Multi-Agent Systems

An Introduction

ASSIGNMENTS (due October 21 at class)

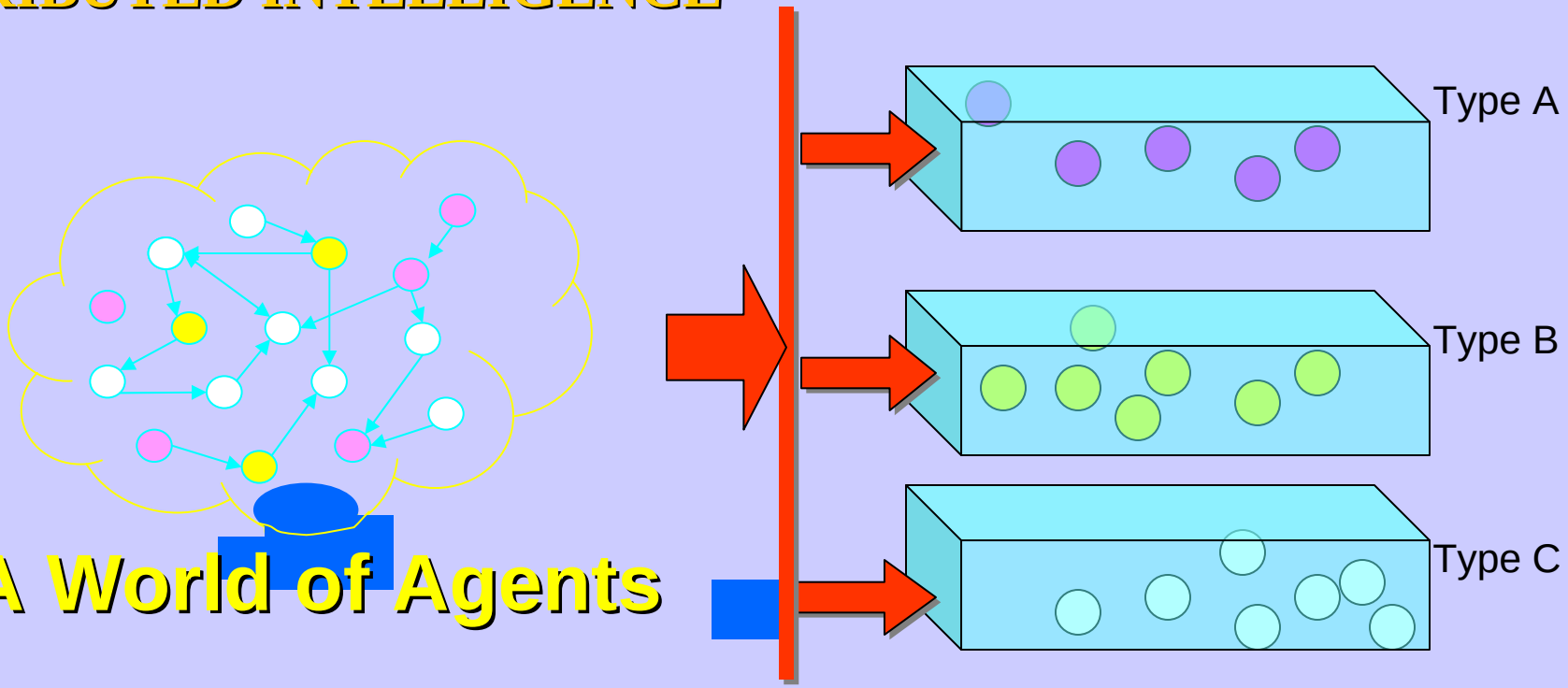
- Give an example of negotiation protocol
- Write in words the interpretation of KQML example on slide 25
- Take a simple example from your project and ‘agentify’ using the three methods from slide 54
- Using slide 74 select max 3 appropriate modes of interaction for your project and give examples for each of them
- Similar to slides 85 and 86 prepare a dialogue diagram for an example from your project
- Good luck with your midterms next week!

MASs definition

It is rare for an agent to act in isolation and it is even more rare for an agent to be useful on its own

Definition*: A multi-agent system (MAS) consists of a network of loosely-coupled computational autonomous agents who can perform actions, they have resources at their disposal and they possess knowledge, capabilities or skills. They are situated in a common environment and they can interact through a set of rules, namely an* ***interaction protocol****.*

DISTRIBUTED INTELLIGENCE



A World of Agents

- An agent is any entity that can send or receive a message.
- An agent can be hardware or software or both (and can be a human)
- An agent can be simple or complex with any desired functionality

Single agent-based systems differ from multi-agent systems:

- Environment: agents need to take into account others who may **interfere** with their plans and goals. They need to **coordinate** with others in order to ***avoid conflicts***.
- Knowledge/expertise/skills: these are ***distributed***
- Design: agents in a MAS need not be homogeneous and may be designed and implemented using different methodologies and languages
- Interaction: agents interact following rules of interaction (***interaction protocols***)

Interaction protocols

- Interaction protocols govern the exchange of a series of messages among agents
- **Communication** protocols provide rules that structure message-passing and produce meaningful dialogues or conversations
- **Cooperation** protocols provide a framework within which agents can coordinate their actions to achieve a complex task or solve a difficult problem in a cooperative way
- **Negotiation** protocols are used in situations where agents have incompatible goals to enable the parties involved to reach a compromise and resolve conflict

Agent communication

Why is communication important?

- Communication is required in MASs where agents have to cooperate, negotiate etc., with other agents
- Agents need to communicate in order to understand and be understood by others
- Diversity introduces heterogeneity

Can we use natural language as an Agent Communication Language (ACL)? No, natural language is ambiguous

Aspects of communication

Communication has three aspects

- Syntax: how the symbols of communication are structured
- Semantics: what the symbols denote
- Pragmatics: how the symbols are interpreted

Meaning: A combination of semantics and pragmatics

Speech Act Theory

- Introduced by Austin (1962) and extended by Searle (1969)
- Communication exchange is a form of action: a speech act
- An utterance has three aspects
  - Locution: the physical utterance with context and reference
  - Illocution: the act of conveying intentions
  - Perlocution: the action that occurs as a result of the illocution
- Identifying the intent or illocutionary force of an utterance is important

Illocutionary Force Categories

- **Assertive**: statements of fact
- **Commissive**: commitments
- **Directive**: commands in a master-slave structure
- **Declarative**: statements of fact
- **Expressive**: expressions of emotion
- Illocutionary (Webster's) relating to or being the communicative effect (as commanding or requesting) of an utterance **<"There's a snake under you" may have the *illocutionary* force of a *warning*>**

Communication protocols

An agent communication language is defined at three levels:

- Lower level: specifies *method of interconnection*
- Middle level: specifies format/*syntax*
- Top level: specifies meaning/*semantics*

Communication protocols need to separate:

- semantics of the protocol which must be domain-independent
- semantics of the enclosed message which may be domain-dependent

ACL - Example

- **The information content will be communicated through the existing message types.**
- Ex: (request

  :sender OrderAgent

  :receiver OperationPlanningAgent

  :content(action d (produce 50 parts A (product ABC))(due date: Dec 12, 2000))

  :protocol fipa-request

  :reply-with order 132

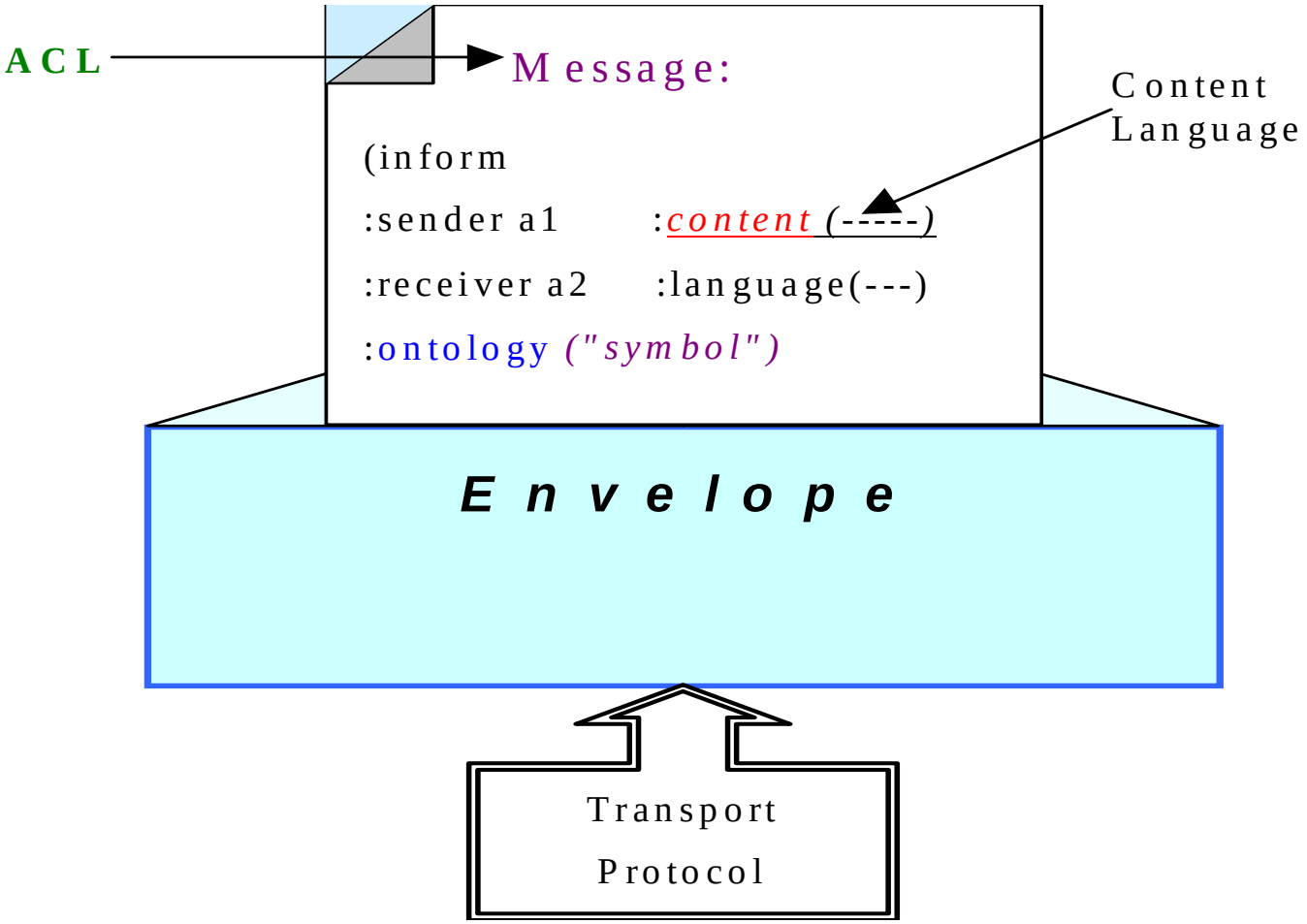
)

ACL Message structure

- Contains parameters=>attributes /values:
- *:language* representation language for content

- *:content* the information about which a message is sent
- *:receiver* agent that receives the message
- *:protocol* interaction protocol used by sender
- *ontology* ontology to which the content refers to

Message Encoding



ACL components

An agent communication language consists of three components:

- An ‘outer’ language (pragmatics): the language used to express the primitives, i.e. the *performatives*
- An ‘inner’ language (syntax): the logical or representation language which is used to write the message itself
- Vocabulary (semantics): describes the domain of discourse in terms of concepts and their relationships and prescribes meaning to the terms used

FIPA ACL

- FIPA ACL was part of FIPA’s standardization efforts for interaction protocols
- Based on speech acts, messages are communicative acts
- It upkeeps the distinction between outer and inner languages and is independent of the inner language
- A number of communication exchange protocols based on the FIPA ACL communicative acts have been defined

Features of FIPA ACL

- All communicative acts can be defined compositionally in terms of two core ones: *inform* and *request*

```
(inform
    :sender Agent435
    :receiver Agent450
    :language Prolog
    :ontology companyx
    :content “salary(9876543,32765)”)
```

FIPA ACL semantics

- The agents' states are defined via SL (Semantic Language)
- Semantics is based on mental attitudes such as belief, desires and intentions etc.
- The meaning of primitives is given in terms of Feasibility Conditions (*FP*s) and Rational Effects (*RE*s)

$\langle i, \text{inform}(j, \phi) \rangle$

FP: $B_i(\phi) \wedge \neg B_i(B_i f_j(\phi) \vee U_i f_j(\phi))$

RE: $B_j(\phi)$

- *FP*(*a*) describes the necessary conditions that need to hold true in order for the sender to send communicative act *a*
- *RE*(*a*) represents the effect that an agent can expect to occur as a result of performing the act (the purpose of the message) – the *RE* of a communicative act cannot be guaranteed as the receiver of the message is autonomous and may not conform

Protocols for Agent Interactions

- FIPA ACL deals with ***intention interpretation*** and supports the communication between agents through *explicit linguistic actions*
- specifies **information content transmitted** and the **protocols**
- Each ACL message
- Represents a communicative act (***Policy***) that can be simple (inform ,request,confirm) or composed request(propose(action))
- Has its corresponding semantics and **protocol**
- Contains parameters=>attributes /values:
- *:language* representation language for content
- *:content* the information about which a message is sent
- *:receiver* agent that receives the message
- *:**protocol*** **interaction protocol used by sender**
- *ontology* ontology to which the content refers to

Knowledge and Query Manipulation Language

- KQML is a high level, message-oriented, communication language and protocol for information exchange
- An outer language
- There is a separation of protocol semantics from content semantics
- All the information required to understand the message is included in the message
- KQML is independent of:
  - the transport mechanism (tcp/ip, corba etc.)
  - the content language (KIF, SQL etc.)
  - the ontology assumed by the content

KQML message structure

Message structure

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

Example

```
(tell
    :sender Agent435
    :receiver Agent450
    :in-reply-to id-msg-005
    :language KIF
    :ontology cycbase
    :content (salary 9876543 32765))
```

- Reserved keywords for performatives and parameters
- The performatives are organized into several categories

KQML semantics

- Semantics to KQML performatives is given in terms of preconditions, postconditions and completion conditions
- Semantics is described in a language which includes operators for expressing an agent's attitudes: beliefs (*BEL*), knowledge (*KNOW*), desires (*WANT*) and intentions (*INT*)

If A and B are the sender and the receiver respectively

- *Pre*(A) and *Pre*(B) identify the necessary states for sending and receiving and processing a performative respectively
- *Post*(A) and *Post*(B) determine the states of the agents after the successful utterance of the performative and its receipt
- *Completion* describes the final state after the conversation has taken place and the original intention for uttering the performative has been accomplished

tell(A,B,X)

Semantics of tell

Pre(A): *BEL*(A,X)$\wedge$*KNOW*(A,*WANT*(B,*KNOW*(B,S)))

Pre(B): *INT*(B,*KNOW*(B,S))

where S may be any of *BEL*(B,X) or $\neg$ *BEL*(B,X)

Post(A): *KNOW*(A,*KNOW*(B,*BEL*(A,X)))

Post(B): *KNOW*(B,*BEL*(A,X))

Completion: *KNOW*(B,*BEL*(A,X))

Criticisms of KQML

KQML has been criticized on a number of fronts:

- Some of its performatives (achieve, broker, stream-all) are not really speech acts
- Lack of commissive performatives
- No security model

Criticisms of FIPA ACL

- Lack of security mechanism (though a proposal had been submitted)
- No facilitation primitives

KQML vs FIPA ACL

- Syntactically similar and both ascribe to the same theory
- Semantics differ:
  - different ways to describe the primitives, i.e. pre, post, completion conditions for KQML, Feasibility Conditions and Rational Effects for FIPA ACL
  - different language to describe the propositional mental attitudes, e.g. KQML's *BEL* operator is not the same as FIPA ACL's *B* operator
- Different treatment of the 'administration' primitives

Agent Interactions

- FIPA ACL deals with ***intention interpretation*** and supports the communication between agents through *explicit linguistic actions*
- The information content transmitted and the protocols are those described and defined in FIPA specifications
- www.fipa.org

ACL Message structure

- Represents a communicative act (***Policy***) that can be simple (inform ,request,confirm) or composed request(propose(action))
- Has its corresponding semantics and protocol
- Contains parameters=>attributes /values:
- *:language* representation language for content
- *:content* the information about which a message is sent

- *:receiver* agent that receives the message
- *:protocol* interaction protocol used by sender
- *ontology* ontology to which the content refers to

ACL - Example

- **The information content will be communicated through the existing message types.**
- Ex: (request

  :sender OrderAgent

  :receiver OperationPlanningAgent

  :content(action d (produce 50 parts A (product ABC))(due date: Dec 12, 2000))

  :protocol fipa-request

  :reply-with order 132

)

Content Languages

- Agents need to be able to represent and reason about:
  - models of other agents, their beliefs, knowledge, perceptions
  - tasks, task structure, planning, goals
- A common language for representing these things is needed (content language)
- Examples of content languages: SQL, Prolog, KIF

Knowledge Interchange Format (KIF)

- KIF is a content language based on first order logic
- Highly expressive and can be used to represent knowledge in declarative form in knowledge bases
- Human readable

KIF syntax and semantics

- KIF is a prefix version of first order predicate calculus
- Standard logical operators as well as quantification

  (exists ((?x bird) (?y tree)) (on ?x ?y)
- One can define objects, relations between objects and rules

  (defrelation mother(?x ?y):= (and (female ?x) (parent ?x ?y)))
- ***Knowledge about knowledge*** can be expressed via the (‘) and (,) operators

  (interested agent435 ‘(distance ,?x ,?y ,?z))

- Model-theoretic semantics

Issues

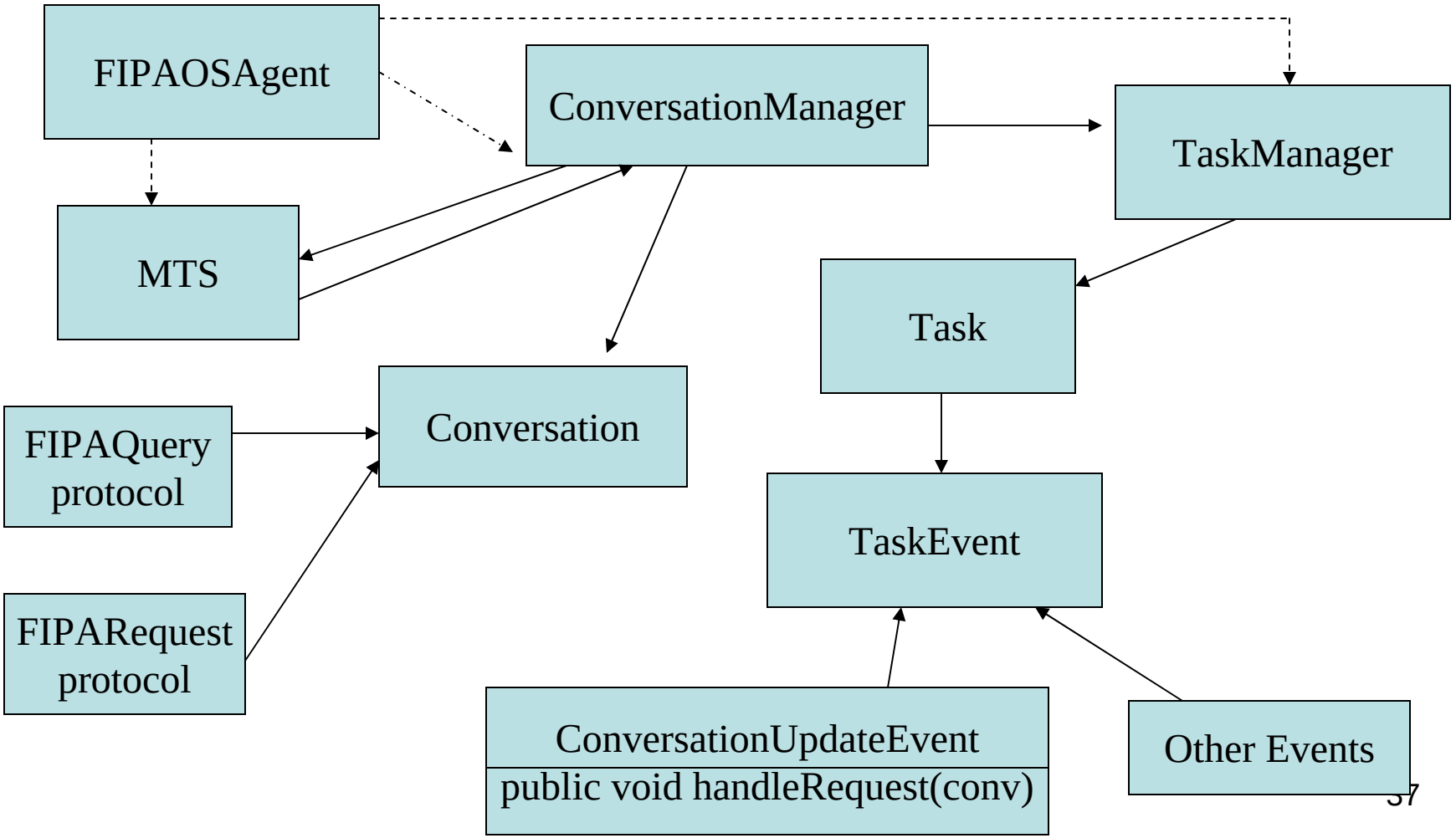
- KIF is probably the most widely used content language for knowledge bases
- Highly expressive, but:
  - complicates the building of fully conforming systems
  - resulting systems tend to be heavyweight
- Its logic-oriented format may not be acceptable to a wider community

Dialogues

A series of messages being exchanged comprise a dialogue or conversation

Type	Initial situation	Participant's goal	Main goal
Persuasion	conflict of opinions	persuade other agent	clarify/resolve issue
Inquiry	need for information	find evidence or proof	knowledge acquisition
Negotiation	conflict of interests	get best deal possible	reach a deal
Information seeking	personal ignorance	acquire or provide personal knowledge	distribute information
Deliberation	need to decide on action	influence outcome	reach a decision
Eristic	conflict	provoke other agent	reach a compromise
Mixed	various	various	various

Tasks and Conversations

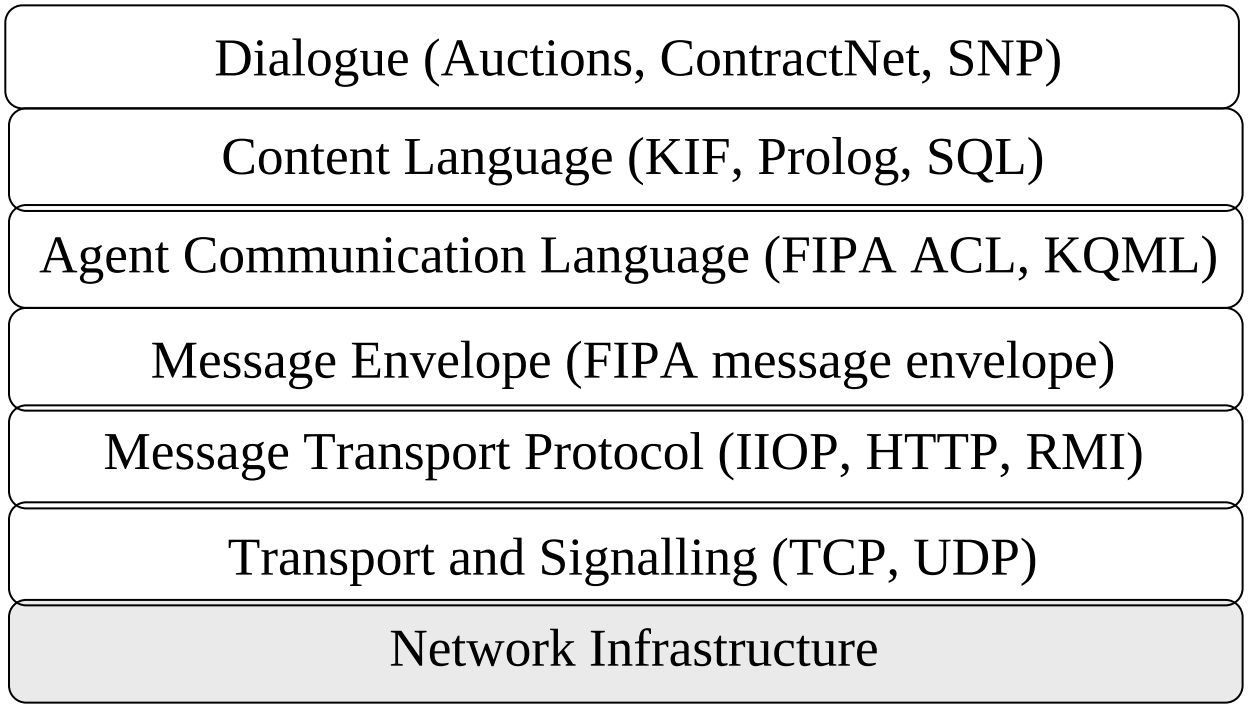


Constructing a Task

```
 public class FirstTask extends Task{
  public FirstTask(OrderAgent owner){
  }
public void startTask(){
}
public void sendRequest( Conversation conv ){
ACLMessage acl= getNewConversation(String protocol)
     acl.setContent(…)
     acl.setLanguage(…)
     acl.setOntology(…)
     forward(acl)
 }
}
```

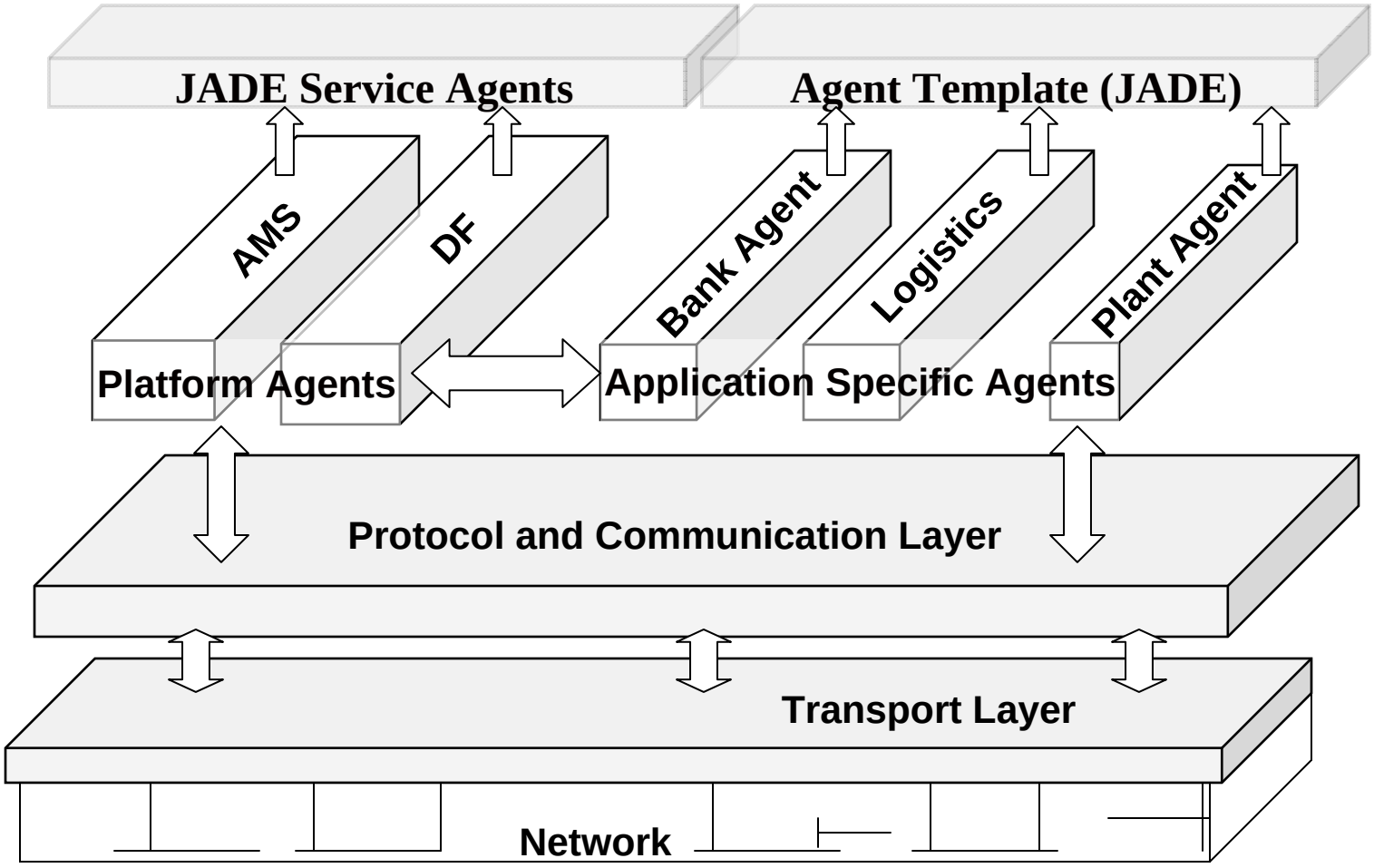
A layered model of communication

To enable effective agent communication among agent platforms a layered approach is required



Agency Design

- **System Architecture**
- Agent Definitions
- Knowledge Modeling
- Agent Class Structure



Ontologies: The problem

- Different agent developers $\Rightarrow$ diversity/heterogeneity of agents
- Agents may refer to the same concept under different terms
  - Client: anyone who buys products
  - Customer: anyone who buys products
- Agents may refer to different concepts under the same term
  - Client: a computer in the client-server architecture
  - Client: anyone who buys products

- A shared representation is essential for successful communication and interaction between perhaps heterogeneous agents to take place
- An ontology is a specification of the objects, concepts, and relationships in an area of interest
- An ontology represents a domain from a particular perspective

Ontologies are useful:

- They define a common vocabulary for those who need to share information
- They enable the reuse of domain knowledge
- They allow us to encode assumptions about a domain explicitly – these can then easily be changed by simply changing the ontology
- They allow us to separate domain from operational knowledge
- They can be used by people, software agents, databases and other computer systems

Explicit ontologies

- Domain knowledge can be encoded implicitly in procedures or explicitly in declarative statements
- An explicit ontology is a formal description usually in a logical language of the concepts in a domain of discourse, their properties and relations
- Focus is on classes which describe concepts

Three aspects

- Conceptualization: describes the underlying model of the domain in terms of classes, objects, attributes and relations
- Vocabulary: assigns symbols or terms to refer to the classes, objects, attributes and relations
- Axiomatization: encodes rules and constraints capturing significant aspects of the domain model

Developing ontologies

Ontology development is done in a number of steps:

1. Determine the domain and the scope of the ontology as well as the purpose for using it
2. Organize the ontology and design the overall conceptual structure of the domain (pivotal in ontology design and development)
   - Top-down approach: starts with general concepts and proceeds with specialization
   - Bottom-up approach: starts with defining specific classes and proceeds with generalization
   - Mixed

3. Check the ontology for syntactic, logical and semantic inconsistencies
4. Publish and deploy the ontology
5. Ontologies need to be maintained, updated and perhaps linked with other ontologies

Ontology Web Language

- Used to describe the terminology of documents on the web
- OWL builds on the Resource Definition Framework (RDF) and proposes a specific vocabulary that provides selected frame and description logic primitives to capture ontologies

Three sublanguages with increasing expressiveness:

- OWL Lite: mainly intended for class hierarchies and limited constraints. Lowest complexity
- OWL DL (includes OWL Lite): intended when completeness and decidability are important
- OWL Full (includes OWL DL): maximum expressivity, but no computational guarantees

OWL example

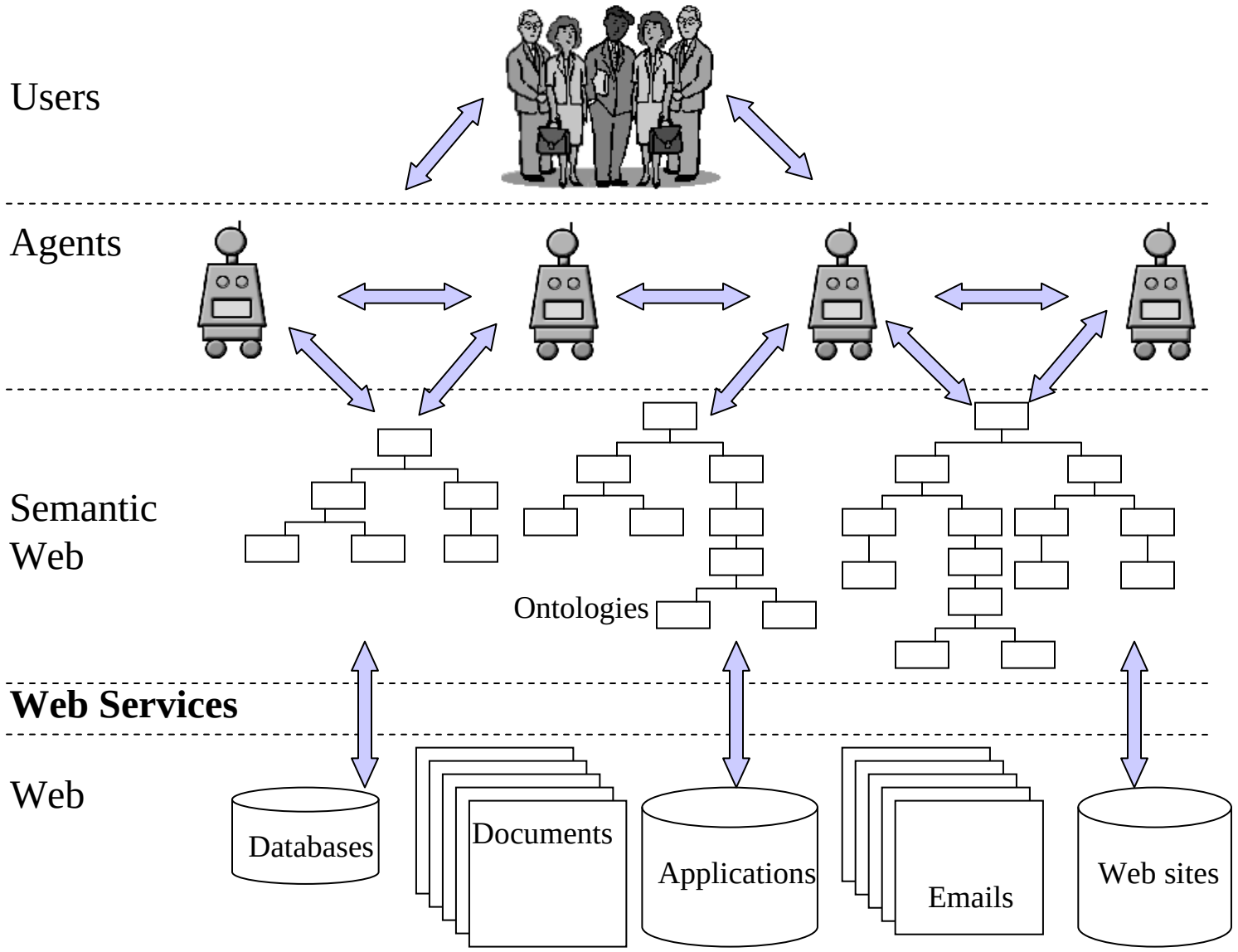
An ontology describes classes and relations between classes

Example

```
<owl:Class rdf:ID="Automobile">
   <rdfs:label>Automobile </rdfs:label>
   <rdfs:comment>Automobile class</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="Truck">
   <rdfs:subClassOf rdf:resource="#Automobile"/ >
   <rdfs:comment>Class Truck is a subclass of Automobile</rdfs:comment>
</owl:Class>
```

Agents, Web Services and the Semantic Web

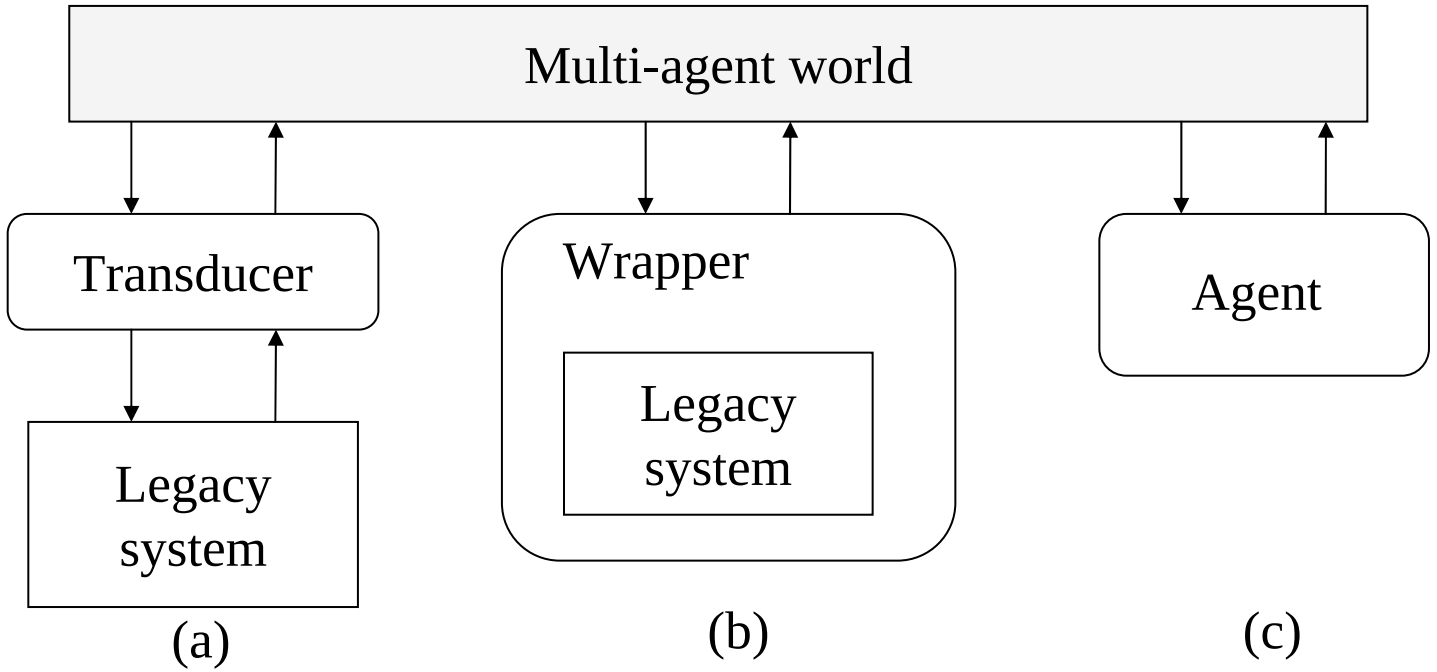
- Semantic Web vision: enrich web pages with meaning so that information can be accessed by programs
- To facilitate automated processing on the Web and enable applications to communicate and work with each other, we need programmatic interfaces, namely web services
- A web service is a collection of functions that are packaged as a single entity and published on the network to be used by others
- Web services can be combined to offer complex functionality
- Users delegate their needs to agents who can then deploy web services


Users
Agents
Semantic
Web
Ontologies
Web Services
Web
Databases
Documents
Applications
Emails
Web sites

Both web services and agents encapsulate functionality, but:

- Web services are static, unable to adapt
- Services are passive until they are invoked – once invoked the service will execute given the correct parameters
- Communication between services is at a low syntactic level; services can exchange data but they do not understand them; they do not ‘understand’ ontologies to resolve incompatibilities
- Composed functionality of web services can only be achieved when orchestrated by a third party, e.g. an agent; no social ability to coordinate with other web services or entities

From legacy systems to agents



Agentification methods
(a) Transducer; (b) Wrapper; (c) Rewrite the system

ACL – SOA message example

- **The information content will be communicated through the existing message types.**
- Ex: (**request** (… a service)
  :sender OrderAgent (service consumer)
  :receiver OperationPlanningAgent (provider)
  :content(action d (produce 50 parts A (service (product ABC))(due date: Dec 12, 2000)) description)
  :protocol fipa-request
  :reply-with order 132
)

Service Oriented Architecture

- **SOA** is an architectural style whose goal is to achieve *loose coupling* among interacting software agents.
- A service is a unit of work done by a *service provider* to achieve desired end results for a *service consumer*.
- Both provider and consumer are *roles* played by software agents on behalf of their owners.

SOA vs. Object-Oriented

- Object oriented programming: bind data and its processing together. (every CD would come with its own player and they are not supposed to be separated).
- SOA: loose coupling among interacting software agents by employing two *architectural constraints*:
- A small set of simple and ubiquitous interfaces to all participating software agents. Only generic semantics are encoded at the interfaces. The interfaces should be universally available for all providers and consumers.
- Descriptive messages constrained by an extensible schema delivered through the interfaces. No, or only minimal, system behaviour is prescribed by messages. A schema limits the vocabulary and structure of messages. An extensible schema allows new versions of services to be introduced without breaking existing services.

TIPS FOR SOA INTERFACES

- Reuse a few generic ones for all applications.
- Since only a few generic interfaces are available, express application-specific semantics in messages.

SOA RULES: messages

1. Messages must be ***descriptive***, rather than instructive, because the service provider is responsible for solving the problem. (e.g. Restaurant: you tell your waiter what you would like to order and your preferences but you don't tell their cook how to cook your dish step by step).
2. Service providers will be unable to understand your request if your messages are not written in a format, structure, and vocabulary that is understood by all parties. Limiting the vocabulary and structure of messages is a necessity for any efficient communication. The more ***restricted*** a message is, the easier it is to understand the message, although it comes at the expense of reduced extensibility.

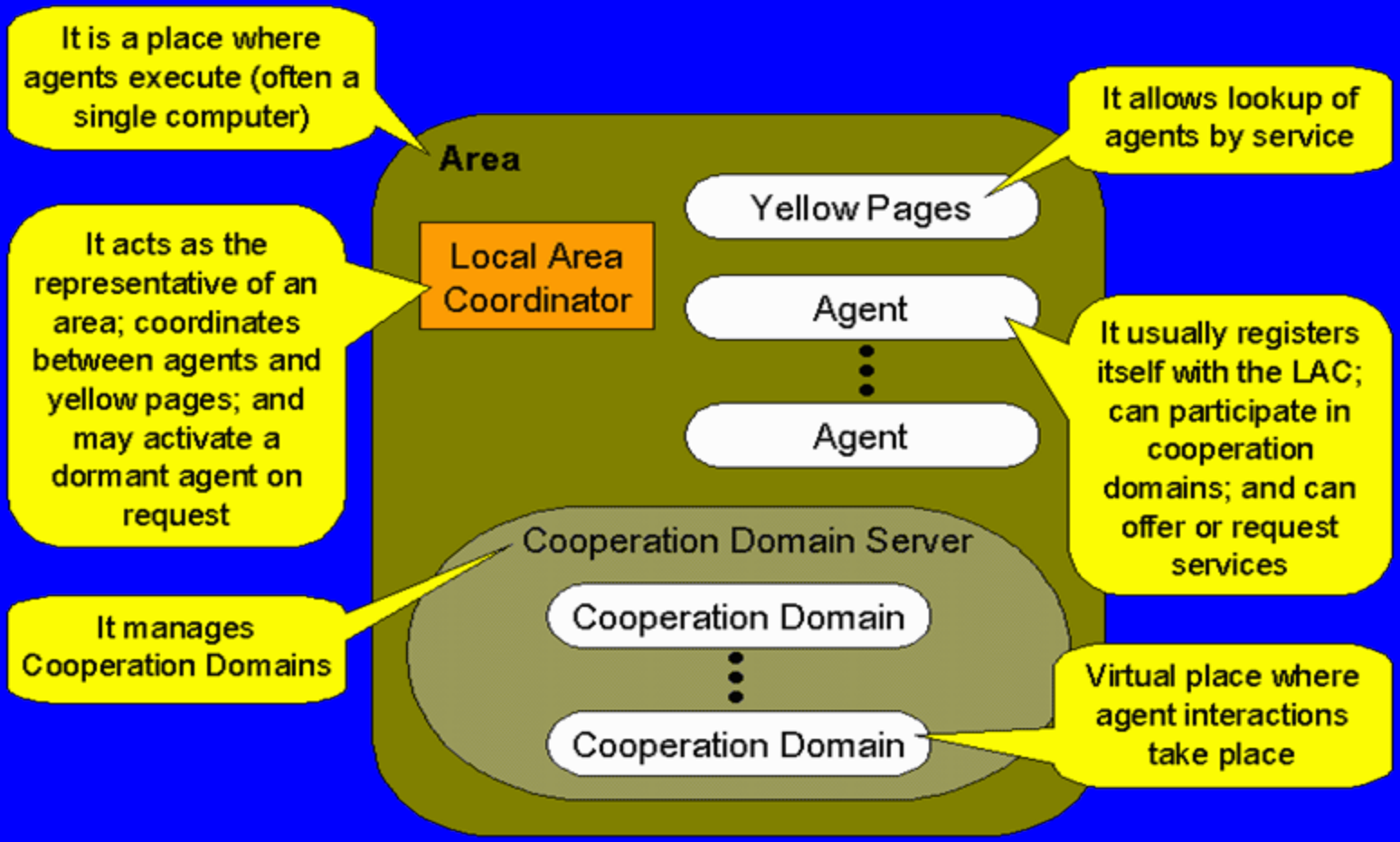
- **Restriction and extensibility** are deeply entwined. You need both, and increasing one comes at the expense of reducing the other. The trick is to have a right balance.

SOA

- SOA must have a mechanism that enables a consumer to discover a service provider under the context of a service sought by the consumer. The mechanism can be really flexible, and it does not have to be a centralized registry.

THE COOPERATION-COMMUNICATION LAYER

- URL: [protocol://] [id@] host : port [/path]

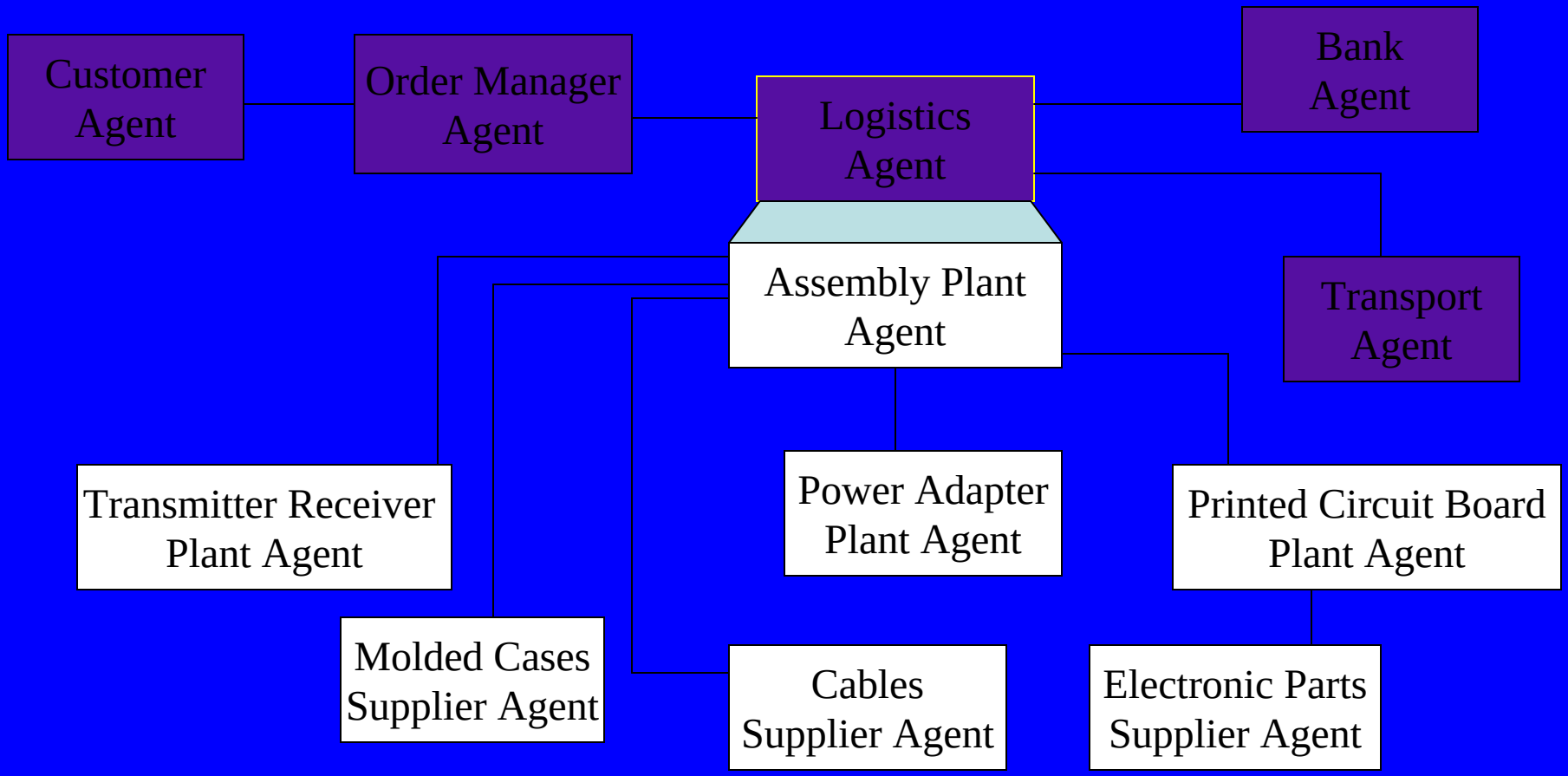


Interaction

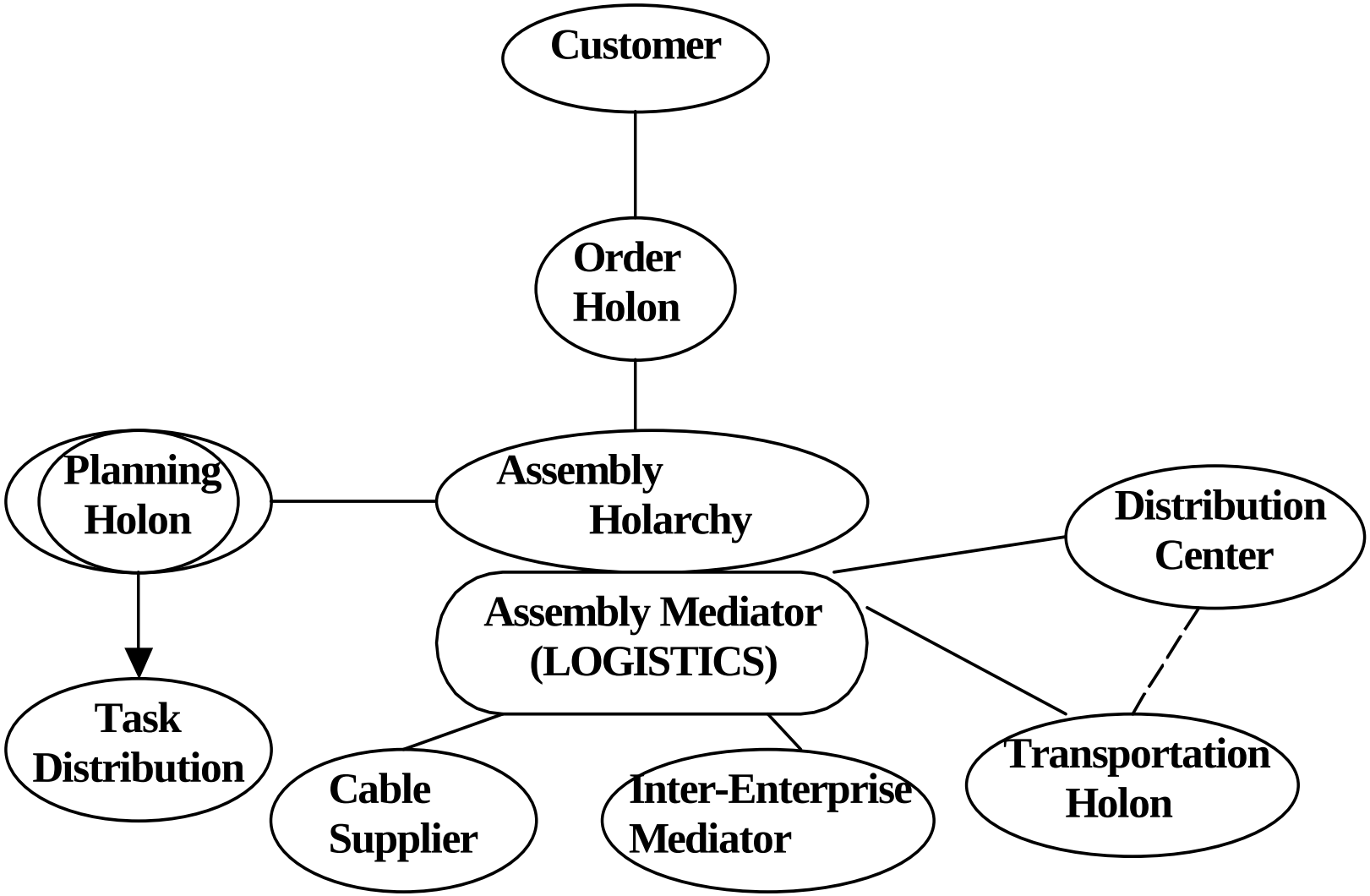
Definition: *An interaction occurs when two or more agents are brought into a dynamic relationship through a set of reciprocal actions* (Ferber 1999)

- Interactions develop as a result of a series of actions whose consequences influence the future behavior of agents
- May be direct or indirect, intended, or unintended
- Interaction assumes:
  - Agents that are capable of acting and/or

Supply Chain Scenario



Agents in concert create a network…



‘Middle’ Agents

Locating agents

- In open systems, agents do not necessarily know others
- The connection problem: finding an appropriate agent to do a given task or provide a service
- Facilities and appropriate infrastructure are required to enable software agents to locate and come into contact with others

Matching

A MAS consists of

- End agents which can be
  - Providers of services/products
  - Requesters
- Middle agents
  - Enable interactions among end agents
  - Purpose: to match requesters with providers

- Providers send advertisements or capability specifications which may include additional information on conditions of service
- Requesters send request specifications for required services which may include preference parameters
- Matching: the middle agent’s task is to match advertisements with request specifications as closely as possible

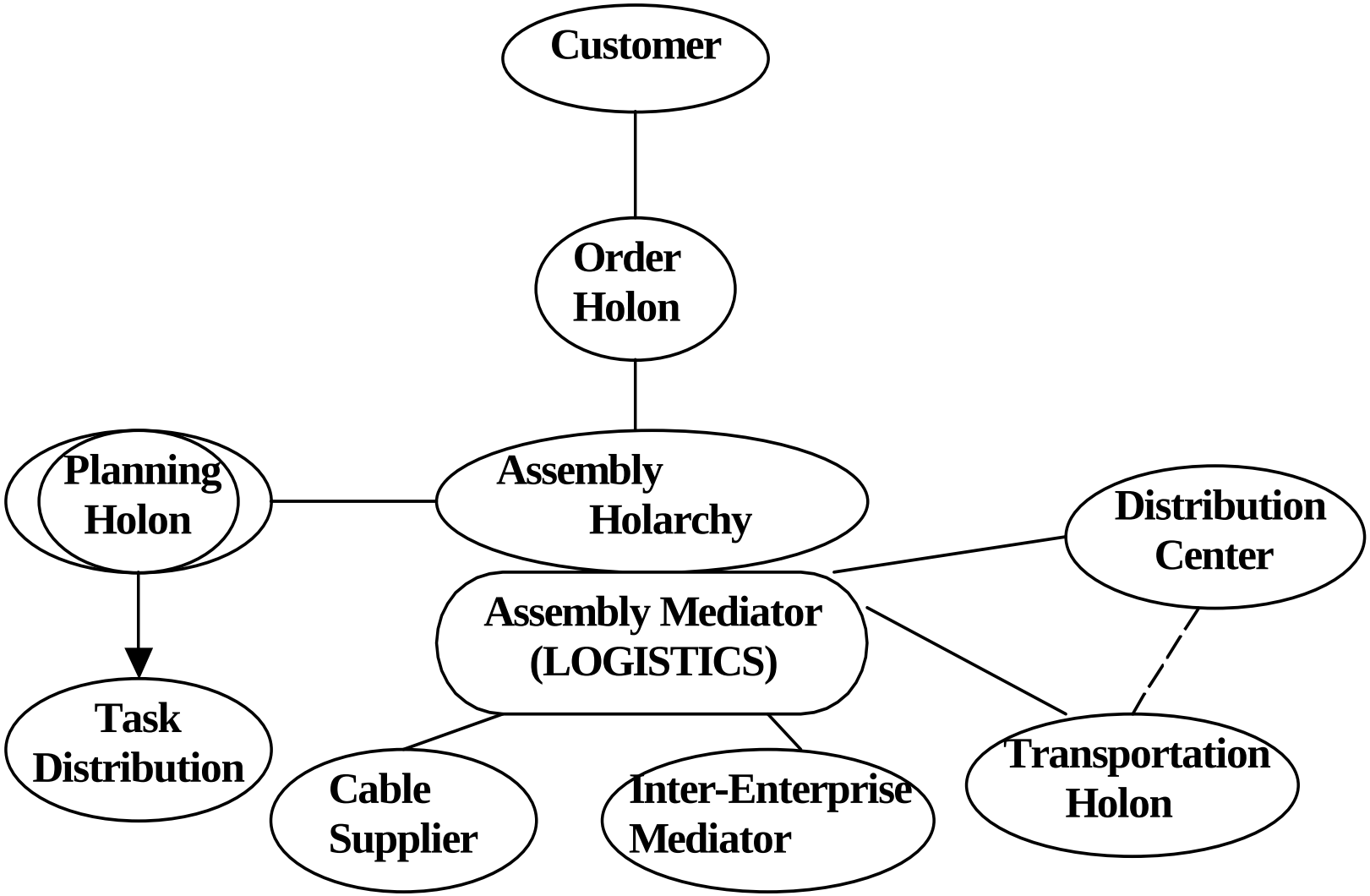
Two issues:

(i) Agents need to be able to communicate their advertisements and requests in a suitable language

(ii) Efficient matching mechanisms are required

- An advertisement matches a request when the service described by the former is *sufficiently similar* to the service

Agents in concert create a network…



… running on The Internet

Communication

(Uses TCP/IP but could use other protocols)

Message Intent

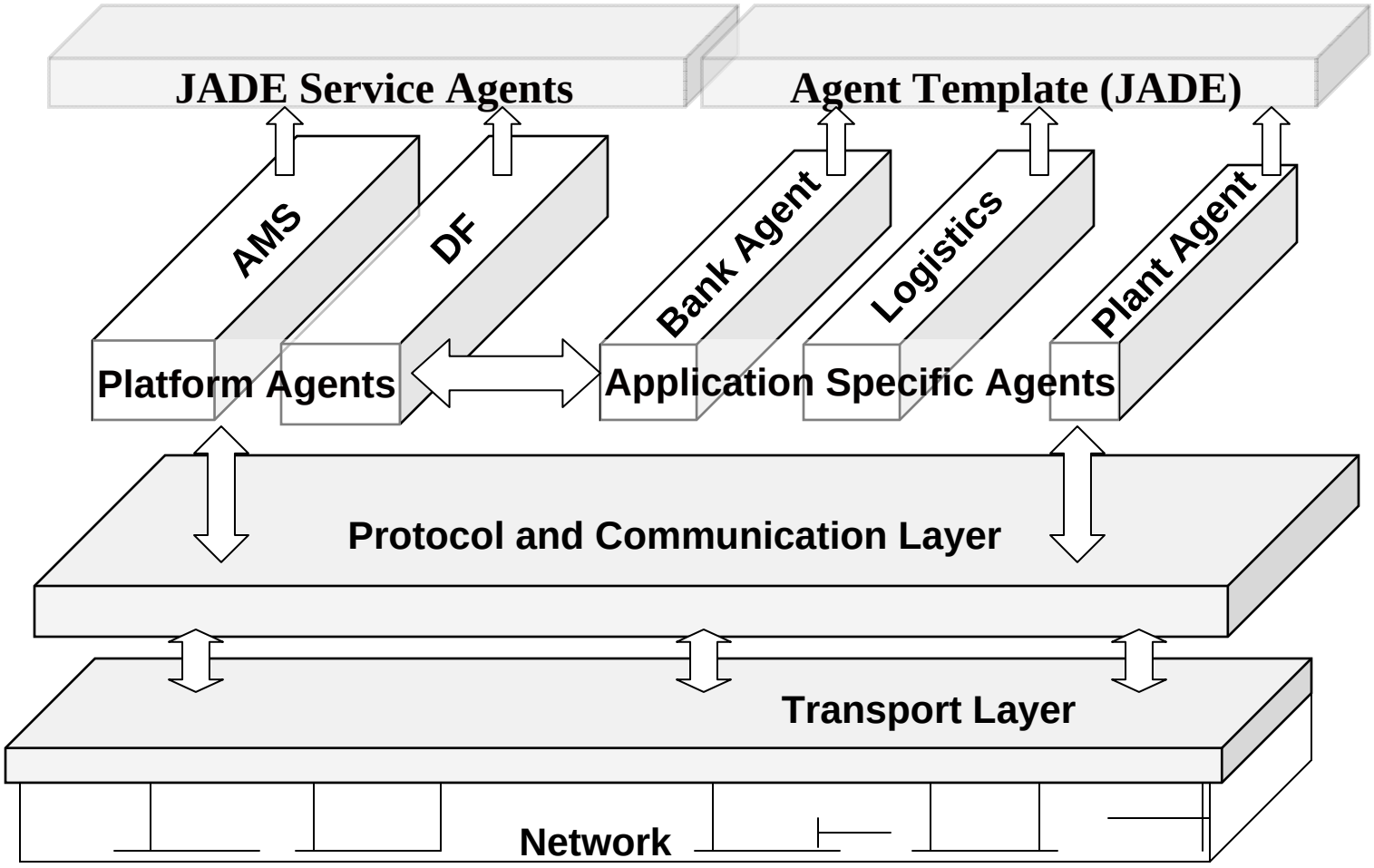
(Each expression is a speech act described by a performative)

Content

(Expressed in KIF or KRSL or other language)

Agency Design

- **System Architecture**
- Agent Definitions
- Knowledge Modeling
- Agent Class Structure



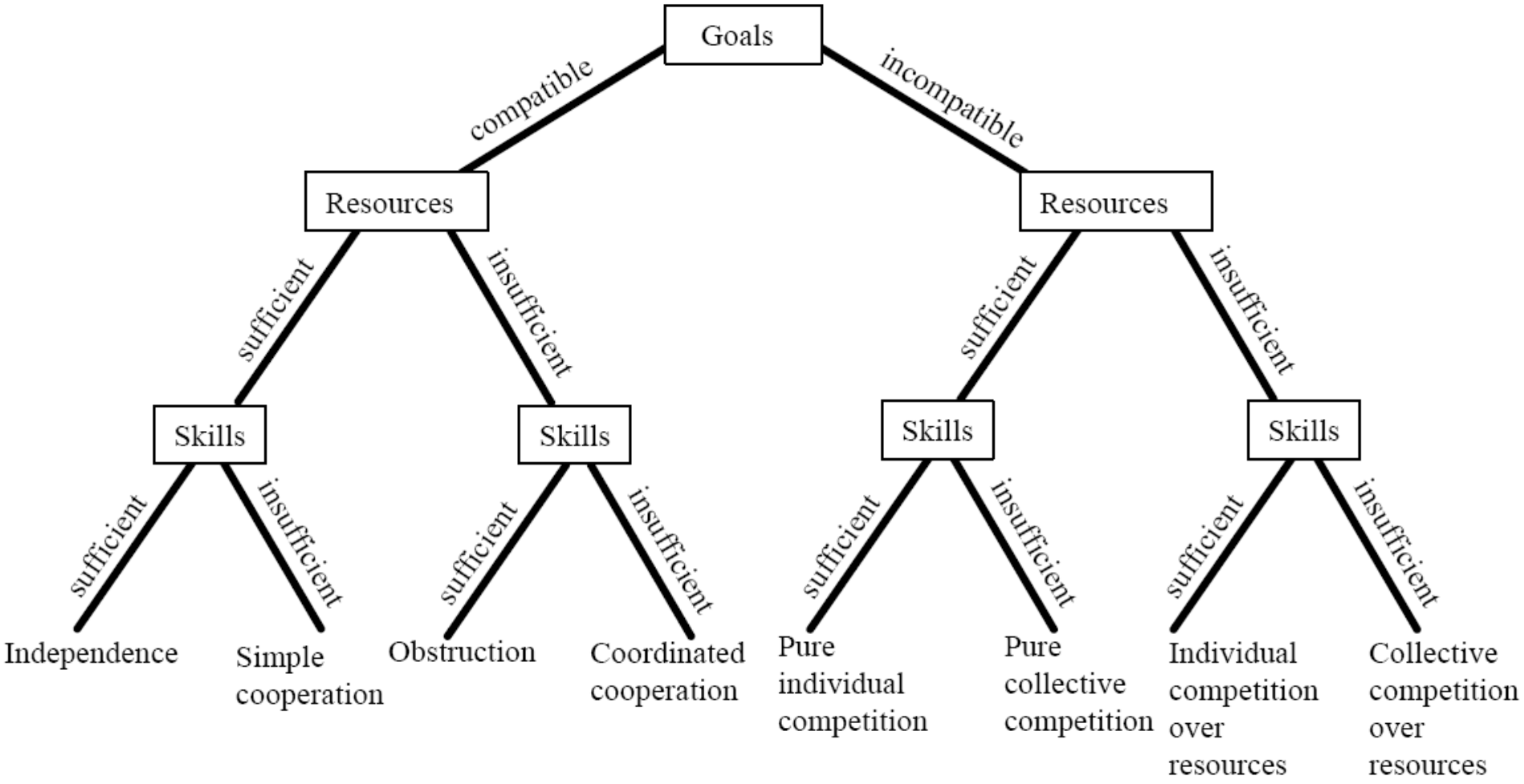
Elements of interactions

- **Goals**: The agents' objectives and goals are important. Goals of different agents can be conflicting
- **Resources**: agents require resources to achieve their objectives. Resources are not infinite, other agents may want to use them as well. Conflicts may arise
- **Expertise/skills/capabilities**. Agents may lack the necessary skills, expertise of capabilities for accomplishing one or more

Agent characterization

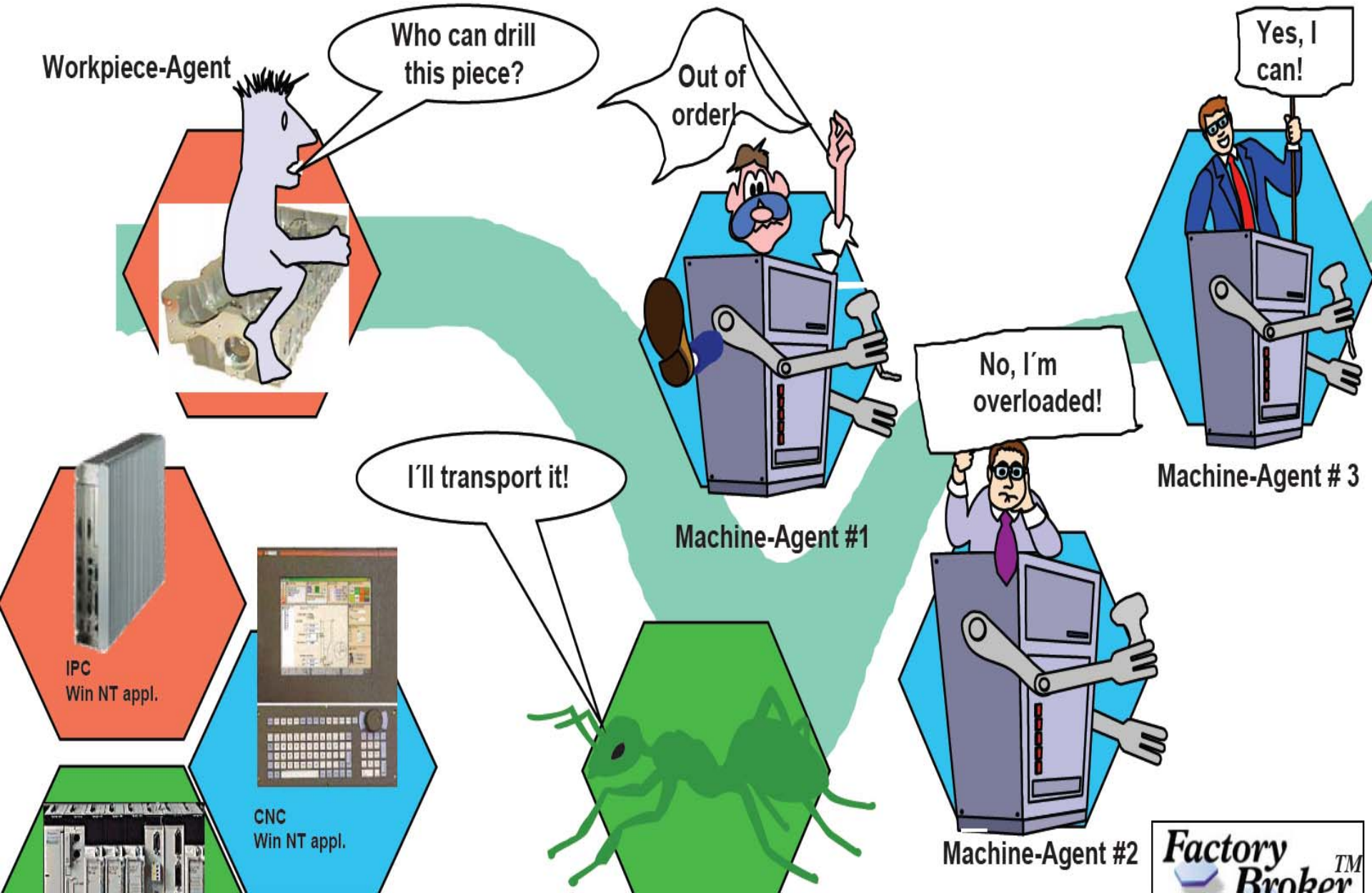
- Self-interested/antagonistic agents: have incompatible goals with others and they are interested in maximizing their own utility, not necessarily that of the society as a whole
- Cooperative/nonantagonistic agents: have usually compatible goals; they act to maximize their own utility in conjunction with that of the entire system

Modes of interaction



Collaborative Automation and Service Oriented Architectures in the Industry

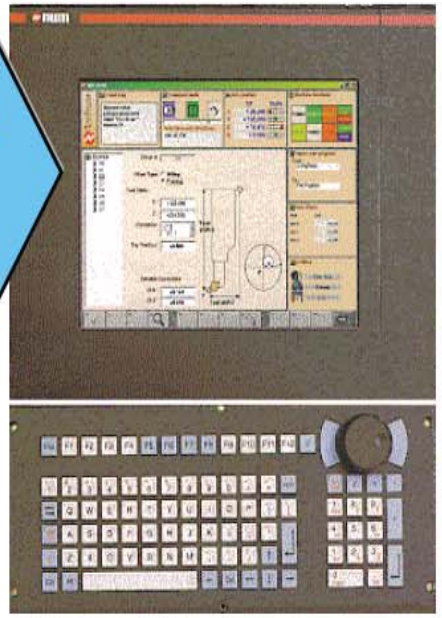
The FactoryBroker™ Solution: What is happening at the DaimlerChrysler shop floor?



Work-
piece-
Agent

Factory Broker™

Machine-
Agent

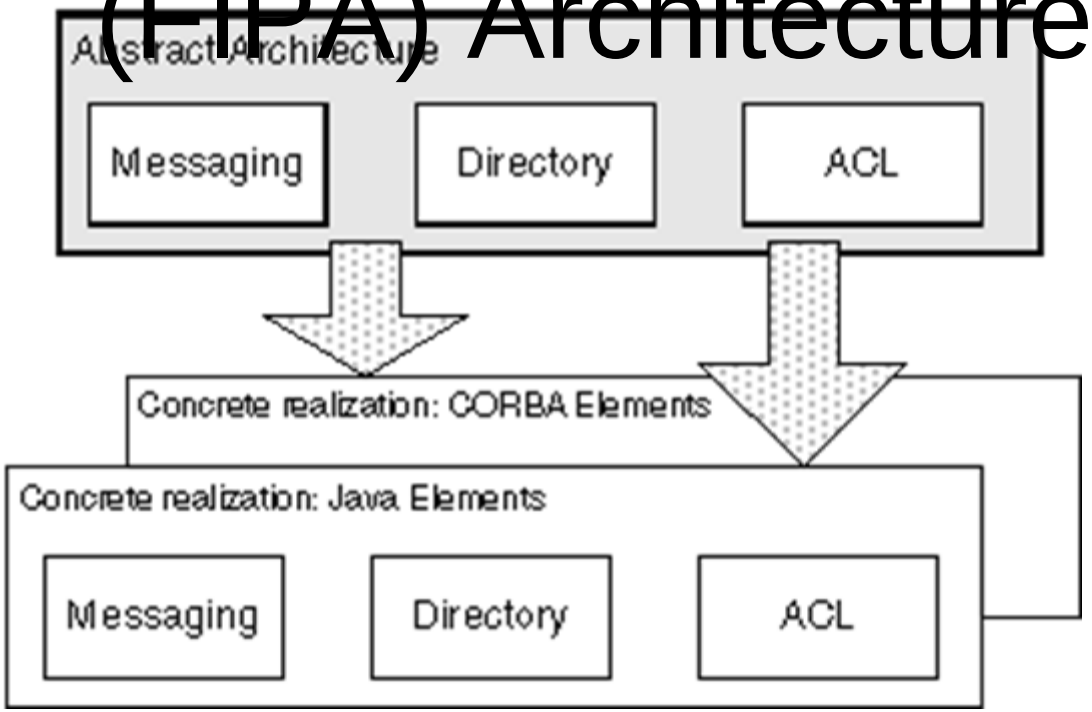


FIPA ACL

Transport
Agent

Fraunhofer
IPA
Agent system

Foundation for Intelligent Physical Agents (FIPA) Architecture



(ACL = Agent Communication Language)

- System Architecture
- **Agent Definitions**
- Knowledge Modeling
- Agent Class Structure

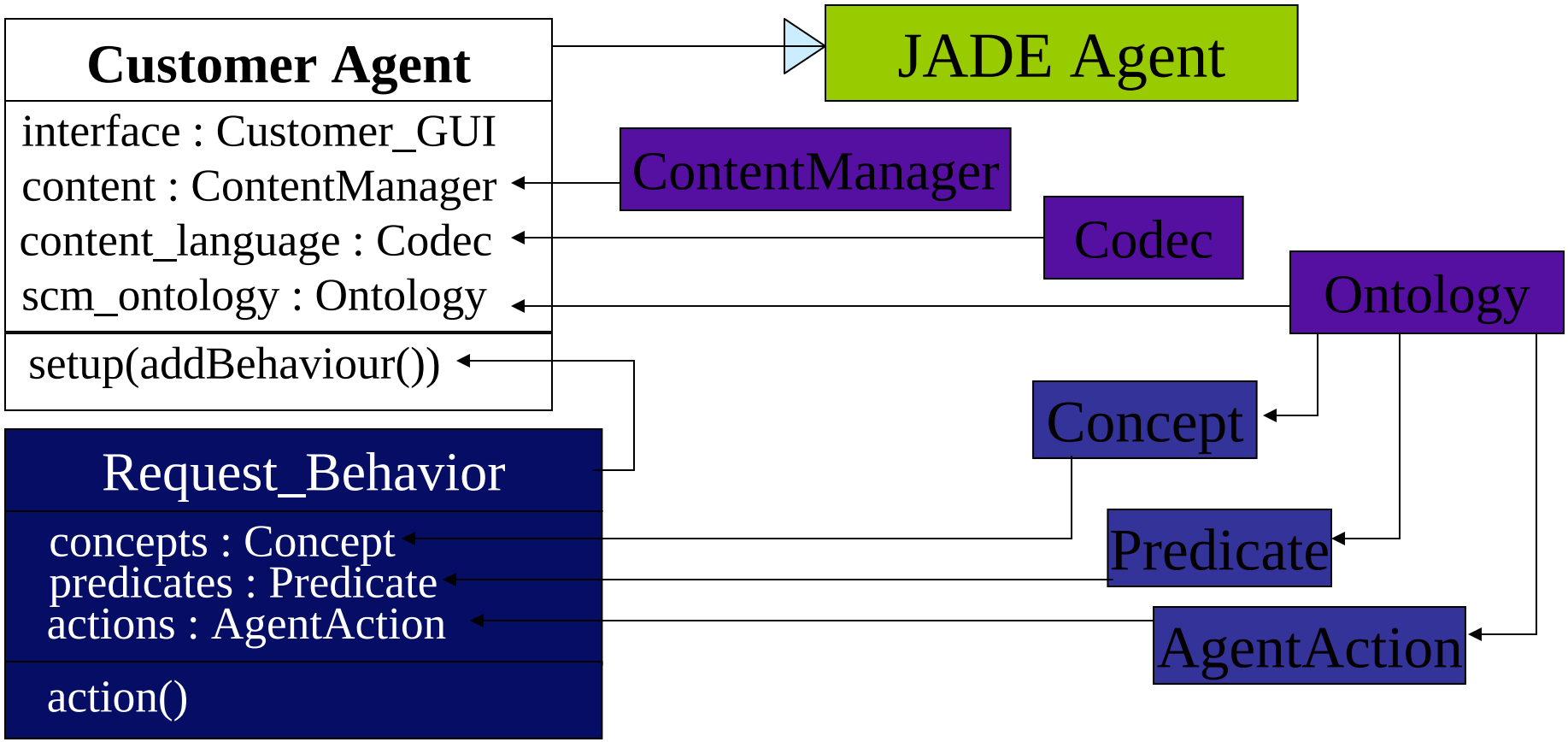
Agency Design

- The agents are defined and their responsibilities are presented.
- Example:

Role Schema:	**Manufacturer Agents**
Description: Manufacturers represent providers of components to the Assembly plant consumer of resources (parts) provided by suppliers	
Protocols: FIPA REQUEST and FIPA ITERATED CONTRACT NET	
Responsibilities: To respond to requests to supply the required components, to initiate request for needed parts, to transfer the required components, to receive and consume resources, to be aware of suppliers of parts and their capabilities, to participate in negotiations about the terms of acquiring and/or supplying a specified resource	

Agency Design

- System Architecture
- Agent Definitions
- Knowledge Modeling
- **Agent Class Structure**



Application Deployment

- **Ontology Creation**
- Inference Engine
- Agent Creation

The creation of Ontology implies :

- Define the schema for concepts, predicates, actions
- Create Java classes
- Selection of a content language
- Register the ontology and the content language to the agent

Deliver (Agent Action)				
Slots (Attributes)	**Description**	**Type**	**Presence**	**Cardinality**	**Restricted Values**
order	The order to be delivered	Order	Mandatory	single	
customer	Who places the order	AgentID	Mandatory	single	

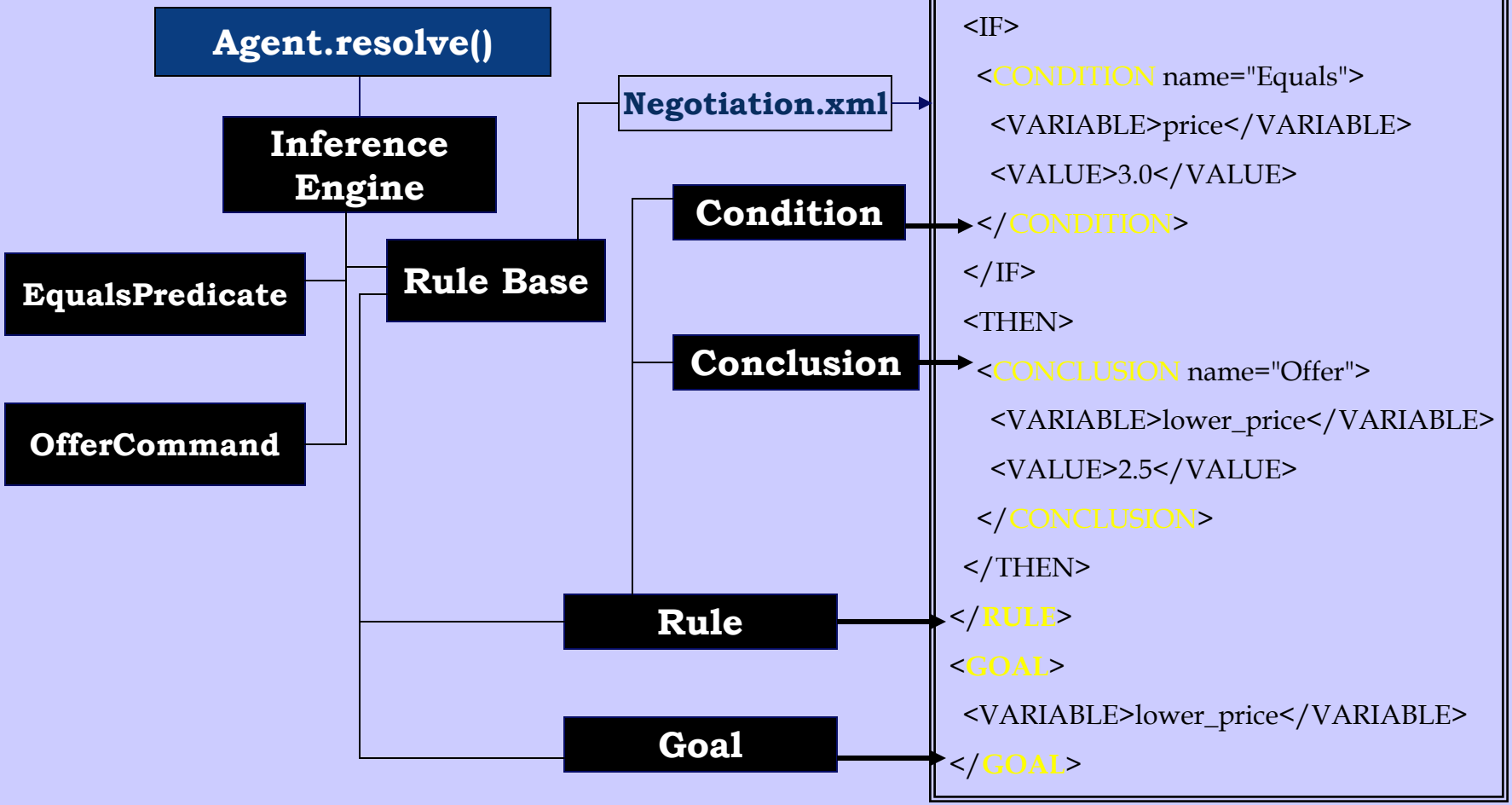
Application Deployment

- **Ontology Creation**
- Inference Engine
- Agent Creation

Order (Concept)				
Slots (Attributes)	**Description**	**Type**	**Presence**	**Cardinality**	**Restricted Values**
priority	Priority of the order	String	Optional	single	“HIGH” “LOW”
orderNo	Reference order number	integer	Mandatory	single	>0
ordered_products	A list of ordered products	List of Products	Mandatory	multiple	
destination	The destination of the order	String	Mandatory	single	
issueDate	The date when the order was placed	Date	Mandatory	single	

Application Deployment

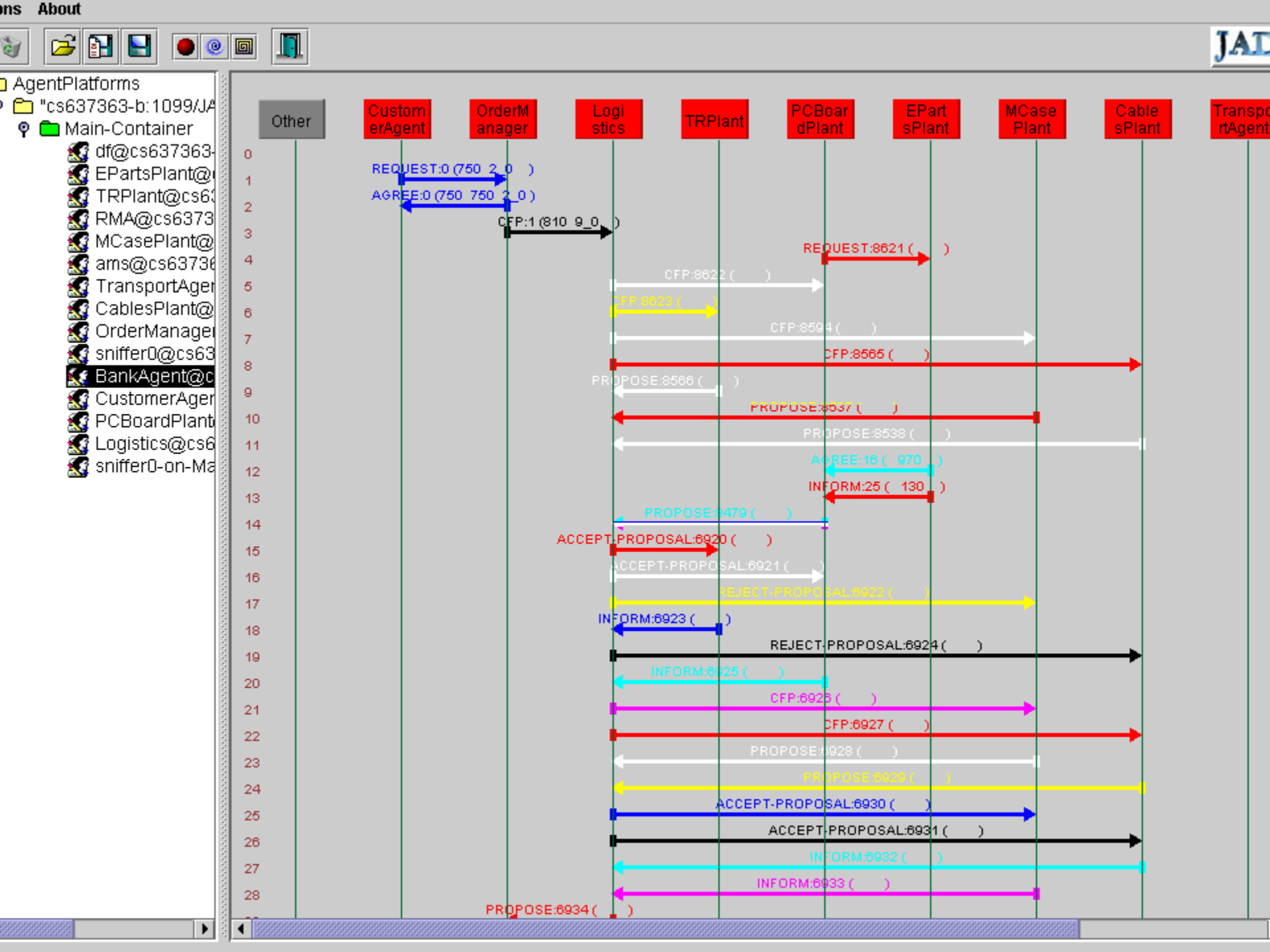
- Ontology Creation
- **Inference Engine**
- Agent Creation



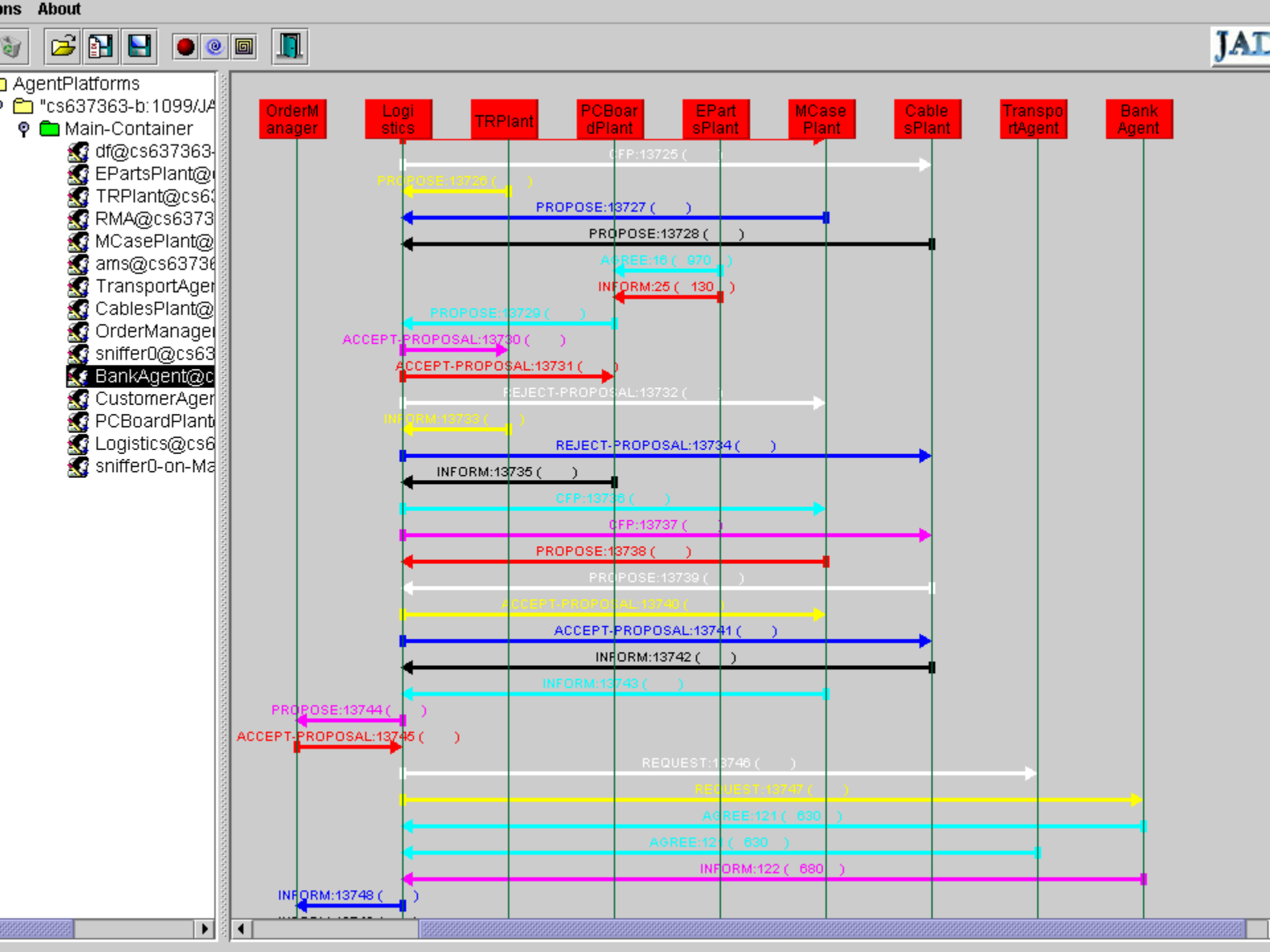
```
public class CustomerAgent extends Agent {
private ContentManager manager  = (ContentManager)getContentManager();
private Codec  codec    = new SLCodec();
private Ontology   ontology1 = ScmOntology.getInstance();
    protected void setup() {
                    manager.registerLanguage(codec);
                    manager.registerOntology(ontology1);
                    // Setting slots in ontology concepts
               Order ord = new Order();
               ord.setIssueDate(new Date(System.currentTimeMillis());
               Telephones tele1 = new Telephones();
               Answering_Machine am = new Answering_Machine();
               tele1.setTel_type(order_gui.telTypeField.getText());
                       ord.addProduct(tele1);
                       ord.addProduct(am);
                      addBehaviour(new RequestBehaviour(this, ord));
                            }//end of setup()
```

```
class RequestBehaviour extends OneShotBehaviour{
        public RequestBehaviour(Agent a, Order or) { super(a);}
public void action() {
// Prepare message
ACLMessage msg = new ACLMessage(ACLMessage.REQUEST);
msg.addReceiver(new AID("OrderManager", AID.ISLOCALNAME) );
msg.setLanguage(codec.getName());msg.setOntology(ontology1.getName());
// Fill content
Deliver del = new Deliver();
del.setCustomer(getAID());
del.setOrder(or);
Action a = new Action(); a.setAction(del);
ContentElementList cont = new ContentElementList();
cont.add(a); manager.fillContent(msg, cont);
//Send message
send(msg);   } }
```

ons About


AgentPlatforms
"cs637363-b:1099/JA
Main-Container
df@cs637363-
EPartsPlant@
TRPlant@cs6
RMA@cs6373
MCasePlant@
ams@cs6373
TransportAger
CablesPlant@
OrderManage
sniffer0@cs63
BankAgent@c
CustomerAger
PCBoardPlant
Logistics@cs6
sniffer0-on-Ma
Other
CustomerAgent
OrderManager
Logistics
TRPlant
PCBoardPlant
EPartsPlant
MCasePlant
CablesPlant
TransportAgent
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
REQUEST:0 (750 2 0)
AGREE:0 (750 750 2 0)
CFP:1 (810 9 0)
REQUEST:8621 ( )
CFP:8622 ( )
CFP:8623 ( )
CFP:8624 ( )
CFP:8565 ( )
PROPOSE:8566 ( )
PROPOSE:8537 ( )
PROPOSE:8538 ( )
AGREE:16 (970)
INFORM:25 (130)
PROPOSE:6479 ( )
ACCEPT-PROPOSAL:6920 ( )
ACCEPT-PROPOSAL:6921 ( )
REJECT-PROPOSAL:6922 ( )
INFORM:6923 ( )
REJECT-PROPOSAL:6924 ( )
INFORM:6925 ( )
CFP:6926 ( )
CFP:6927 ( )
PROPOSE:6928 ( )
PROPOSE:6929 ( )
ACCEPT-PROPOSAL:6930 ( )
ACCEPT-PROPOSAL:6931 ( )
INFORM:6932 ( )
INFORM:6933 ( )
PROPOSE:6934 ( )

ons About


AgentPlatforms
"cs637363-b:1099/JA
Main-Container
df@cs637363-
EPartsPlant@
TRPlant@cs6
RMA@cs6373
MCasePlant@
ams@cs6373
TransportAger
CablesPlant@
OrderManage
sniffer0@cs63
BankAgent@c
CustomerAger
PCBoardPlant
Logistics@cs6
sniffer0-on-Ma
OrderManager
Logistics
TRPlant
PCBoardPlant
EPartsPlant
MCasePlant
CablesPlant
TransportAgent
BankAgent
CFP:13725 (
PROPOSE:13726 (
PROPOSE:13727 ( )
PROPOSE:13728 ( )
AGREE:16 (970)
INFORM:25 (130)
PROPOSE:13729 ( )
ACCEPT-PROPOSAL:13730 ( )
ACCEPT-PROPOSAL:13731 ( )
REJECT-PROPOSAL:13732 (
INFORM:13733 ( )
REJECT-PROPOSAL:13734 ( )
INFORM:13735 ( )
CFP:13736 ( )
CFP:13737 ( )
PROPOSE:13738 ( )
PROPOSE:13739 ( )
ACCEPT-PROPOSAL:13740 ( )
ACCEPT-PROPOSAL:13741 ( )
INFORM:13742 ( )
INFORM:13743 ( )
PROPOSE:13744 ( )
ACCEPT-PROPOSAL:13745 ( )
REQUEST:13746 ( )
REQUEST:13747 ( )
AGREE:121 (630)
AGREE:121 (630)
INFORM:122 (680)
INFORM:13748 ( )