Distributed Artificial Intelligence and Intelligent Agents

Agent-Oriented Software Engineering

Mihhail Matskin

How to build the multiagent systems from a software engineering point of view

Lecture Outline

1. When is an agent-based solution appropriate?
2. AOSE Methodologies:
    1. AAII (*Kinny*)
    2. Gaia (*Wooldridge et. al*.)
    3. Agent UML (*Bauer et. al*.)
    4. Other extentions
3. Formal Methods for AOSE
4. Pitfalls in agent development

References - Curriculum

- **Wooldridge: ”Introduction to MAS”, Chapter 9**

- **Additional reading**:

– M. Wooldridge, N. R. Jennings, and D. Kinny. The Gaia Methodology for Agent-Oriented Analysis and Design. In *Journal of Autonomous Agents and Multi-Agent Systems*. 3(3):285-312. 2000. http://www.csc.liv.ac.uk/~mjw/pubs/jaamas2000b.pdf

– Odell et al, “Representing Agent Interaction Protocols in UML

– http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.83.4611&rep=rep1&type=pd

– Bauer, ”UML Class Diagrams Revisited in the Context of Agent-Based Systems”

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.86.5561&rep=rep1&type=pdf

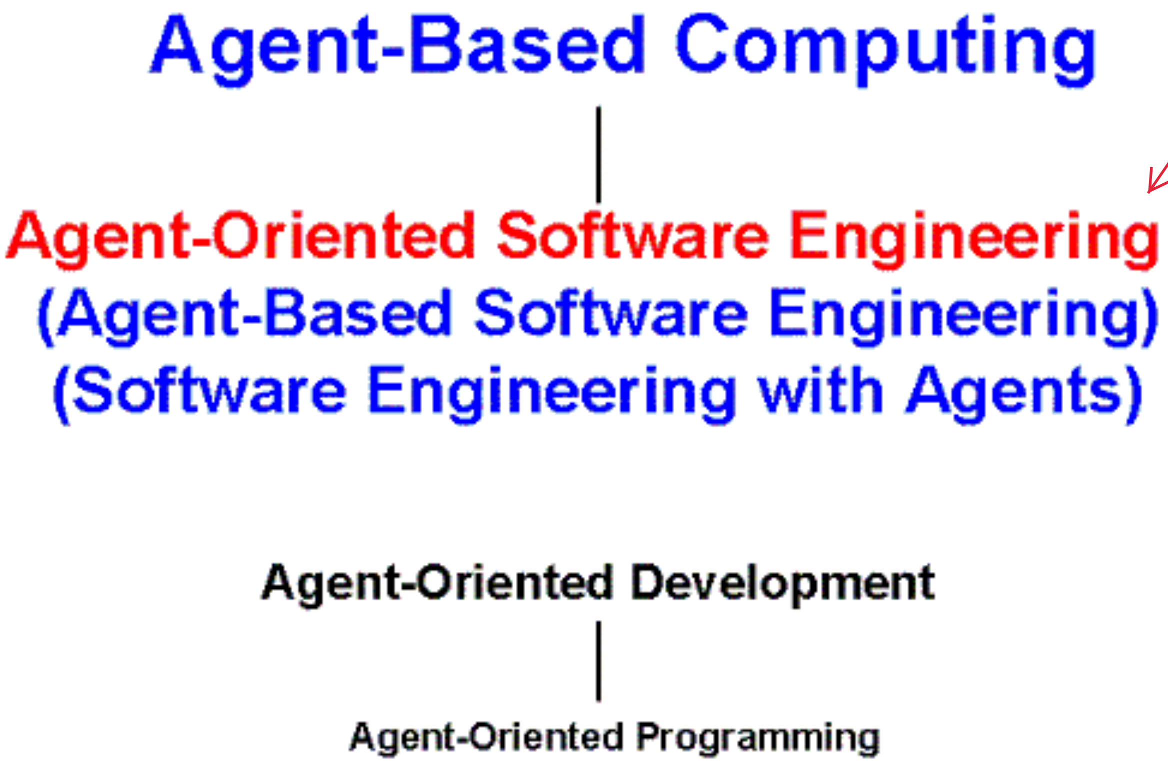
- Yan et al, “romas: a role-based modeling method for multi-agent system “

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.2.7706&rep=rep1&type=pdf

- M. Wooldridge and N. R. Jennings. Pitfalls of Agent-Oriented Development.

http://agents.umbc.edu/introduction/paod.pdf

Terminology of Agent-Oriented Software Engineering


Agent-Based Computing
Agent-Oriented Software Engineering
(Agent-Based Software Engineering)
(Software Engineering with Agents)
Agent-Oriented Development
Agent-Oriented Programming

how to build multiagents systems

Agent-oriented Software Engineering (AOSE)

- Concerns methodologies to support the development and maintenance of agent systems.
- This is similar to methodologies that have been successful in the development of OO systems:
  - e.g. OMT, UML

When is an agent-based solution most appropriate?

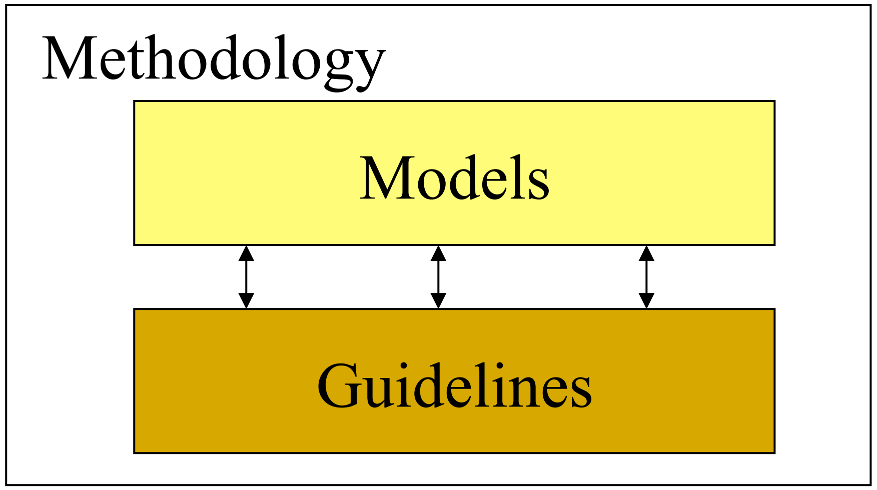
- The environment is open, or at least highly dynamic, uncertain or complex
- Agents are a natural metaphor
- Distribution of data, control and expertise
- Legacy systems
- …

Criticisms of Existing Approaches

- Approaches such as object-oriented design fail to capture:
  - An agent's flexible, autonomous problem-solving behavior
  - The richness of an agent's interactions

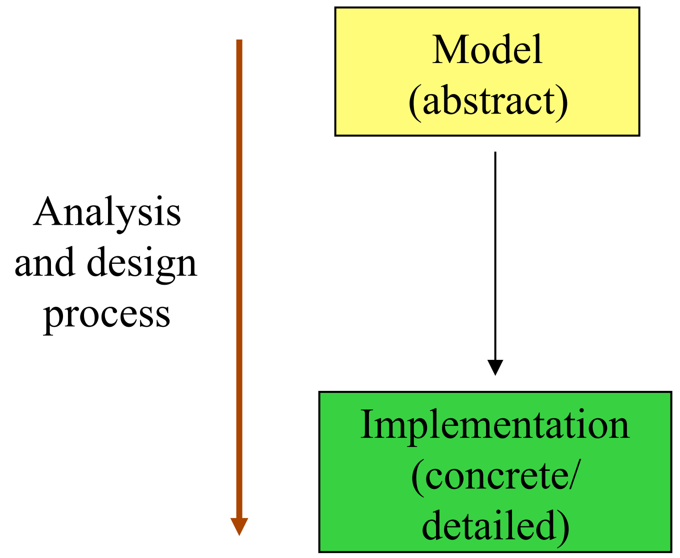
Agent-oriented Analysis & Design Techniques

- An analysis and design methodology is intended to assist in:
  - Gaining an understanding of a particular system
  - Designing the system.



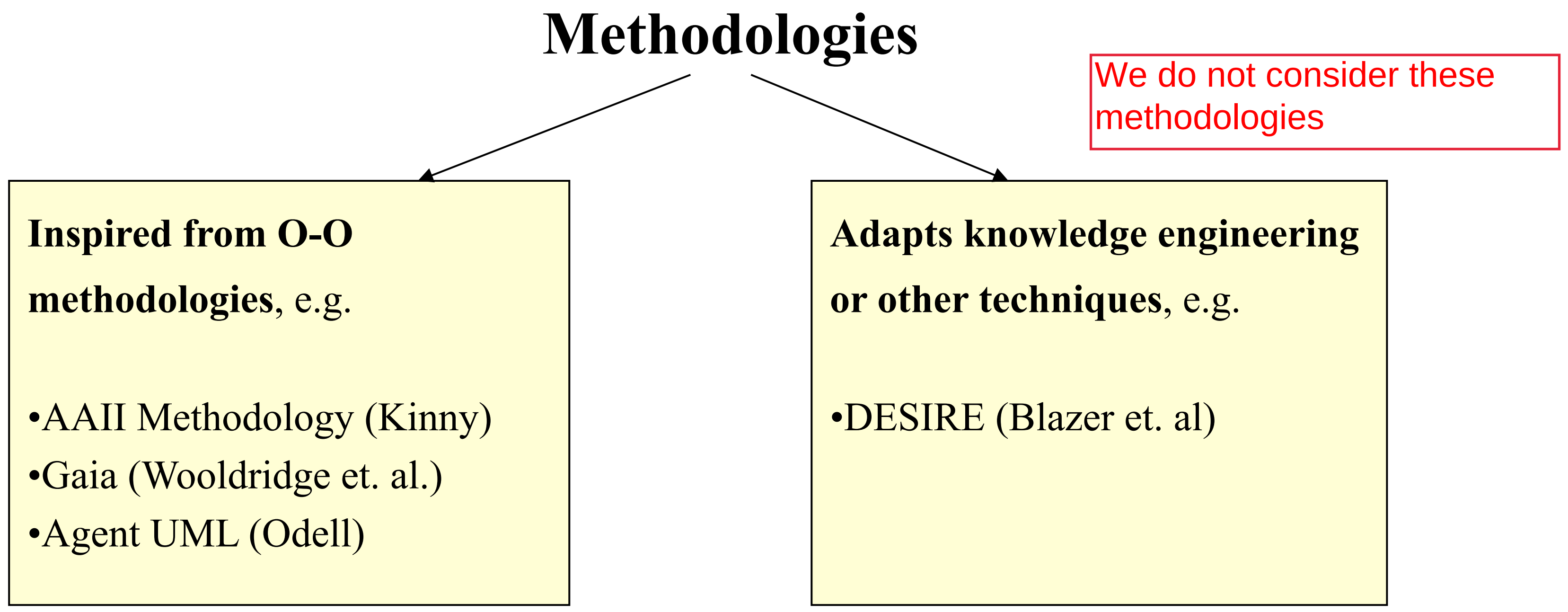
Agent-oriented Models

- Agent-oriented models are intended to formalize understanding of a system under consideration.



Methodologies

- Methodologies for the analysis and design can be divided into 2 groups.



Methodologies

We do not consider these methodologies

Inspired from O-O methodologies, e.g.

•AAII Methodology (Kinny)
•Gaia (Wooldridge et. al.)
•Agent UML (Odell)

Adapts knowledge engineering or other techniques, e.g.

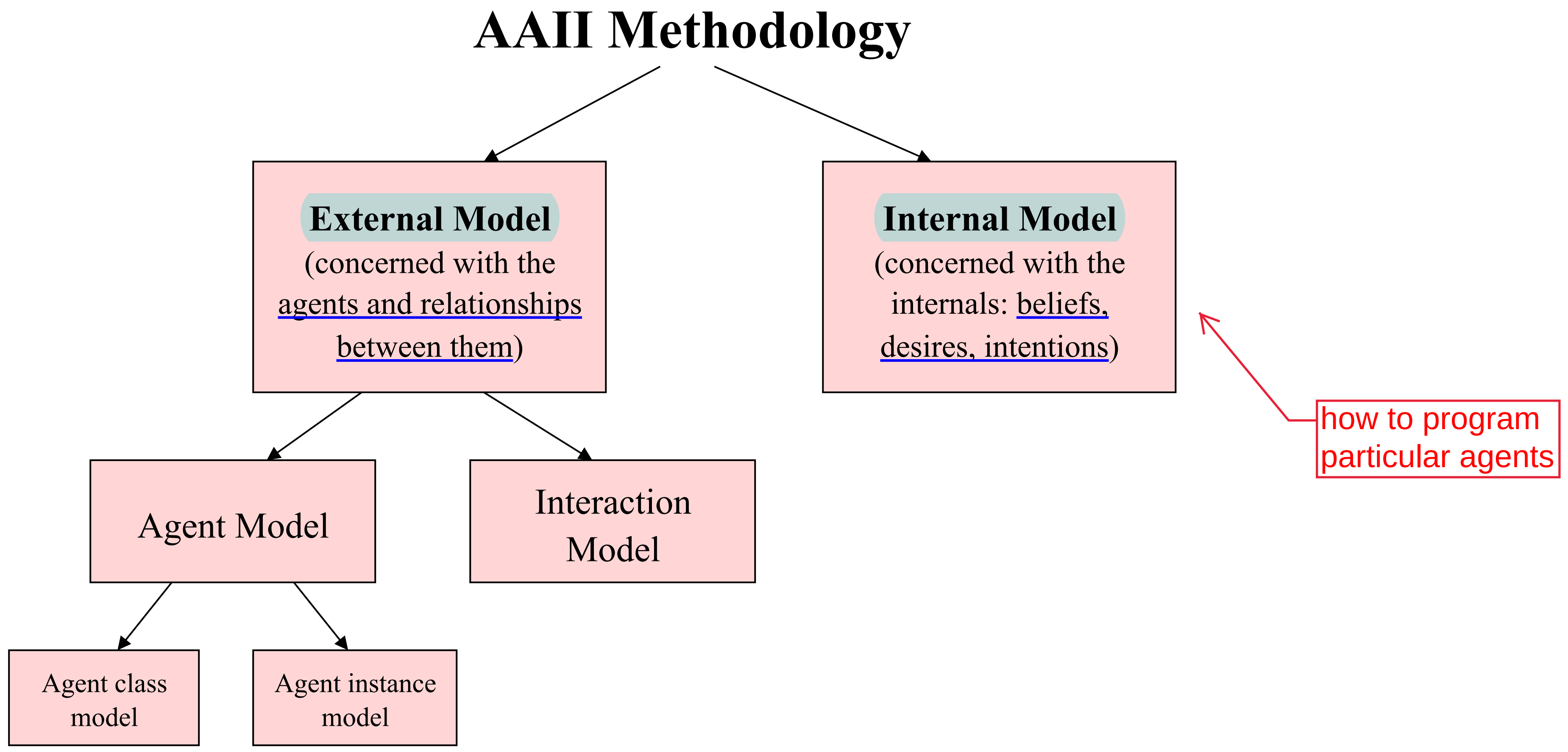
•DESIRE (Blazer et. al)

AAII Methodology 1
(Kinny et. al.)

Australian
Artificial
Intelligence
Institute

- **Aim**: to construct a set of models, which when fully elaborated, define an agent system specification.

AAII Methodology 2


AAII Methodology
External Model
(concerned with the agents and relationships between them)
Internal Model
(concerned with the internals: beliefs, desires, intentions)
how to program particular agents
Agent Model
Interaction Model
Agent class model
Agent instance model

AAII Methodology 3

Role: some functionality that can be performed by different agents

Methodology

•Identify relevant roles in domain

•Develop an agent class hierarchy

•Identify responsibilities for each roles:

- •Services required
- •Services provided

•Determine goals associated with each service

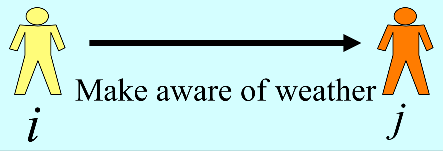
•For each goal, determine plan and context conditions

how will the agent reach the goal?

•Determine belief structure of system

Example

•Role: weather monitor



what should this role do?

•Goals:

- •Find out current weather
- •Make agent *j* aware of it

•Plan for " Make agent *j* aware of it":

•Send message to agent *j*

•How to represent agent *i*'s and agent *j*'s beliefs: temperature(x)

Gaia Methodology
(Wooldridge et. al.)

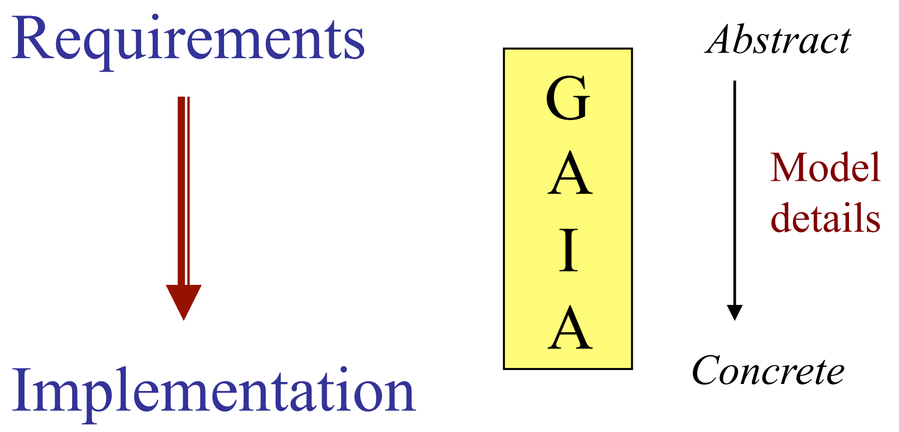
system =
organization

- **Motivation**: Existing software methodologies don't support agents, in particular, interactions and agent organisations.
- It borrows terminology from object-oriented analysis and design.
- Encourages the analyst to think of building agent-based systems as a process of **organisational design**.
  - Societal view of agents.

Reference: Wooldridge et. al. 2000.

Gaia Methodology

- **Gaia** is:
  - **General**: applicable to a range of multi-agent systems
  - **Comprehensive**: macro-level (societal) and micro-level (agent) aspects of systems.
- Allows an analyst to go systematically from a statement of requirements to a design that is sufficient for implementation.

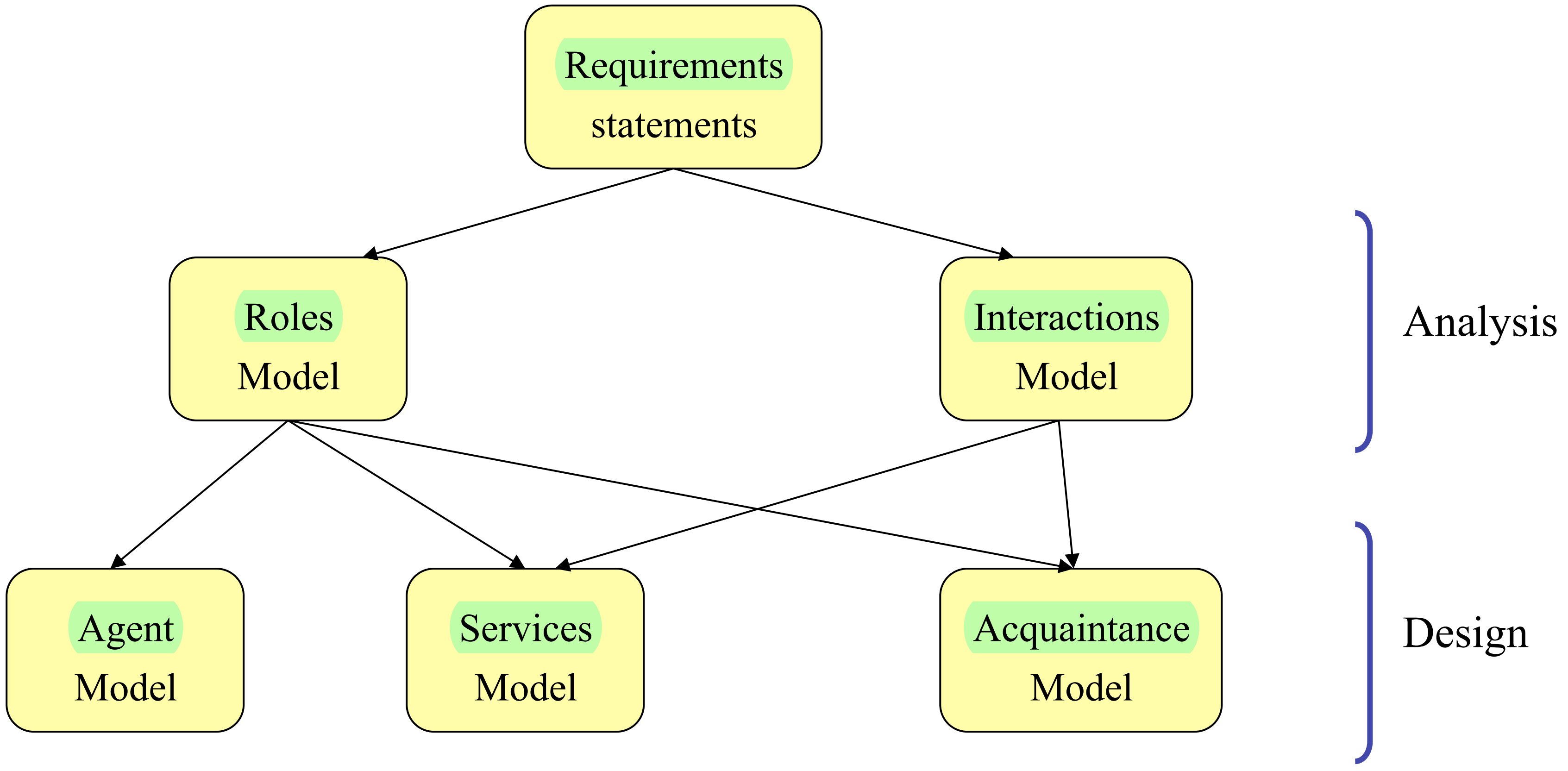


Requirements for an OO system

Reference: Wooldridge et. al. 2000.

Gaia Methodology

Relationships between Gaia Models



At the end, it's still a model, not a running system

Reference: Wooldridge et. al. 2000.

Gaia Methodology

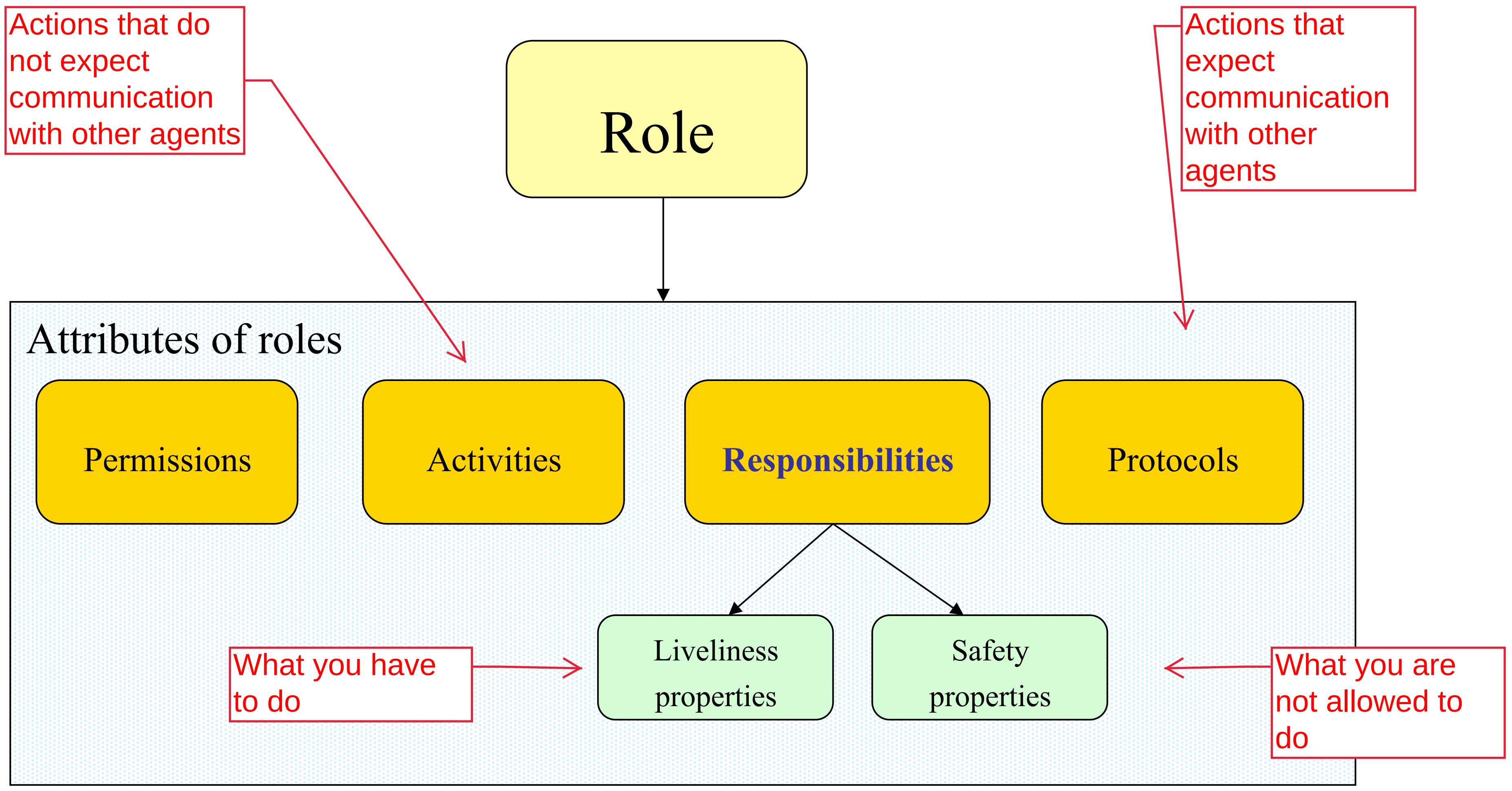
Concepts

Abstract Concepts *(used during analysis to conceptualise the system)*	**Concrete Concepts** *(used within the design process)*
•Roles •Permissions •Responsibilities Liveness Properties Safety properties •Protocols •Activities	•Agent types •Services •Acquaintances

Needed to implement the concepts

Reference: Wooldridge et. al. 2000.

Gaia Methodology
Roles Model



Actions that do not expect communication with other agents
Role
Actions that expect communication with other agents
Attributes of roles
Permissions
Activities
Responsibilities
Protocols
What you have to do
Liveliness properties
Safety properties
What you are not allowed to do

Reference: Wooldridge et. al. 2000.

Gaia Methodology
Role Schema

- A **roles model** is comprised of a set of **role schema**

Role Schema:	***Name of Role***
•Description	•*short description of the role*
•Protocols and activities	•*protocols and activities in which the role plays a part*
•Permissions	•*"rights" associated with the role*
•Responsibilities	
–Liveliness	•*liveliness responsibilities*
–safety	•*Safety responsibilities*

Reference: Wooldridge et. al. 2000.

Gaia Methodology
Interaction Model

- The links between the roles are represented in the interaction model.

communication can be simply a message to be sent and received or particular protocols

- It contains a set of **protocol definitions**:
  - an institutionalised pattern of interaction, e.g. a Dutch auction.

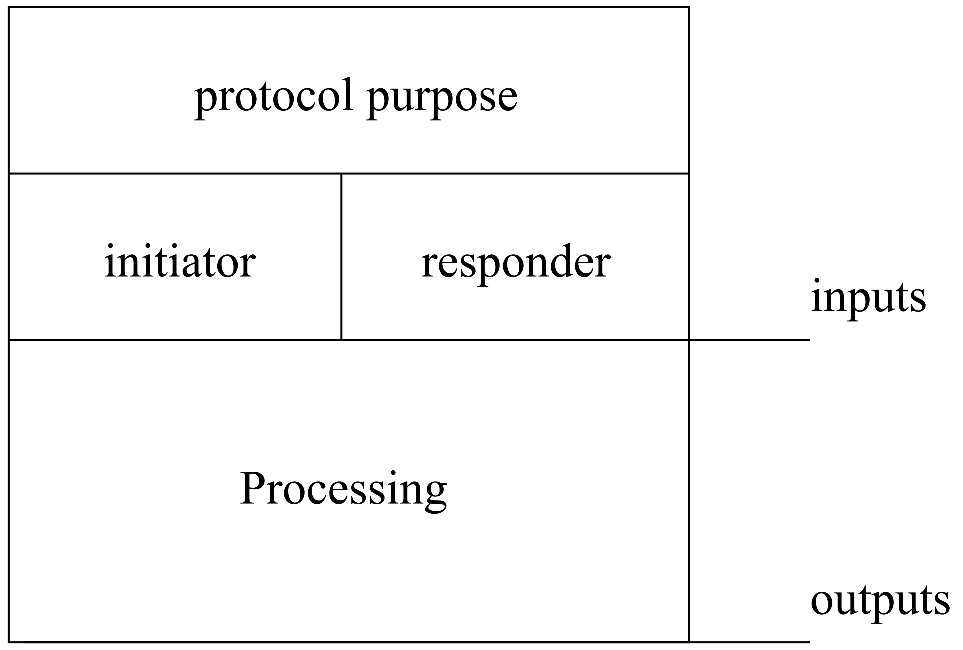
Reference: Wooldridge et. al. 2000.

Gaia Methodology
Protocol attributes

- ***purpose***: description of the interaction
- ***initiator***: the role(s) responsible for starting the interaction;
- ***responder***: the role(s) with which the initiator interacts;
- ***inputs***: information for enacting the protocol;
- ***outputs***: information supplied by/to the protocol responder;
- ***processing***: description of any processing that is performed during the interaction.

Reference: Wooldridge et. al. 2000.

Gaia Methodology
Protocol attributes



Reference: Wooldridge et. al. 2000.

Gaia Methodology

Analysis Process

we have to develop the two models: the role model and the interaction model

- **Objective**: to develop an understanding of the system and its structure.

Steps	**Output**
1. Identify roles	Roles model
2. For each role, identify associated protocols	Interaction model
3. Using the protocol/interaction model as basis, elaborate role model	Fully elaborated roles model (with permissions, responsibilities, etc.)
4. Iterate 1-3	

Reference: Wooldridge et. al. 2000.

Gaia Methodology

Design Process

- **Objective**: to transfer the abstract models from analysis stage into models of sufficiently low abstraction.

Step	**Output**
1. Create agent model	Agent model *(identifies agent types)*
2. Develop service model	Service model *(identifies main services required to realise agent's role)*
3. Develop acquaintance model from interaction model and agent model.	Acquaintance model *(documents the lines of communication between the agents)*

Reference: Wooldridge et. al. 2000.

it can happen that one agent can have multiple role, that's why we introduce types

Gaia Methodology

Design Process

- agent model – used for documenting various
  - *agent types* that will be used in the system under development,
  - *agent instances* that will realize these agent types.

Instance qualifiers

Qualifier	*Meaning*
n	there will be exactly n instances
m .. n	there will be between m and n instances
*	there will be 0 or more instances
+	there will be 1 or more instances

Reference: Wooldridge et. al. 2000.

Gaia Methodology
Design Process (Service model)

- The services model - identifies the *services* associated with each agent role, and specifies the main properties of these services.
- By a service, a *function* of the agent is meant.
- Service properties: *inputs*, *outputs*, *pre-conditions*, and *post-conditions*
- *Pre-* and *post-conditions* represent constraints on services derived from the safety properties of a role.
- The services are usually derived from the list of protocols, activities, responsibilities - the liveness properties of a role.

Reference: Wooldridge et. al. 2000.

Gaia Methodology

Design Process

- Acquaintance models defines the communication links that exist between agent types.

- Represented as a graph

there are links if agents are expected to communicate

Reference: Wooldridge et. al. 2000.

Gaia Methodology
Example

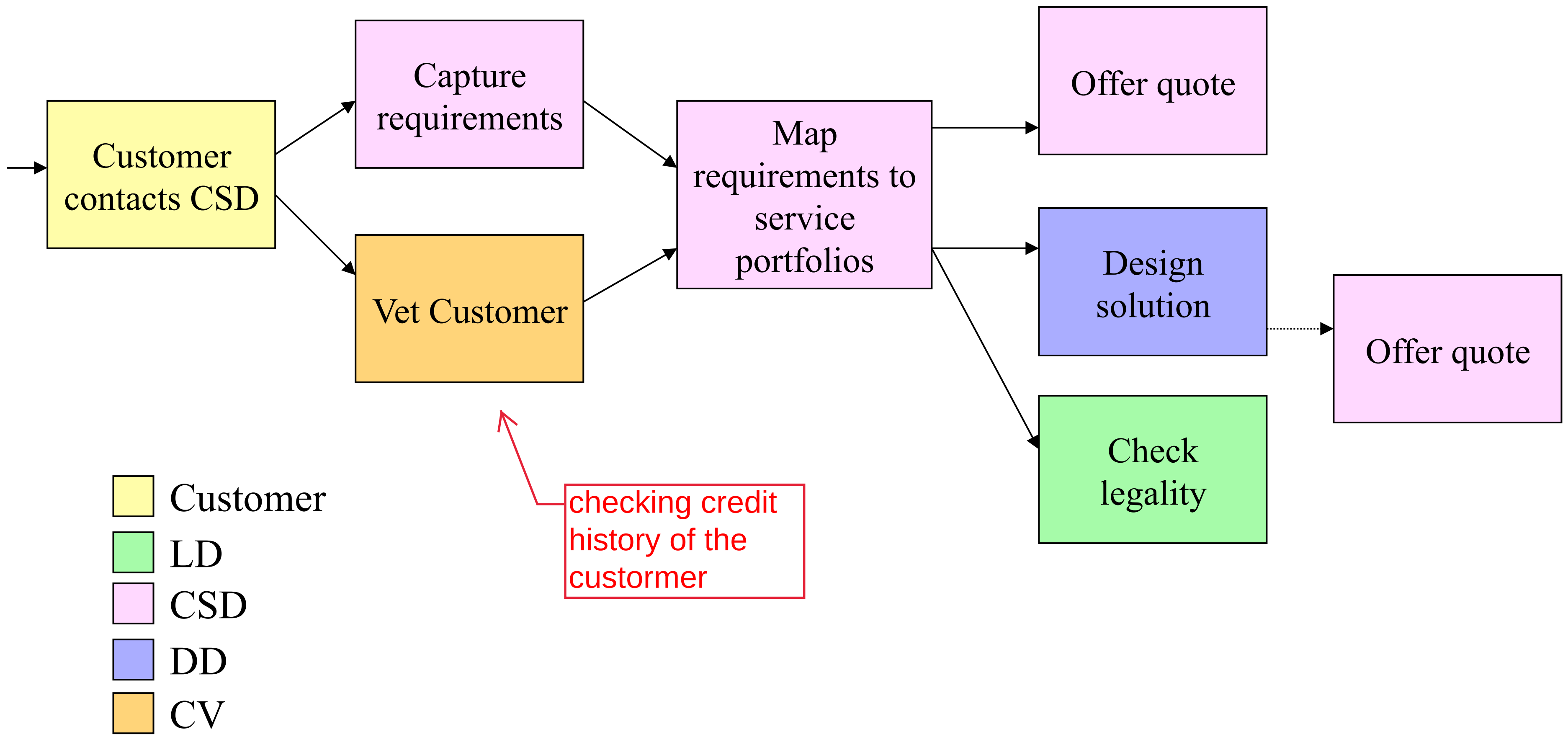
- To provide customers with a quote for installing a network to deliver a particular type of telecommunications service.
- Several departments are involved:
  - Customer Service Dept. (CSD)
  - Design Dept. (DD)
  - Legal Dept. (LD)
  - Customer Vetting (CV)

As a customer, you go to a telecom company to have a service.
As the telecom company, you can propose a standard solution to the customer, if you have it, after checking the credits history of the customer. Other solutions can be proposed to the customer.

Reference: Wooldridge et. al. 2000.

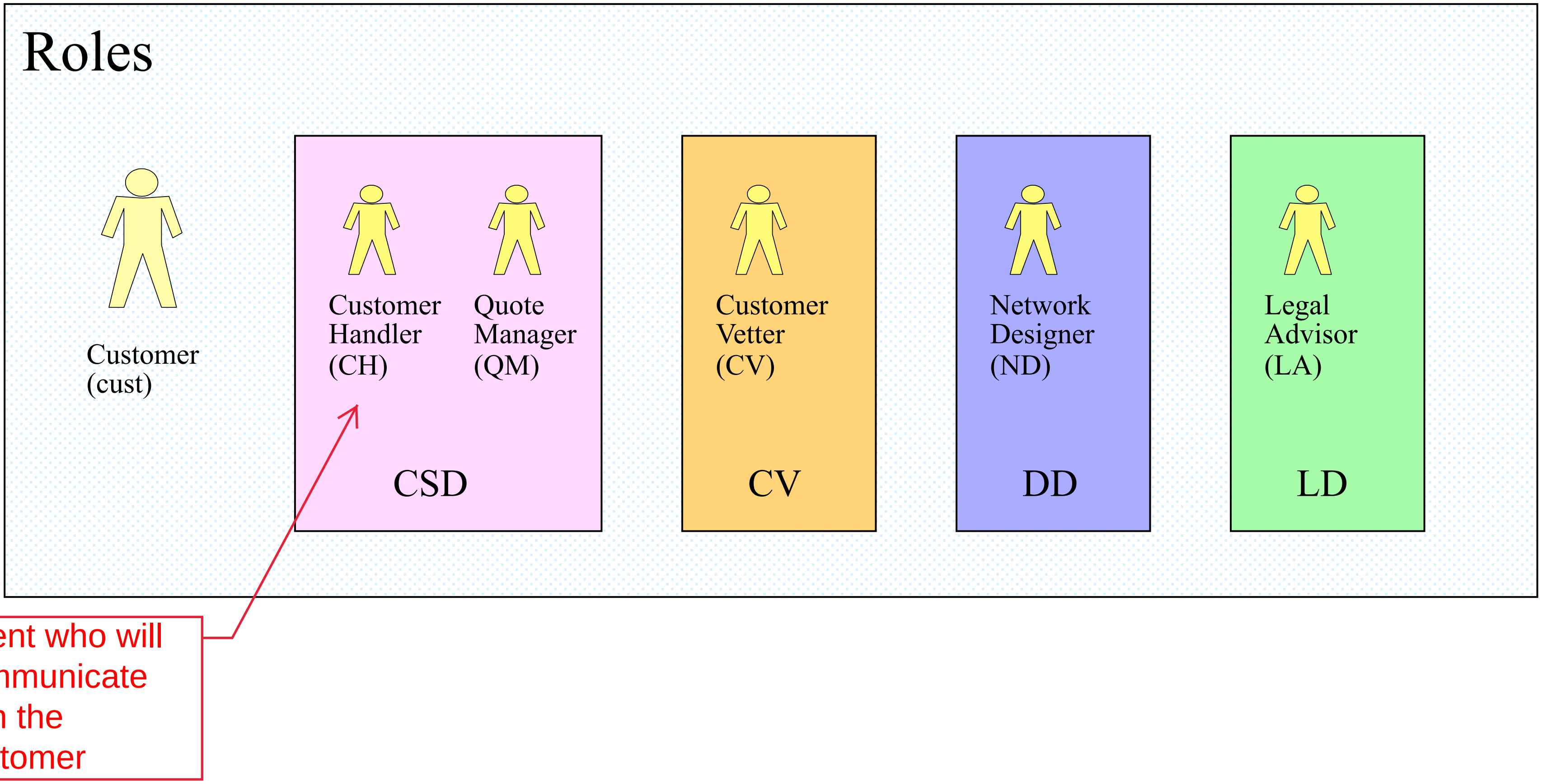
Gaia Methodology

Example



Gaia Methodology
Example

- **Analysis stage**: Identify roles and interaction models



Operators for liveness expressions

Operator	Interpretation
x . y	x followed by y
x | y	x or y occurs
x^*	x occurs 0 or more times
x^ω	x occurs infinitely often
x+	x occurs 1 or more times
[x]	x is optional
x || y	x and y interleaved

Gaia Methodology

Example-role description

Role Schema: `CustomerHandler(CH)`

Description:

```
   Receives quote request from the customer and oversees process
   to ensure appropriate quote is returned.
```

Protocol and Activities:

```
   AwaitCall, ProduceQuote, InformCustomer
```

Permissions:

```
   reads supplied customerDetails // customer contact information
         supplied customerRequirements // what customer wants
         quote                         // completed quote or nil
```

Responsibilities

Liveness:

$$\texttt{CustomerHandler} = (\texttt{AwaitCall}.\ \texttt{GenerateQuote})^{\omega}$$

$$\texttt{GenerateQuote} = (\texttt{ProduceQuote}.\ \texttt{InformCustomer})$$

Safety:

`true` ← no constraints

Reference: Wooldridge et. al. 2000.

Gaia Methodology

Example-role description

Role Schema: QuoteManager (QM)

Description:

```
Responsible for enacting the quote process. Generates a quote
or returns no quote (nil) if customer is inappropriate or
service is illegal.
```

activity (underlined)
The others are protocols

Protocols and Activities:

```
VetCustomer, GetCustomerRequirements, CostStandardService,
CheckServiceLegality, CostBespokeService
```

Permissions:

```
reads supplied customerDetails //customer contact information
      supplied customerRequirements // detailed service
                                     //requirements
      creditRating        // customer's credit rating
      serviceIsLegal      // boolean for bespoke requests
generates quote           // completed quote or nil
```

Responsibilities

Liveness:

```
QuoteManager = QuoteResponse
QuoteResponse = (VetCustomer ∥ GetCustomerRequirements) |
          (VetCustomer ∥ GetCustomerRequirements).CostService
CostService = CostStandardService | (CheckServiceLegality ∥
          CostBespokeService)
```

provide new services

at the same time, check for legality of these new services

Safety:

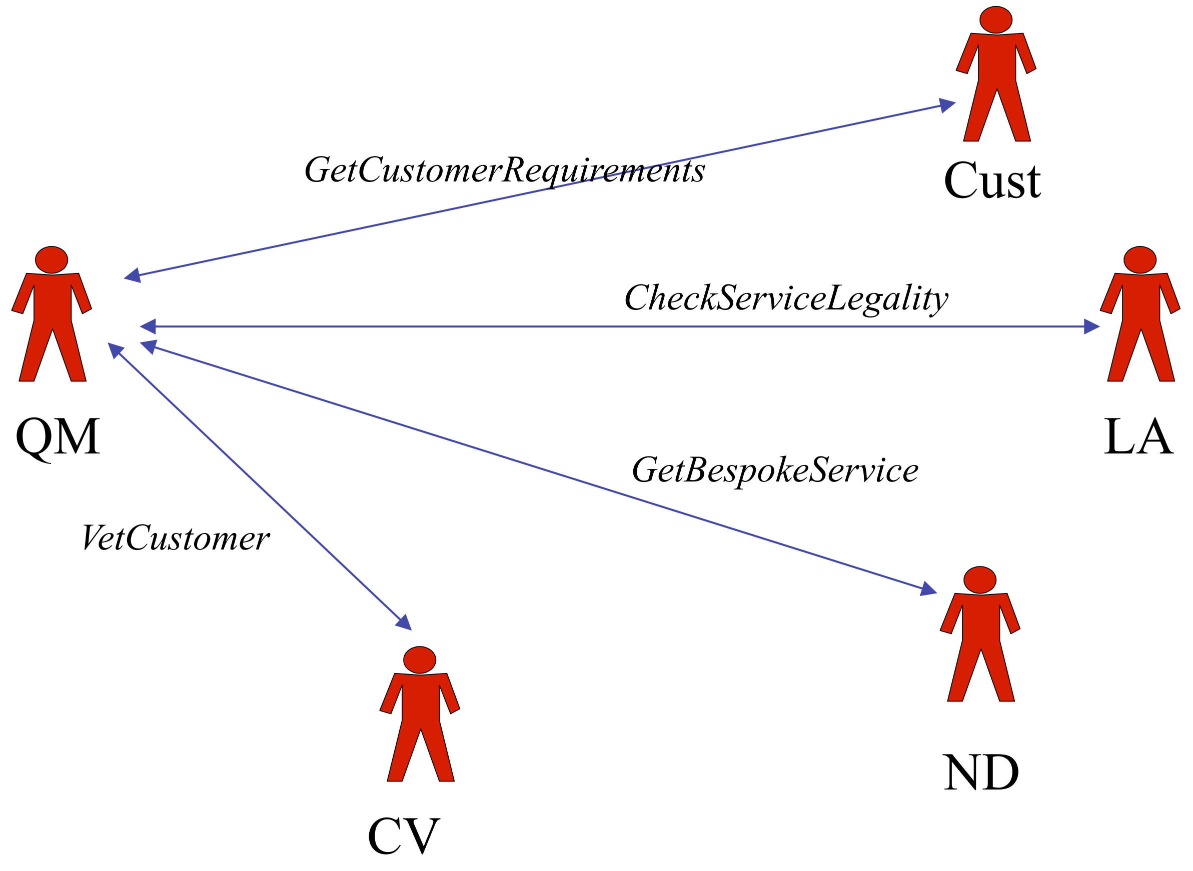
$creditRating = bad \Rightarrow quote = nil$

$serviceIsLegal = false \Rightarrow quote = nil$

ce: Wooldridge et. al. 2000.

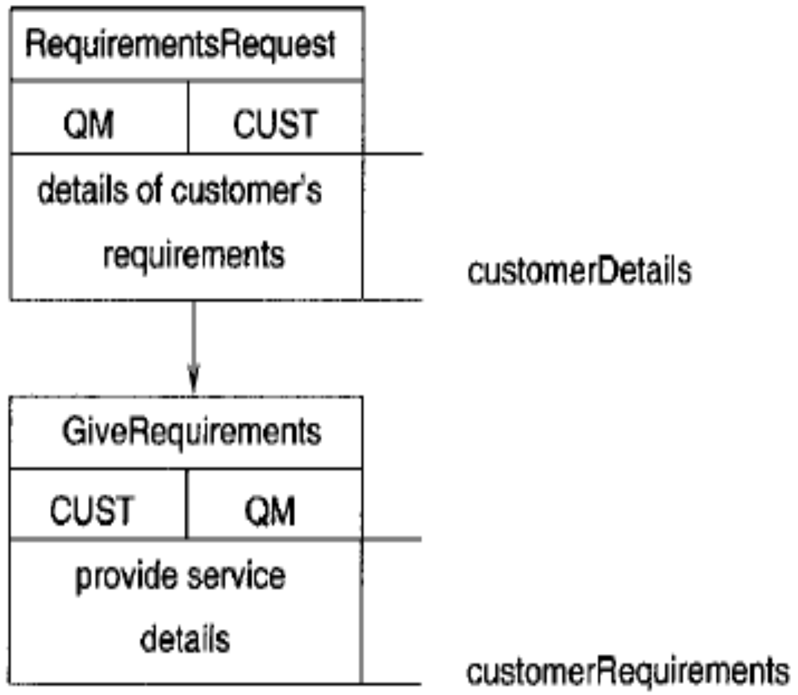
Gaia Methodology
Example

- **Interaction Model**: For role Quote Manager (QM)

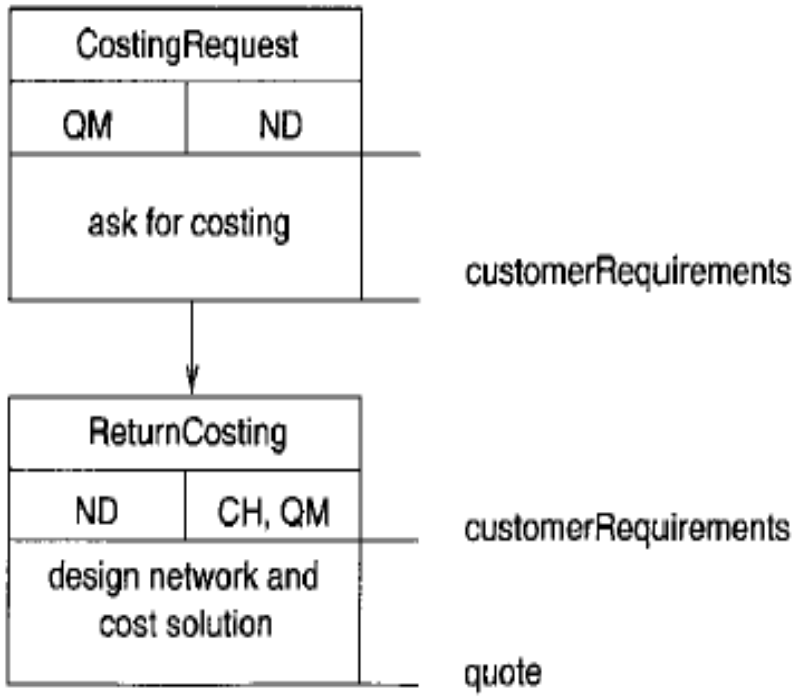


Gaia Methodology
Example-protocol definition

GetCustomerRequirements



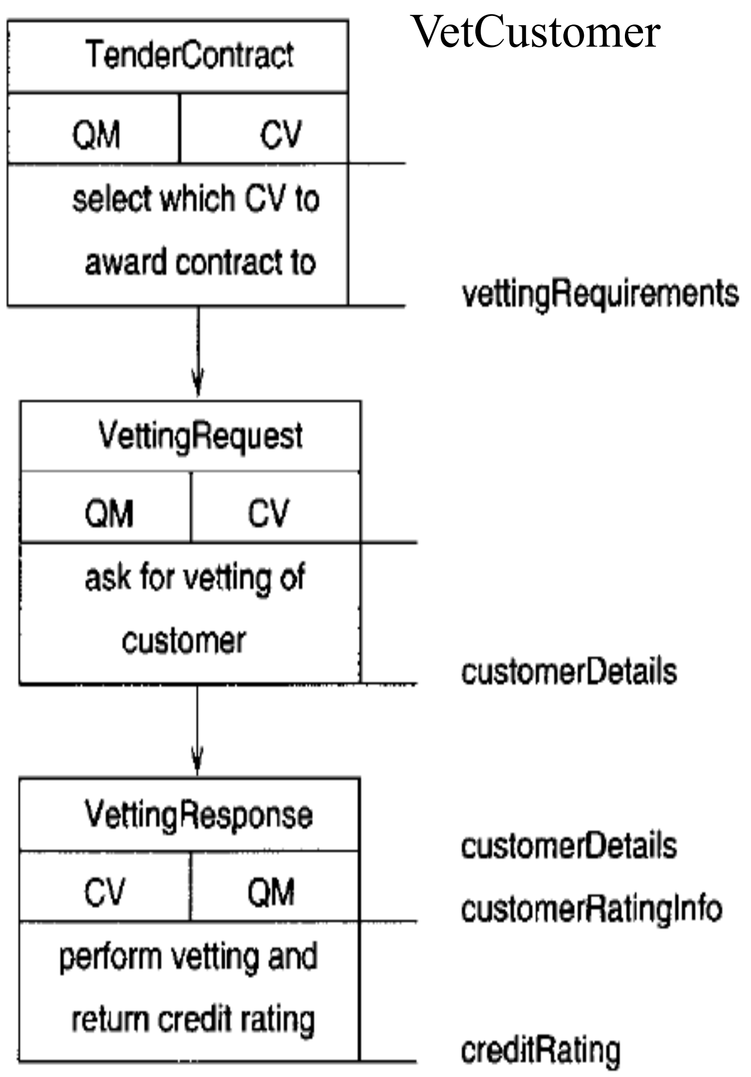
GetBespokeService

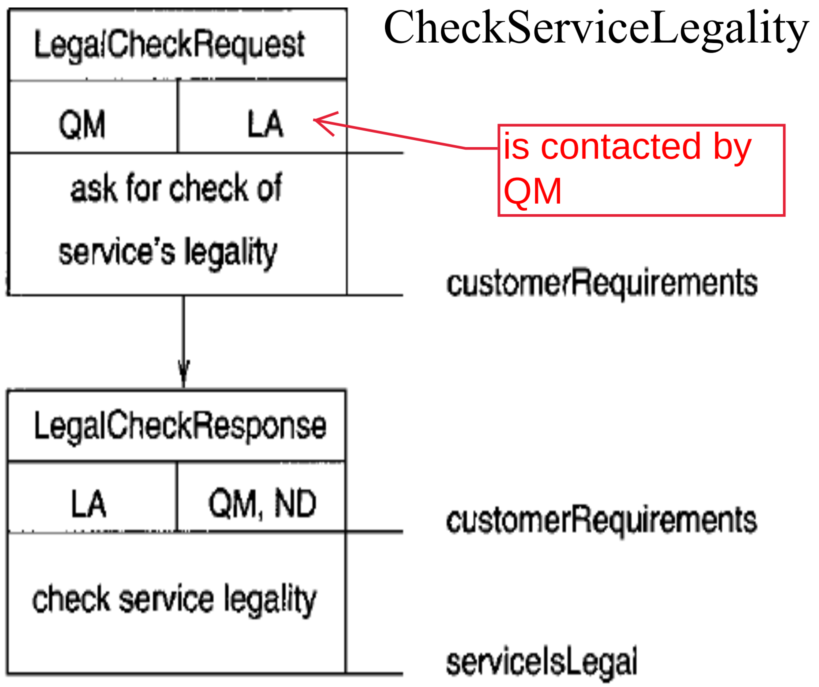


From: Wooldridge et. al. 2000.

Gaia Methodology

Example-protocol definition

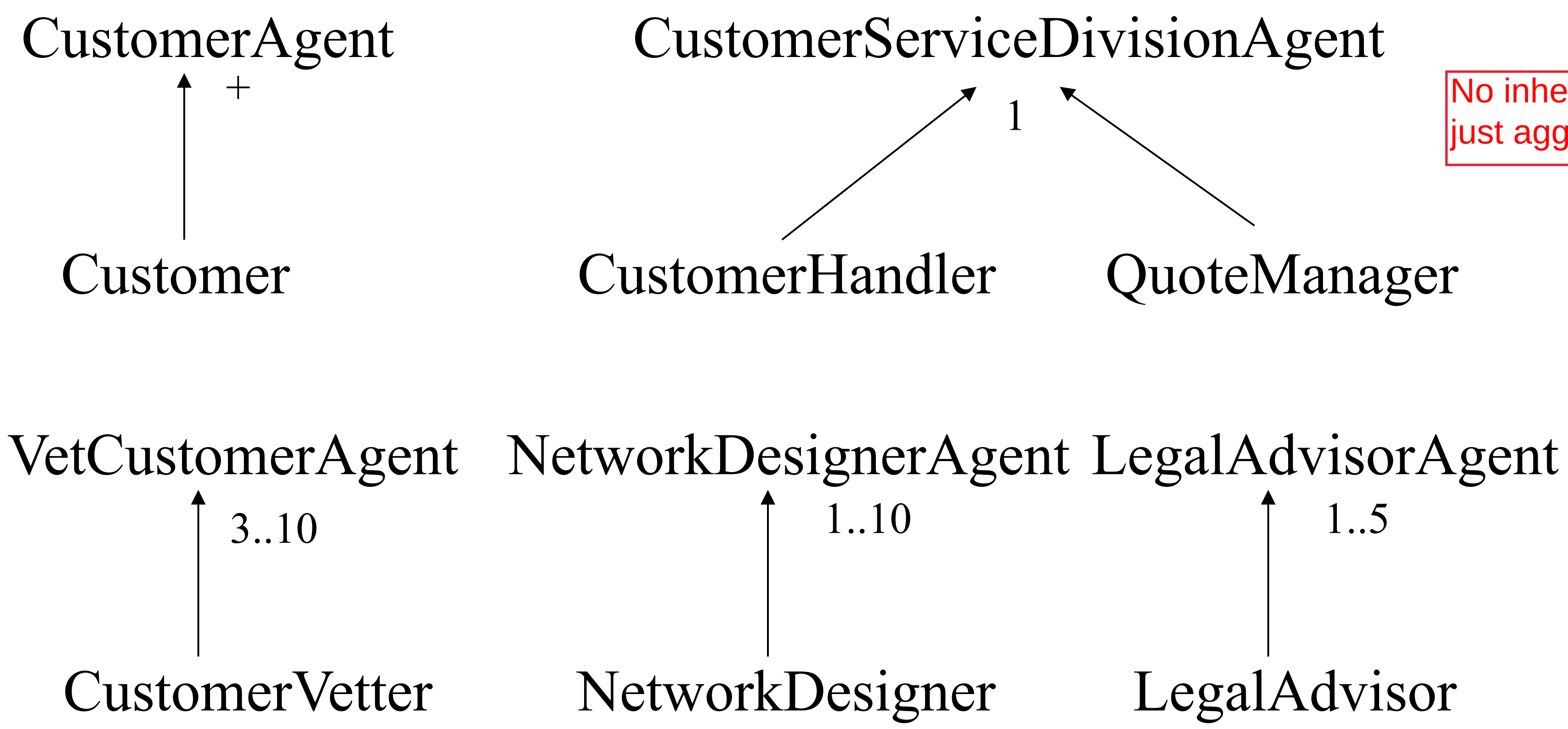




From: Wooldridge et. al. 2000.

Gaia Methodology

Example: agent model



No inheritance,
just aggregation

Reference: Wooldridge et. al. 2000.

Gaia Methodology
Example- service model for QM

Service	Inputs	Outputs	Pre-condition	Post-condition
obtain customer requirements	*customerDetails*	*customerRequirements*	***true***	***true***
vet customer	*customerDetails*	*creditRating*	*customer vetter available*	*creditRating ≠ nil*
check customer satisfactory	*creditRating*	*continuationDecision*	*continuationDecision =nil*	*continuationDecision ≠ nil*
check service type	*customerRequirements*	*serviceType*	*creditRating ≠ bad*	*serviceType ∈ {standard, bespoke}*
produce standard service costing	*serviceType, customerRequirements*	*quote*	*serviceType=standard ^ quote = nil*	*quote ≠ nil*
produce bespoke service costing	*serviceType, customerRequirements*	*quote, serviceIsLegal*	*serviceType = bespoke ^ quote = nil ^ serviceIsLegal*	*(quote ≠ nil) v (quote = nil ^ not serviceIsLegal)*
inform customer	*customerDetails, quote*		***true***	***customers know quote***

From: Wooldridge et. al. 2000.

Gaia Methodology
Example- acquaintance model

who communicates with whom?

$$\begin{tikzcd}
& \text{CustomerAgent} \arrow[d, leftrightarrow] & \\
& \text{CustomerServiceDevisionAgent} \arrow[dl, leftrightarrow] \arrow[d, leftrightarrow] \arrow[dr, leftrightarrow] & \\
\text{VetCustomerAgent} & \text{NetworkDesignAgent} \arrow[r, leftrightarrow] & \text{LegalAdvisorAgent}
\end{tikzcd}$$

Gaia Methodology

Shortcomings of Gaia

- Organisation structure is static – cannot be changed at run-time.
- Abilities of agents and services they provide are also static.
- Suitable for a small number of agent types.
- Protocols do not contain sufficient details

Agent UML

Rationale

Notation:how you present your design

- Agent UML is an extension of UML.
- It is not a methodology, rather a language for documenting models of systems.
- **Rationale for agent UML**: UML is insufficient for modeling agents for the following reasons:
  - Compared to objects, agents are active.
  - Agents do not only act in isolation, but in cooperation and coordination with other agents.

Extensions

- interaction protocols,
- agent roles,
- multithreaded lifelines,
- extended UML message semantics,
- nested and interleaved protocols,
- protocol templates,
- extended class diagrams
- extended statechart diagrams
- …

Agent UML
Proposed Modifications

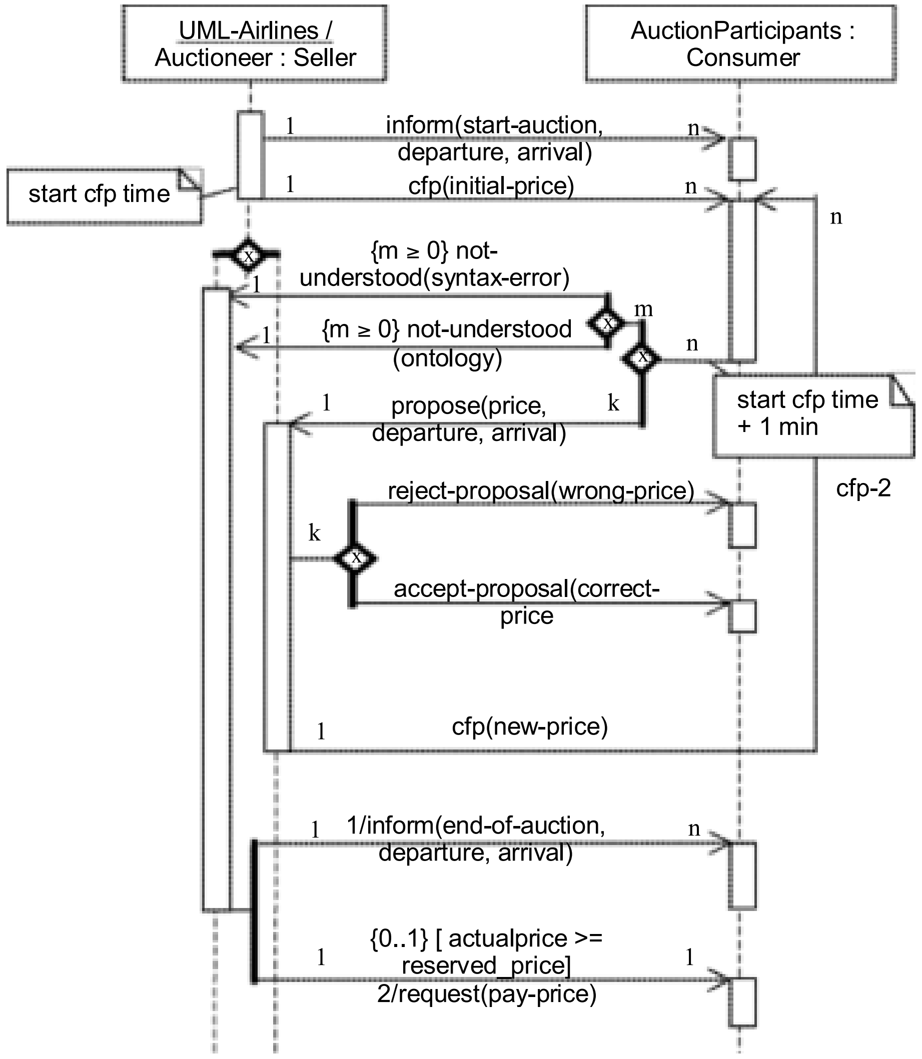
- Support for expressing **concurrent threads of interaction**, enabling UML to model agent protocols such as the CNP.
- A notion of "**role**" to allow an agent playing several roles.

Agent UML
Agent Interaction Protocols

- An **agent interaction protocol** describes
  - A communication pattern with
    - Allowed sequences of messages between agents having different roles
    - Constraints on the contents of messages
  - A semantic that is consistent with the communicative act within a communication pattern

➢ Thus, the proposed modifications are aimed at supporting interaction protocols.

A protocol diagram – an example



UML-Airlines / Auctioneer : Seller
AuctionParticipants : Consumer
1
inform(start-auction, departure, arrival)
n
start cfp time
1
cfp(initial-price)
n
n
{m ≥ 0} not-understood(syntax-error)
1
m
{m ≥ 0} not-understood (ontology)
1
n
start cfp time + 1 min
1
propose(price, departure, arrival)
k
reject-proposal(wrong-price)
k
accept-proposal(correct-price
cfp-2
1
cfp(new-price)
1
1/inform(end-of-auction, departure, arrival)
n
{0..1} [actualprice >= reserved_price] 2/request(pay-price)
1
1

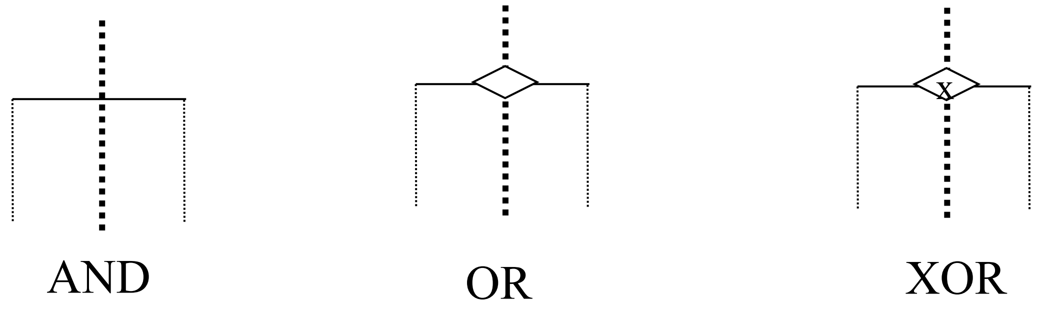
Sequential diagram.
Difference with standard UML:
- concurrent threads
- XOR operations
- roles at he beginning with instances
- conditions
- communicative acts are stated
- order of operations

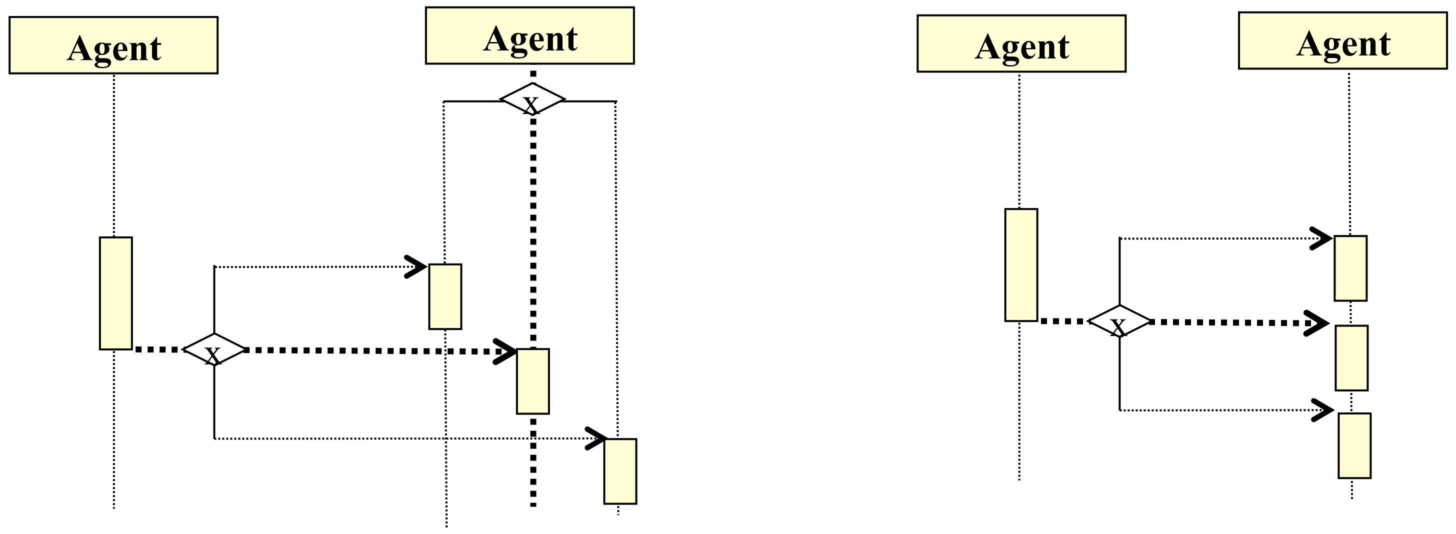
Reference: Bauer, B., Muller, J. P. and Odell, J.

Elements of protocol diagram

<u>**`instance-1 / role-1 ... role-m : class`**</u>

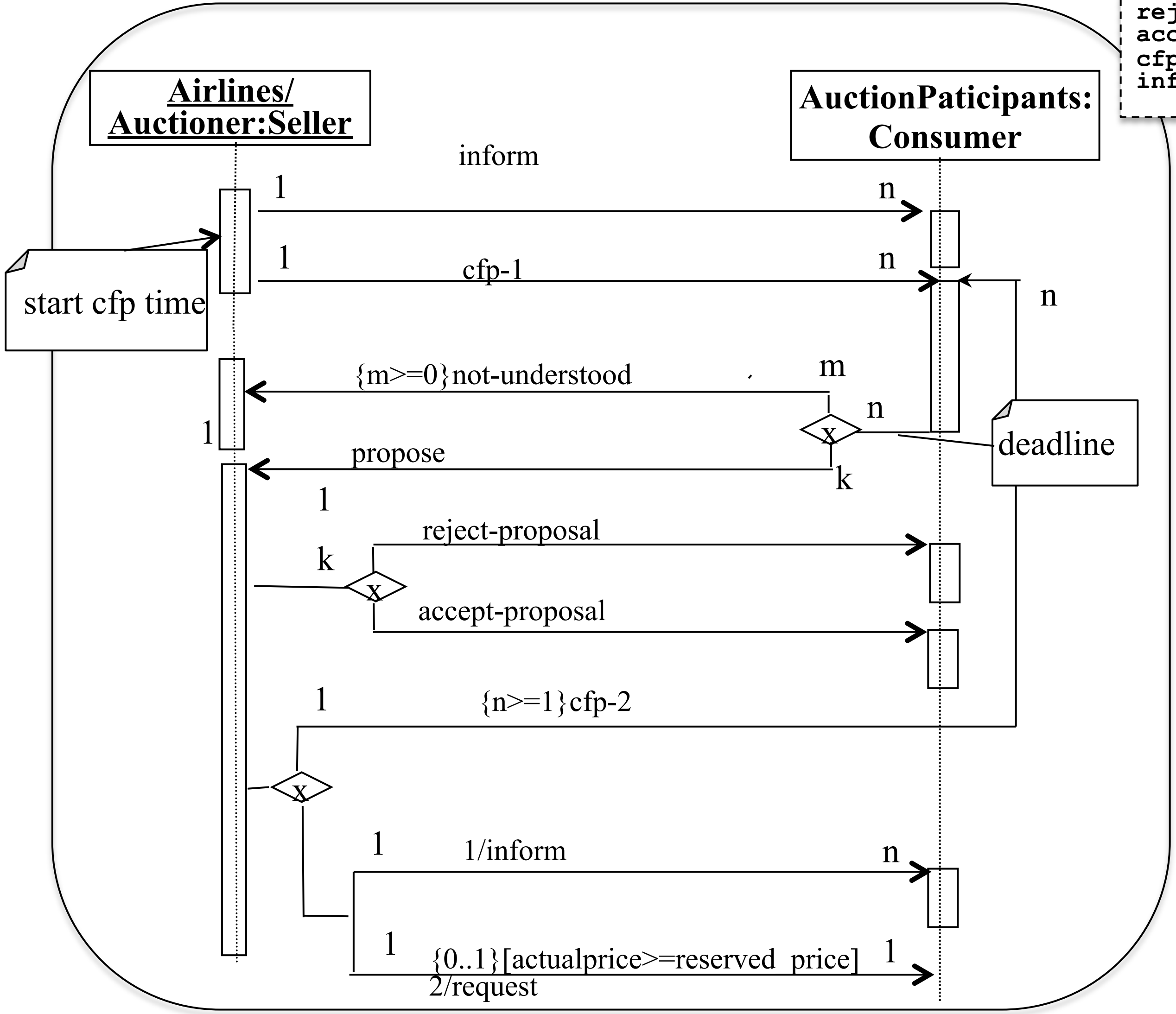
we introduce ROLES in addition to instances and classes





Agent UML (a generic AIP template)

```
Auctioner, Participant
inform-start-of-auction:inform
cfp-1:cfp,
not-understood*:not-understood,
propose*:propose,
reject-proposal*:reject-proposal,
accept-proposal*:accept-proposal,
cfp-2:cfp,
inform*:inform, request*:request
```



We can describe protocols as templates
--> we can parametrize all elements of the protocol

not-understood* : not-understood

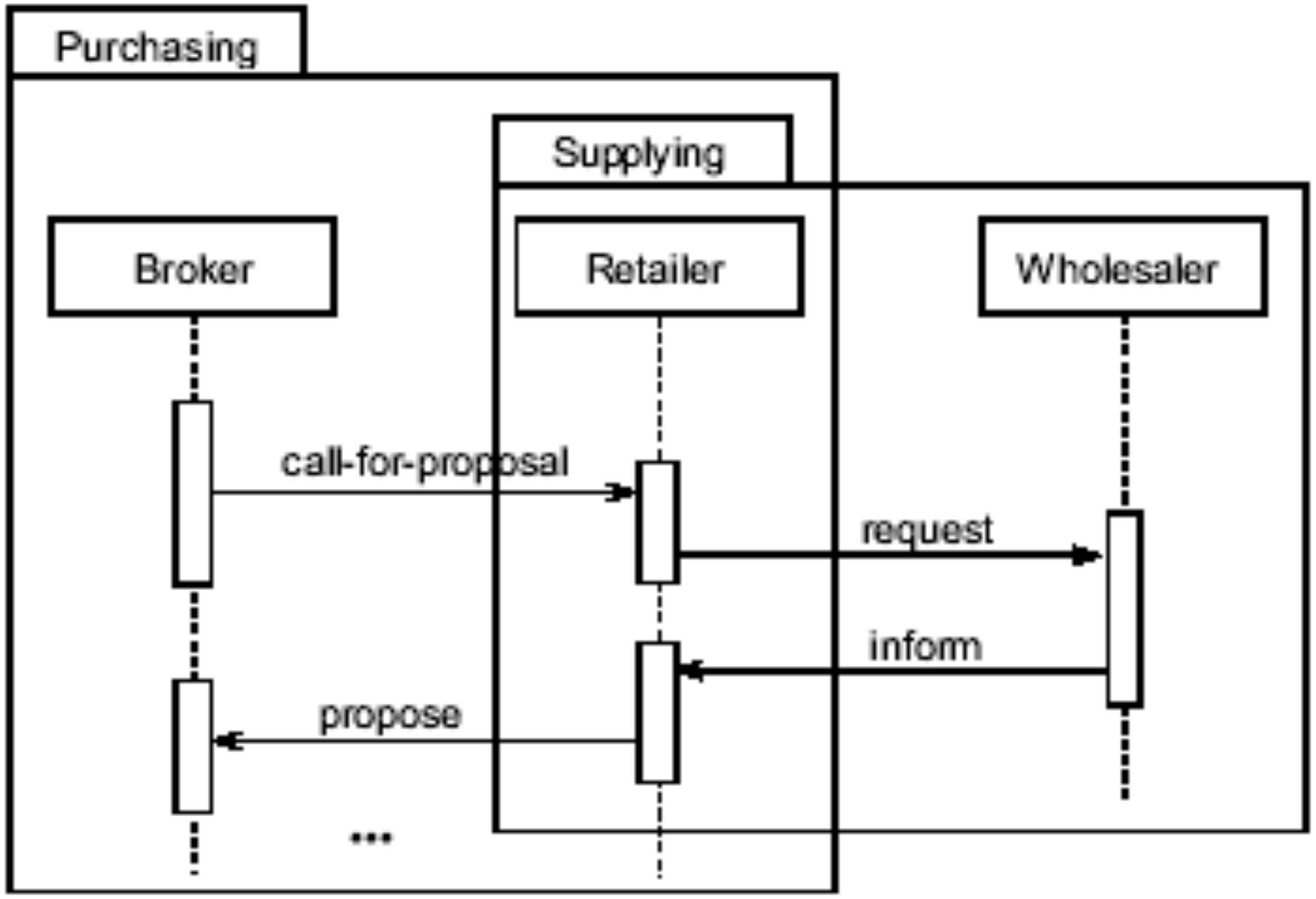
means that it can be different types of "not-understood"

Adopted from: Bauer, B., Muller, J. P. and Odell, J.

Protocol instantiation

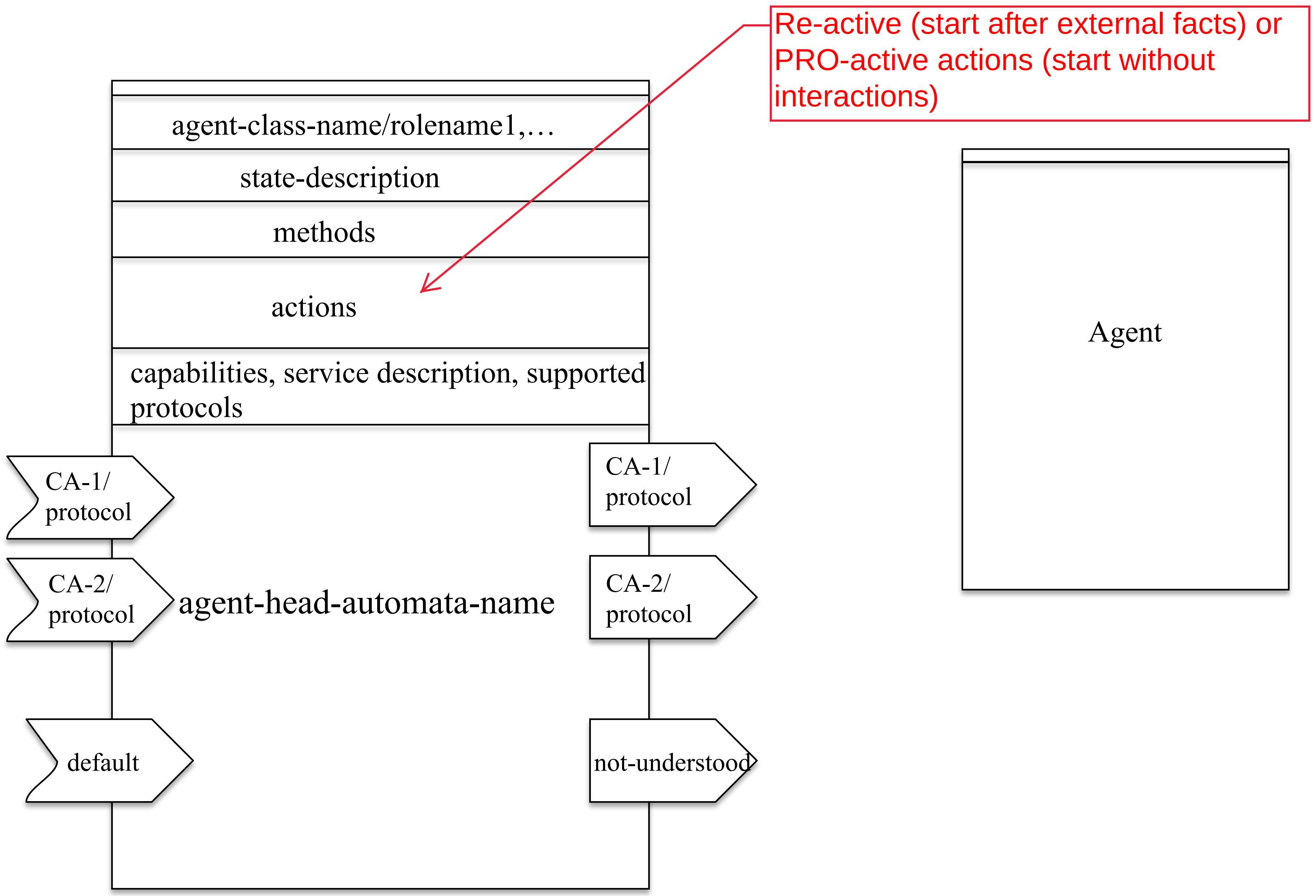
```
FIPA-English-Auction-Protocol <
  UML-Airlines / Auctioneer : Seller,
     AuctionParticipants : Consumer
  inform(start-auction, departure, arrival),
  cfp(initial-price),
  not-understood(syntax-error),
  not-understood(ontology),
  propose(pay-price),
  reject-proposal(wrong-price),
  accept-proposal(correct-price),
  cfp(increased-price),
  inform(end-of-action),
  request(pay-price, fetch-ticket)
>
```

AgentUML (Odel et al) representing interleaved protocols



Reference: Bauer, B., Muller, J. P. and Odell, J.

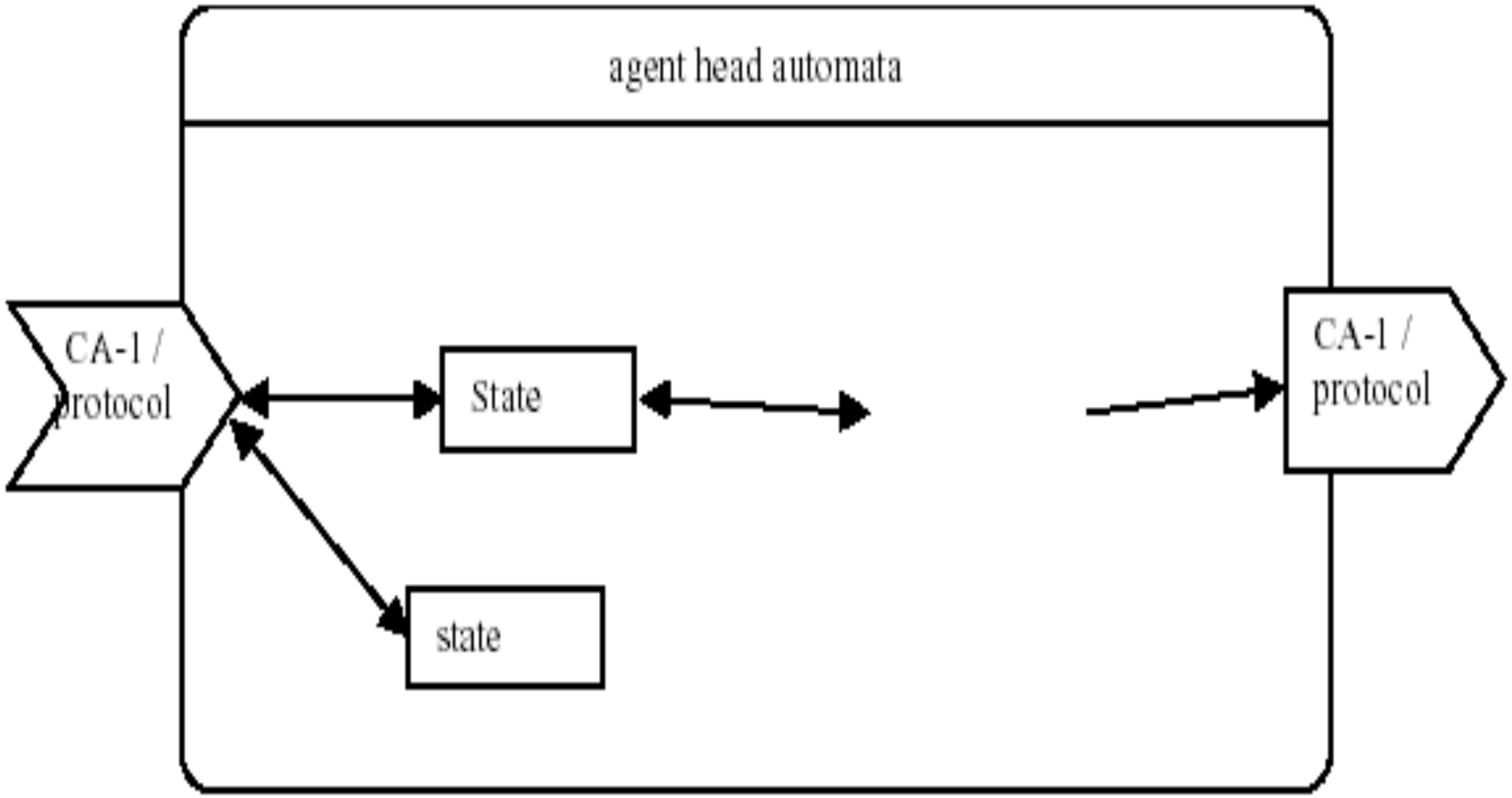
Agent Class Diagrams



Re-active (start after external facts) or PRO-active actions (start without interactions)
agent-class-name/rolename1,…
state-description
methods
actions
capabilities, service description, supported protocols
CA-1/ protocol
CA-2/ protocol
agent-head-automata-name
CA-1/ protocol
CA-2/ protocol
default
not-understood
Agent

Adopted from: Bauer, B.

Agent-head-automata



Adopted from: Bauer, B.

RoMAS: A ROLE-BASED MODELING METHOD FOR MULTI-AGENT SYSTEM

how to find roles?

- The main development process is as follows:

  (1) Capture use cases;

  (2) Identify roles from use cases;

  (3) Construct role organization;

  (4) For each role, if the appropriate agent does not exist, then go to (5); else

    I. Bind roles to agents

    II. Describe dynamic properties of bind relation between agents and roles

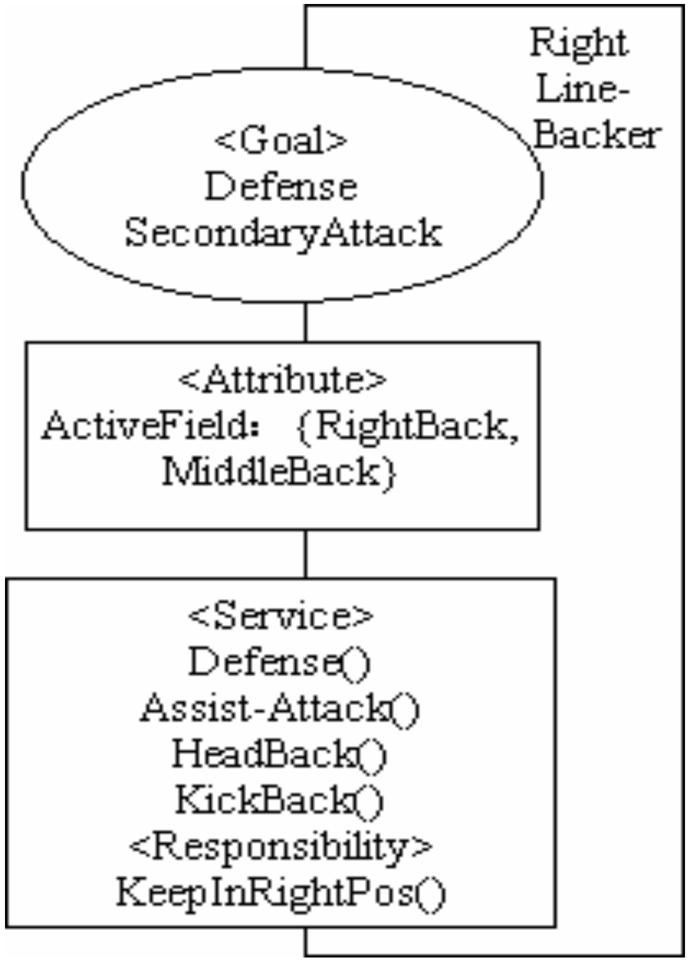
    III. Go to (6)

  (5) Generate agents according to roles; Go to(4).I.

  (6) Generate codes for agents with roles bound;

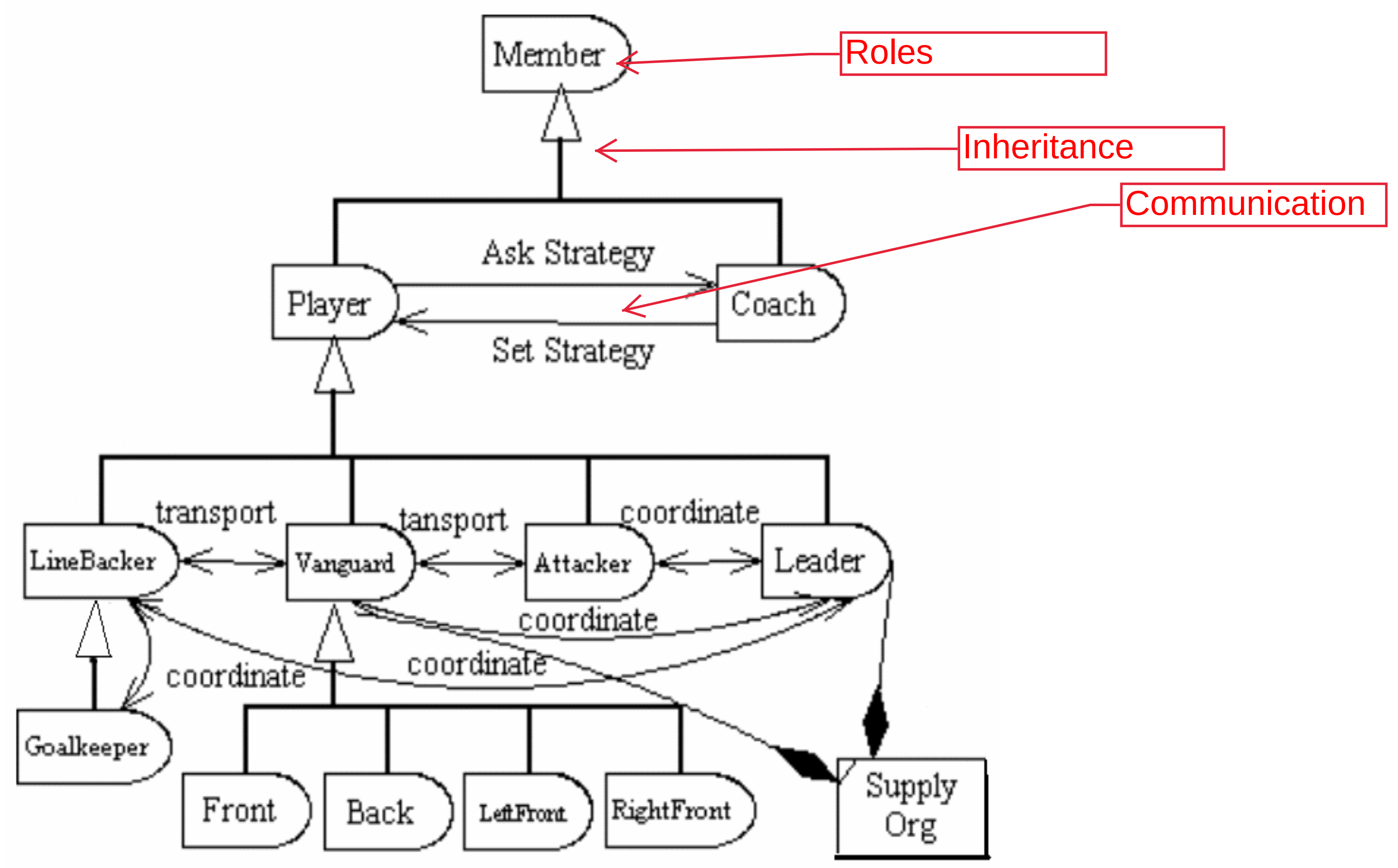
Adopted from: Yan et al.

RoMAS: A ROLE-BASED MODELING METHOD FOR MULTI-AGENT SYSTEM (identify roles)



Adopted from: Yan et al.

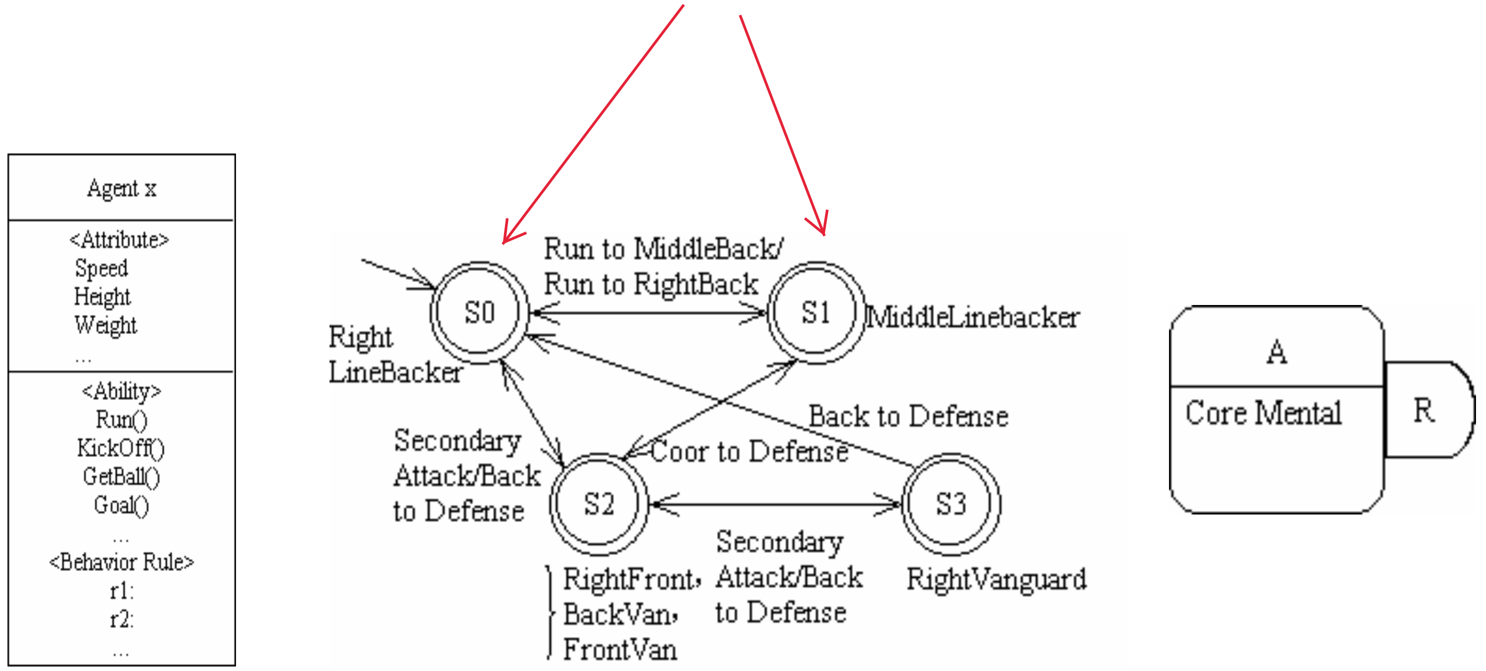
RoMAS: A ROLE-BASED MODELING METHOD FOR MULTI-AGENT SYSTEM (construct role organizations)



Adopted from: Yan et al.

RoMAS: A ROLE-BASED MODELING METHOD FOR MULTI-AGENT SYSTEM (bind roles to agents)

Agents may change role during the execution



Adopted from: Yan et al.

Formal Methods in AOSE

What role is played by logic in the development of agent systems? It can be used for:

- Specification of a system
- Directly (automatically) programming a system
  - Once specified, the system must be implemented. 3 Possibilities:
    1. Manual refinement of the specifications.
    2. Direct execution of the specification.
    3. Compilation, transform specifications into code using some automatic synthesis process.
- Verification of a system
  - Once the system is developed, we need to show that the system is correct with respect to the specification.

Pitfalls of Agent Development 1

- “While agent-based systems are becoming increasingly well understood, multi-agent system development is not.” *(Wooldridge & Jennings, 1999)*
- Little effort has been devoted to understanding the pragmatics of multi-agent systems development.

Pitfalls of Agent Development

Political	You can oversell agents or fail to see where agents may usefully be applied.
Management	You don’t know why you want agents or what your agents are good for. E.g. Starting an agent project without any clear goals.
Conceptual	You believe that agents are a silver bullet.
Analysis and design	While developing an agent system, the % of the system that is agent-specific (e.g. Negotiation or coordination) is very small.
Macro (society) level	You see agents everywhere, too many agents.
Micro (agent) level	You decide you want your own architecture , you use too much AI.
Implementation	You ignore *de facto* standards (e.g. FIPA in agent communication)

Summary

- AOSE methodologies are close to existing Software Engineering methodologies (e.g. OO) methodologies.
- AOSE methodologies are inspired by either OO techniques or knowledge engineering techniques.
- Main difference is **focus on interaction and behavior**

Next Lecture:

Agent Theory

- Wooldridge: ”Introduction to MAS”,
  - Chapter 17