Intelligent Systems

– Agent and Multiagent Technology –

Part 4

Gerhard Weiss

DKE, Maastricht University

Outline

Outline

Standard Classification Schema

Kriterium	Optionen	
Adressierung	direkt	indirekt
Blockierung	synchron	asynchron
Pufferung	ungepuffert	gepuffert, Mailbox
Kommunikationsform	meldungsorientiert	auftragsorientiert

(table from (Weber 1998))

Addressing

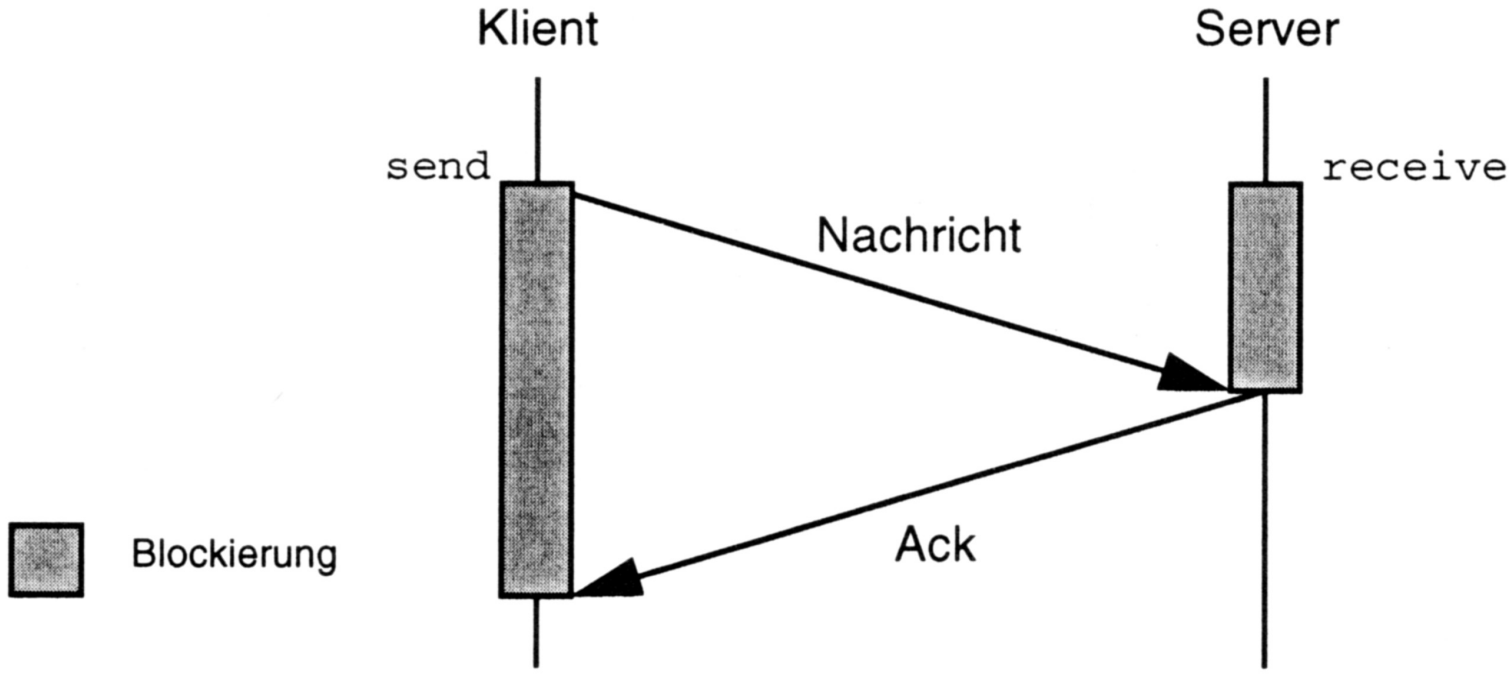
- **Direct addressing**:
  - sender and receiver communicate directly (point-to-point)
  - *symmetric*: both sender and receiver name each other

    `Q:send(P,message) ; P:receive(Q,message)`
  - *asymmetric*: sender (client) names server, receiver (client) gets message only

    `Q:send(P,message) ; P:receive(message)`
- **Indirect addressing**:
  - mailbox
  - ports (special I/O points, often provided by operating system)

Blocking

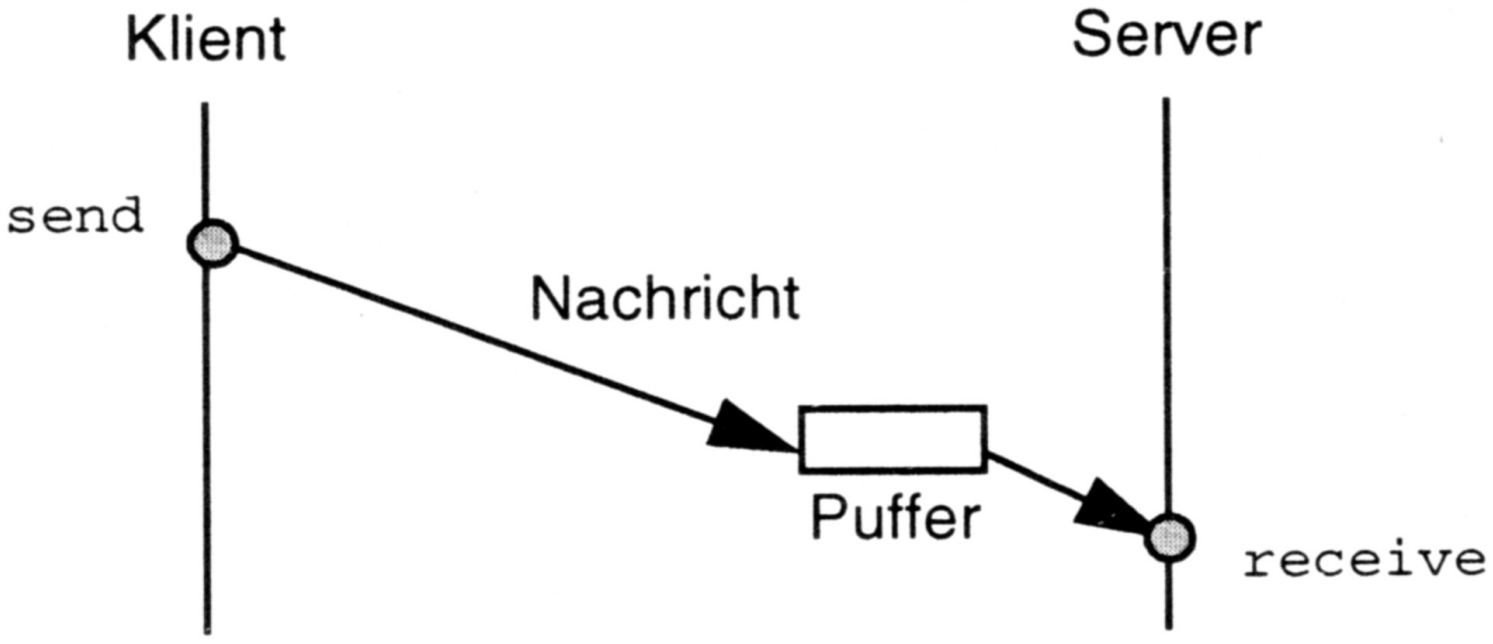
- **synchronous communication**:



(figure from (Weber 1998))

Blocking (Cont'd)

- **asynchronous communication**:



(figure from (Weber 1998))

Buffering

- **non-buffered communication**

  `receive`-command provides memory space (data structure), operating system writes incoming message into this space

- **buffered communication**

  if a receiver is not able to take messages, the operating system kernel saves these messages

Message- vs Task-Oriented

- Message-oriented communication

  sender posts message and expects (i) no reply or (ii) an acknowledgement of receipt

- task-oriented communication

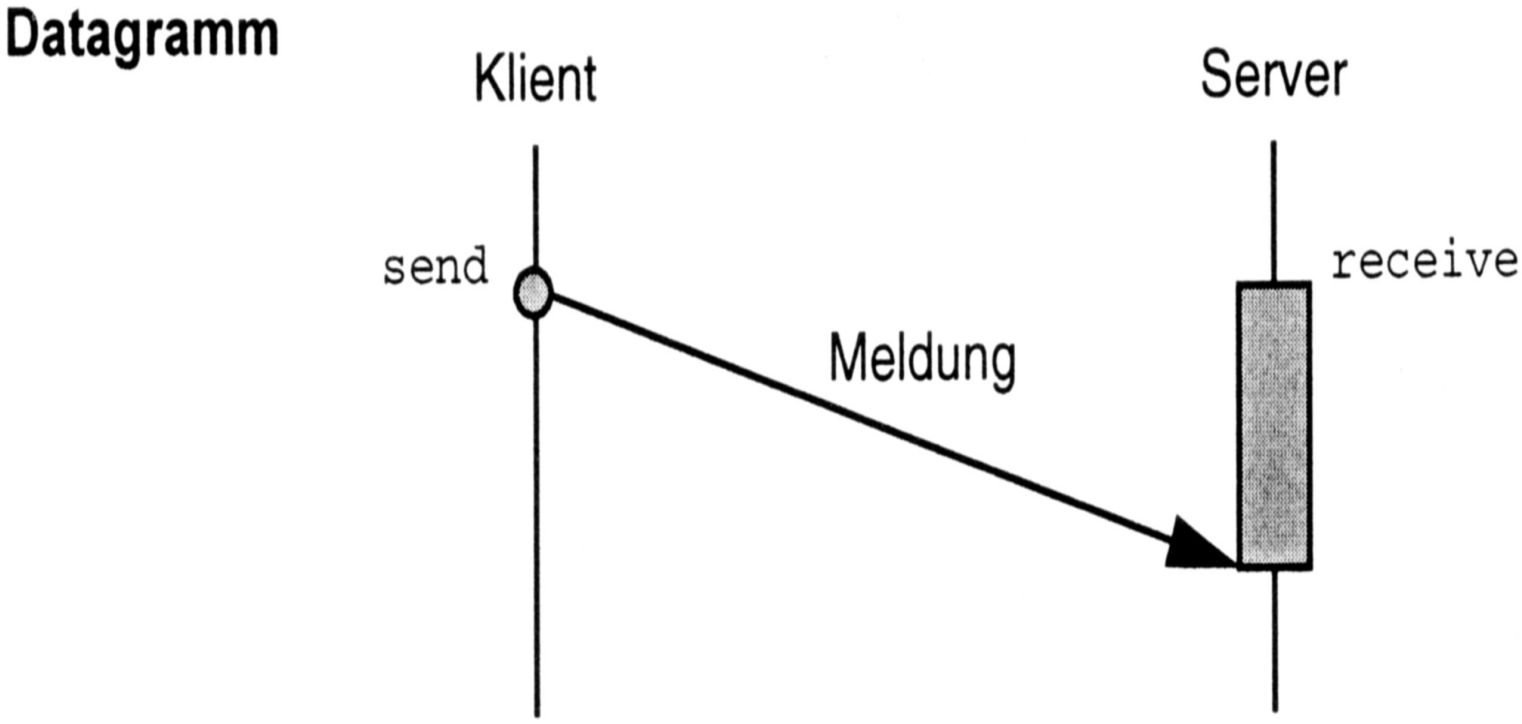
  sender posts task specification, receiver replies with result of task execution

- both forms may occur in asynchronous and synchronous mode:

	asynchron	**synchron**
meldungsorientiert	Datagramm	Rendezvous
auftragsorientiert	asynchroner entfernter Dienstaufruf	synchroner entfernter Dienstaufruf

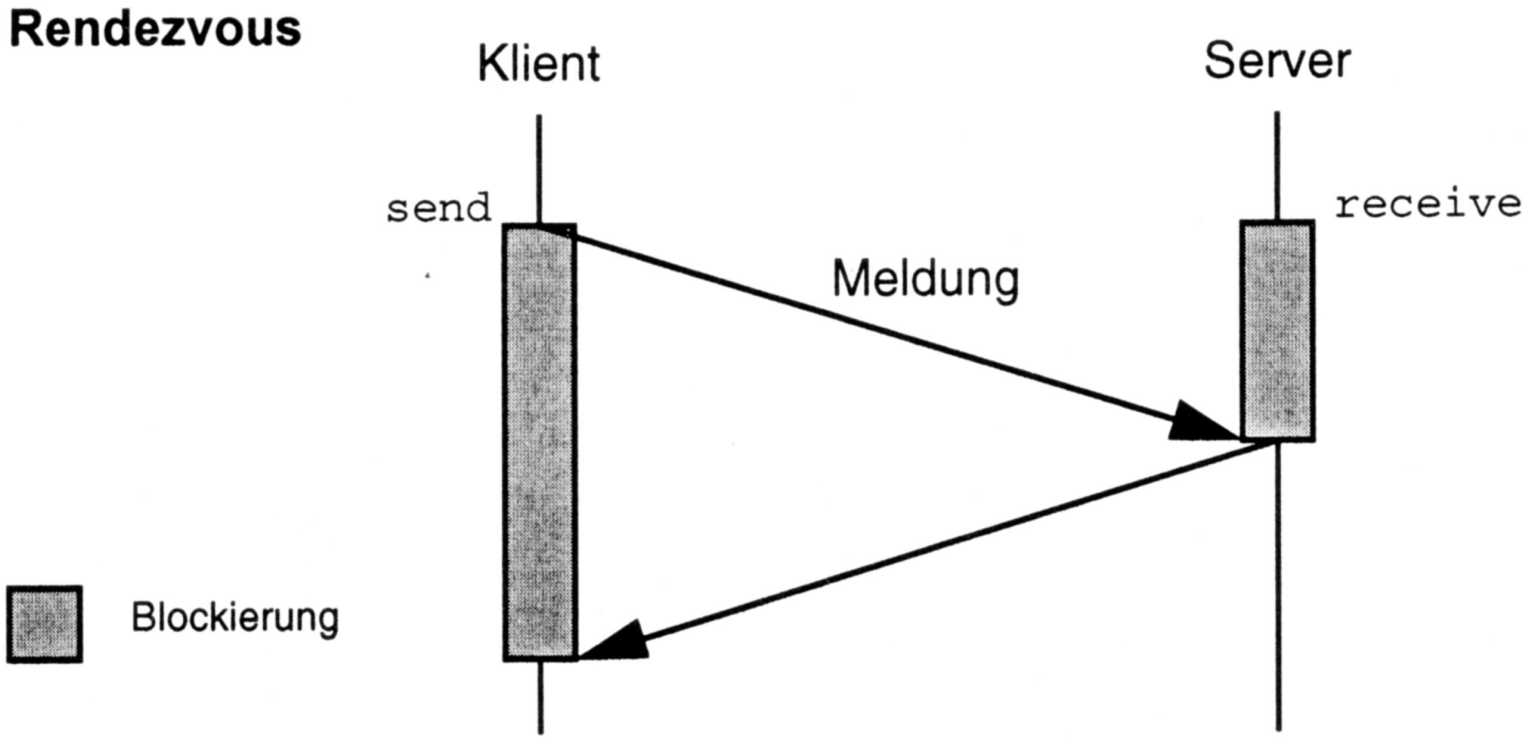
(figure from (Weber 1998))

Message- vs Task-Oriented (Cont'd)



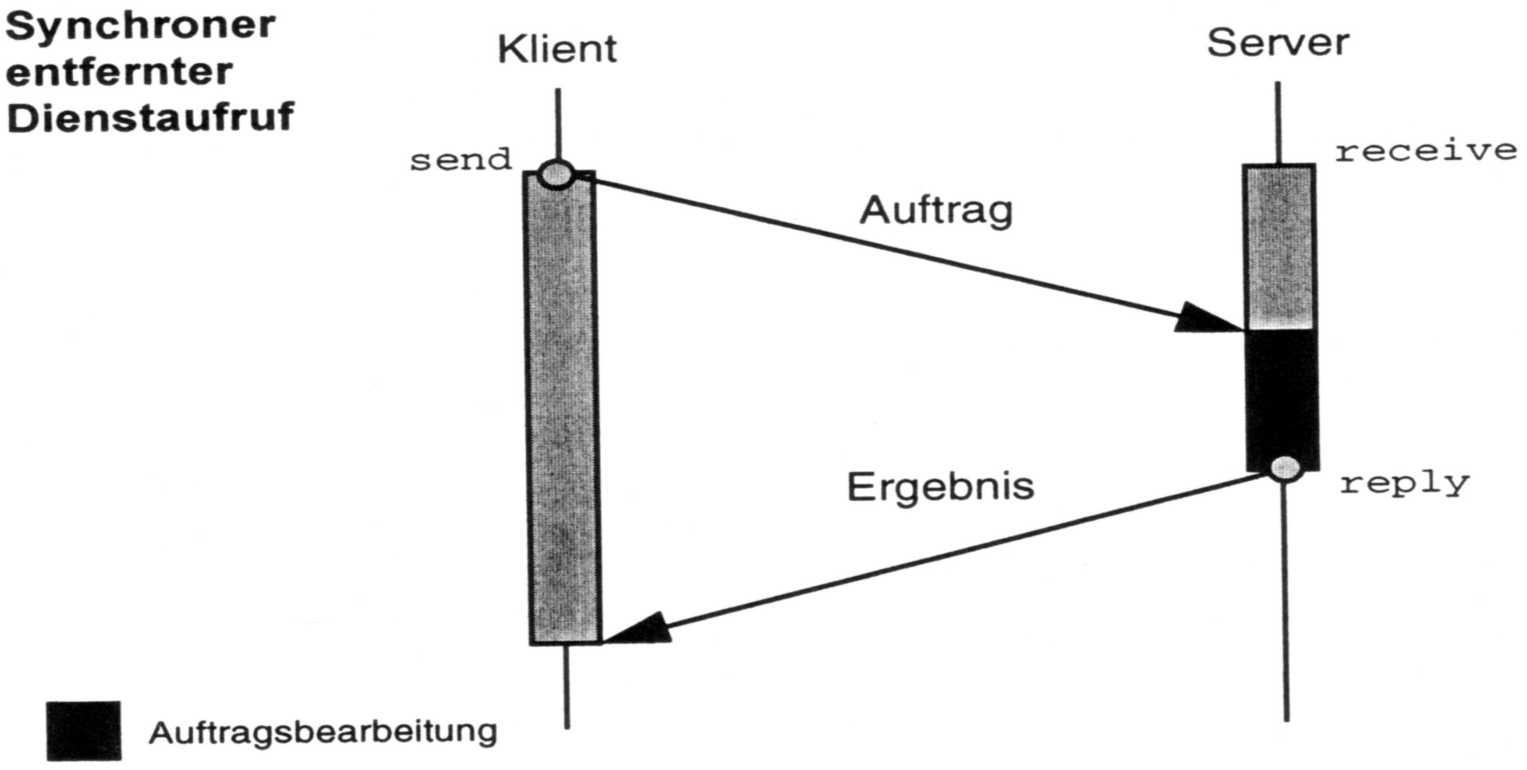
(figure from (Weber 1998))

Message- vs Task-Oriented (Cont'd)



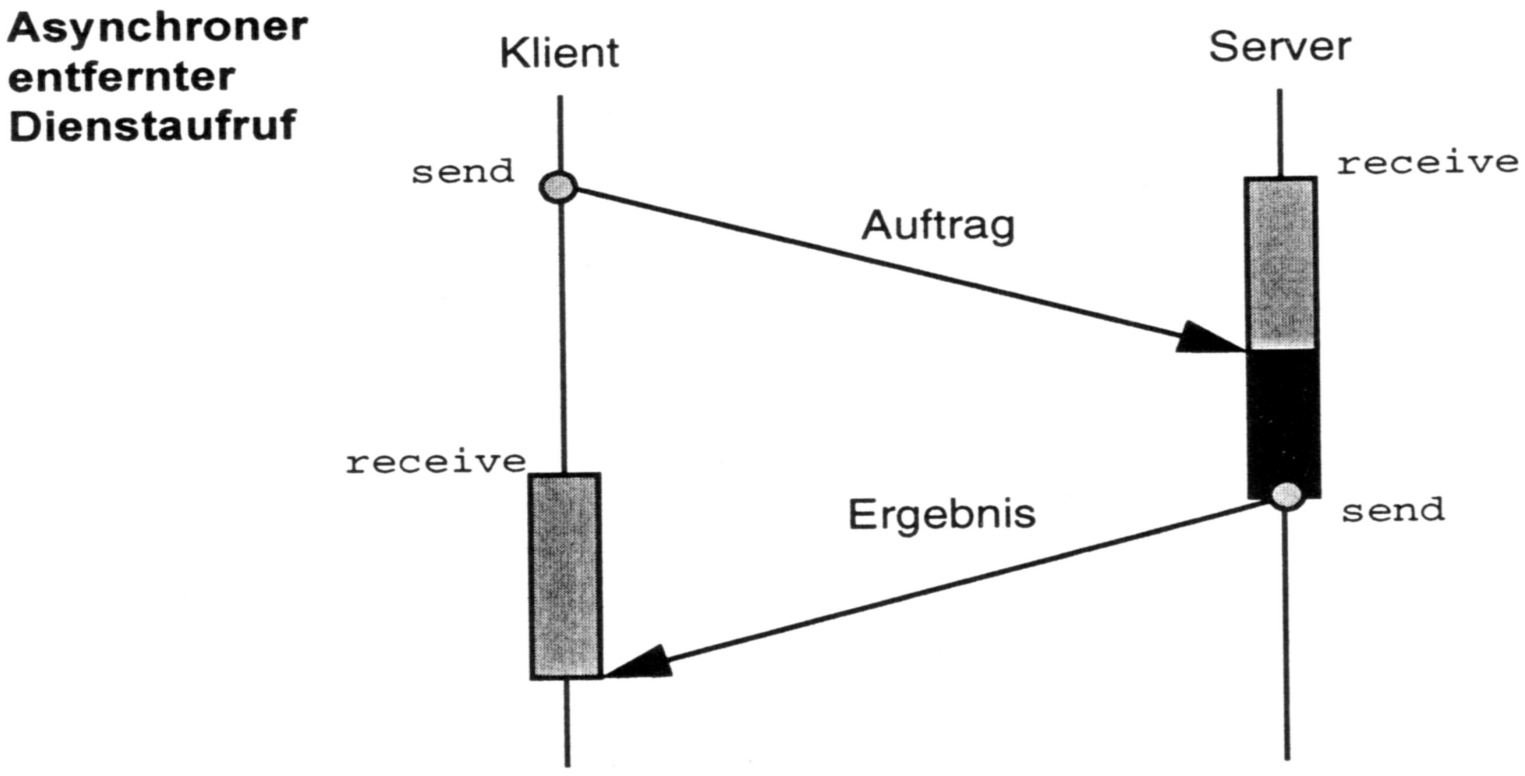
(figure from (Weber 1998))

Message- vs Task-Oriented (Cont'd)



(figure from (Weber 1998))

Message- vs Task-Oriented (Cont'd)



(figure from (Weber 1998))

Message- vs Task-Oriented (Cont'd)

- A note on terminology:
  - synchroner entfernter Dienstaufruf = Remote-Procedure-Call (RPC)
  - asynchroner entfernter Dienstaufruf = Remote-Service-Invocation (RSI)
  - Datagramm = no-wait-send

  - RSI, in contrast to RPC: sender explicitly awaits result by means of `receive`

Outline

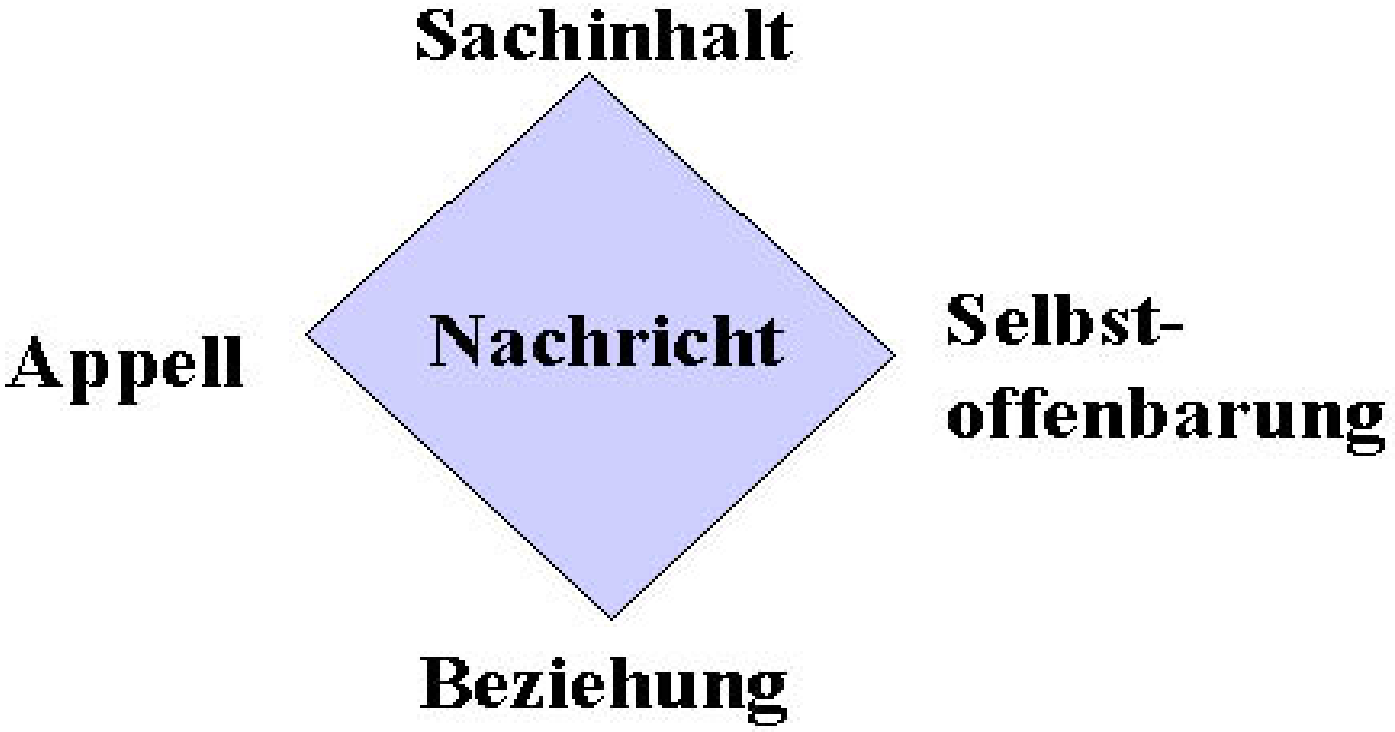
Richness

- verbal vs non-verbal

  miming, gestures, intonation, accent, ...
- intentional
- emotional
- a means for coordination
- speaking = physical activity + transfer of information (knowledge and belief)

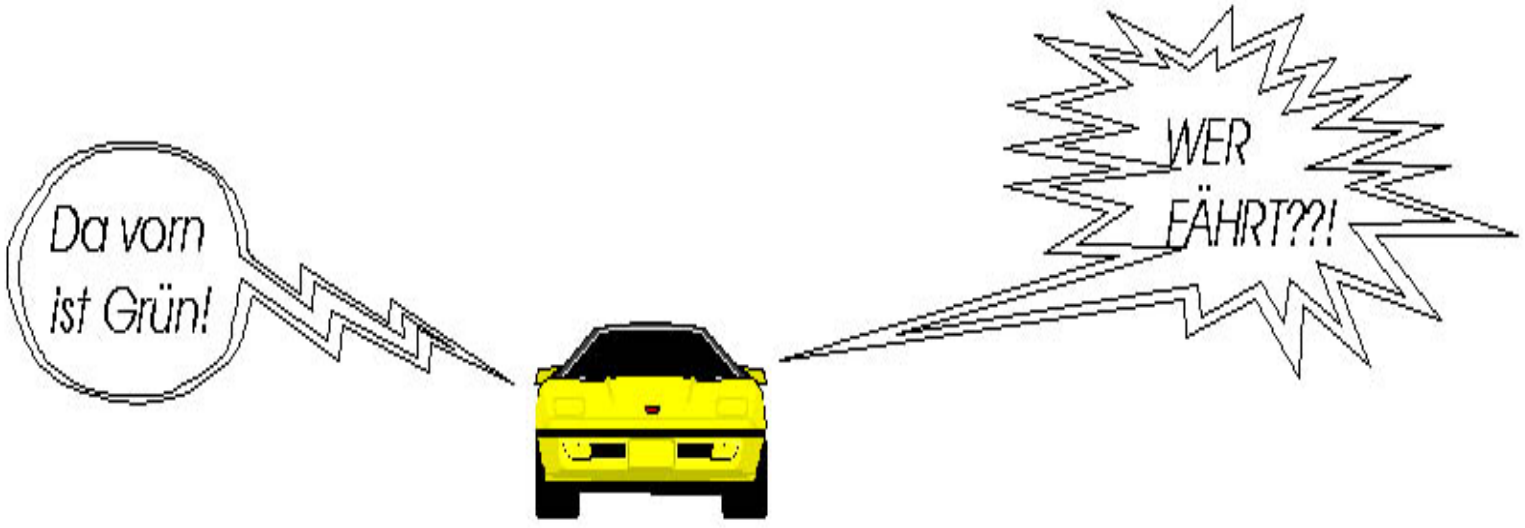
Complexity

- (Psychological) dimensions of a message (Schultz von Thun 1996):



Complexity (Cont'd)

- Example (Schultz von Thun 1996):



Complexity (Cont'd)

- Example (Cont'd):

Outline

Ingredients of a Speech Act

- Austin 1962, Searle 1970
- viewing messages as *actions*
- a speech act consists of:
  - Locution (physical utterance),
  - Illocution (intended meaning) and
  - Perlocution (resulting action).
- *Example*: “I am cold.”
  $\rightarrow$ ambiguity

Performatives

- Use *performatives* to distinguish illocutionary force: *promise, report, convince, insist, request, demand,* etc.
- Most common categories of performatives:
  1. assertives,
  2. directives,
  3. commissives,
  4. declaratives,
  5. expressives

Performatives (Cont'd)

Commonly used performatives:

Performative	*Messg. Type*	*Illoc. Force*	*Expected Result*
inform	*Assertion*	Declarative	belief revision
ask	*Query*	Directive	reply
reply	*Assertion*	Assertive	acceptance
request	*Query*	Directive	action/information
command	*Assertion*	Directive	action execution
allow	*Assertion*	Directive	acceptance
propose	*Query*	Assertive	counter-proposal?
confirm	*Assertion*	Commissive	acceptance
prefer	*Assertion*	Expressive	belief revision?

Outline

KQML

- *KQML* = Knowledge Query and Manipulation Language (Labrou & Finin 1994)
- Message format:

```
(performative
        :sender          <word>
        :receiver        <word>
        :in-reply-to     <word>
        :reply-with      <word>
        :language        <word>
        :ontology        <word>
        :content         <expression>)
```

- Various kinds of performatives

KQML (Cont'd)

- Example:

```
(advertise
          :sender          Agent1
          :receiver        Agent2
          :in-reply-to     ID1
          :reply-with      ID2
          :language        KQML
          :ontology        kqml-ontology
          :content         (ask
                           :sender      Agent1
                           :receiver    Agent3
                           :language    Prolog
                           :ontology    blocks-world
                           :content     "on(X,Y)"))
```

KIF

- KIF = Knowledge Interchange Format
- KQML does not say anything about *content* of messages
  $\rightarrow$ need content languages/ontologies
- KIF is a logical language to describe contents/ knowledge (first-order logic with some extensions/restrictions)
- Examples of KIF formulae:
  - `(=> (and (real-num ?x) (even-num ?n)) (> (expt ?x ?n) > 0))`
  - `(interested joe '(salary ,?x ,?y ,?z))`

Outline

What is an (Interaction) Protocol?

- Definition:
  - *"[an interaction protocol is] an interaction regime that guides the agents"* (Koning, Francois & Demazeau 1999)
  - *"Interaction protocols govern the exchange of a* series *of messages among agents."* (Huhns & Stephens 1999)
- Restrict the range and ordering of possible messages
- Usually formalized by state diagrams or *Interaction Diagrams* in FIPA-AgentUML
  (FIPA $=$ Foundation for Intelligent Physical Agents)

Types of Agent-specific Protocols

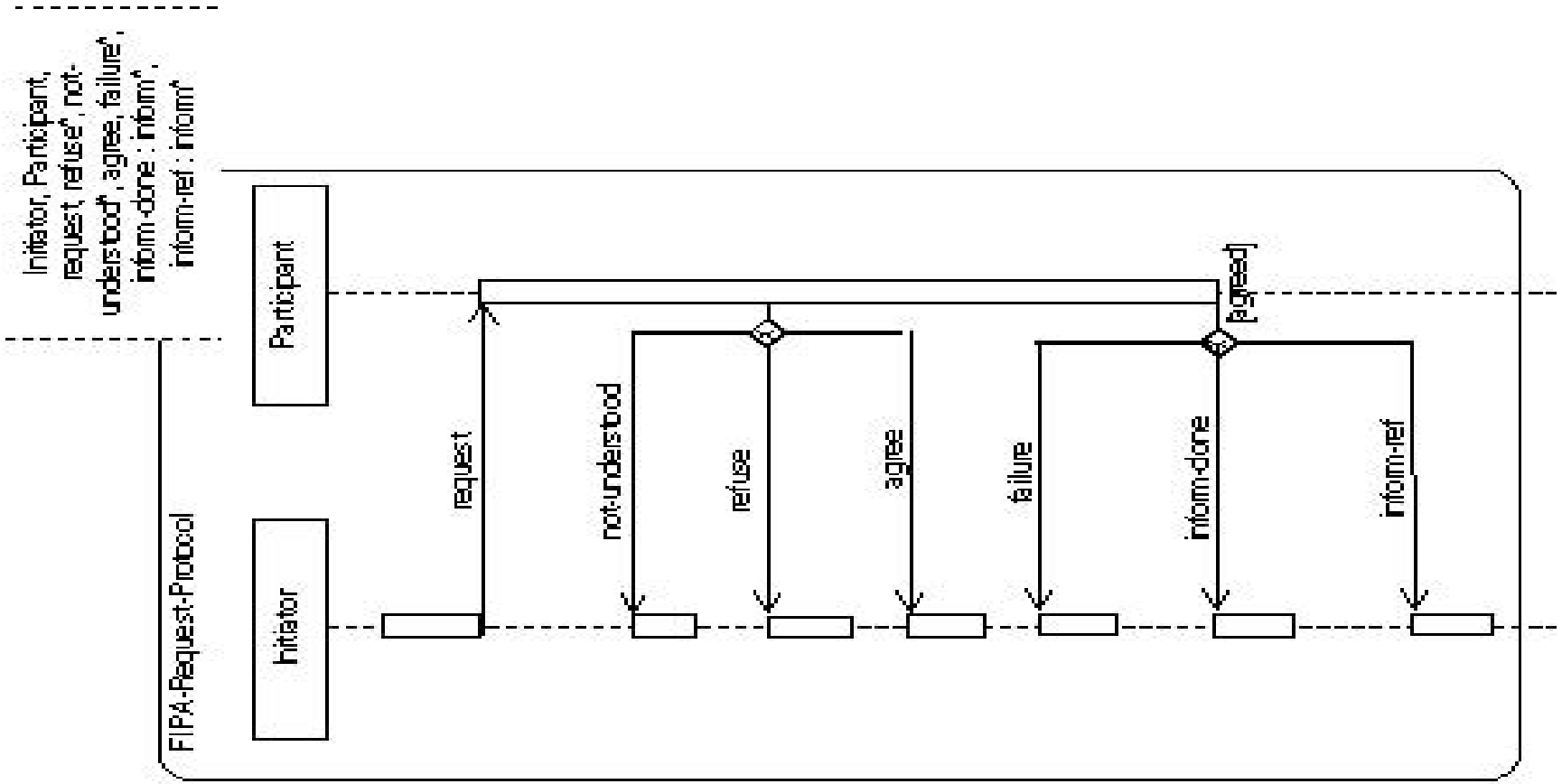
- Argumentation **protocols**
- Contracting **protocols**
- Auctions **protocols**
- Bargaining **protocols**
- Voting **protocols**
- Brokering **protocols**
- Matchmaking **protocols**
- Authentication **protocols**

Protocol Design

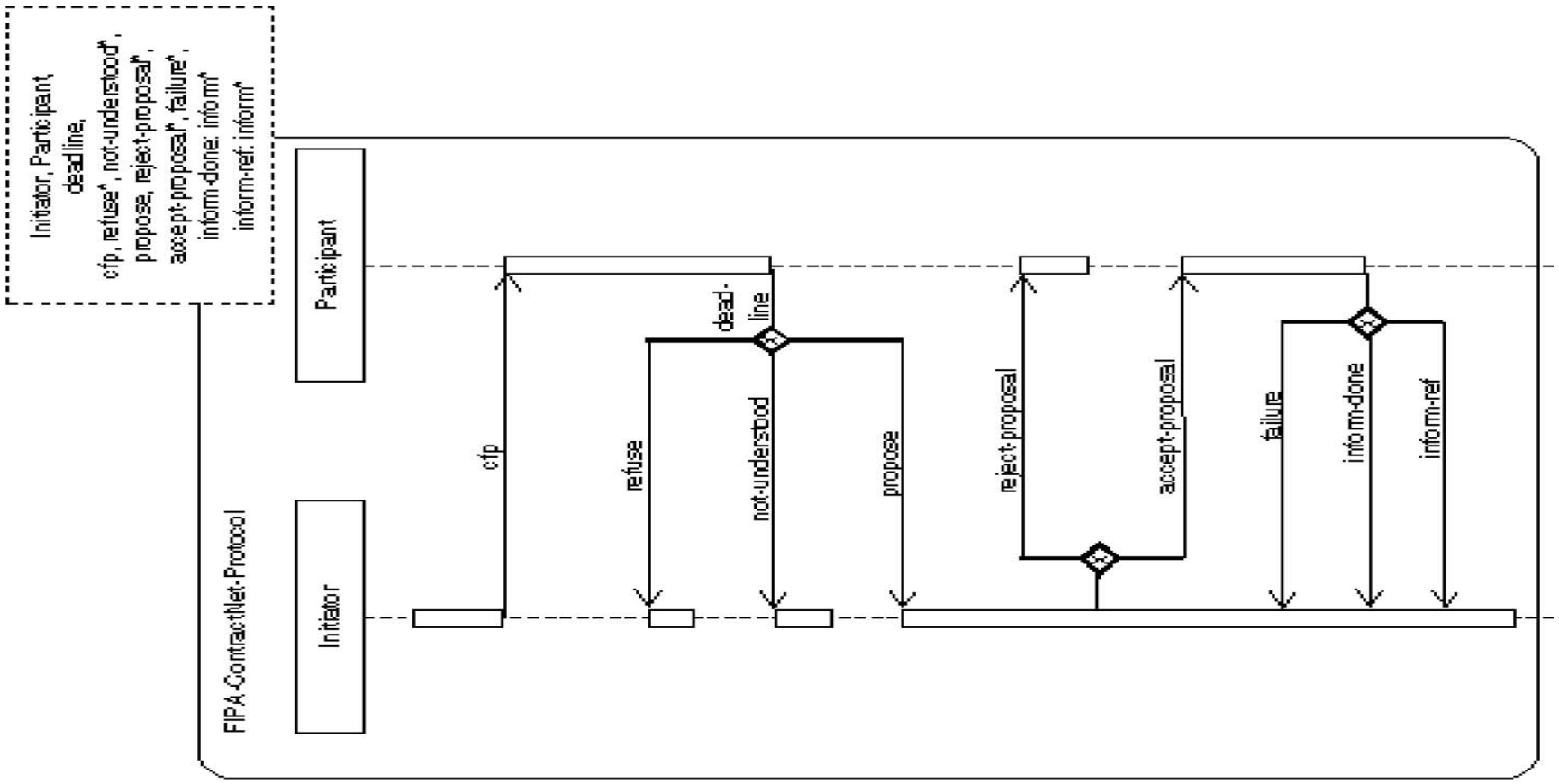
Six-step process (Koning, Francois & Demazeau 1999):

1. describe the *interaction capabilities* of the agents in use,
2. clarify the *types of messages* involved,
3. describe the *agents' behaviours*,
4. explain the *possible message sequences* between agents,
5. clarify the various *internal agent states*,
6. establish the *diagram of the protocol*.)

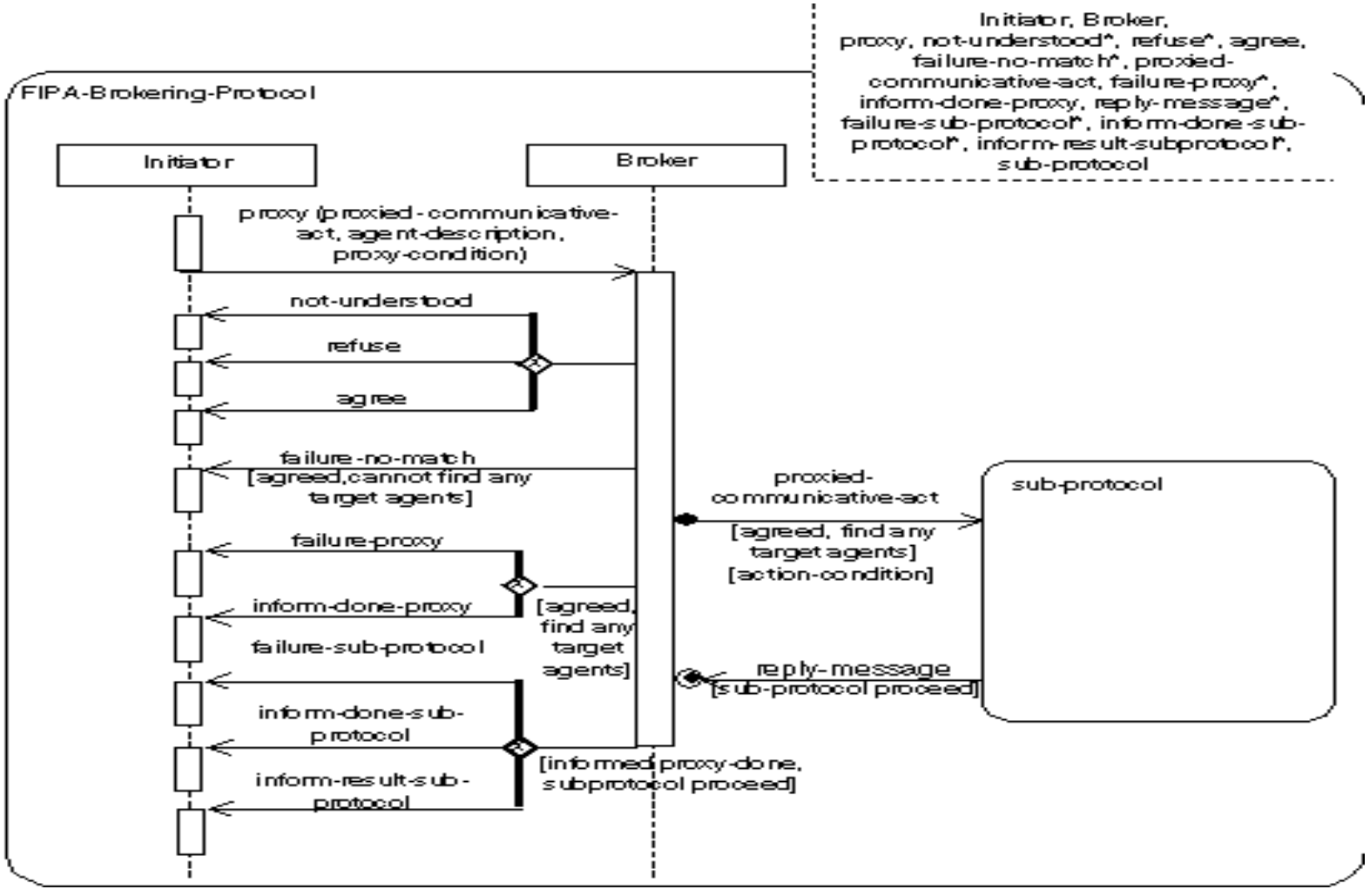
Example 1: A Basic Request Protocol



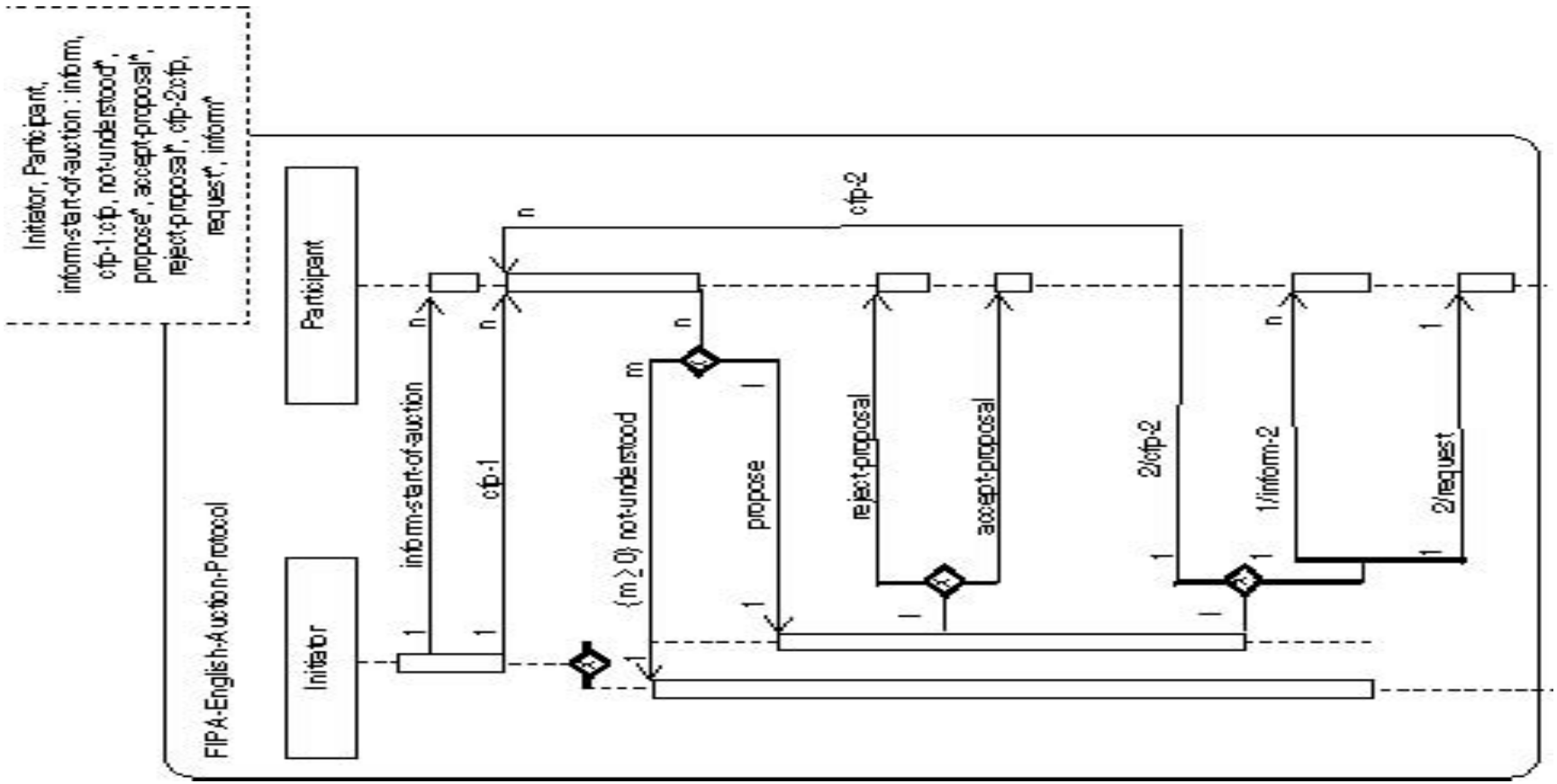
Example 2: Contract Net Protocol



Example 3: Brokering Protocol



Example 4: English Auction Protocol



Outline

What is an ”‘Ontology”’

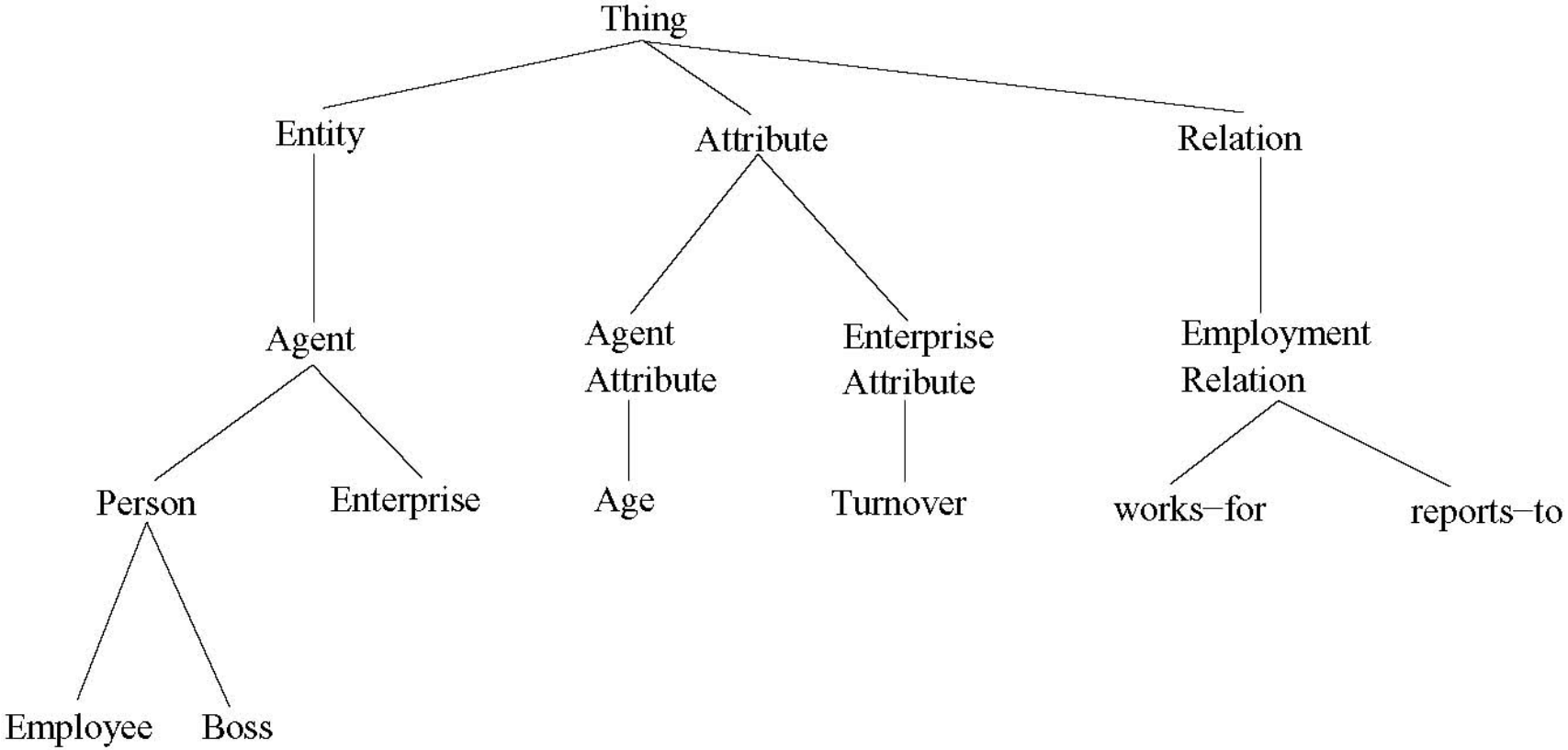
- Ontology =
  - description of relevant objects and relations in a domain
  - a formal, explicit specification of a shared conceptualization (Gruber 1993)
    - conceptualization: abstract model of some phenomenon, identifies its relevant characteristics
    - explicit: model (characteristics) are formulated explicitly
    - formal: machine-readable
    - shared: captures consensual knowledge (i.e., accepted not only by a single individual)

Example 1

- Expressing ontologies in simple statements:
  - `(class Block), (class PhysicalObject), (subclassOf Block PhysicalObject)`
  - $\forall$ `x,y,z (instanceOf x y` $\wedge$ `(subclassOf y z)` $\Rightarrow$ `(instanceOf x z)`
  - `(domain On-Table PhysicalObject)`
  - `(range On-Table PhysicalObject)`

Example 2

- Expressing ontologies in tree structures:



Principles for Ontology Design

- *Clarity*: minimize ambiguity, motivate distinctions, give examples
- *Coherence*: internal consistency
- *Extendibility*: extension of existing terms without need to revise existing definitions
- *Minimal encoding bias*: ideally representation choices are not made for the convenience of notation or implementation
- *Minimal ontological commitment*: ontology should make as few claims as possible about the world being modelled (parties committed to the ontology are free to specialize and instantiate the ontology as needed)

Languages and Tools

- there are not only many “ready-to-use” ontologies available already ...
  - common sense ontologies, domain ontologies, task ontologies, etc.
  - e.g. CYC, WordNet, PIF (business process modeling), PhysSys (knowledge about physical system processes), AIRCRAFT (air-campaign planning knowledge)
- ... but also many languages for ontology specification ...
  - KIF, Ontolingua, Frame Logic, CLASSIC, LOOM, CycL, etc.
  - languages being conform with Semantic Web standards: SHOE, XOL, OIL, DAML-OIL, etc.
- ... and many useful ontology design tools
  - Protege, Webonto, ONTOEDIT, Ontobroker, etc.