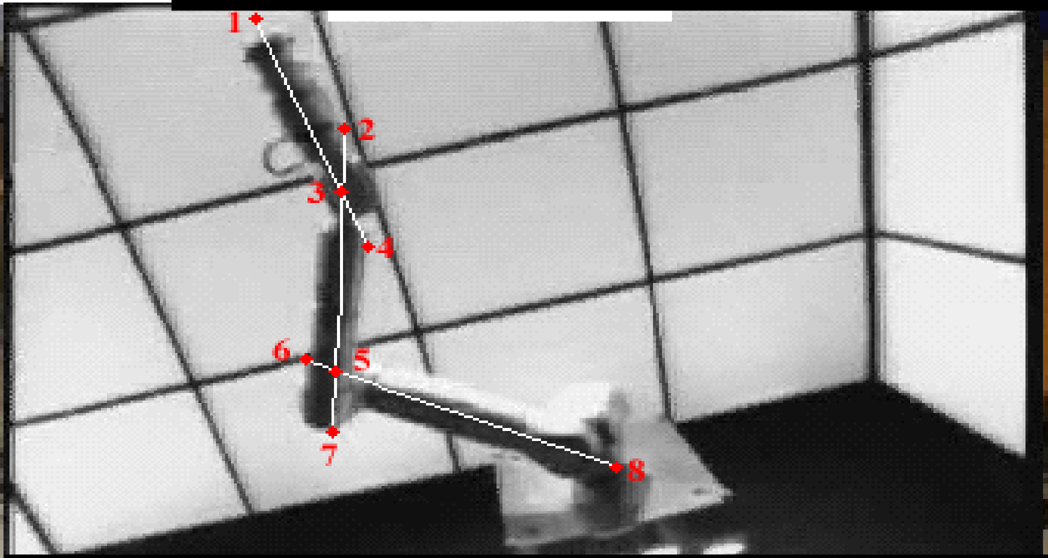
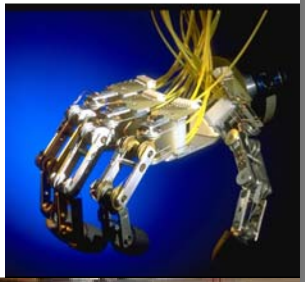
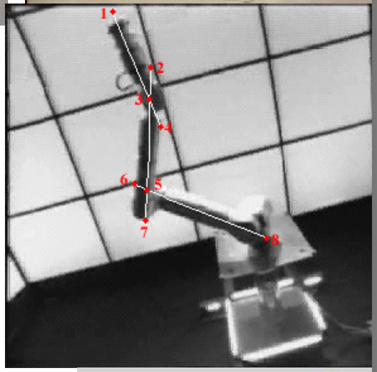
PASSI (Process for Agent Societies Specification and Implementation)



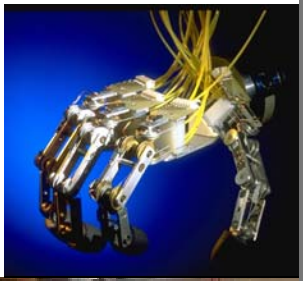
Massimo Cossentino (ICAR-Italian National Research Council)
cossentino@pa.icar.cnr.it

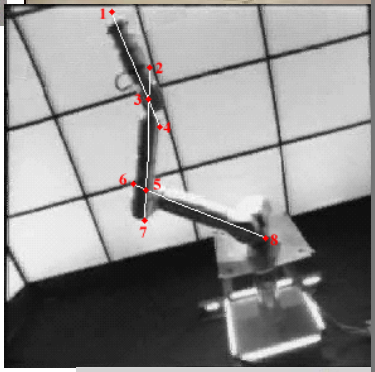




Characteristics of PASSI

- PASSI is a step-by-step requirements-to-code method for developing multi-agent software that integrates design models and philosophies from both object-oriented software engineering and MAS using UML notation
- PASSI refers to the most diffused standards: (A)UML, FIPA, JAVA, RDF, Rational Rose
- PASSI is conceived to be supported by **PTK**, an agent-oriented CASE tool
  - The functionalities of PTK include:
    - Automatic (total or partial) compilation of some diagrams
    - Automatic support to the execution of recurrent operations
    - Check of design consistency
    - Automatic compilation of reports and design documents
    - **Access to a database of patterns**
    - Generation of code and Reverse Engineering
- Ontology design (and its actual Java implementation) has a central role in the process
- Reuse is performed through patterns

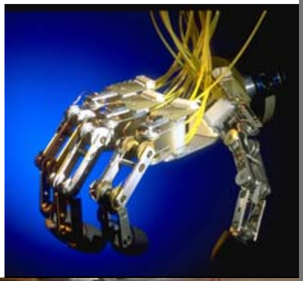


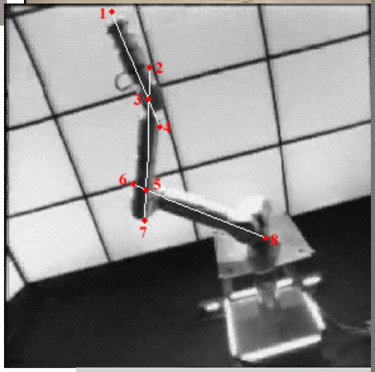


PASSI: process and language

PASSI is composed of a complete design process and a modeling language

- The design process is *incremental* and *iterative*
- The modeling language is an extension of UML. It will evolve towards the results of the FIPA AUML standardization process





Patterns of agents

We consider a pattern of agent as composed of its design level description and the corresponding JAVA code.

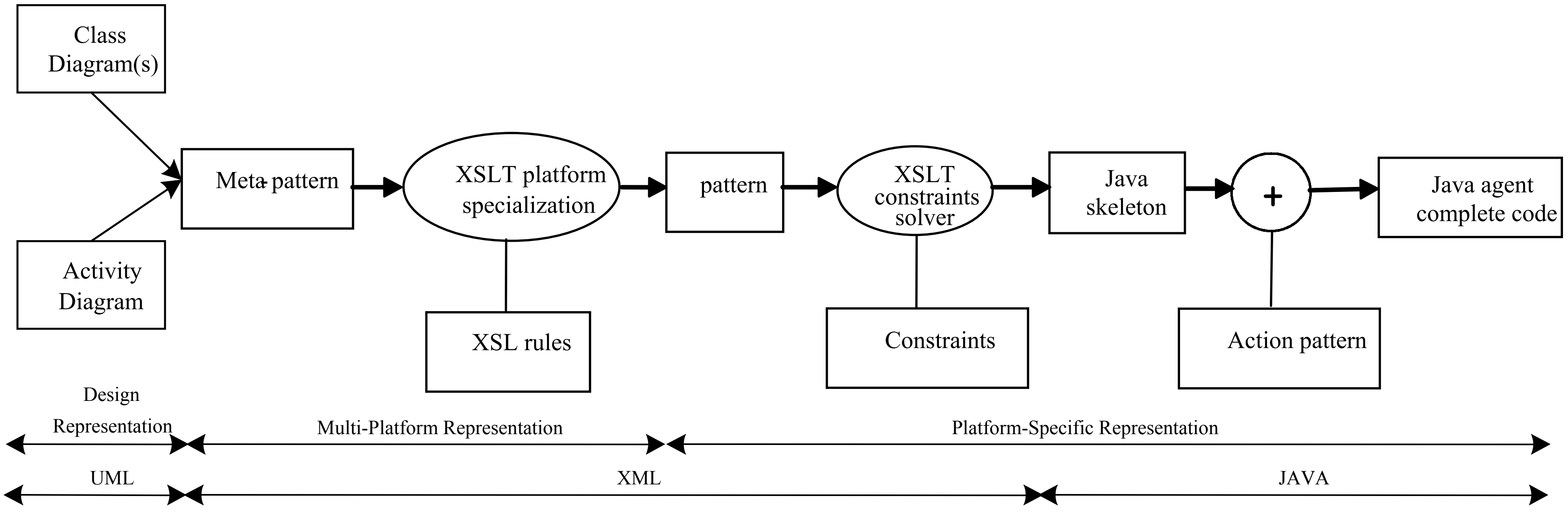
Our patterns are multi-platforms: they can be used in both our supported agent platforms

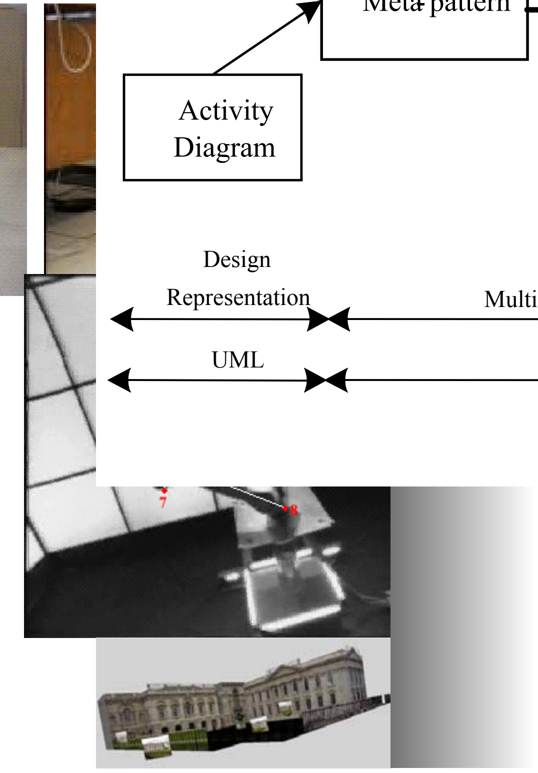
More in detail each pattern is composed of:

- A structure
  - Usually a base agent class and a set of task/behavior classes.
  - Described using UML class diagrams
- A behavior
  - Expressed by the agent using its structural elements
  - Detailed in UML dynamic diagrams (activity/state chart diagrams)
- A portion of code
  - Lines of code implementing the structure and behavior described in the previous diagram

Patterns of agents

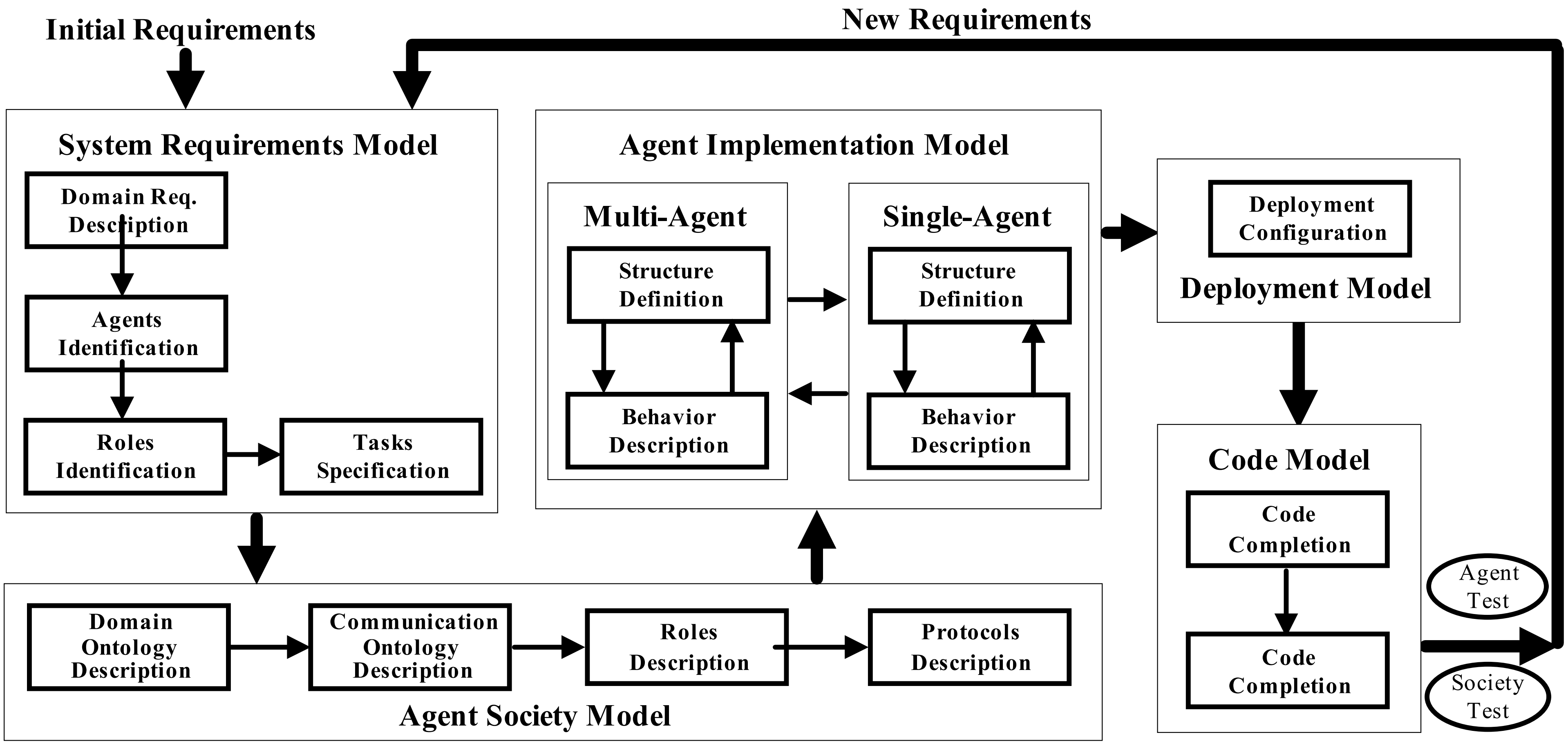
From design diagrams to the JAVA code

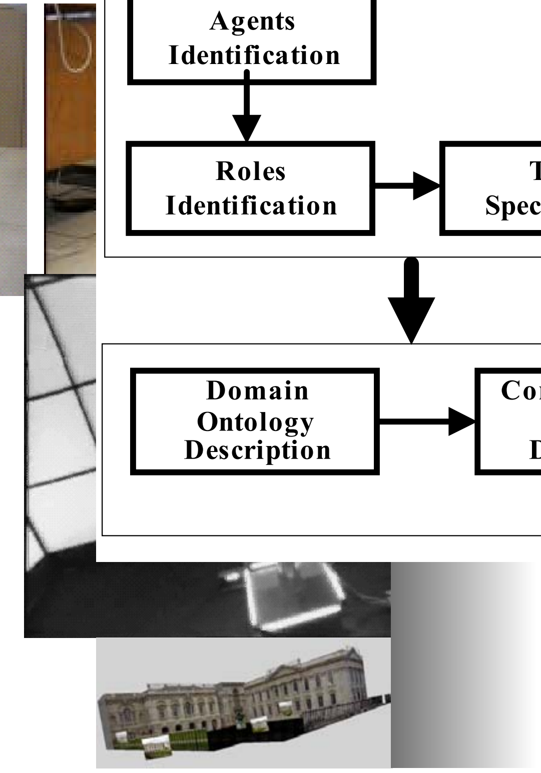


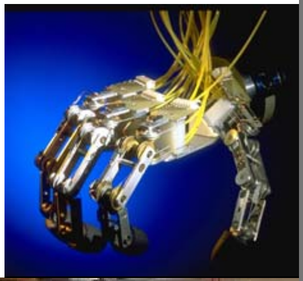


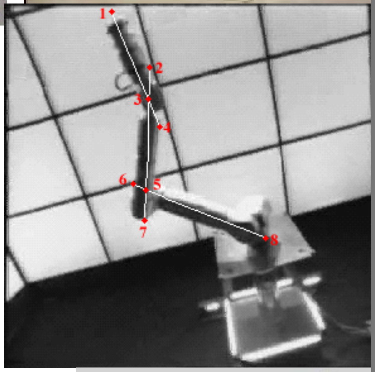
PASSI

(Process for Agent Societies Specification and Implementation)





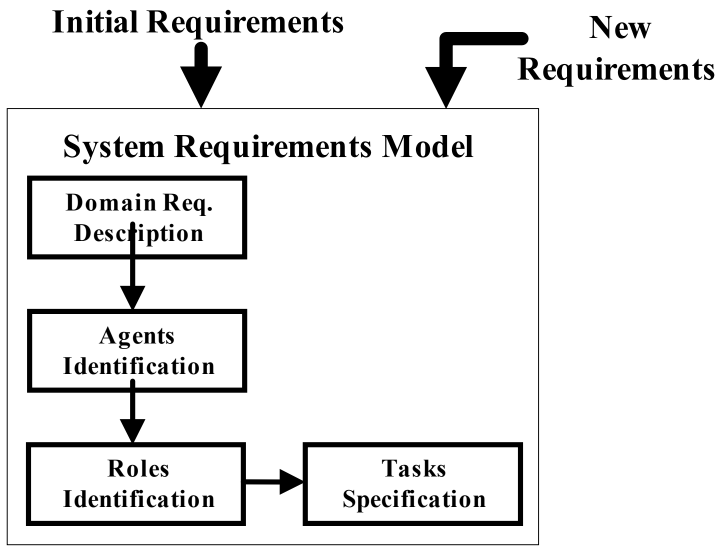


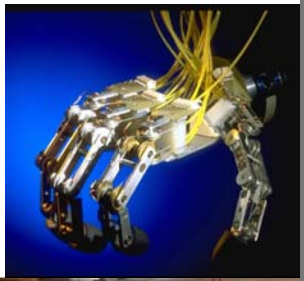


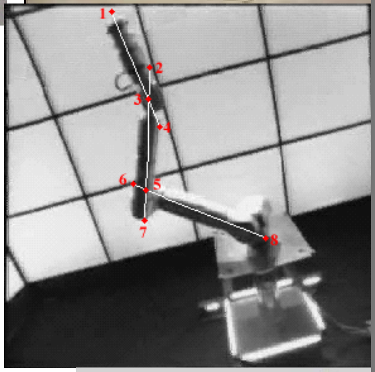
The System Requirements Model

It describes:

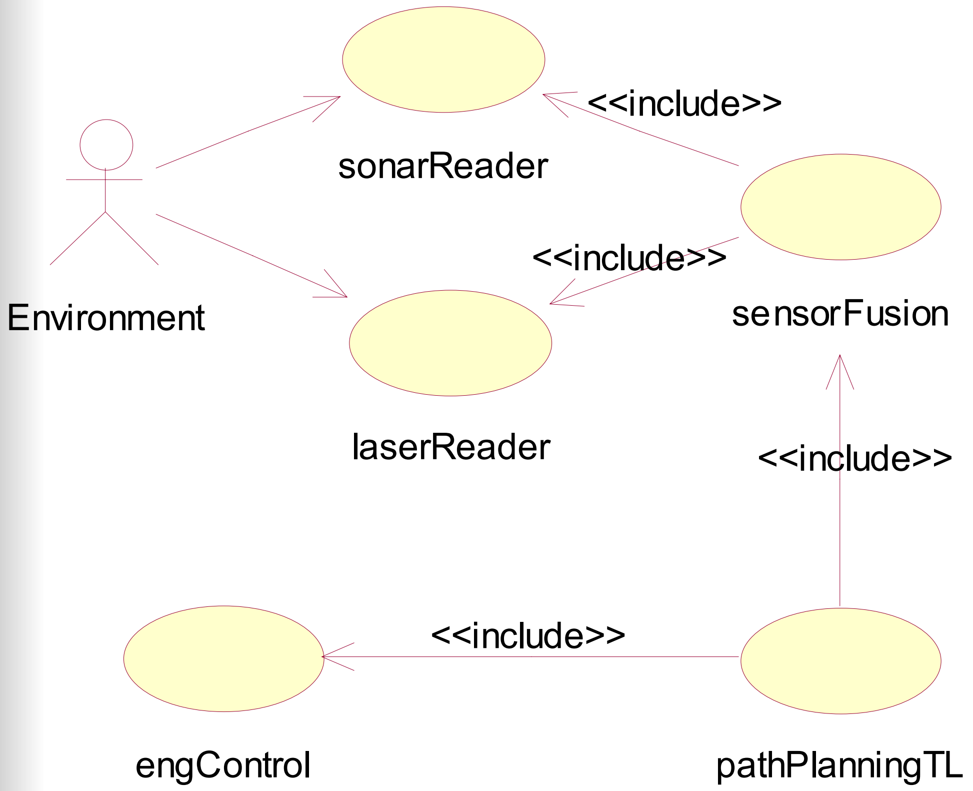
- System requirements
- Agents functionalities
- Roles played by agents in accomplishing their duties
- Tasks performed by agents in their roles



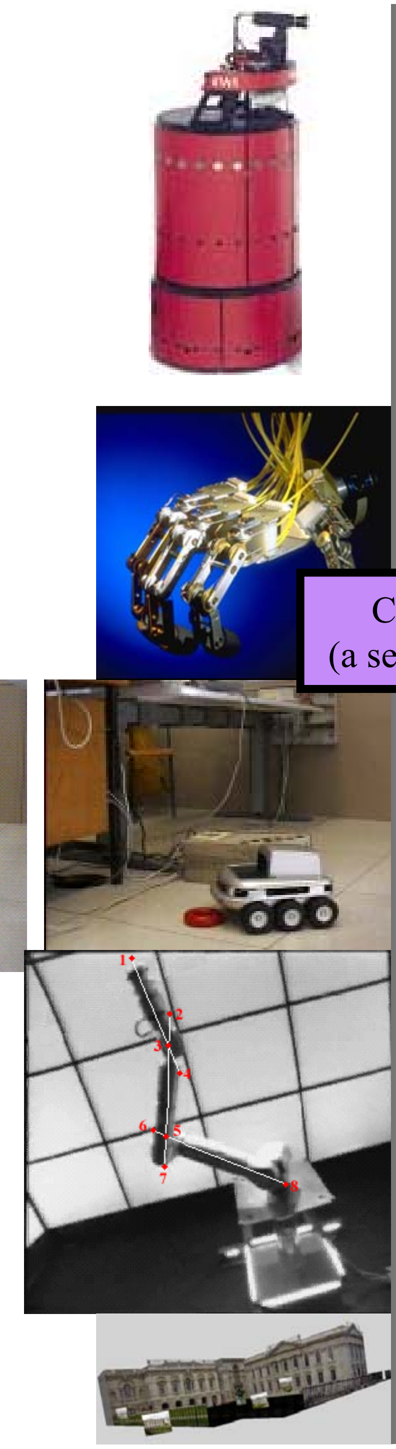




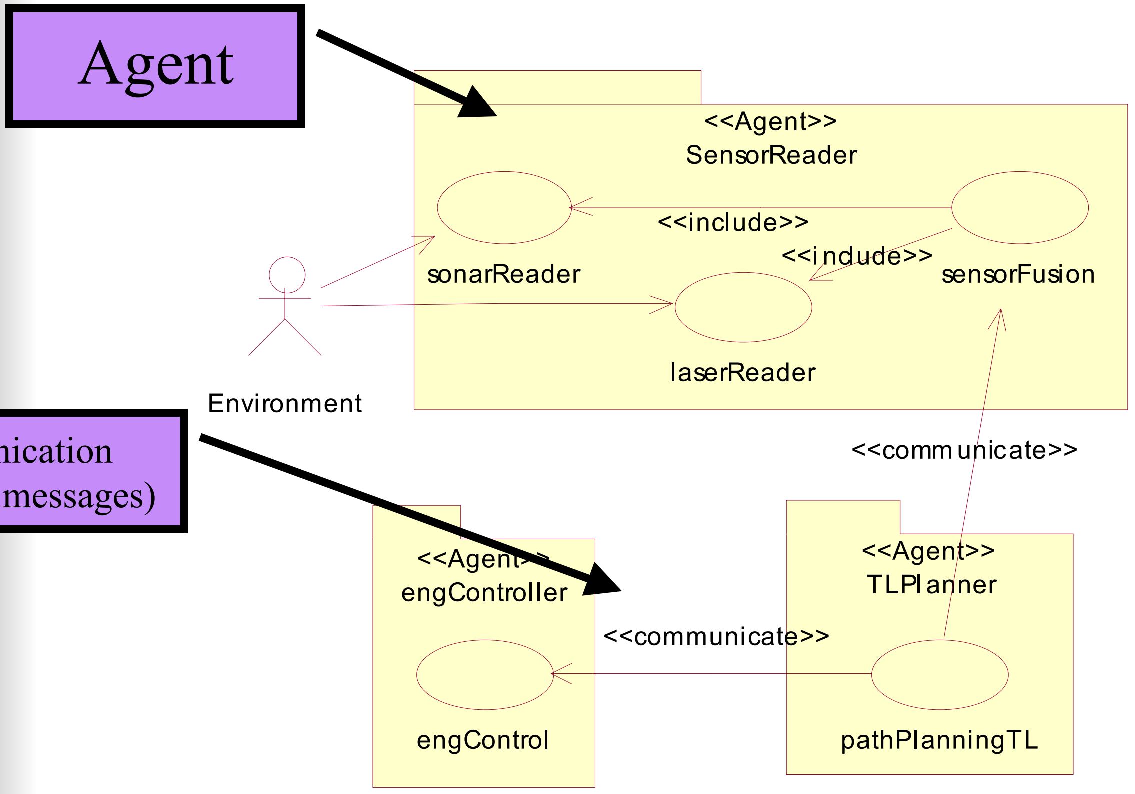
Domain Description Phase



A functional description of the system with conventional use case diagrams.

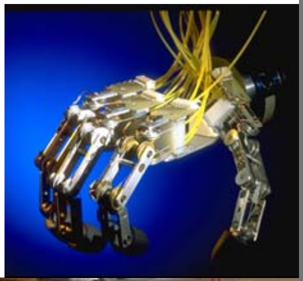


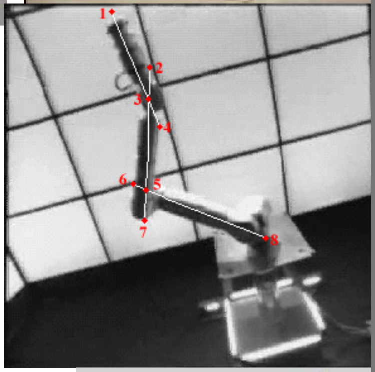
Agents Identification Phase



Packages are used in the previous diagrams to divide the functionalities into different agents.

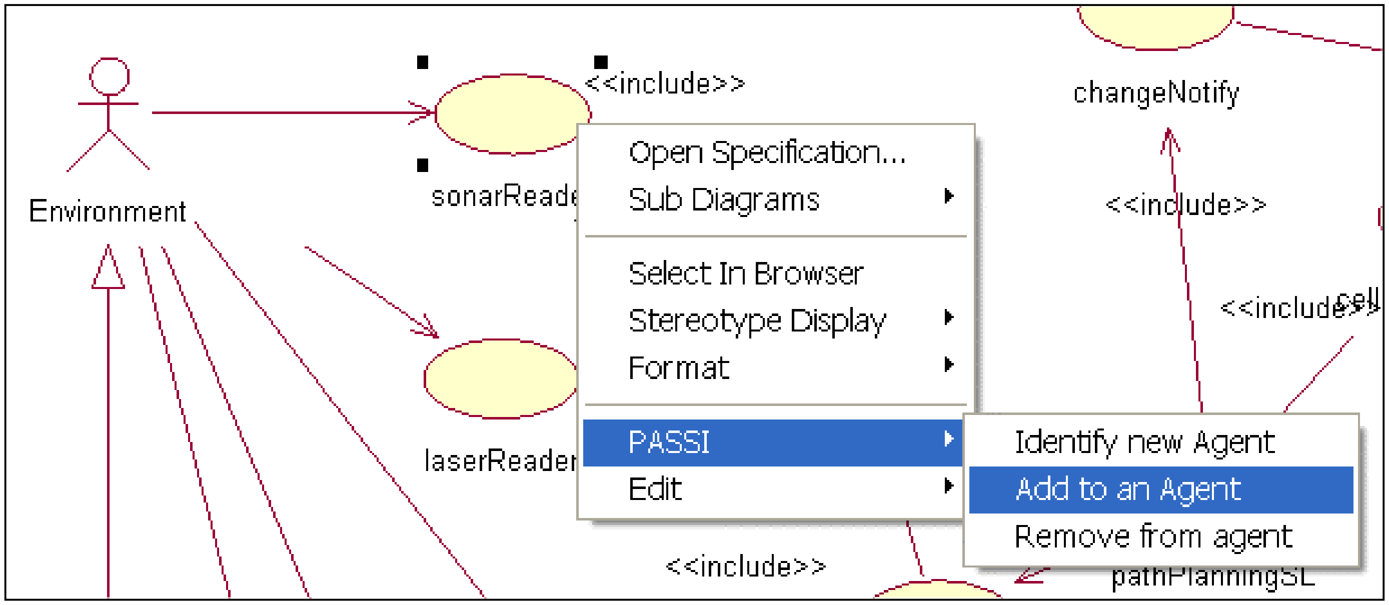
Relationships among different agents are characterized by a "communication" stereotype



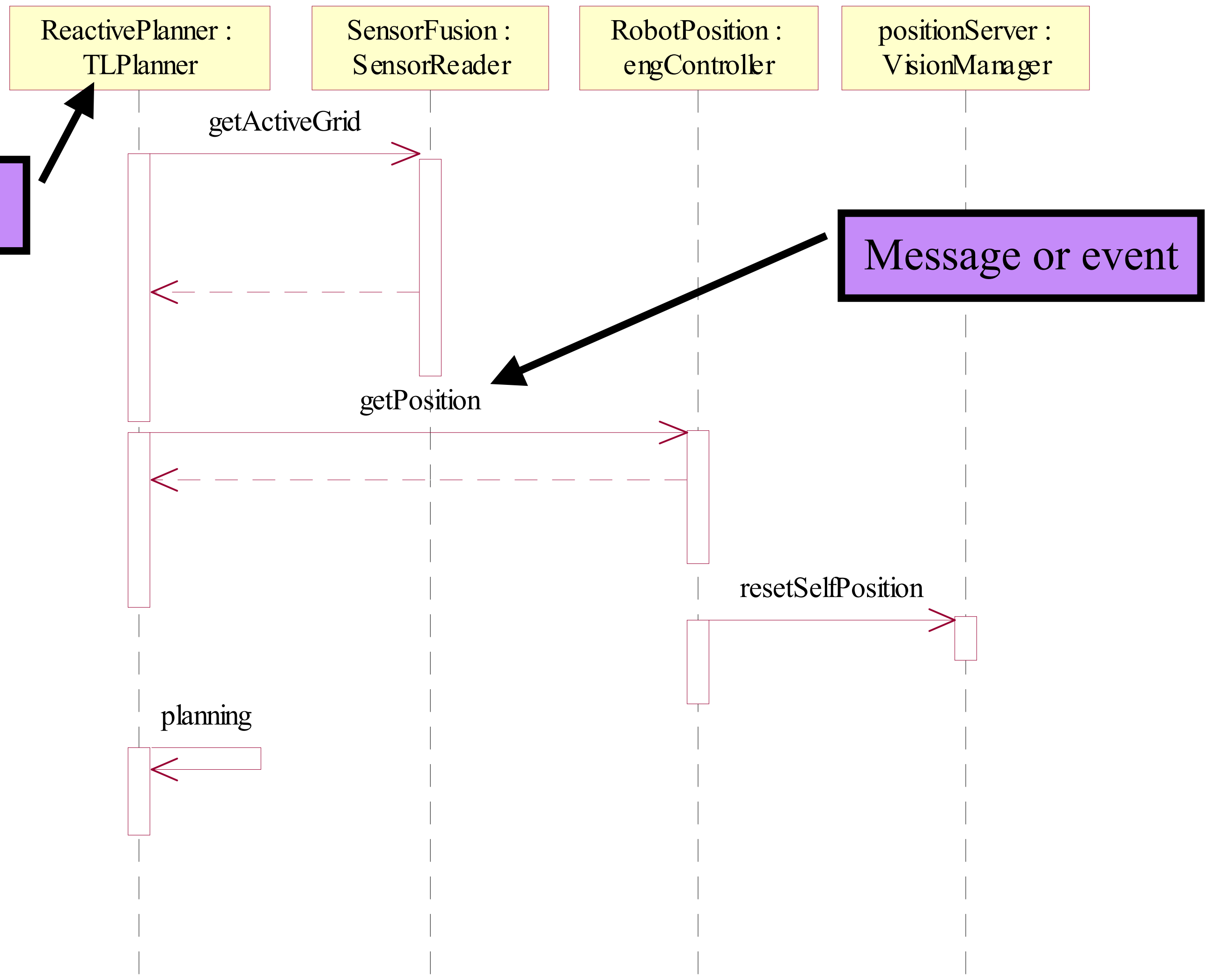


PTK support

- The A.Id. diagram is automatically composed by the tool
- The designer creates new agents and select their use cases operating in the D.D. diagram

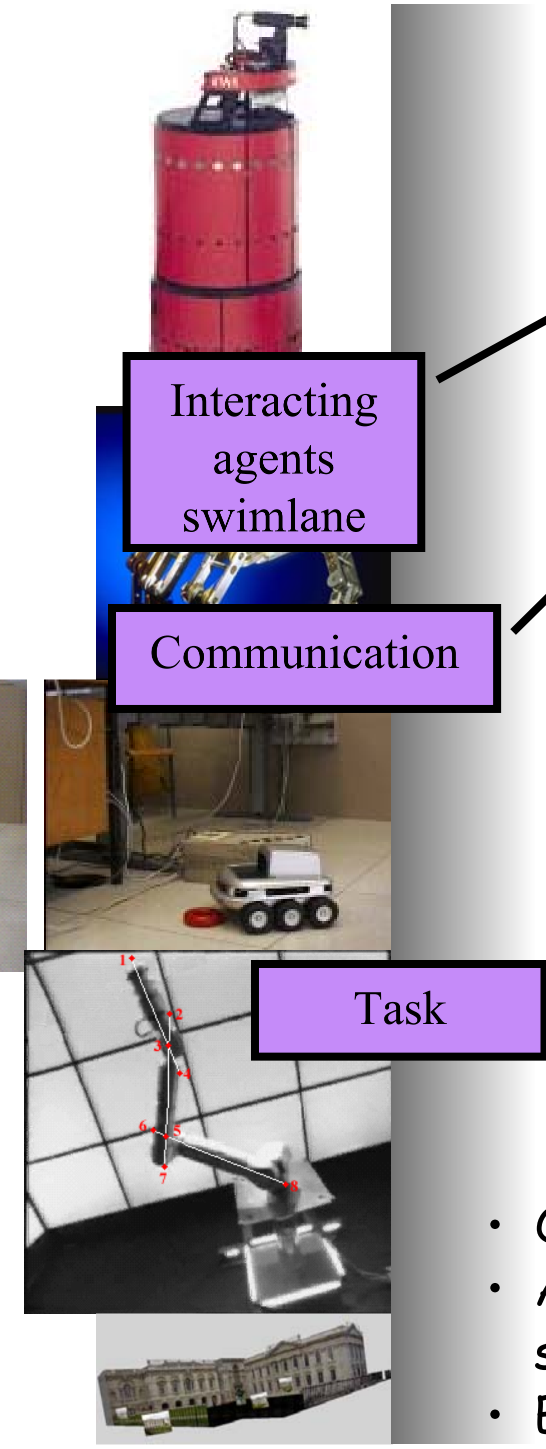


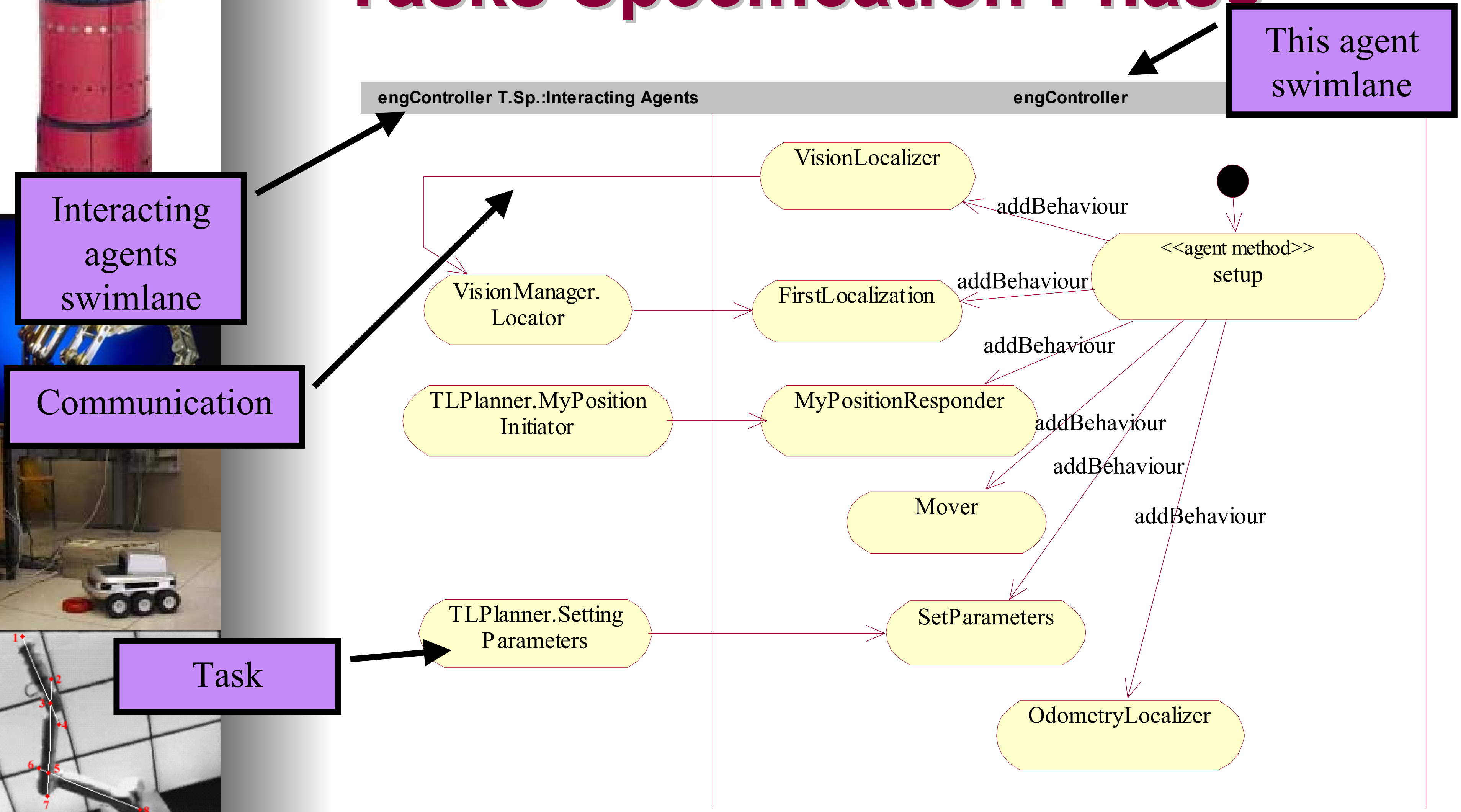
Roles Identification Phase


ReactivePlanner : TLPlanner
SensorFusion : SensorReader
RobotPosition : engController
positionServer : VisionManager
<Role> : <agent name>
getActiveGrid
Message or event
getPosition
resetSelfPosition
planning

Scenarios coming from UC diagram are used to identify agents’ roles

Tasks Specification Phase

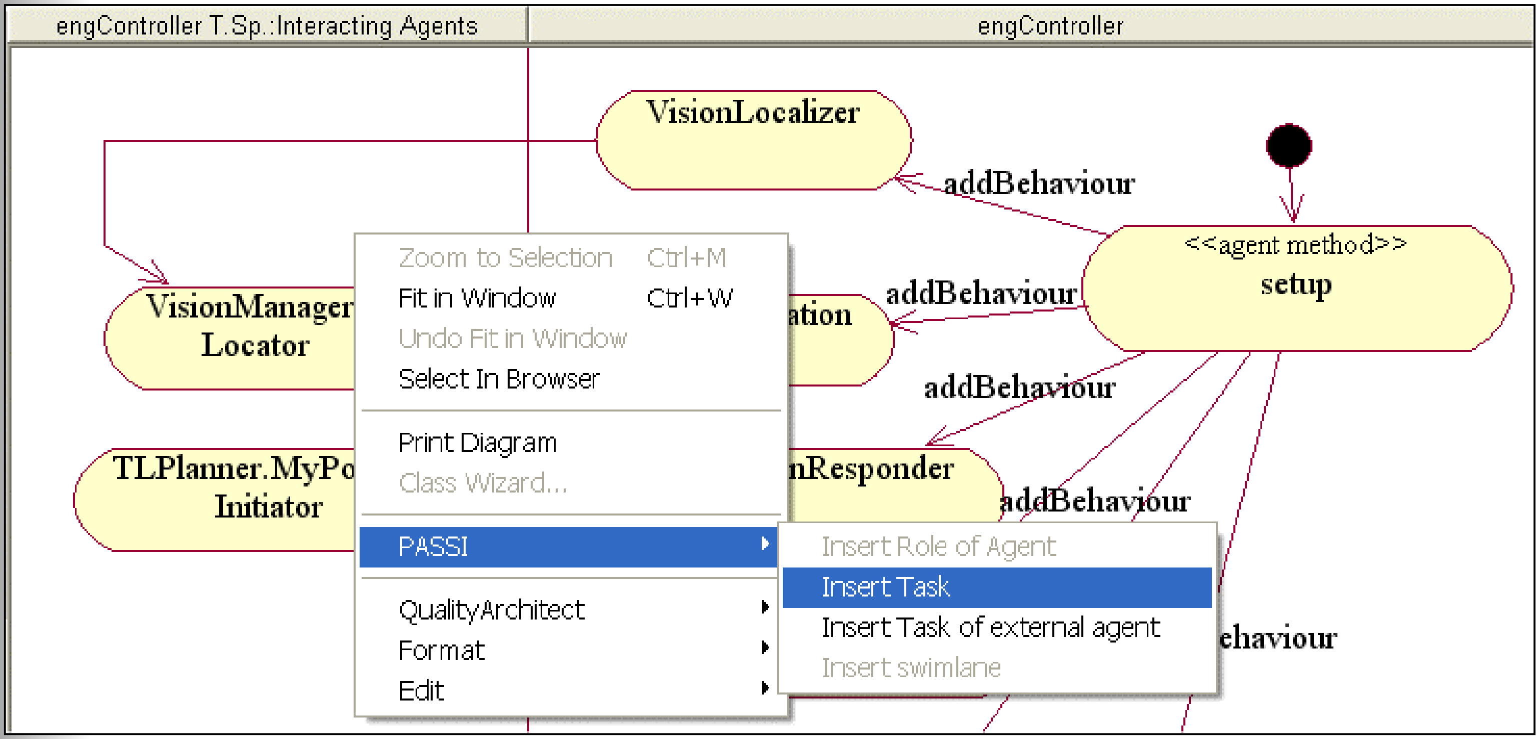


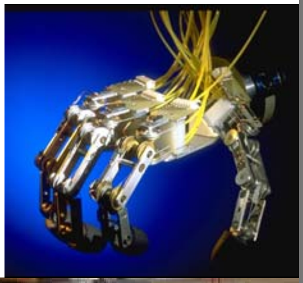


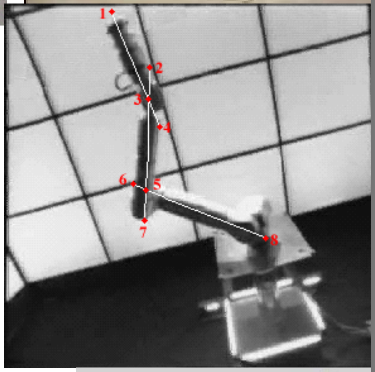
- One diagram for each agent
- A complete description of the agent's behavior in terms of state/activity machine
- Each activity represents a task (or a method) of the agent

PTK support

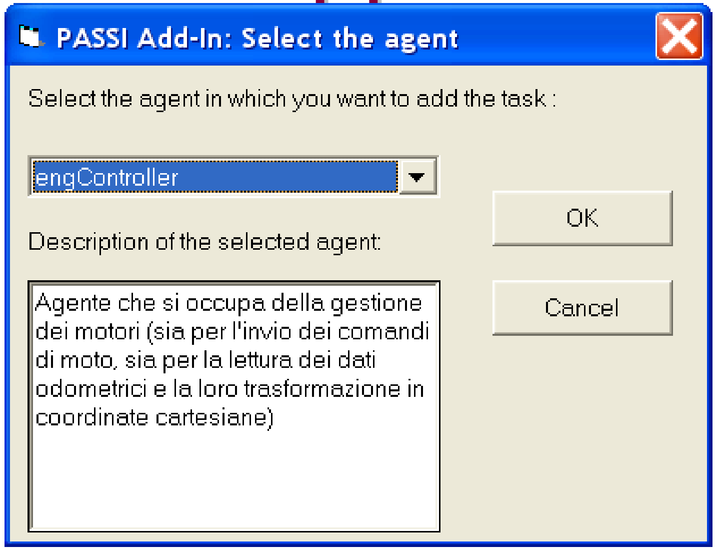
Tasks are introduced by the designer in the T.Sp. diagram and they are automatically reported in the structural diagrams







PTK support



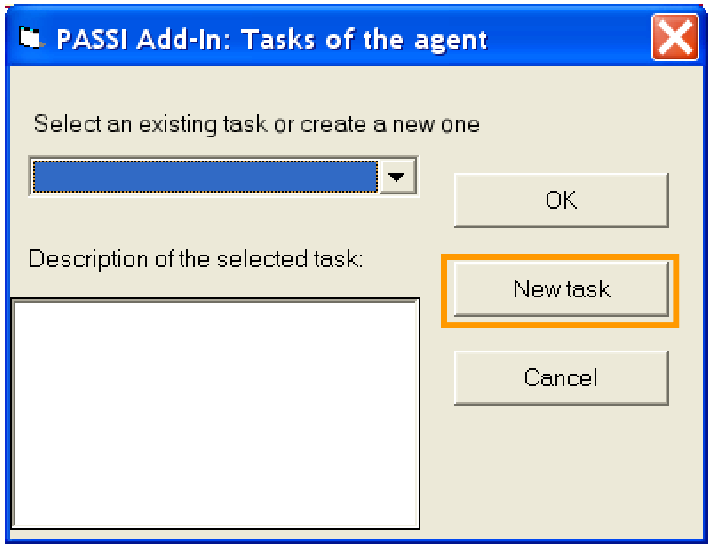
PASSI Add-In: Select the agent

Select the agent in which you want to add the task :

engController

Description of the selected agent:

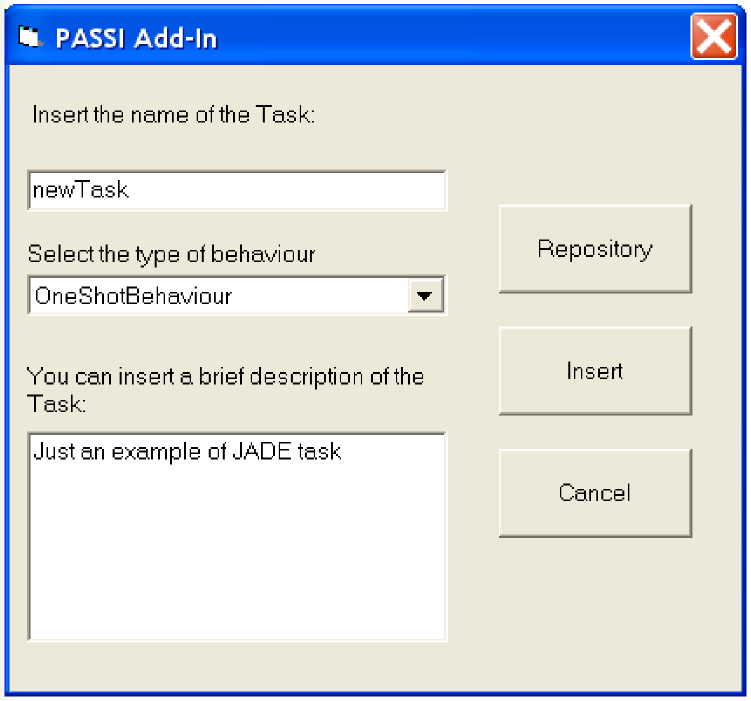
Agente che si occupa della gestione dei motori (sia per l'invio dei comandi di moto, sia per la lettura dei dati odometrici e la loro trasformazione in coordinate cartesiane)



PASSI Add-In: Tasks of the agent

Select an existing task or create a new one

Description of the selected task:



PASSI Add-In

Insert the name of the Task:

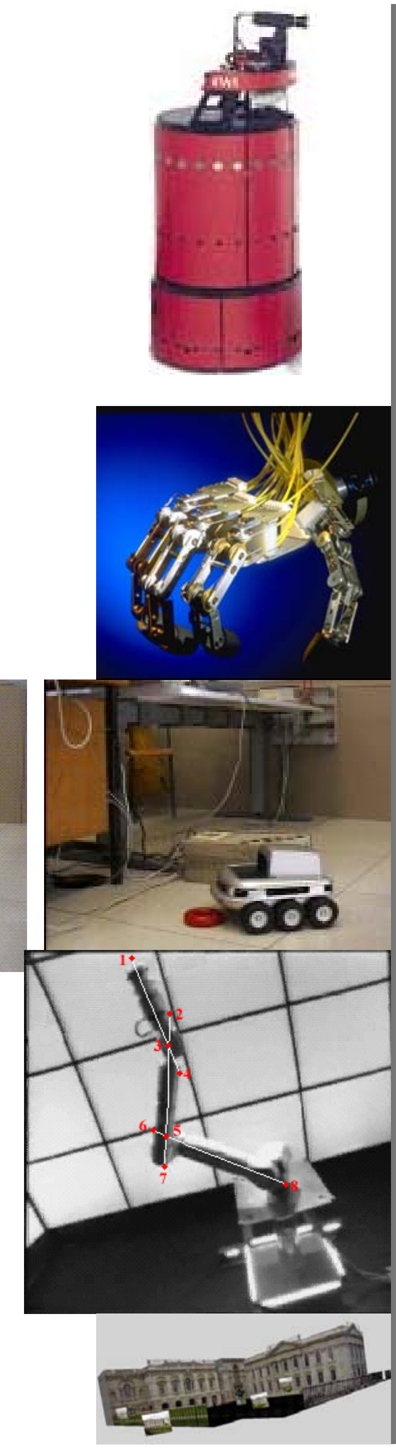
newTask

Select the type of behaviour

OneShotBehaviour

You can insert a brief description of the Task:

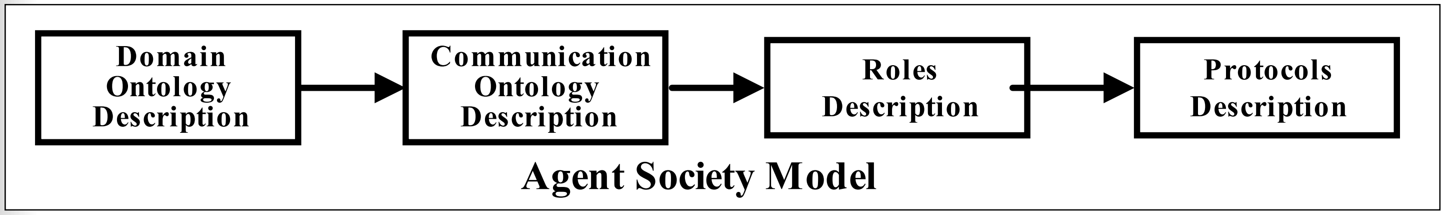
Just an example of JADE task

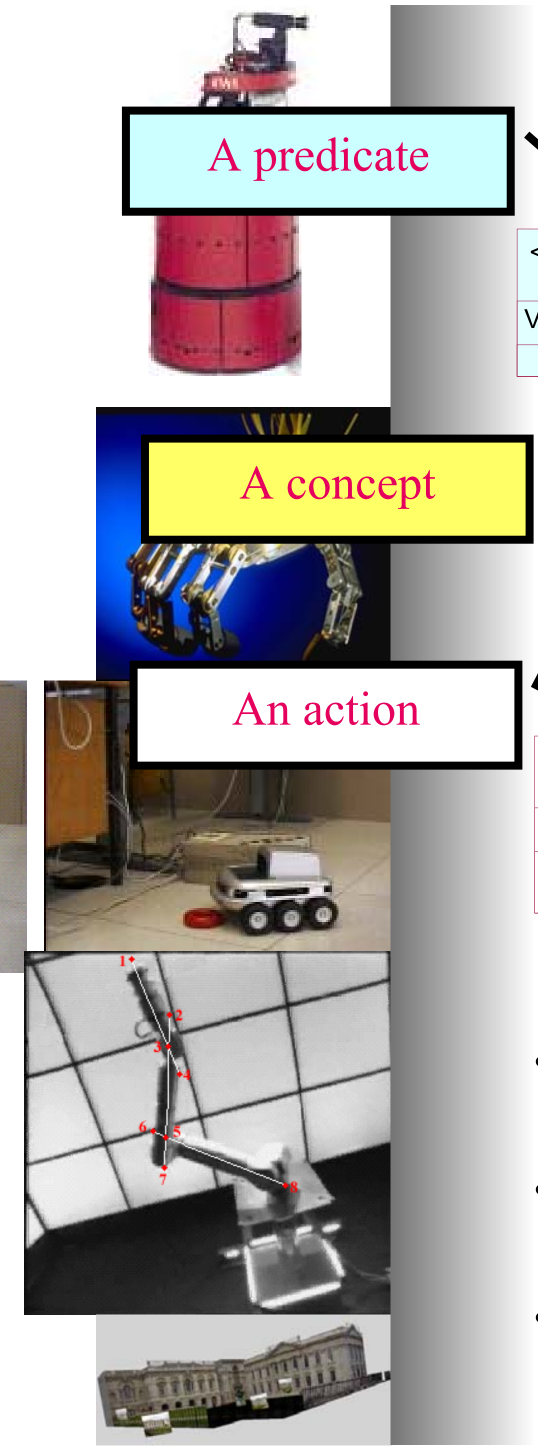


The Agent Society Model

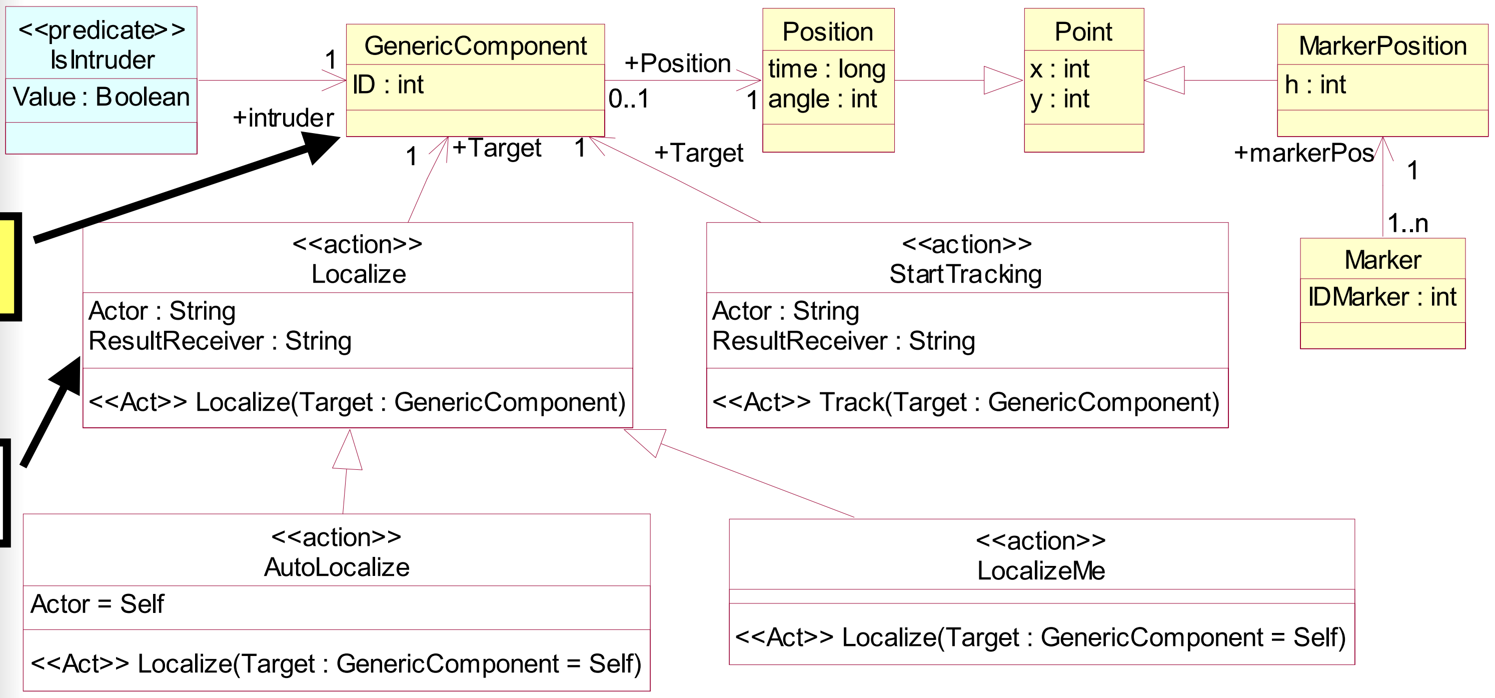
It includes the description of:

- Ontology of the system domain
- Ontology of inter-agents communications
- Services offered by agents
- Agents' communications (in terms of ontology, agent interaction protocol and content language)
- Agent interaction protocols


Domain Ontology Description
Communication Ontology Description
Roles Description
Protocols Description
Agent Society Model



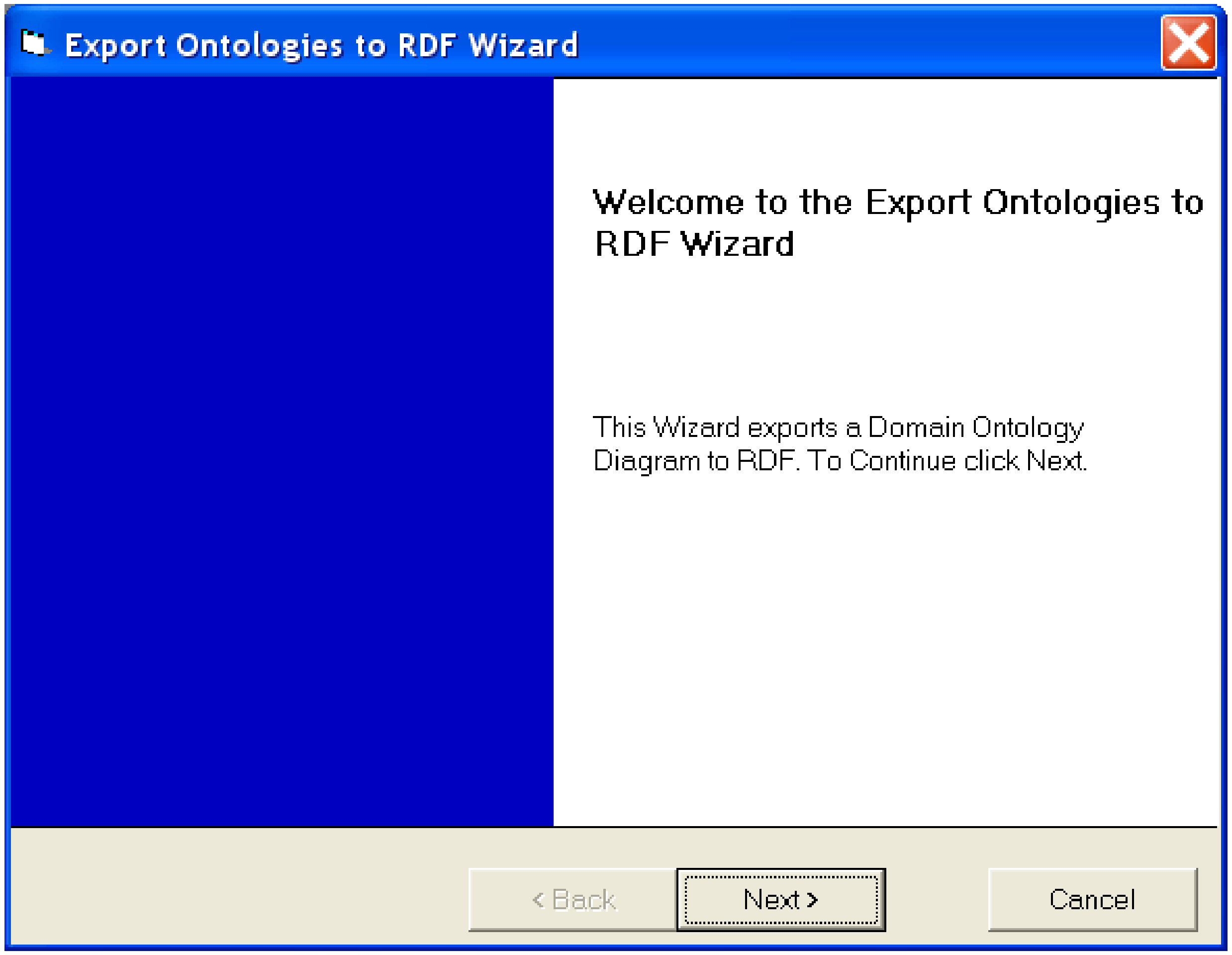
Domain Ontology Description Phase



- We use concepts, predicates and actions to model the ontology of the domain
- We can have aggregation, association and generalization relationships
- From this diagram we automatically obtain an RDF description of the ontology

PTK support

Starting from this diagram, PTK exports the RDF description of the ontology

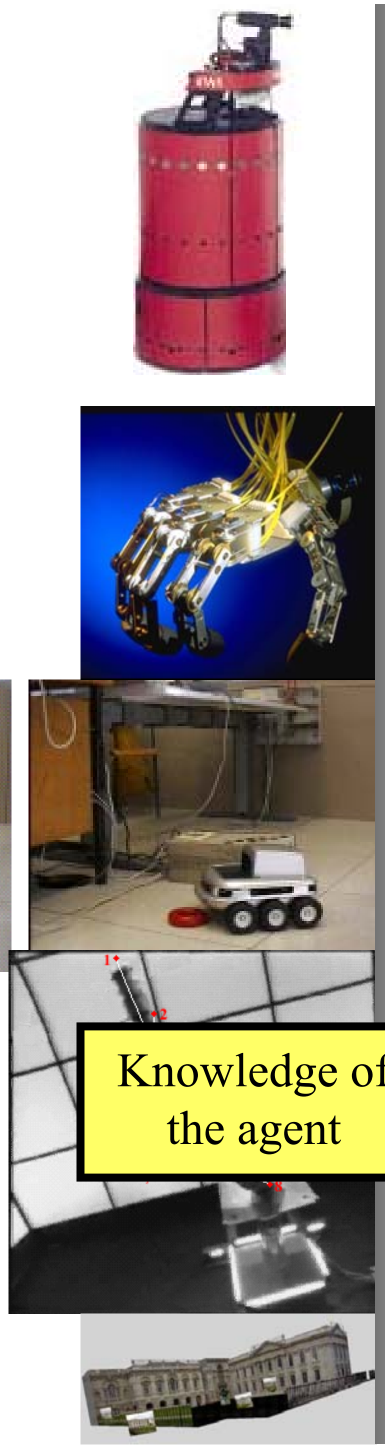


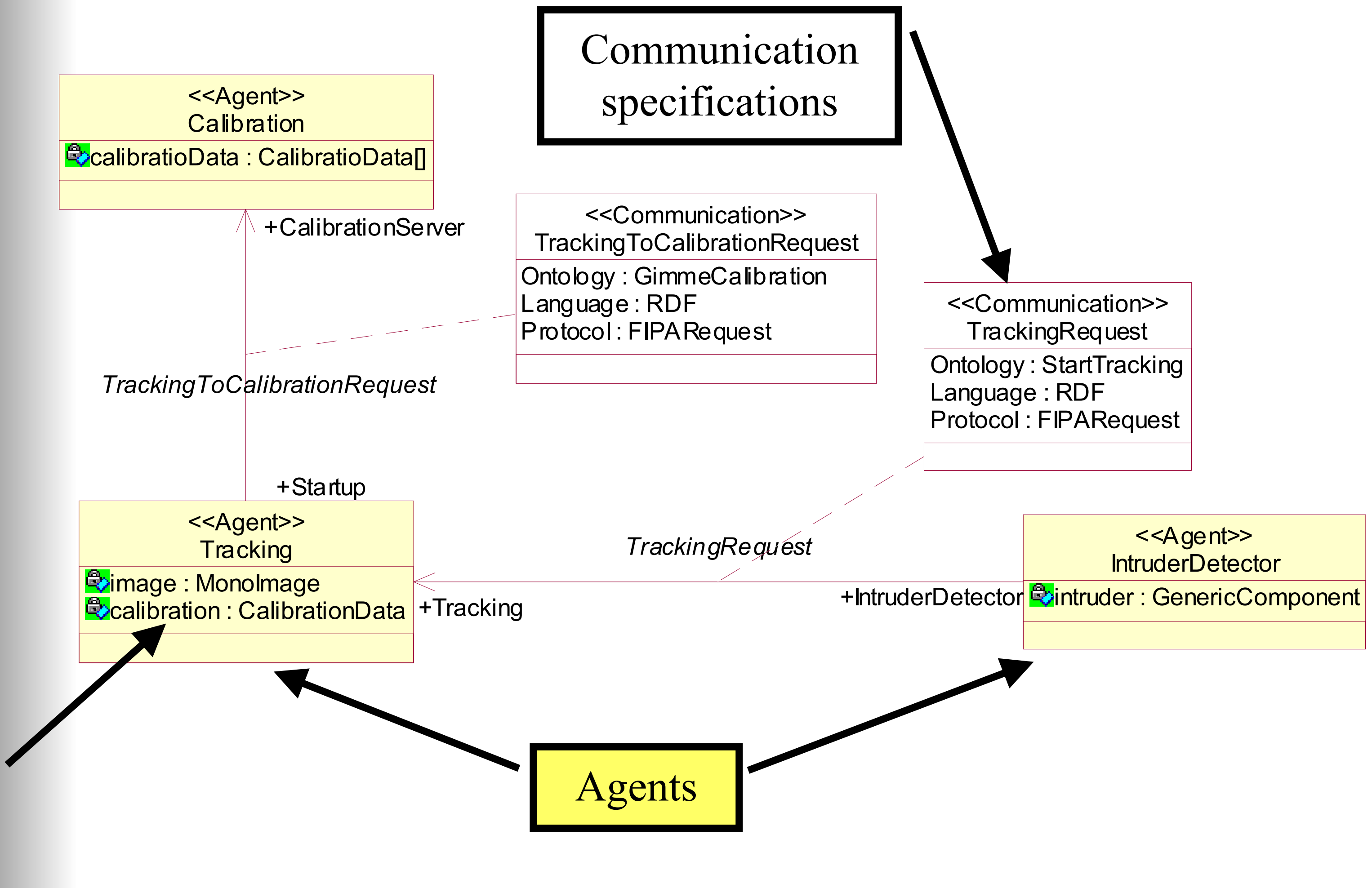
Export Ontologies to RDF Wizard

Welcome to the Export Ontologies to RDF Wizard

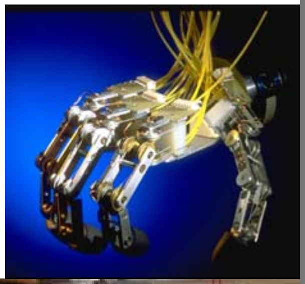
This Wizard exports a Domain Ontology Diagram to RDF. To Continue click Next.

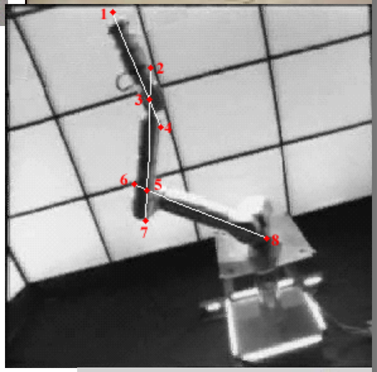
Communication Ontology Description Phase





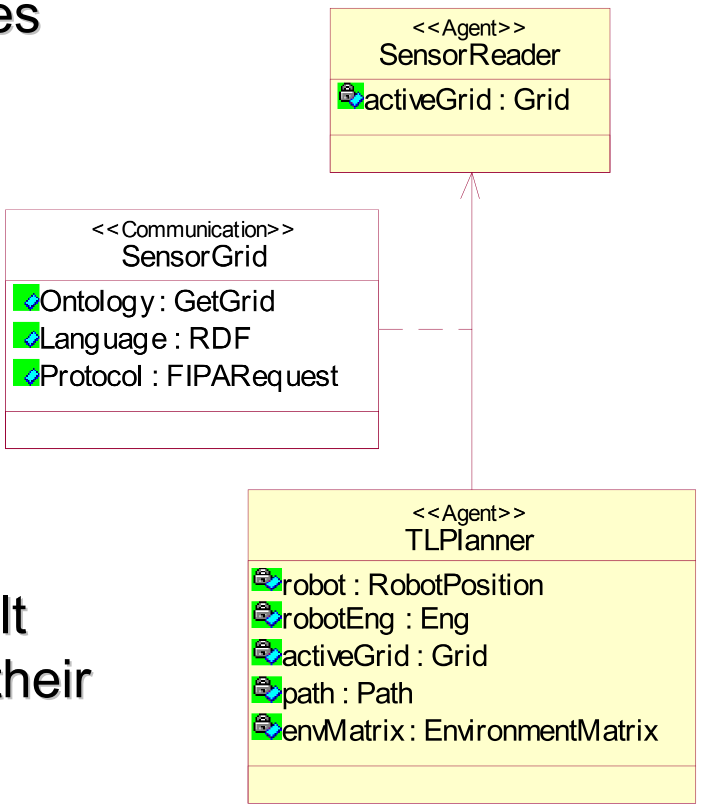
A class diagram is used to specify communications among the agents and the knowledge they need

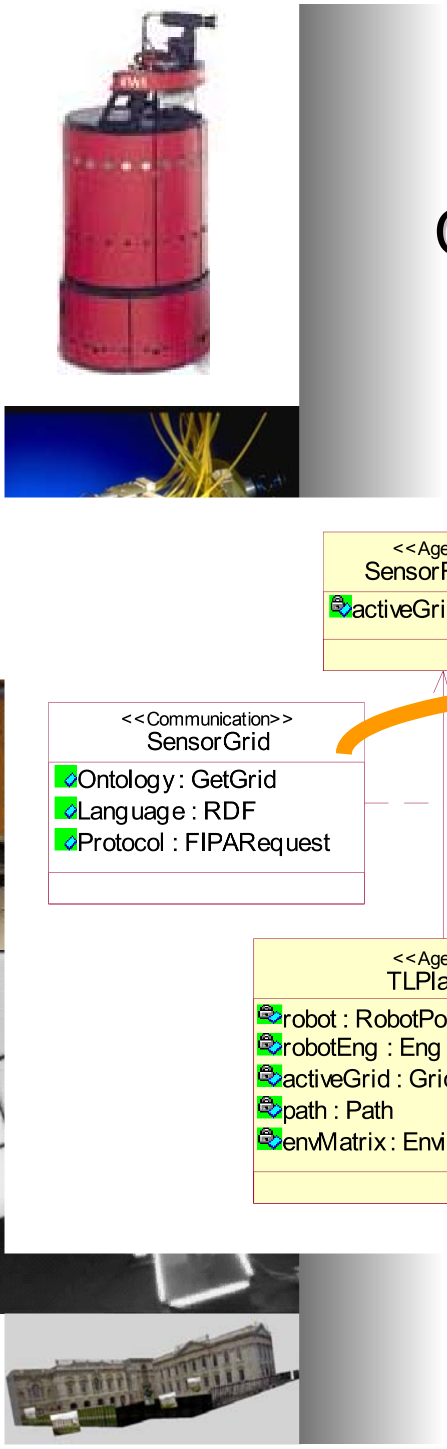




PTK support

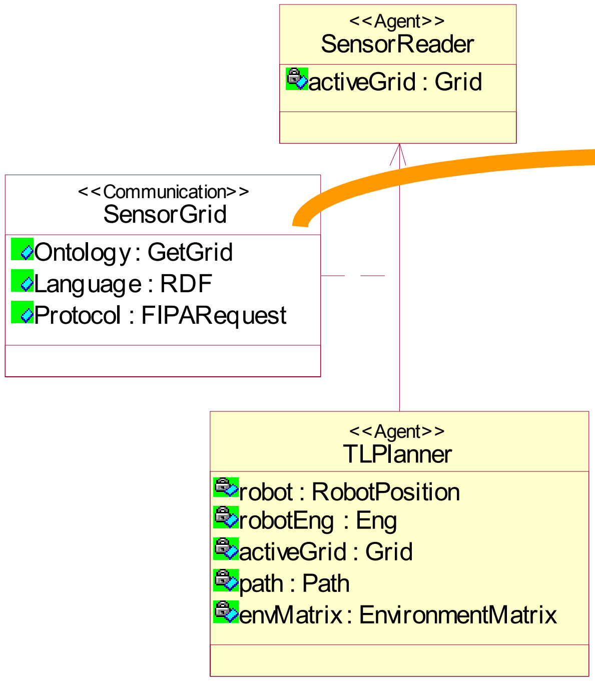
- Communications are automatically created looking at messages exchanged in scenarios
- Knowledge of agents is built considering the content of their communications

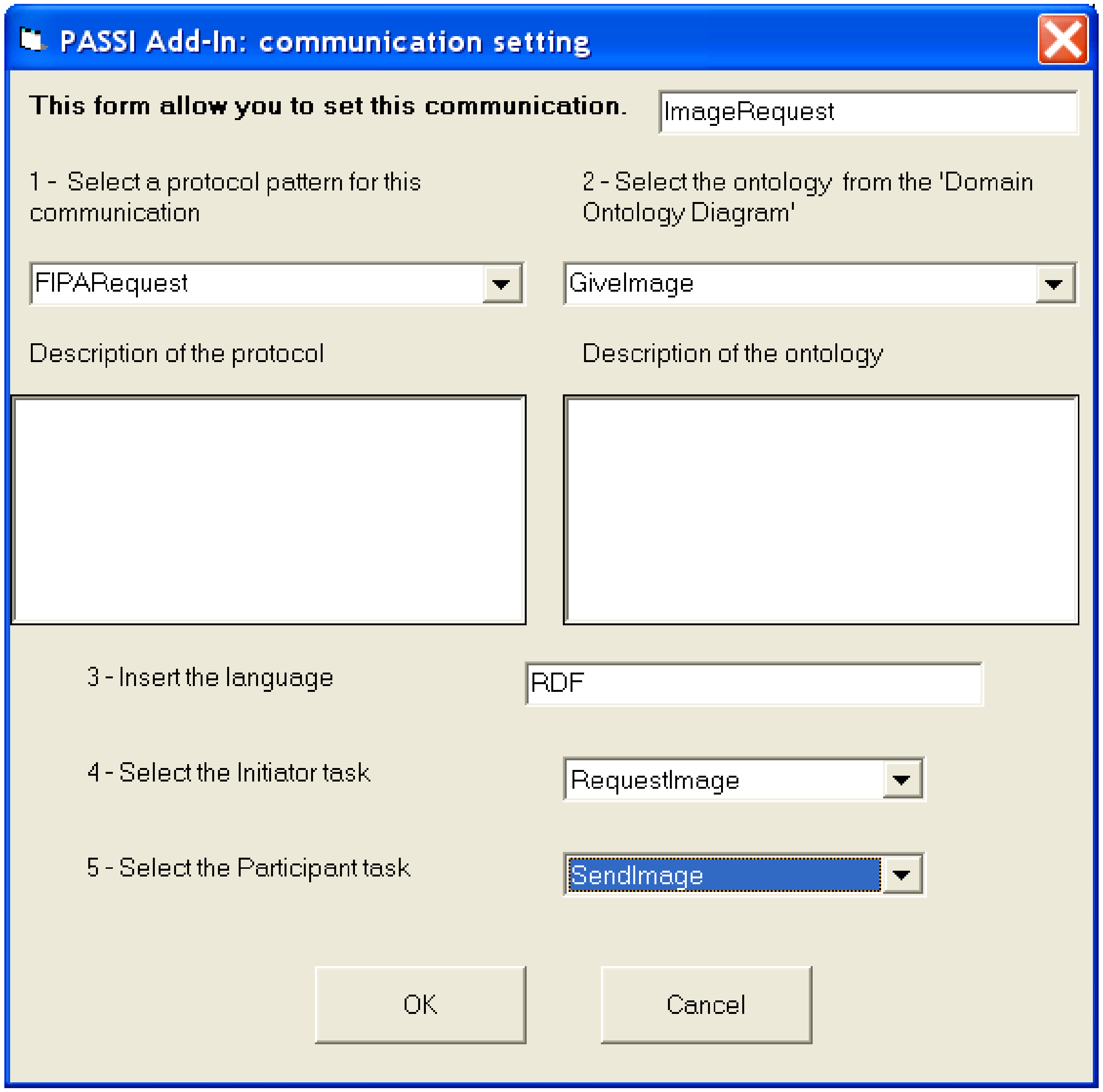




PTK support

Communications are detailed in this form





This form allow you to set this communication. ImageRequest

1 - Select a protocol pattern for this communication: FIPARequest

2 - Select the ontology from the 'Domain Ontology Diagram': GiveImage

Description of the protocol

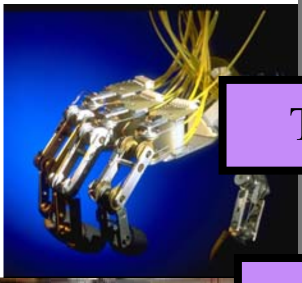
Description of the ontology

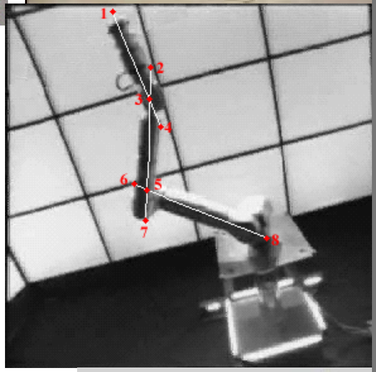
3 - Insert the language: RDF

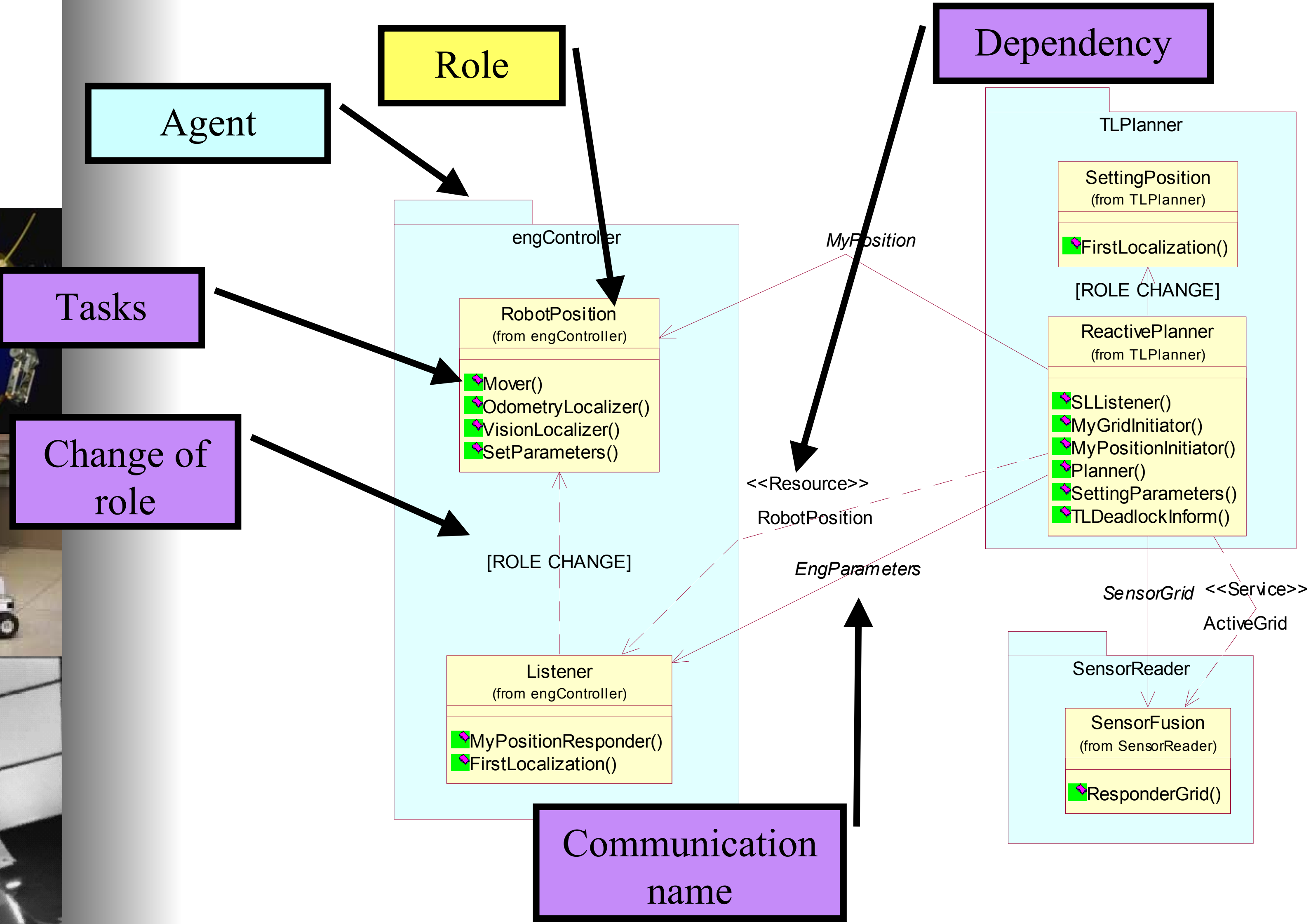
4 - Select the Initiator task: RequestImage

5 - Select the Participant task: SendImage

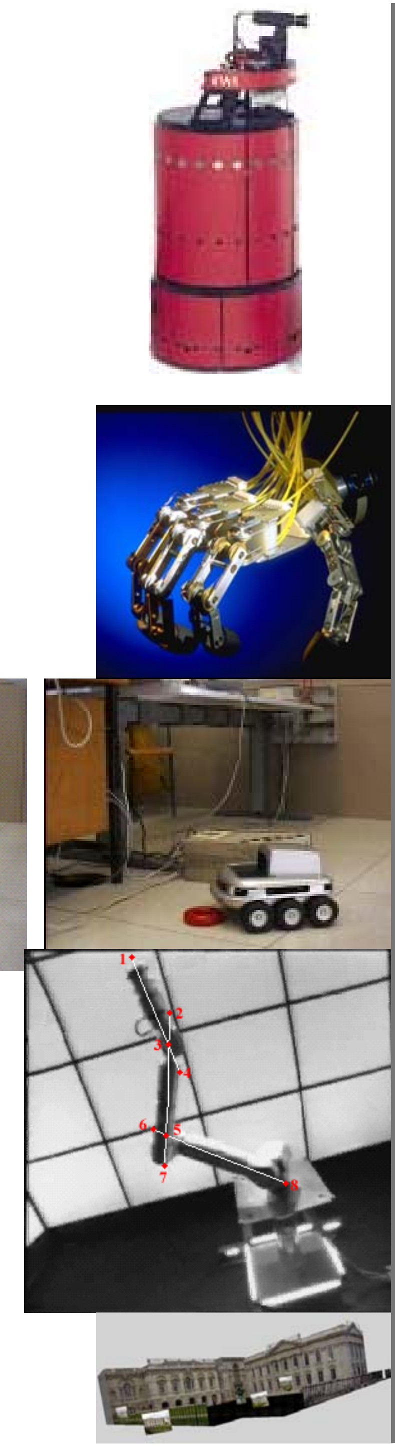
Roles Description Phase







All roles from the R.Id. Diagrams are here detailed together with communications exchanged

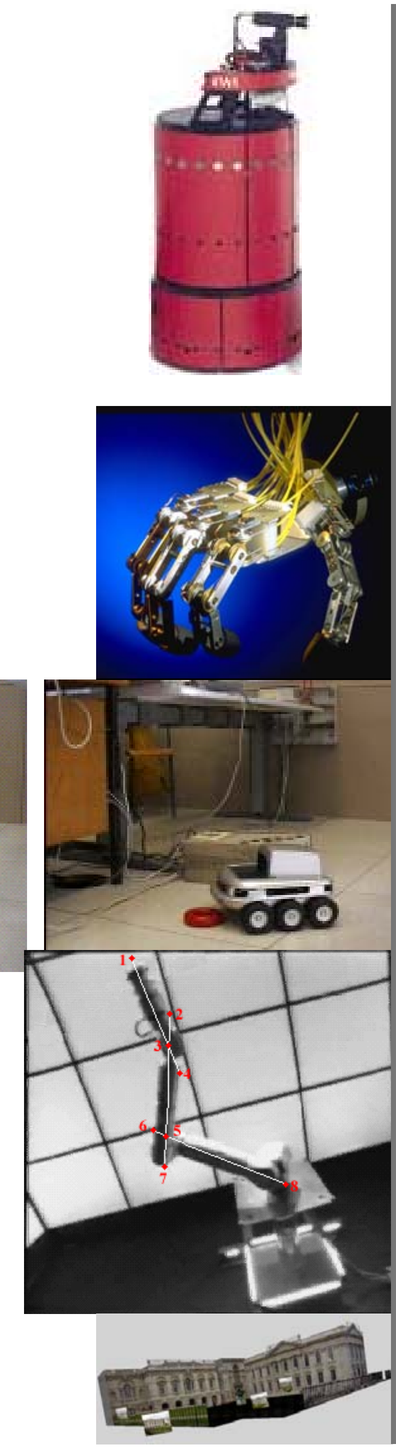


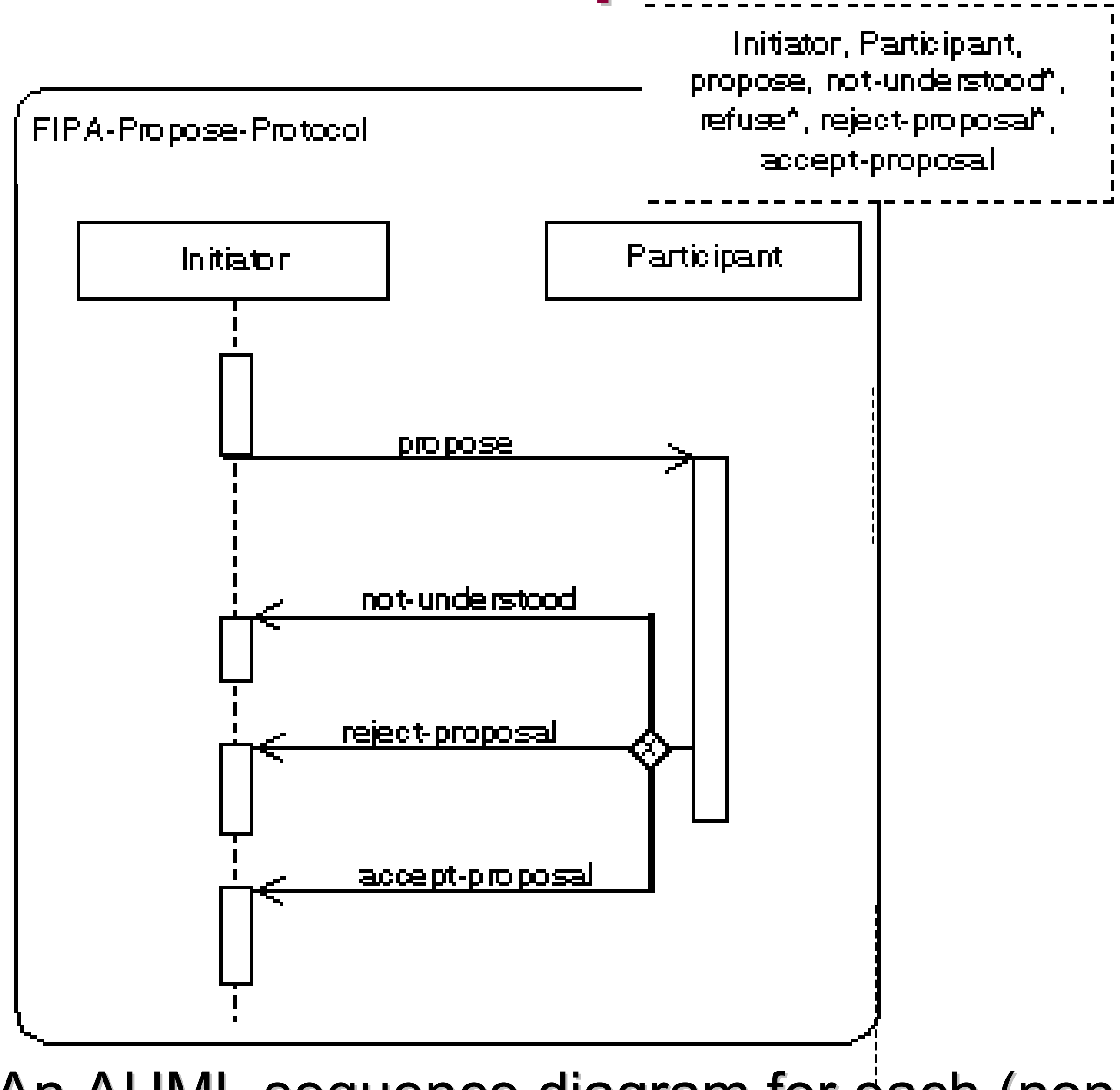
PTK support

A great part of this diagram is automatically built looking at roles identified in scenarios

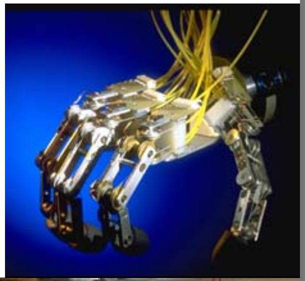
- If an agent plays different roles (in A.Id. scenarios) they are here reported together with communication exchanged (coming from the C.O.D. diagram)
- If an agent plays different roles in the same scenario the *change role* relationship in introduced among them

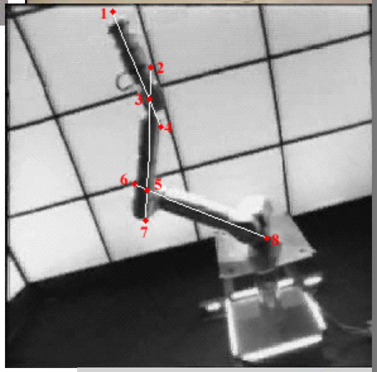
Protocols Description Phase




Initiator, Participant, propose, not-understood*, refuse*, reject-proposal*, accept-proposal
FIPA-Propose-Protocol
Initiator
Participant
propose
not-understood
reject-proposal
accept-proposal

An AUML sequence diagram for each (non standard) protocol



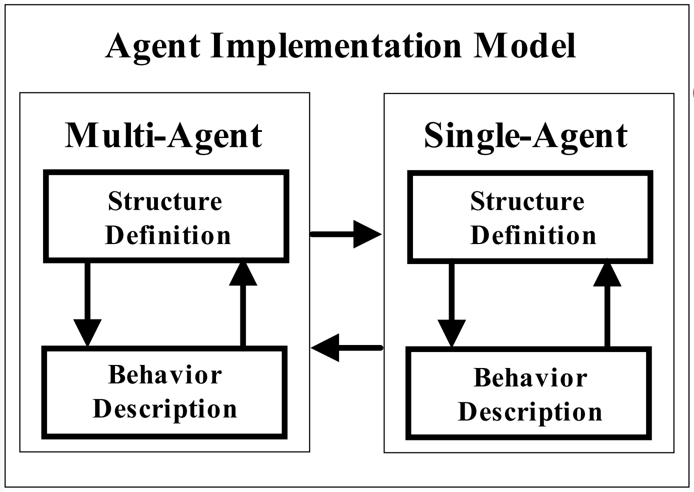


The Agent Implementation Model

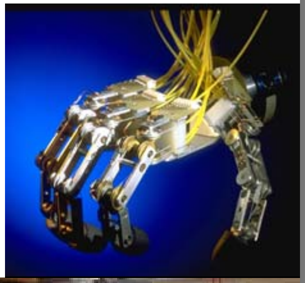
It includes the description of:

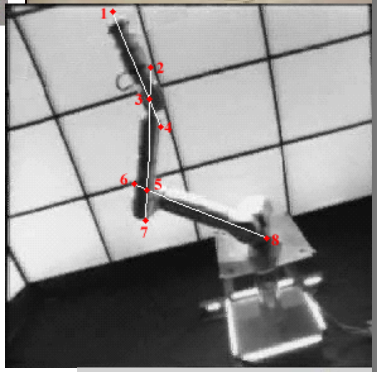
- Agents' structure (society abstraction level)
- Agents' behavior (society abstraction level)

- Agents' structure (single agent abstraction level)
- Agents' behavior (single agent abstraction level)

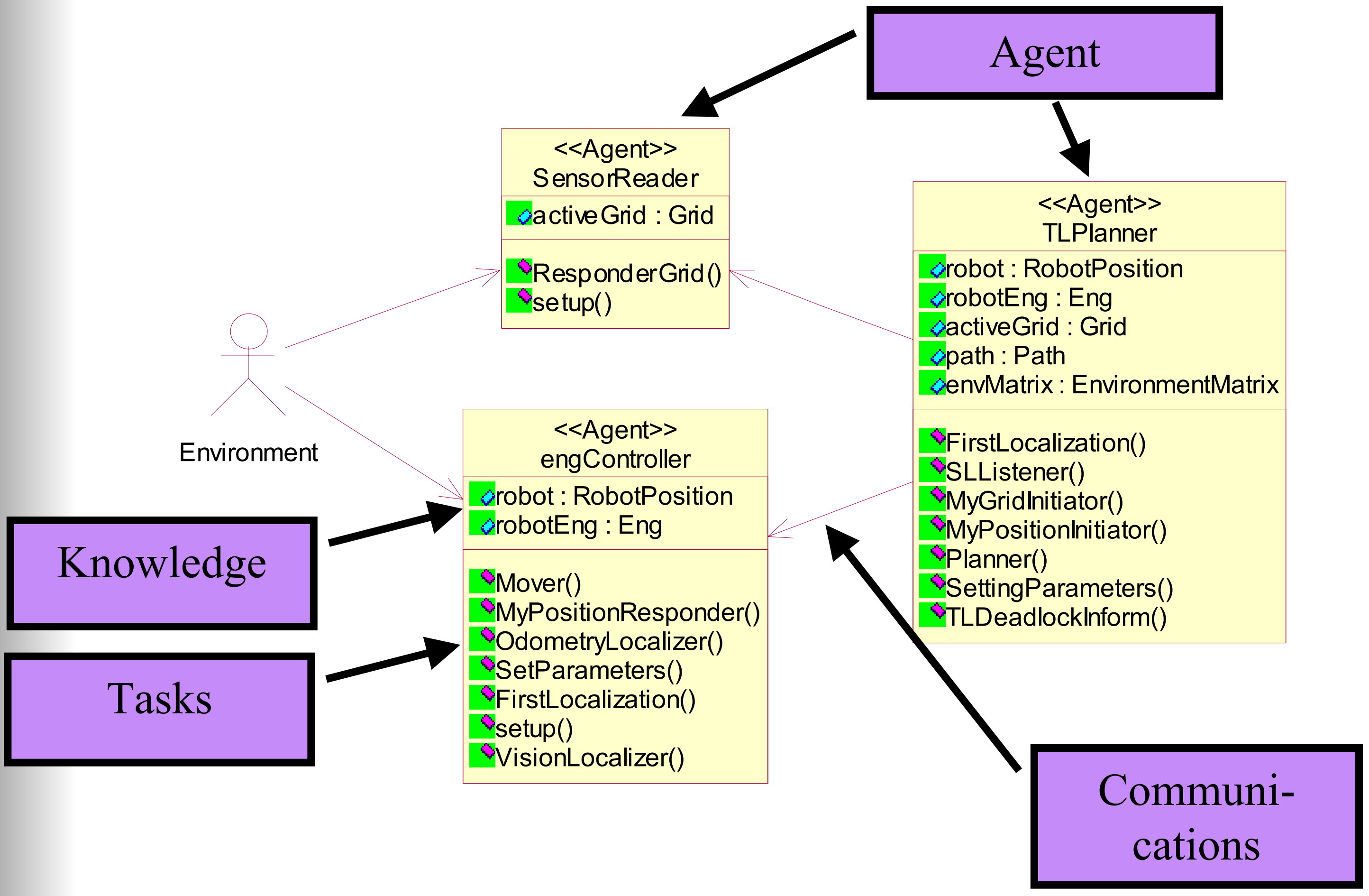


- Parts of structure and behavior provided by pattern reuse



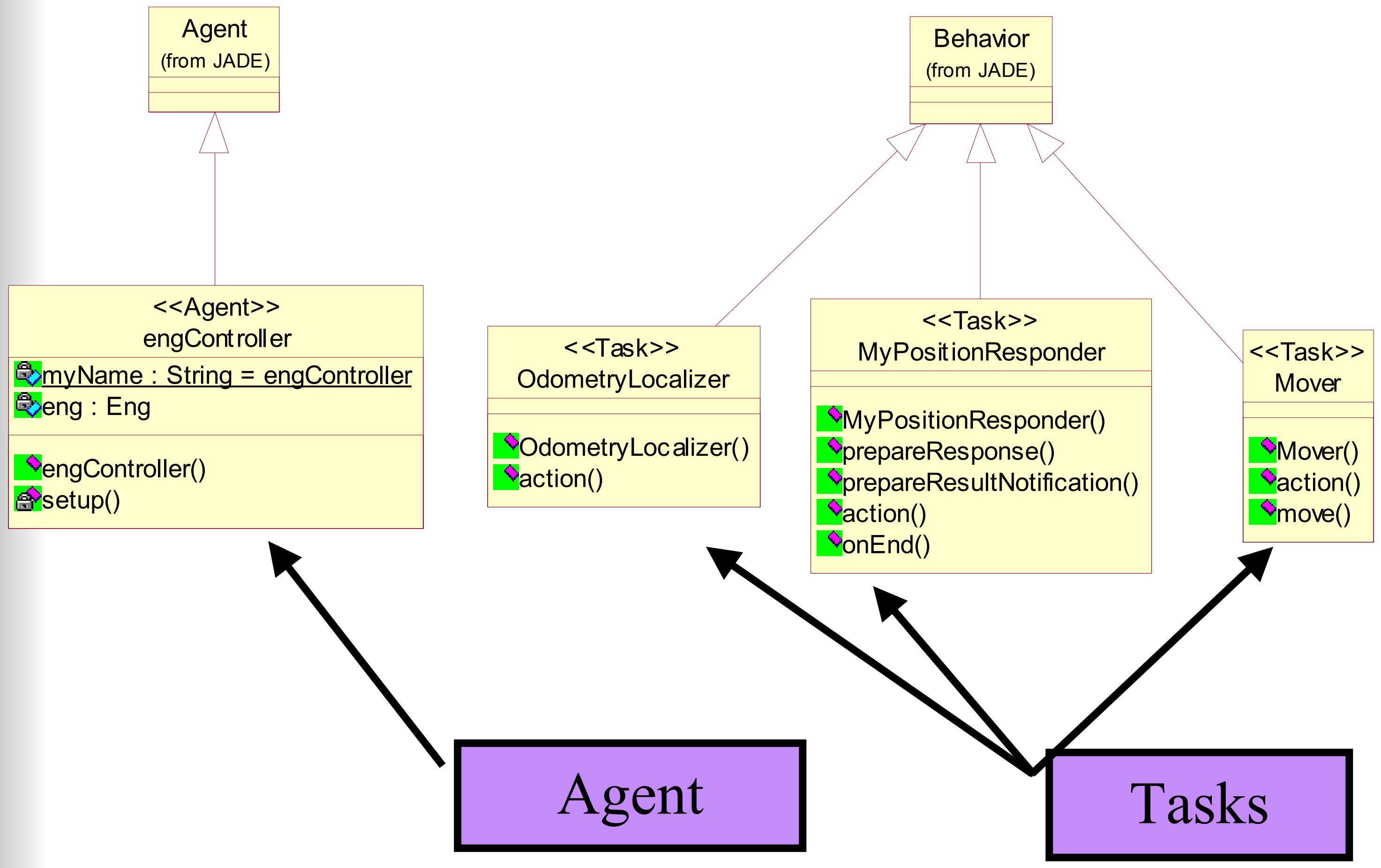


Multi-Agent Structure Definition Phase

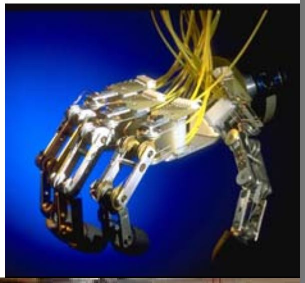


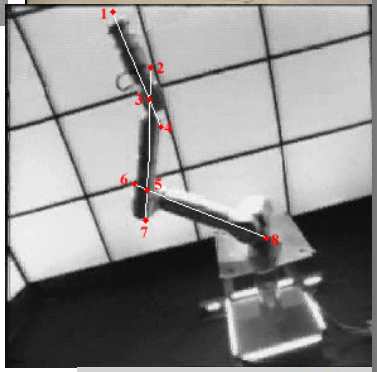
One class diagram is drawn for the whole system

Single-Agent Structure Definition Phase



One class diagram for each agent

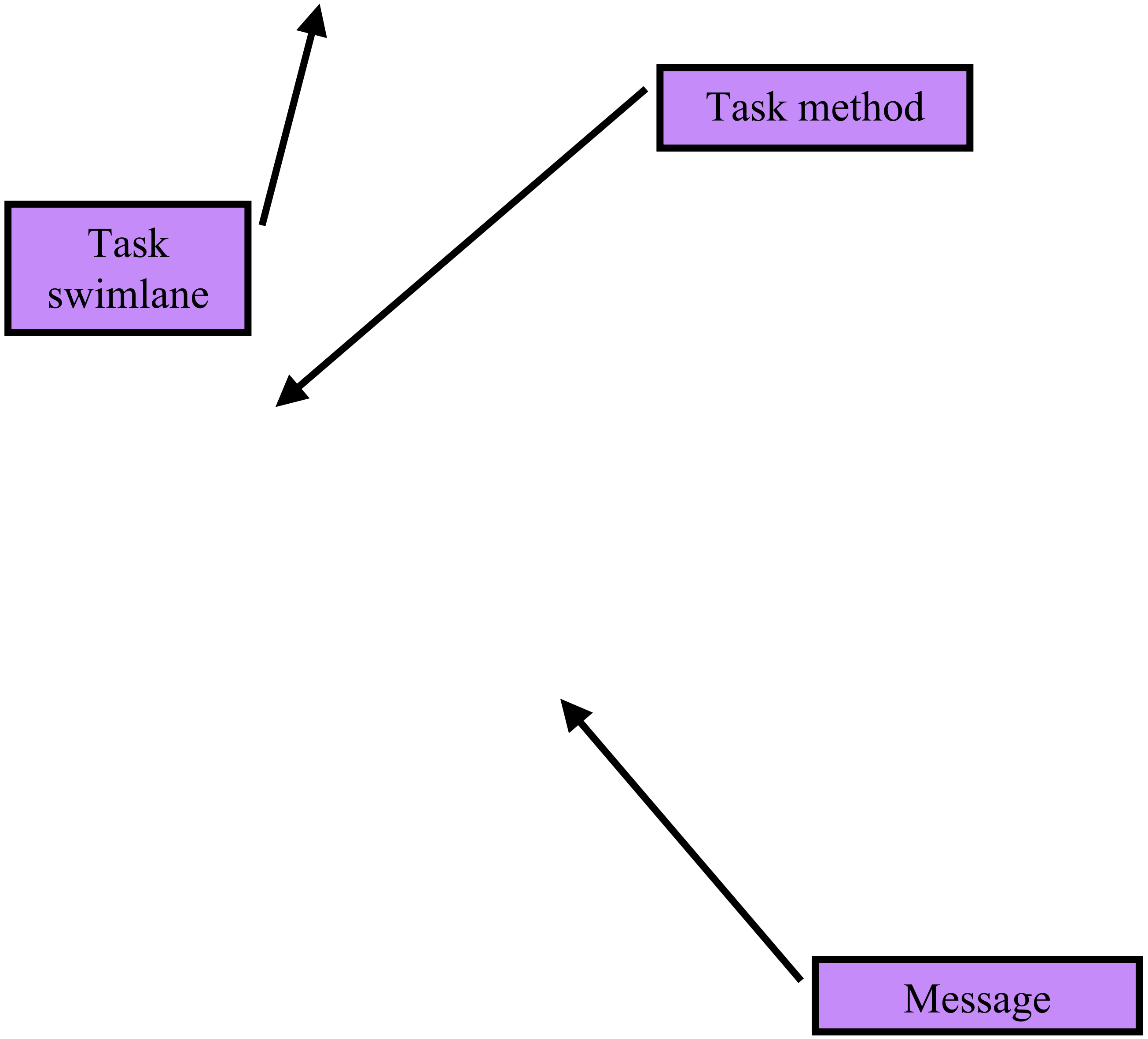


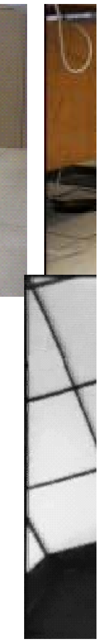


PTK Support

- Automatic compilation of the whole MASD diagram.
- Automatic compilation of part of the SASD diagram (agent skeleton, tasks coming from the T.Sp. phase, patterns of tasks) for each agent.
- Introduction of new tasks (synchronization of T.Sp. –Multi ASD – Single ASD diagrams).

Multi-Agent Behavior Description Phase

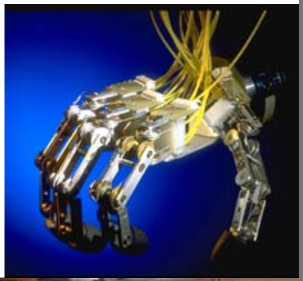


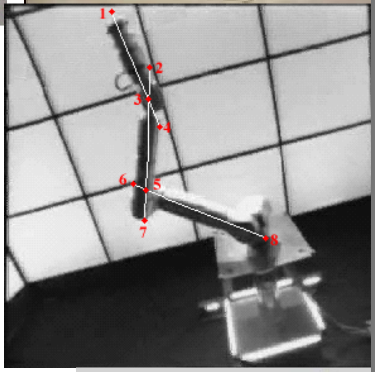


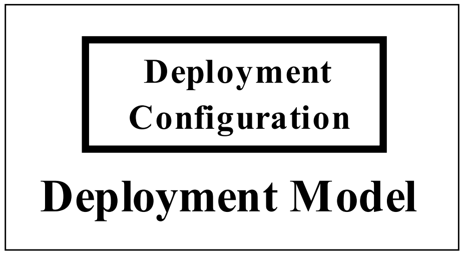
The Deployment Model

It includes the description of:

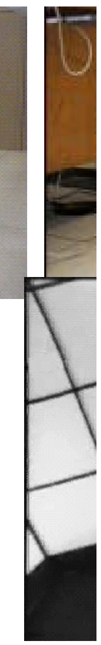
- Agents’ deployment computational units
- Agents’ movements
- Agents’ communication paths

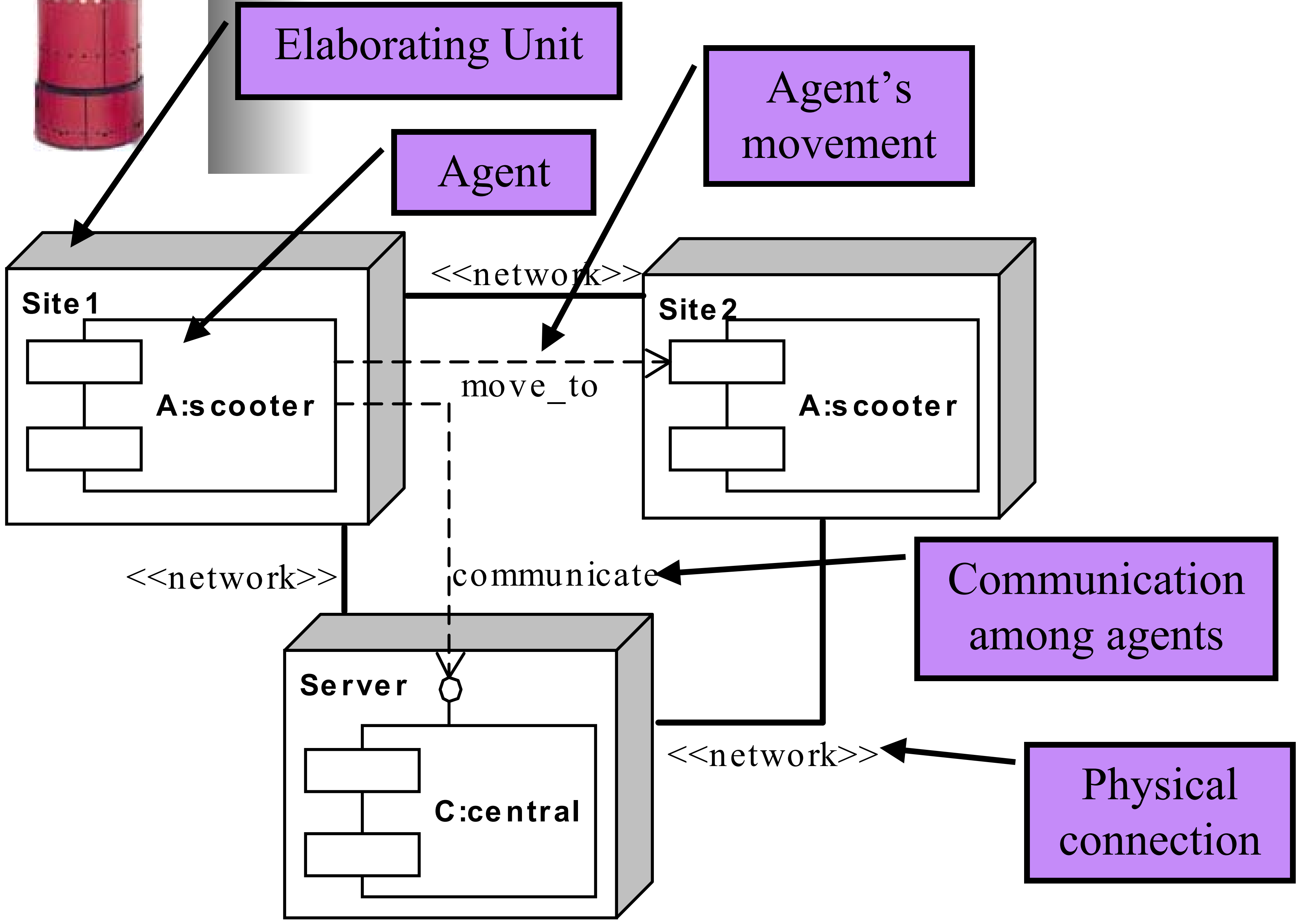


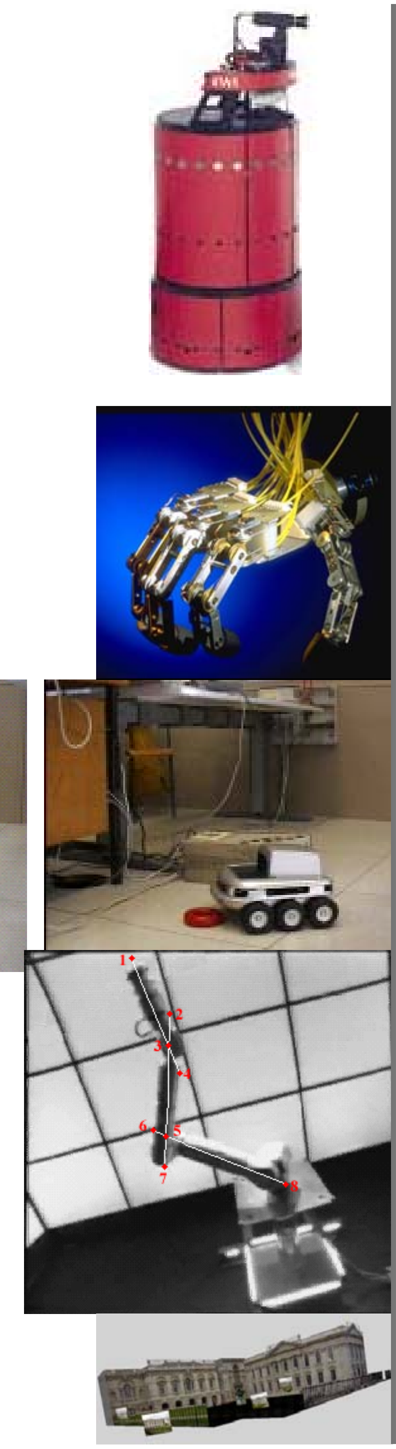




Deployment Configuration Phase



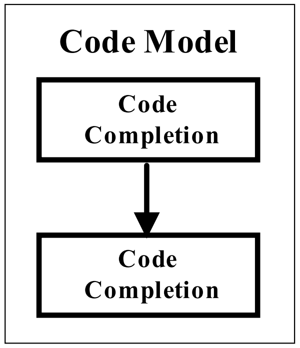




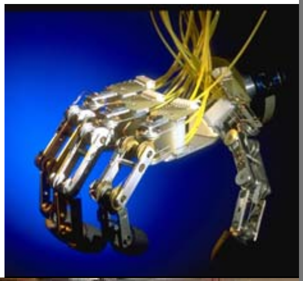
The Code Model

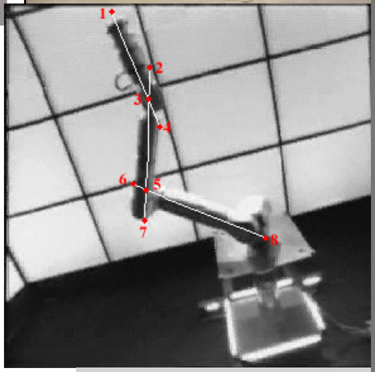
It includes the description of:

- Pattern reused code
- Code of the application



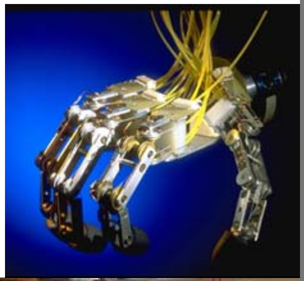
In our applications we used the FIPA-OS and JADE agent platforms therefore code is mainly written in the JAVA language

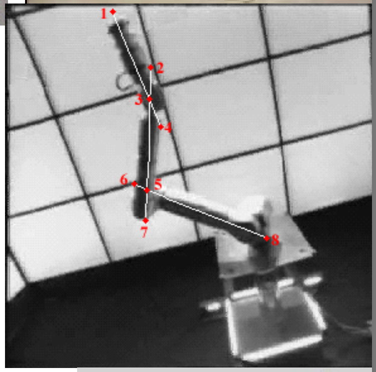




PTK support

- The whole skeleton of the agents is generated
- When patterns have been applied, code includes not only skeletons but also internal parts of methods
- It is possible to reverse engineer code
- In the next release (April 2003) a complete management of communications will be introduced:
  - JAVA data structures for agents' knowledge
  - Code for (RDF) messages management





Future works

- Support for multi-perspective design
- Improvement of ontology design capabilities
- Greater repository of patterns

For more information visit our website:
www.csai.unipa.it/passi