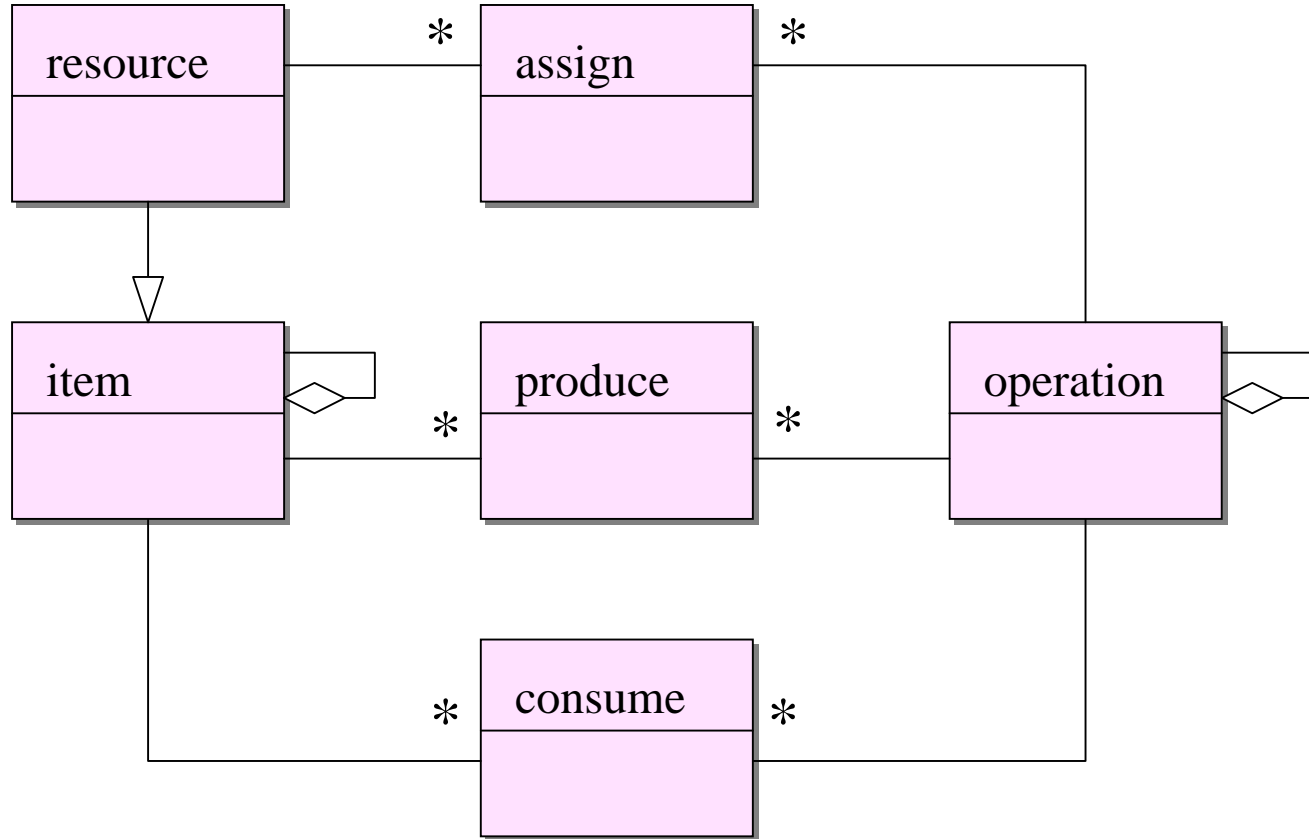


# Core components



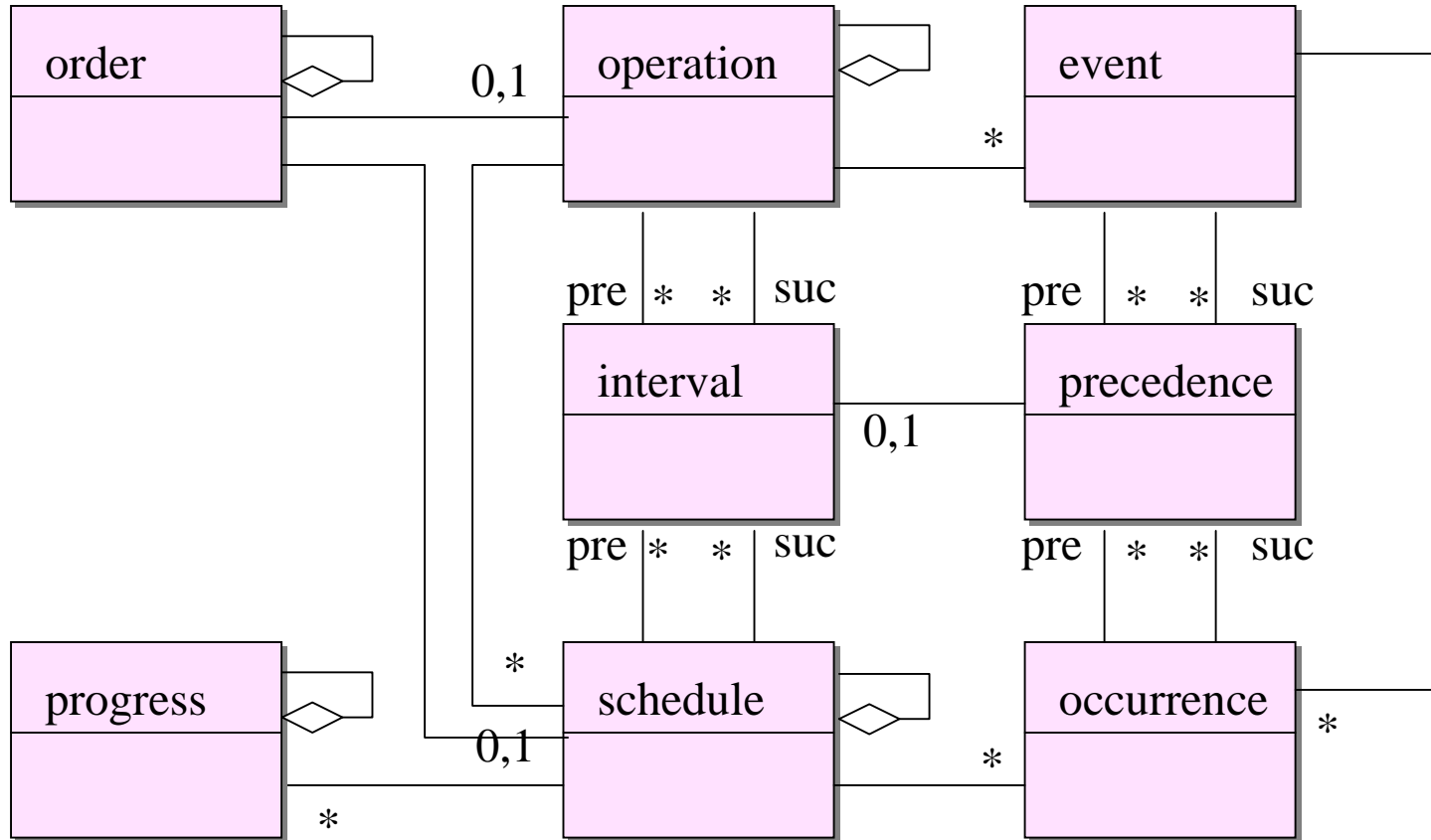


# UML specification (1)



## item and operation

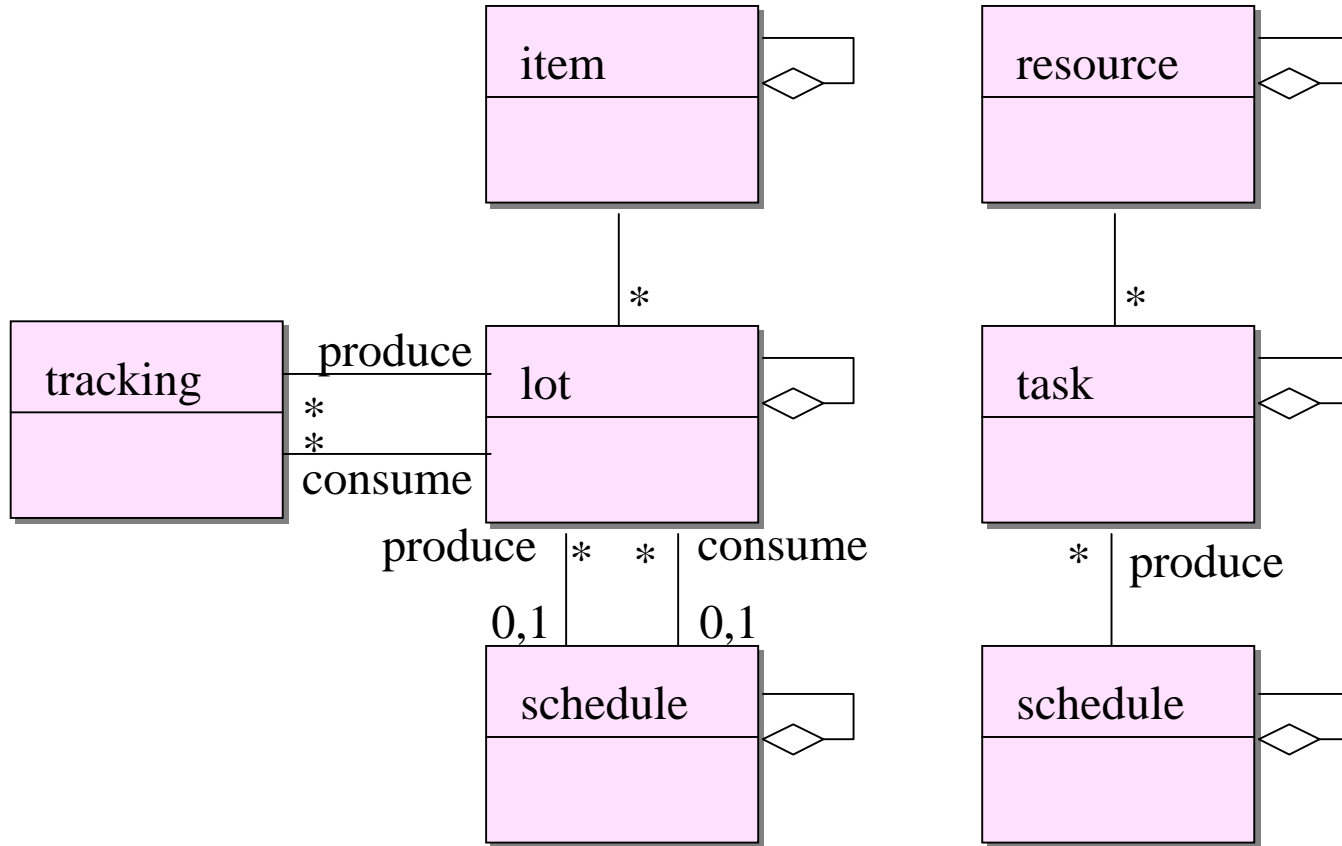
# UML specification (2)



## operation and event



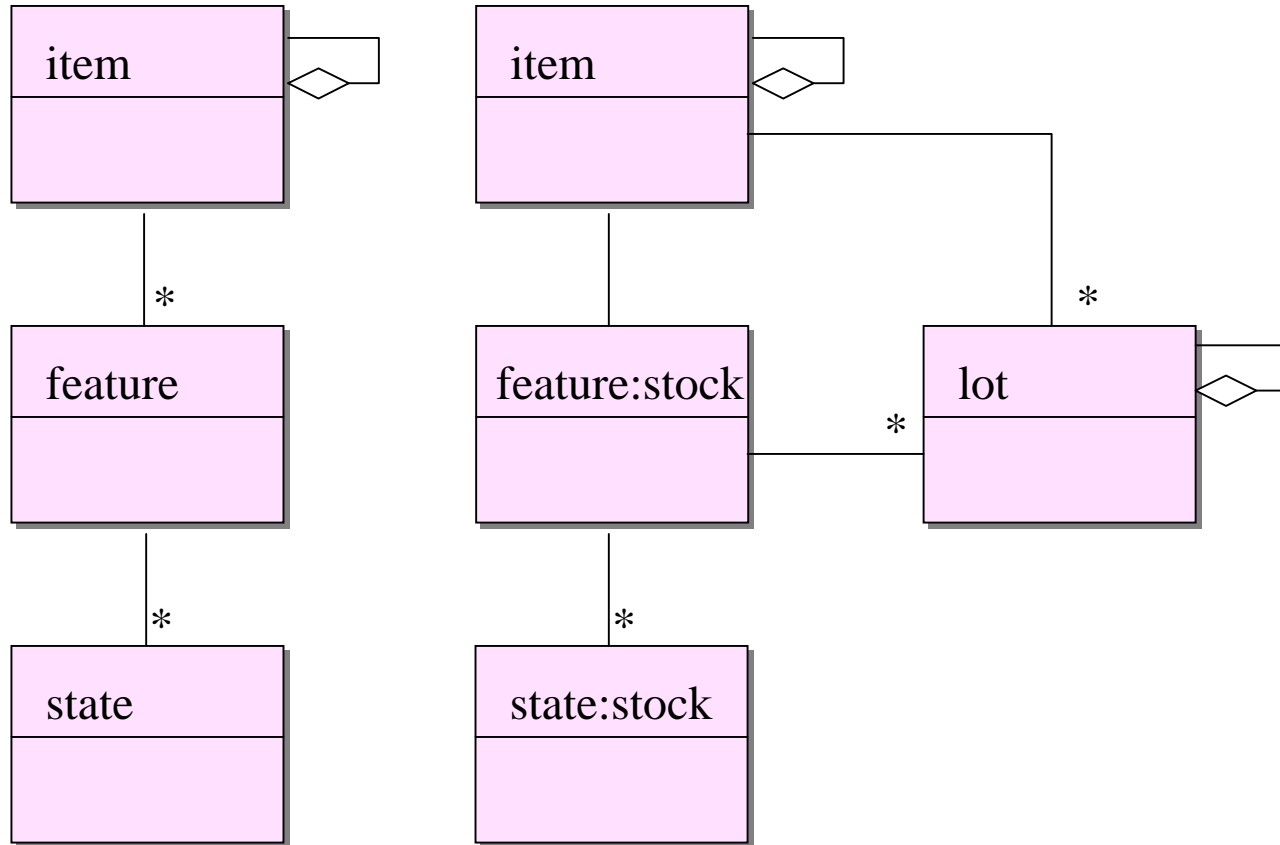
# UML specification (3)



## lot and task

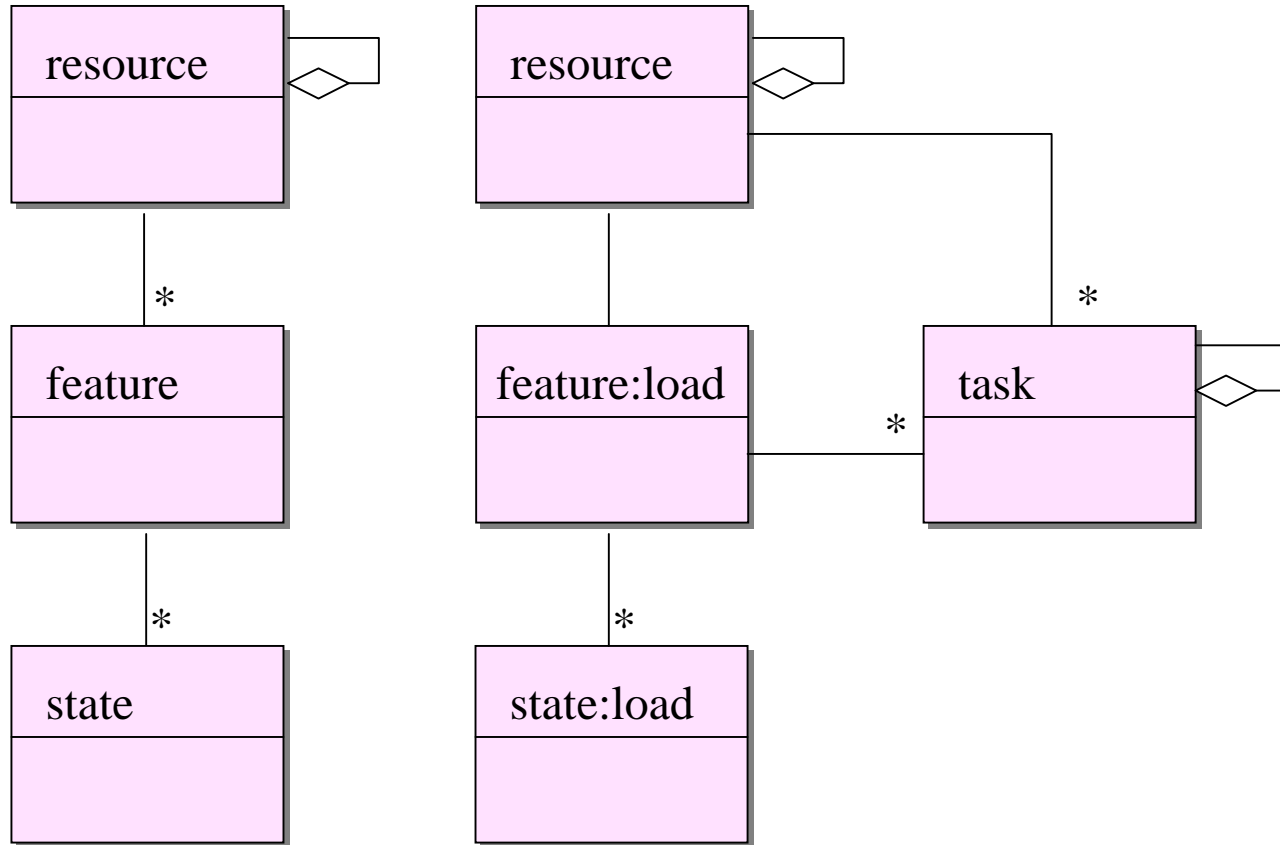


# UML specification (4)



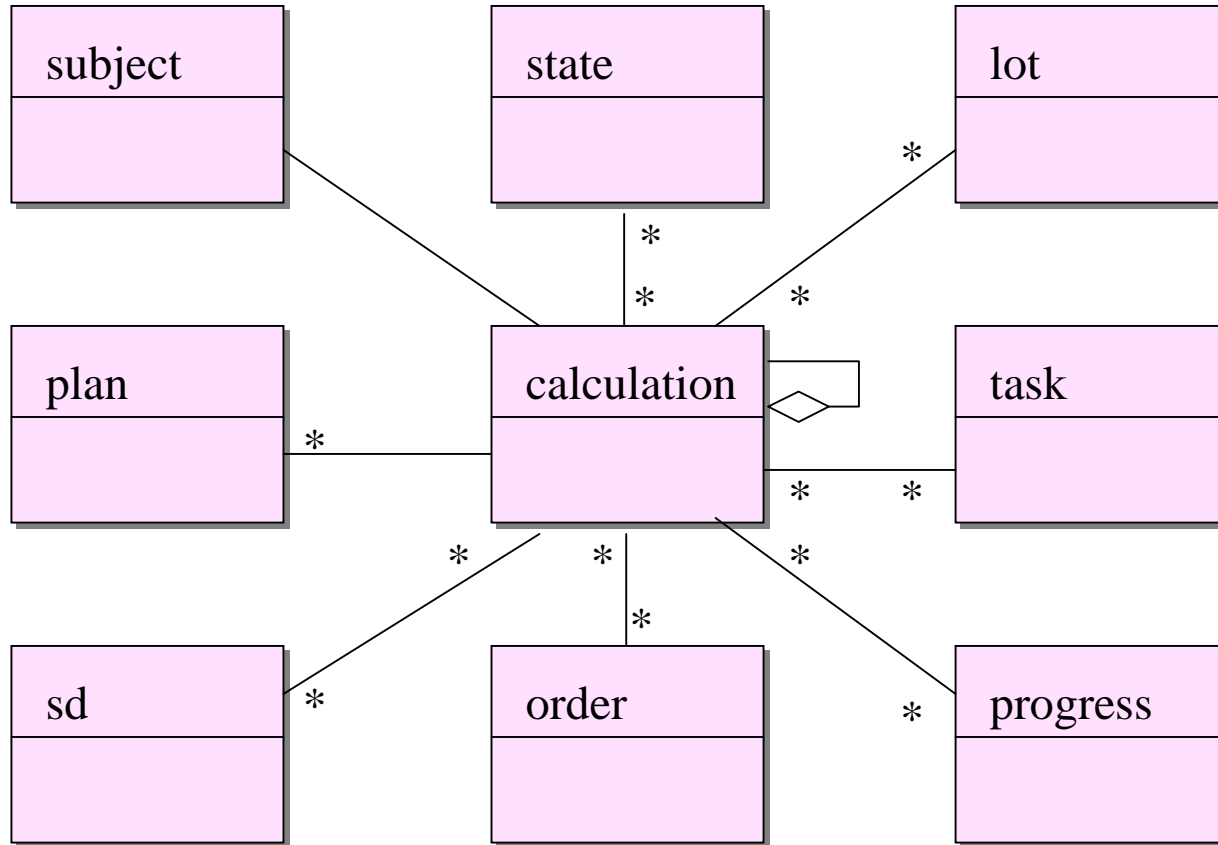
## item and stock

# UML specification (5)



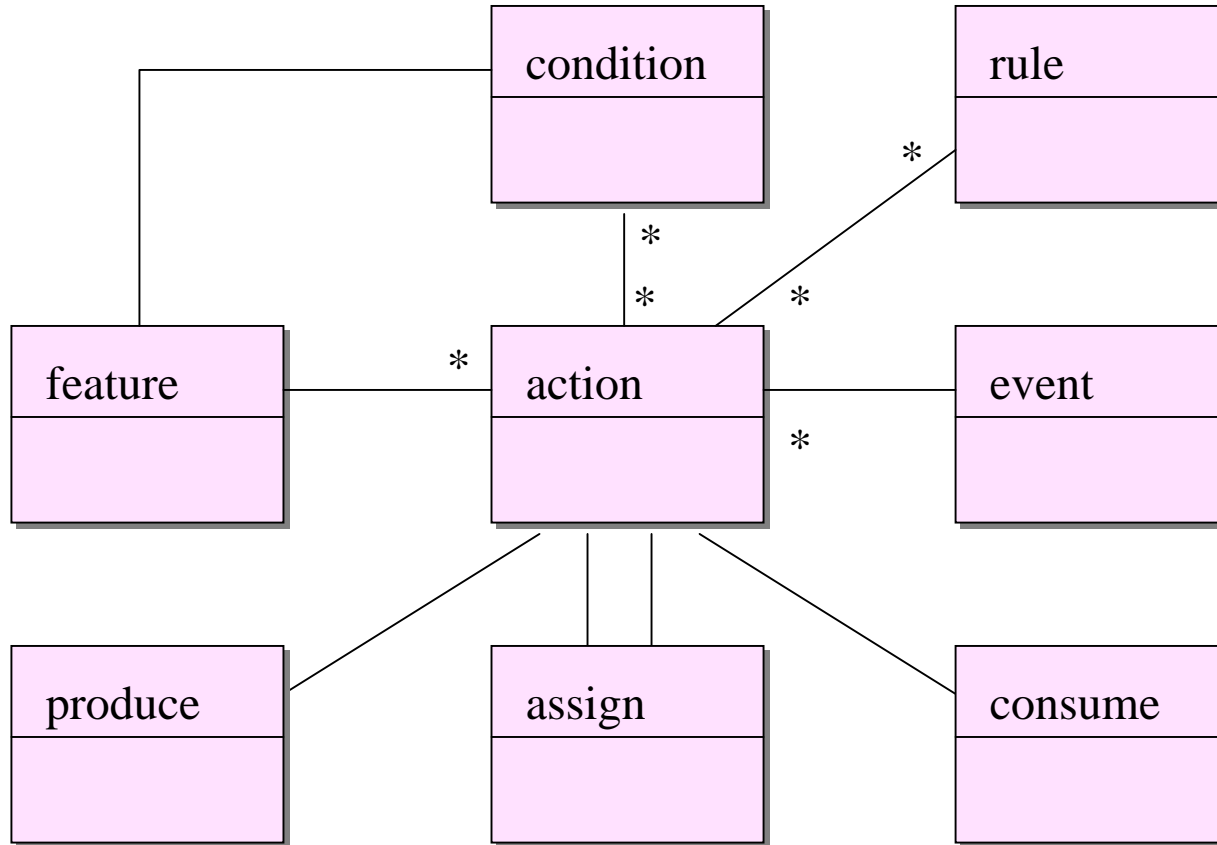
## resource and load

# UML specification (6)



## calculation and plan

# UML specification (7)



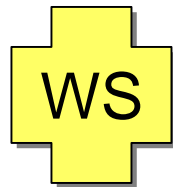
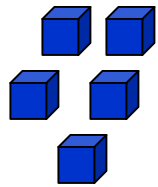
## action and condition



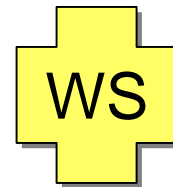
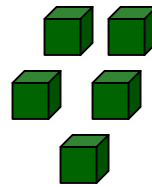


# Sample representation (1)

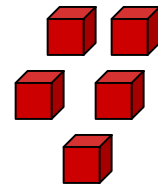
material



WIP

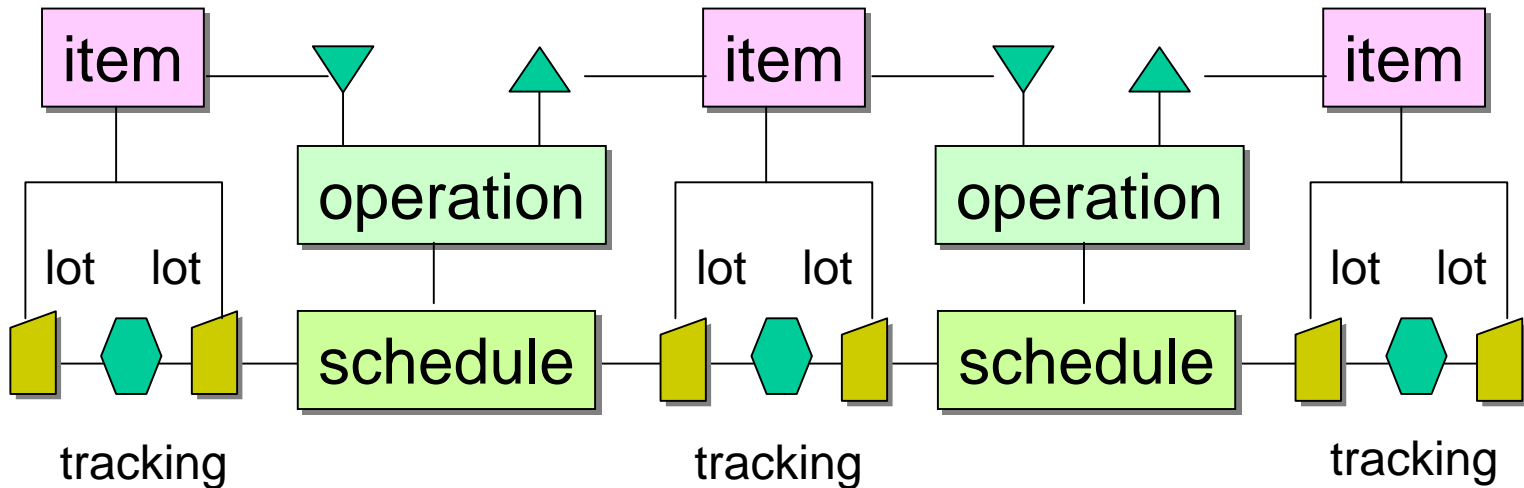


products



consume produce

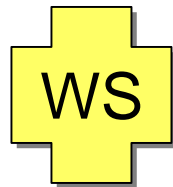
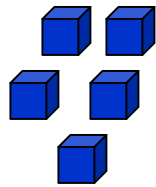
consume produce



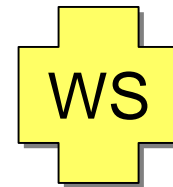
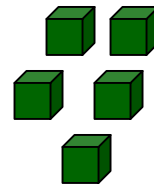


# Sample representation (2)

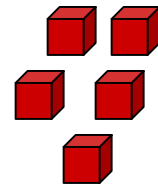
material



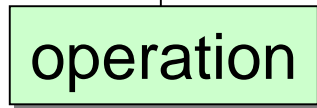
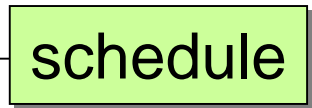
WIP



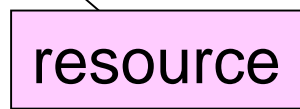
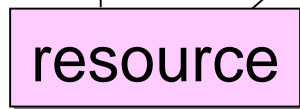
products



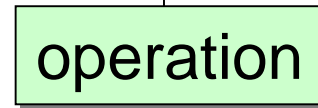
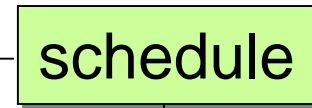
task



assign

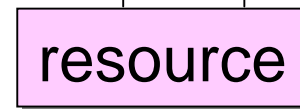
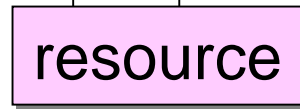


task



assign

assign



task

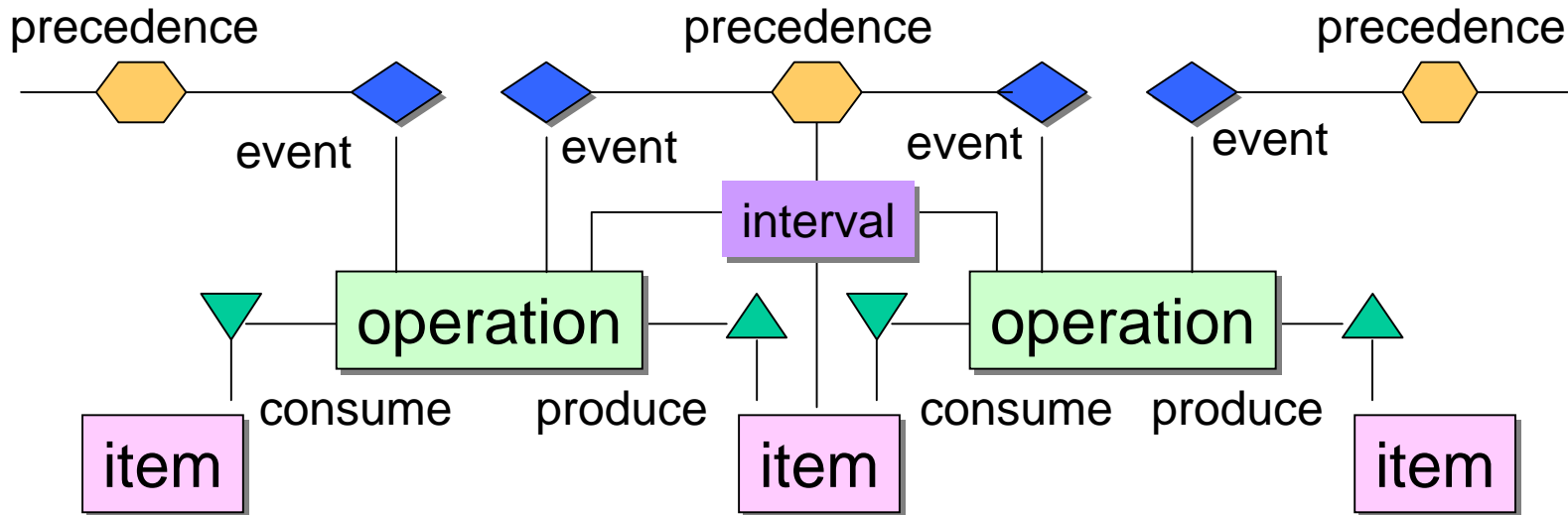
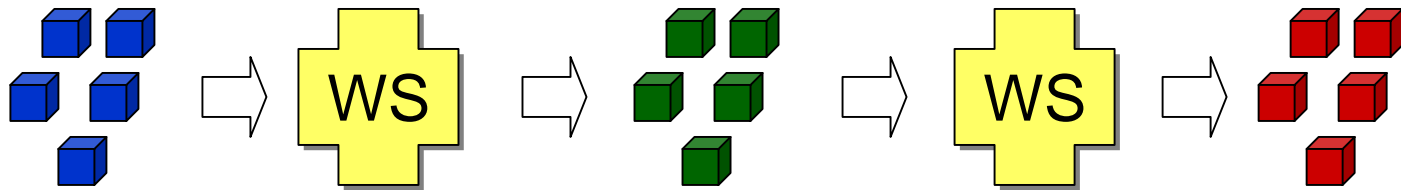


# Sample representation (3)

material

WIP

products





# Outline

- Define core components as ontologies necessary for representing production planning and scheduling problems
- Define naming rules to create new interfaces and describe their functions in accordance with business processes
- Define translation processes from core components to particular object schema used in industrial applications



# General functional terms

- General functional terms are common key words to represent functionality of an interface in an abstract level

## Set

- Register data or input some data to the agent. Change, cancel and delete request are also in this category.

## Get

- Inquiry data or output some data from the agent. This implies calculation if the request data is not exist in the agent.



# Naming rule of Interface

- Set or store any data to the data store of the agent (Set Data Interface)

“set” + “*Noun*”, e.g., setOrder, setProgress

- Get or retrieve any data from the data store of the agent (Get Data Interface)

“get” + “*Noun*”, e.g., getStock, getCalendar

- Do one of any functions the agent allows to be executed (Do Something Interface)

“*verb*” + “*Noun*”, e.g., makeSchedule

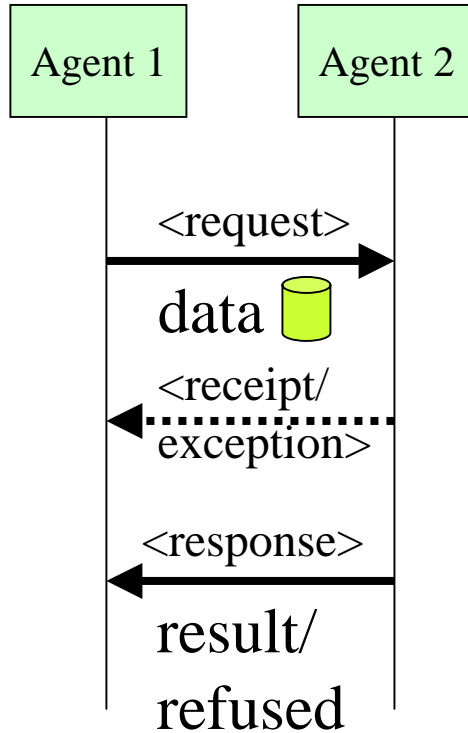


# Communication pattern

- Single interaction with the target, i.e., a business process is divided according to a pair of bi-directional messages.
- Only receipt information can be doubled from a same target in a pattern.
- Bi-directional messages conducted by an interface form a communication pattern, which would be a building block of a business process.

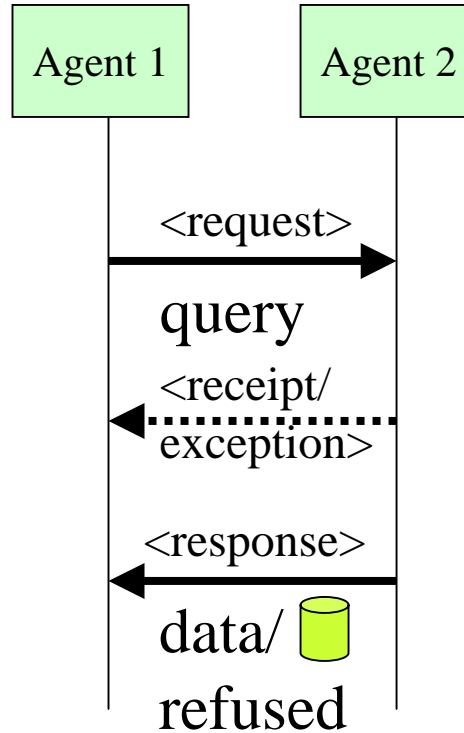
# Communication pattern

initiator



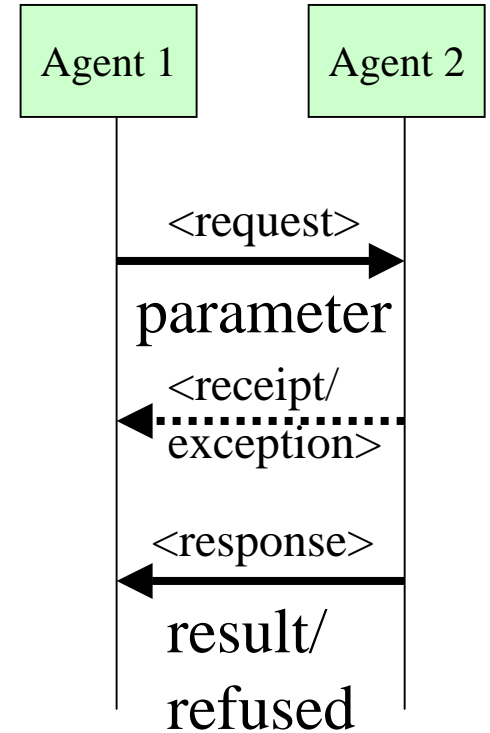
set data

initiator



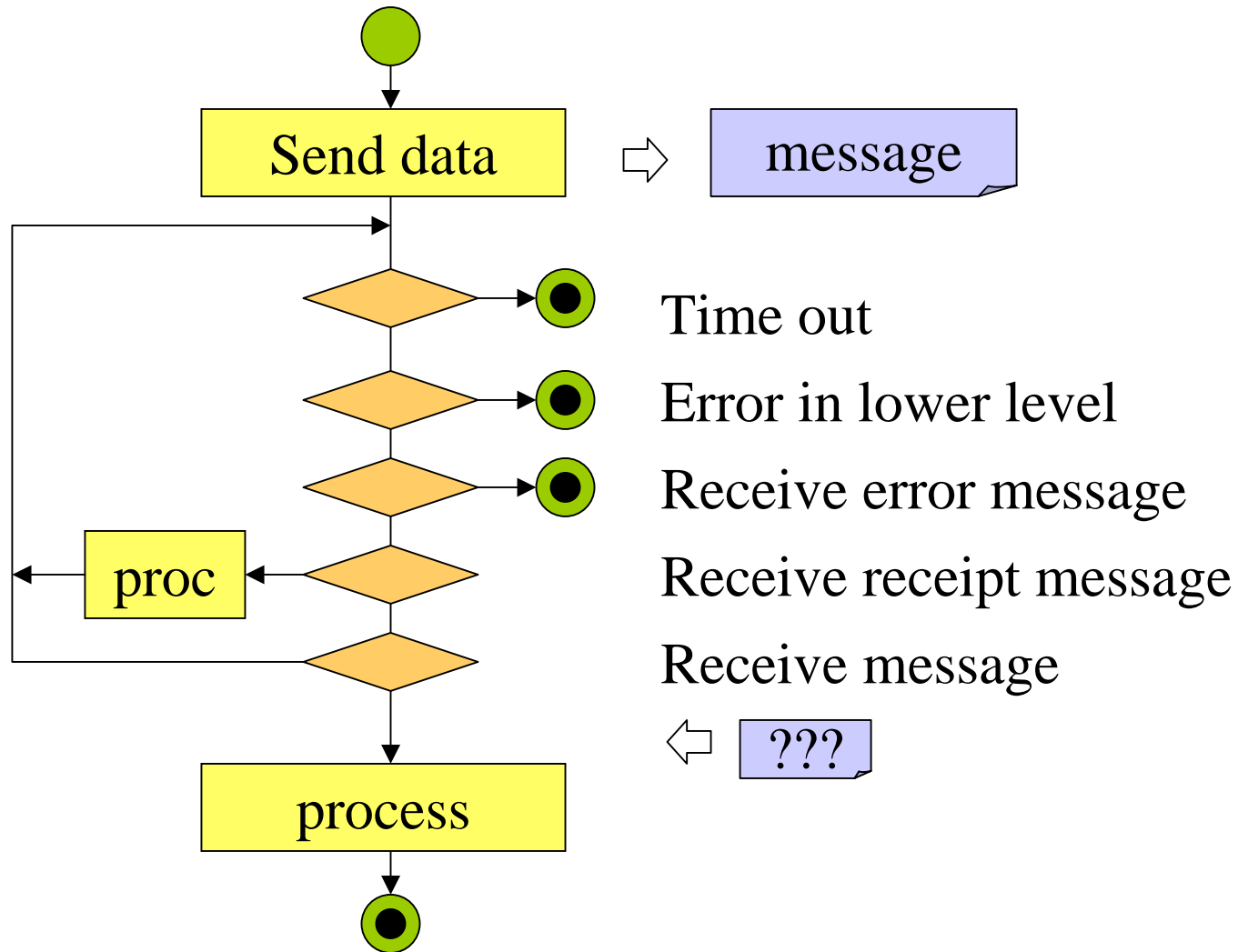
get data

initiator



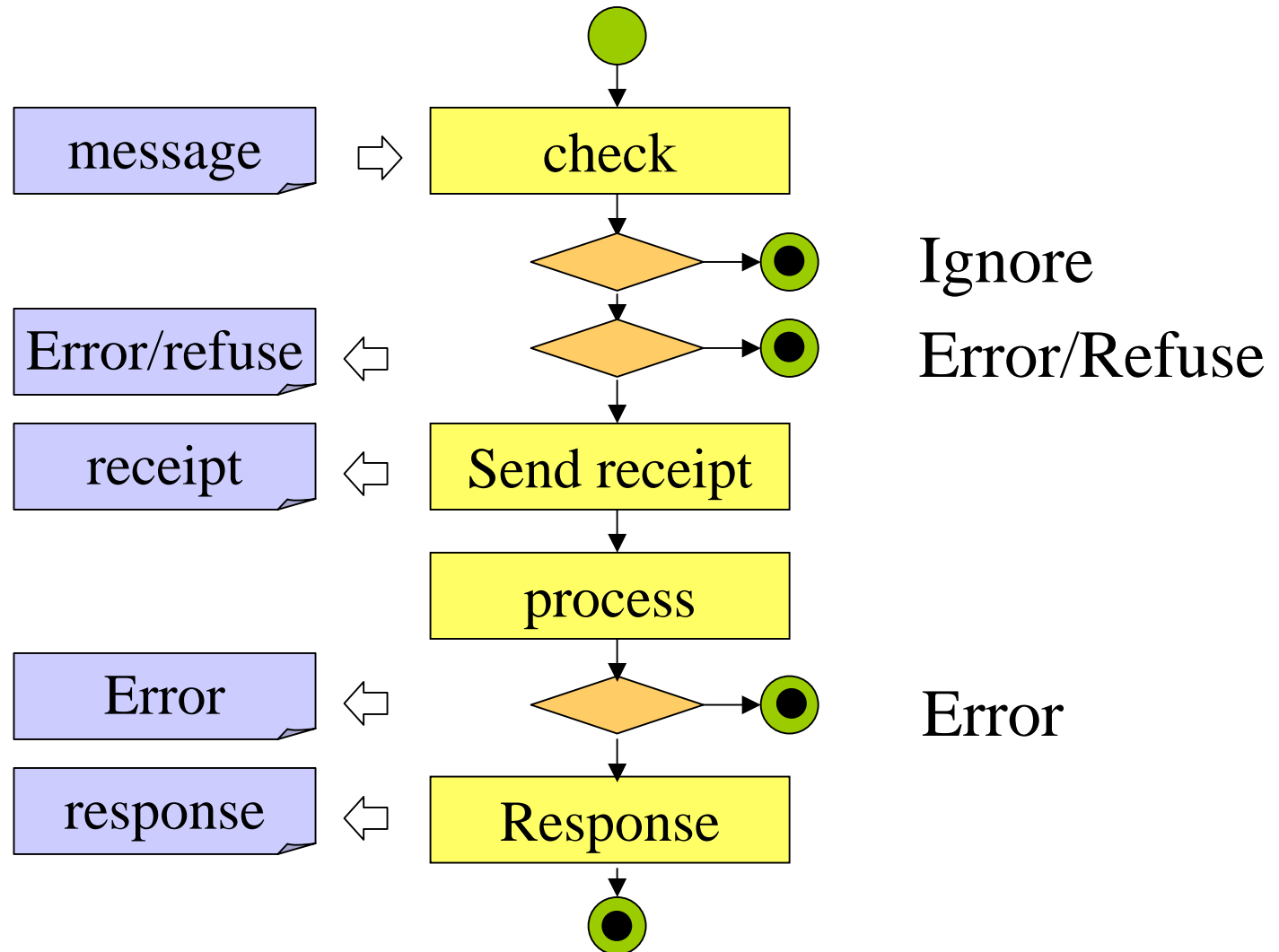
do something

# How should an agent wait



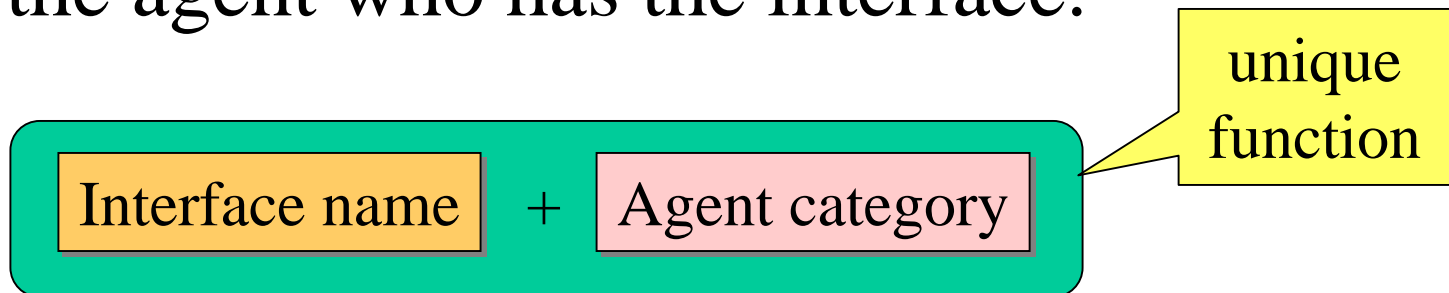


# How should an agent react



# Interface identification

- Function should be defined by an interface name and the class of the target agent, i.e., one interface name can represent several functions corresponding to each agent.
- Therefore, an interface name itself should be independent from the characteristics of the agent who has the interface.





# Specification of Interface

- For each interface name:
  - Message schema of request
  - Message schema of response
- For each interface design (name & agent)
  - Business function in terms of response
  - Control parameters in the message
  - Exception type and their behavior
- For each interface implemented
  - Detail behavior to expectable situations



# How to extend the interfaces

- Check the PSLX interface repository in order to avoid the interface name duplication.
- Use General functional terms and PSLX ontologies as a vocabulary. You can add such vocabulary as a candidate to new interface name.
- Not touch existing other interfaces and their structure. Any Interface never eliminated
- All interface name can be used by all kinds of agents in the APS domain.



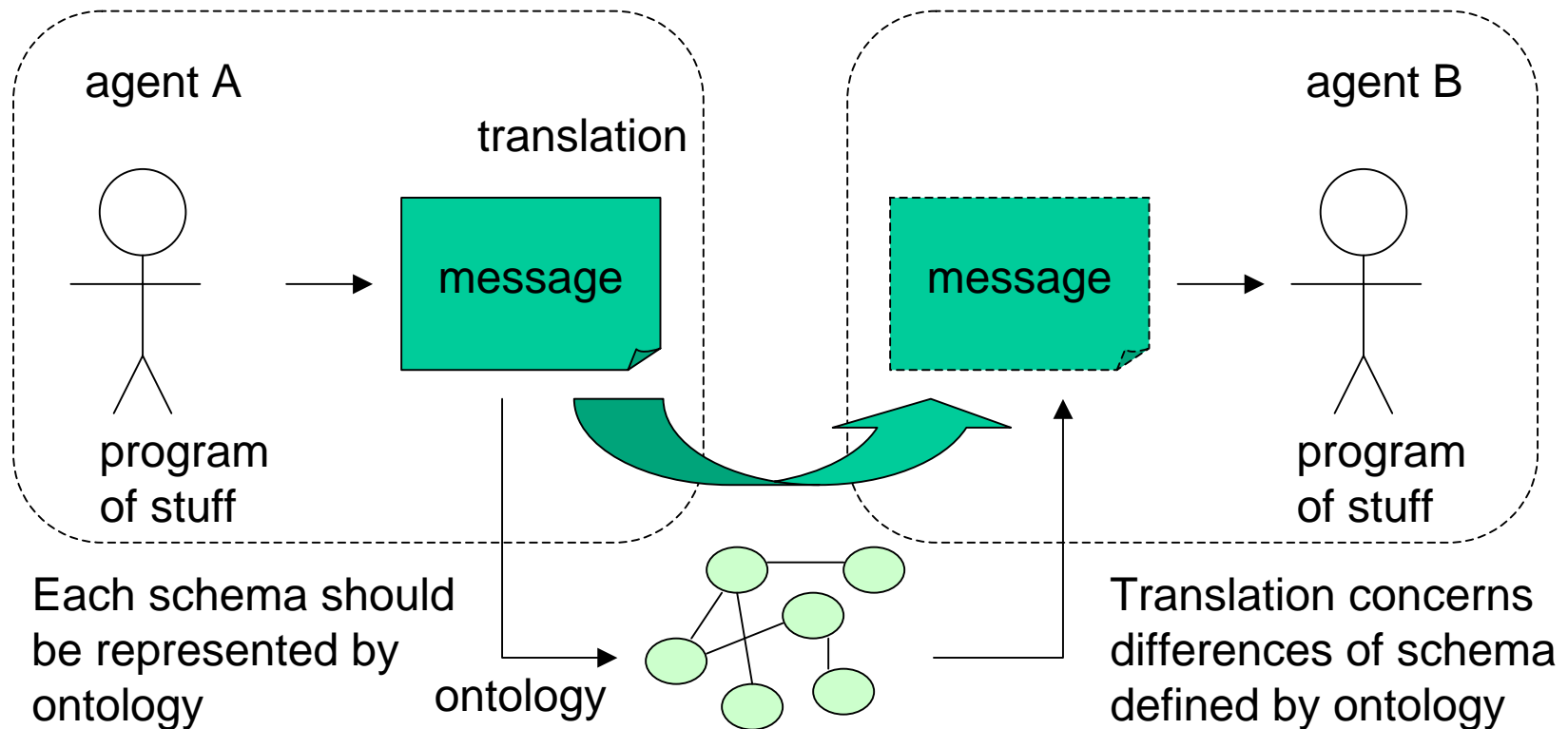
# Outline

- Define core components as ontologies necessary for representing production planning and scheduling problems
- Define naming rules to create new interfaces and describe their functions in accordance with business processes
- Define translation processes from core components to particular object schema used in industrial applications

# Communication between two agents

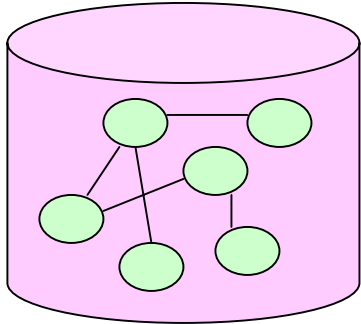
Message that agent A can only understand the meaning

Message that agent B can understand the meaning



# Schema and meta schema

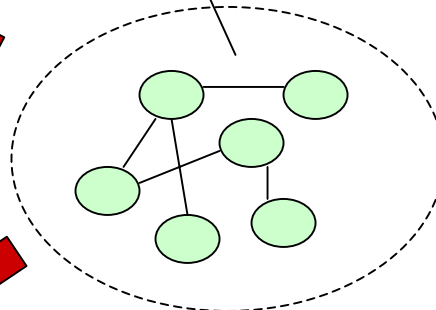
Data base A



Schema 1

PSLX domain  
Object (CC)

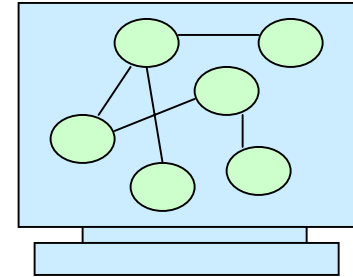
translation  
rule



Meta  
schema

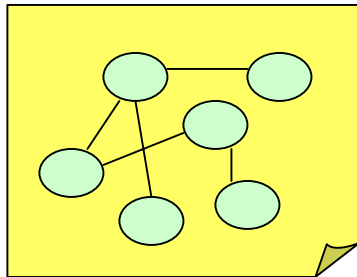
translation rule

Schema 4



On memory program

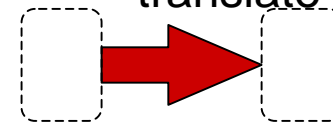
XML document



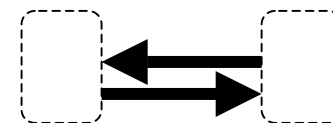
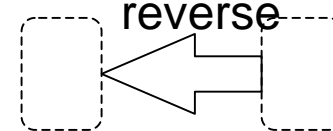
Schema 2

trans  
rule

translate



reverse



Schema  
level

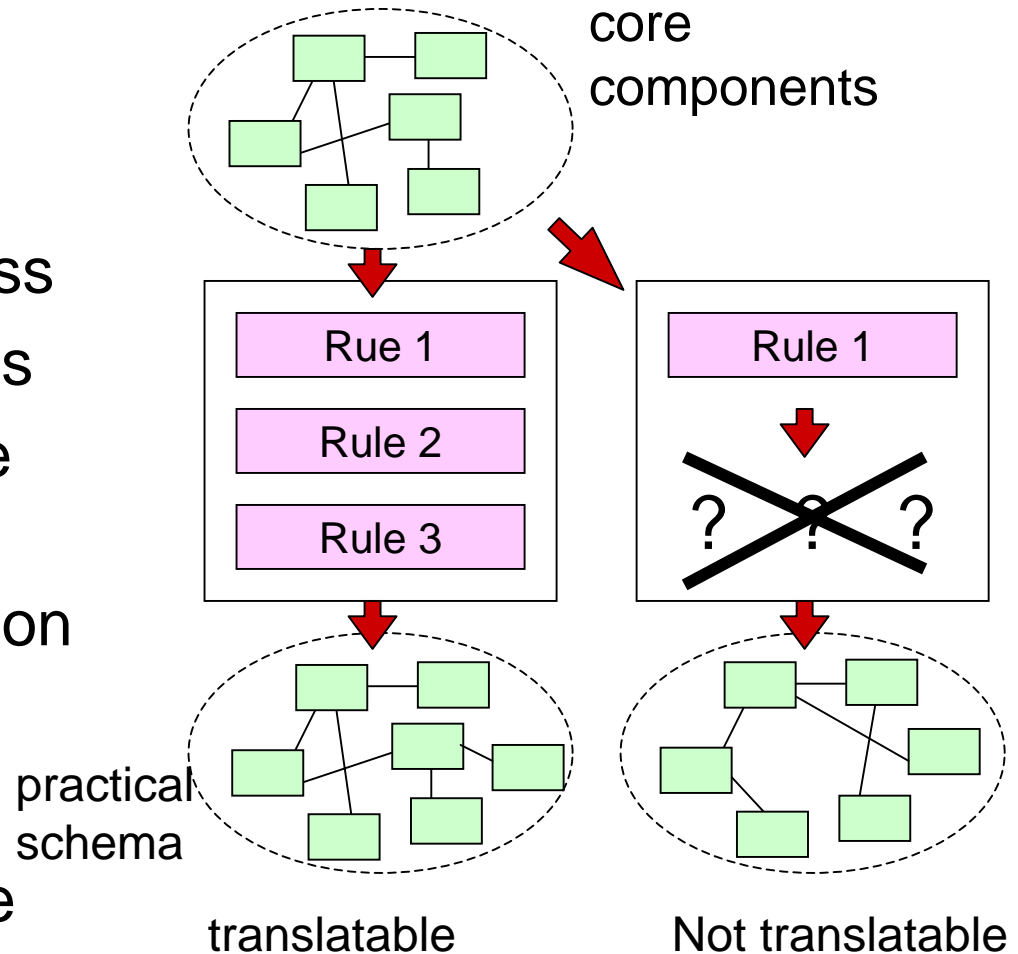
Data  
level

Domain objects

Particular schema

# Rules of schema translation

- create subclass
- divide class
- merge class
- create attribute class
- create relation class
- add/delete attribute
- move attribute through class relation
- add/delete relation
- add constraints
- change class name



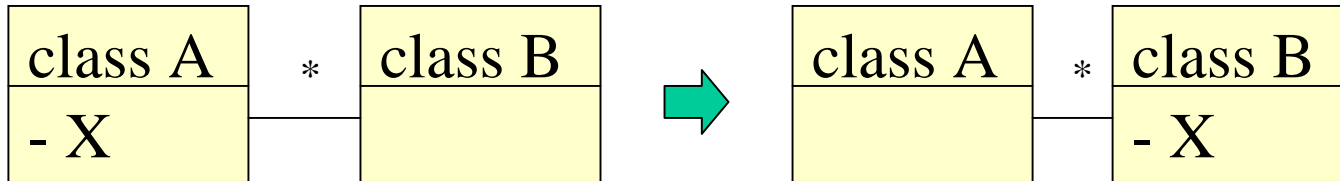


# Translation type

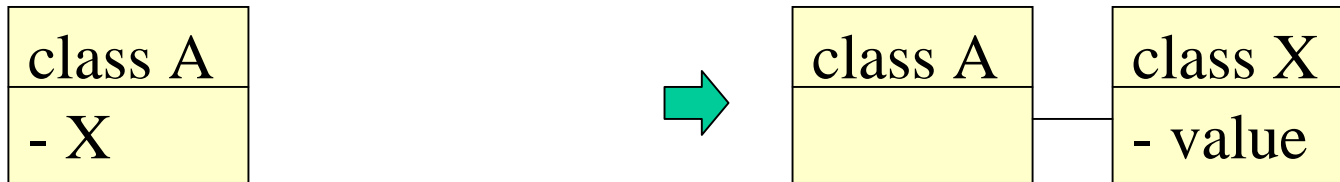
- Equivalent translation
  - move attribute, create attribute class, create relation class, rename
- Expand the target area
  - create class, delete relation, , add constraint add concurrence,
- Reduce the target area
  - delete class, add relation, add constraint, reduce concurrence,
- Add complexity to the model
  - add attribute, create sub-class, divide class
- Eliminate complexity from the model
  - delete attribute, merge classes

# Equivalent translation

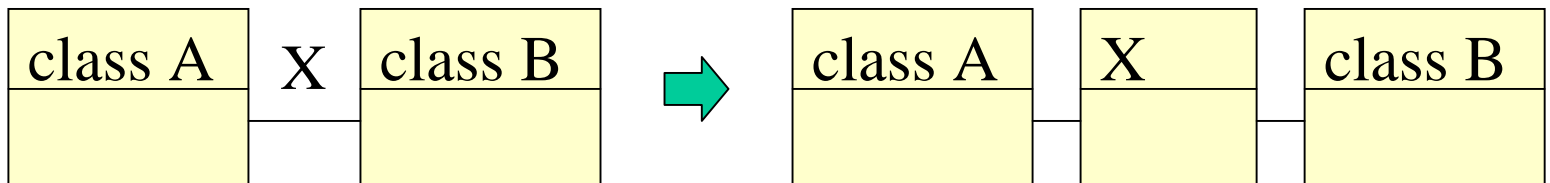
move attribute



create/delete attribute class



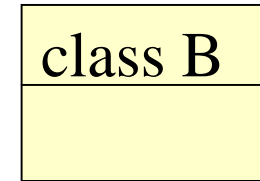
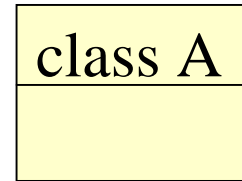
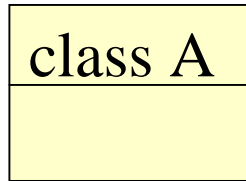
create/delete relation class



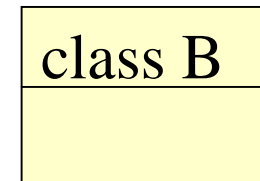
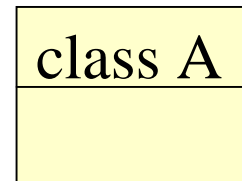
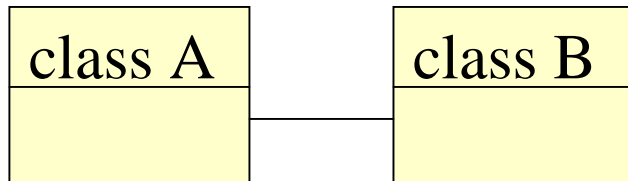


# Expand/reduce target area

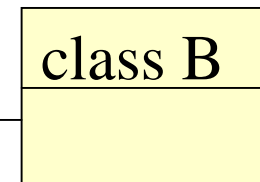
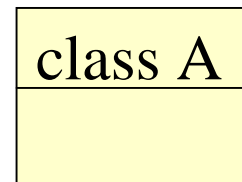
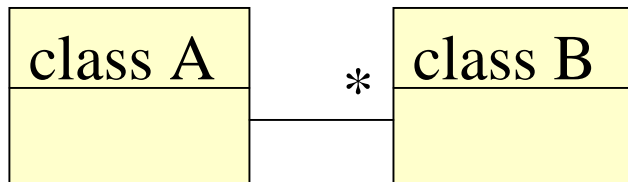
create class



delete/add relation



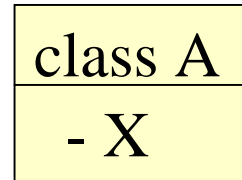
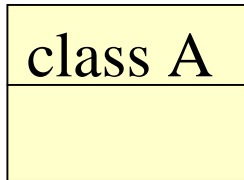
add/reduce concurrency



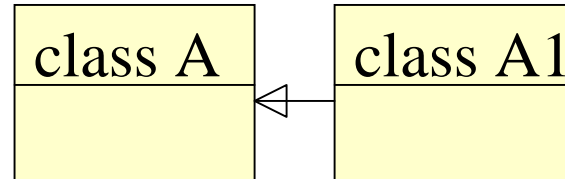
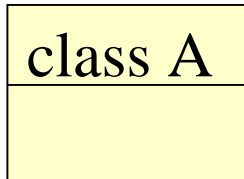


# Add/eliminate complexity

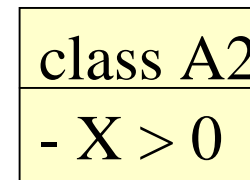
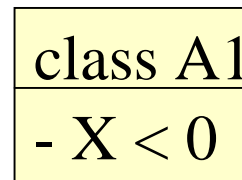
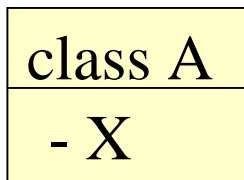
add/delete attribute,



create/delete sub-class



divide/merge class





# Sub classes (example)

Class name	Sub class name
item	resource, product, material, wip, subassy
resource	equipment, tool, labor, ws, shop, site
feature	stock, load, capacity, location
event	start, end, halt, resume
operation	fabrication, transportation, storage, inspection, setup, maintenance
party	customer, supplier
calculation	cost, profit



Thank you

<http://www.pslx.org>