Multi-Agent System Platforms

Olivier Boissier

Olivier.Boissier@emse.fr

SMA/SIMMO

ENS Mines Saint-Etienne

Statement

- **Interest in using the MAS paradigm is increasing !!!**
- **BUT**
  - MAS is not (yet?) an implementation model and MAS oriented tools are usually specific
  - Agents themselves just begin to have their own languages
  - MAS Design relies on existing languages and programming paradigms
  - It's often hard to develop MAS (implementation, distribution, communications, …)

➔ **The trend of the work is towards Multi-Agent Oriented Programming, meaning programming MAS with MAS tools**

"Everyday" a new platform is announced on the Web. A Bunch of MAS development tools exist.
Which one to choose ? How to choose ? How to compare ?

What's the problem ? (1) *Agent's* point of view

- **Agents** require support to **perform** the tasks they are required to do. They must be able to :
  - sense events from the environment, act or manipulate it (software, hardware, real world),
  - communicate
    - uniqueness of names is important
    - search facilities
    - security, …
  - install cooperation, negotiation, coordination, organizations, …
- **Solution** ?
  - Agent Platform, MAS Platform, Agent Middleware

What's the problem ? (2) *Designer's* point of view

- **Designers** require support to **develop** the software they are required to do. Four stages must be covered :
  - **Analysis** : discovering, separating and describing the type of problem and the surrounding domain;
  - **Design** : defining the solution architecture of the problem (UML, …)
  - **Development** : constructing a functional solution to the problem
  - **Deployment** : effecting the solution to the real problem in the given domain (launching, maintaining, …)
- **Solution** ?
  - MAS or Agent Development Tools, MAS CASE Tools

What's the problem ? (3)
Final user's point of view

- **Final Users** require support to use the application they have payed for !!!
  - Security : enforcing trust, truth-telling, system integrity
  - Certification : third-party security services
  - Economic : charging and managing economic interactions,
  - …
- What about Education ? Research ?

Outline

- ✓ Introduction
- **MAS Middleware**
- Standards in MAS : FIPA
- Multi-Agent Oriented Software Engineering
- Some Multi-Agent Platforms

Middleware

- Software layer that resides between the underlying host and network operating system on the one hand, and application layer on the other.
- Distributed computing
  - linking large numbers of different components to perform joint tasks
- Providing support services
  - common set of programming interfaces, reuse of code, …

➔ defines (part of …) the environment, the communication infrastructure, … of a Multi-Agent System

Client vs Middleware

Specific applications and domains.

Generic mechanisms fundamental level
- generic protocols of cooperation, communication, negotiation, etc, …
- description of agents' links, organisations, etc,
- description of agents languages ontologies,
- generic behaviour of agents,

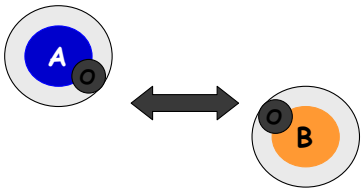
Low-level layers of MAS
- primitives of communication between remote agents KQML, ACL, …
- servers that allow agents to enter and to leave the system (check-in, check-out procedures, …)
- engines that implement the basic cycle of agent behaviour

Available resources
- mechanisms of low-level communication (sockets, TCP/IP protocols, …)
- mechanisms for distributed processing (threads, …)

Fat Client - Thin Middleware

Agents have many general purpose components

Little is left to the environment



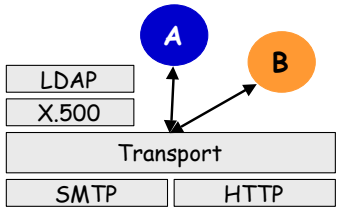
+ Advantages
- More control over behavior
- Good sense of the environment
- Agent is more “independent”

- Disadvantages
- Little abstraction
- Little interoperability (many different implementations)
- (Potentially) Less efficient due to multiple specialization.

Thin Client - Fat Middleware

Agents have only behavior code

Everything else is a “service” in the middleware



+ Advantages
- Reduction in code that needs to be written
- Greater reuse of code
- Stronger Interoperability (more predictable environment)
- Lightweight agents

- Disadvantages
- Less control of behavior
- No low level environment sensors
- (Potentially) Less efficient due to generalization

Agent Middleware

Often quite “fat”
- Using agents as an abstraction
- Focus on high level behavioral details
- Many systems are prototypes so :
  - Efficiency is less of a concern
  - The domain is restricted (less worried about general survival)

Often Incorporate
- Naming services
- Message Transport
- Communication mechanisms and sometimes even coordination mechanisms such as auction interfaces
- Agent architectures including reasoning systems

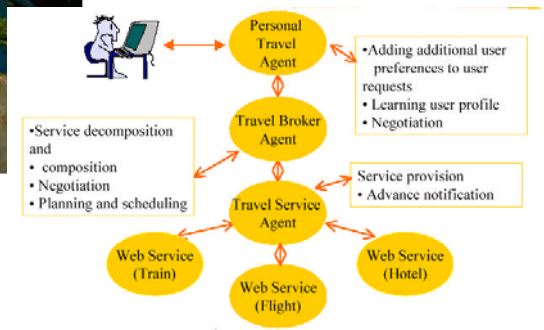
Outline

- ✓ Introduction
- ✓ MAS Middleware
- ➢ **Standards in MAS : FIPA**
- Multi-Agent Oriented Software Engineering
- Some Multi-Agent Platforms

Why do we need Standards in MAS ? (1)

- Why ?
  - Standards favor development of the market.
  - Market development speed up development of technology.
- Why now ?
  - Technology is under development
    - Lack of large agreement on definitions, models, theories, …
    - Several research axes
  - Needs are recognized
  - Stress on the expected features : *openness, heterogeneous systems, emergent properties, …*

Why do we need Standards in MAS ? (2)



CLIMATE Industrial Workshop 26/4/99

Standards in MAS (1)

- **FIPA** Foundation for Intelligent Physical Agents
  - www.fipa.org
- **MASIF** - OMG (Object Management Group) : OMG effort to standardize mobile agents - middleware services and internal middleware interfaces
  - www.omg.org
- **Knowledge Sharing Effort** The DARPA Knowledge Sharing Effort
  - citeseer.nj.nec.com/pati192darpa.html

Standards in MAS (2)

- **Ontology : DAML, OIL, OWL, …**
  - http://www.daml.org
  - http://www.ontoknowledge.org/oil/
- **Other Standards (De Facto)**
  - dynamic discovery of services :
    - Jini (www.sun.com/jini),
    - UPnP (www.upnp.org),
    - UDDI (www.uddi.org),
    - Salutation (www.salutation.org)
  - mobility : Aglets (www.trl.ibm.com/aglets/)
  - coordination rules : JavaSpace (www.sun.com/jini)

FIPA Overview (1)

- Aim :
  - to create International Standards body in order to promote the development of agents applications
- Structured in :
  - FIPA Architecture Board, Technical Committees, Working Groups
- Funded in 1996
  - 62 member companies with heavy involvement from telecommunications companies in particular.
    BT, EPFL, France Télécom, Fujitsu, HP, Hitachi, IBM, Imperial College, Intel, Motorola, NASA, Nec, NHK, Nortel Networks, NTT, Philips, Siemens, SNCF, SUN Microsystems, Telecom Italia, Toshiba …
- First standards in 1997 - FIPA97, since then - FIPA98 and FIPA2000

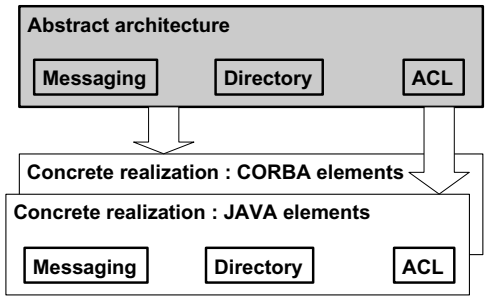
FIPA Overview (2)

- FIPA specifies the interfaces of the different components in the environment with which an agent can interact, i.e. humans, other agents, non-agent software and the physical world. FIPA produces two kinds of specifications:
  - **normative** specifications mandating the external behaviour of an agent and ensuring interoperability with other FIPA-specified subsystems;
  - **informative** specifications of applications providing guidance to industry on the use of FIPA technologies.

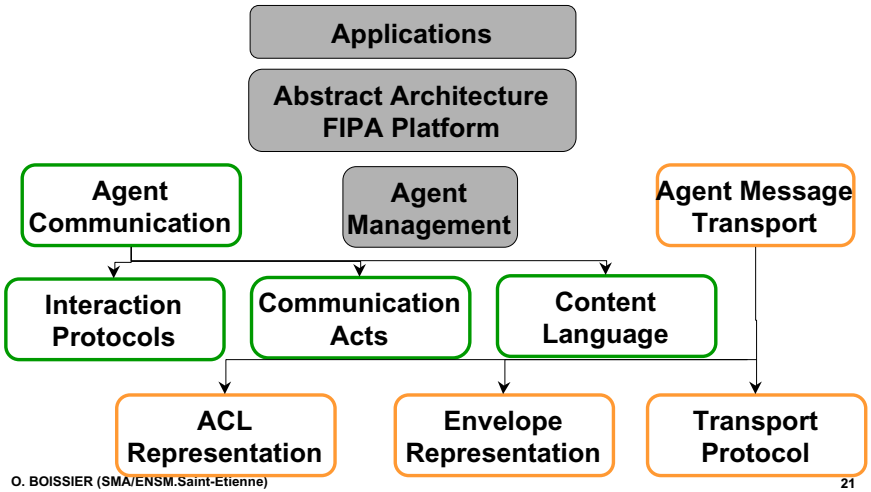
FIPA Overview (3)

- Definition of an abstract Architecture
  - Message transport interoperability.
  - Supporting various forms of ACL representations.
  - Supporting various forms of content language.
  - Supporting multiple directory services representations.
- Modelling of the abstract elements and their relationships.

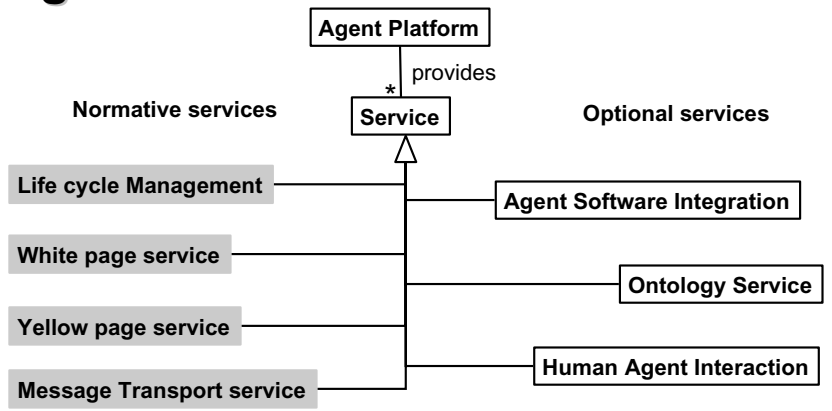
FIPA Overview (4)



FIPA Specifications (#80)

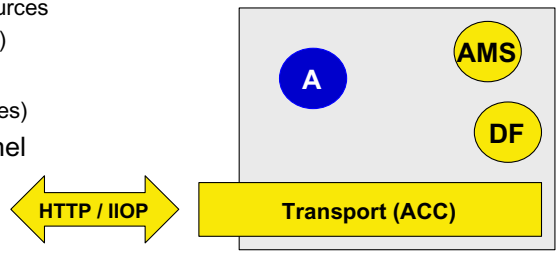


FIPA : conceptual model of an Agent Platform



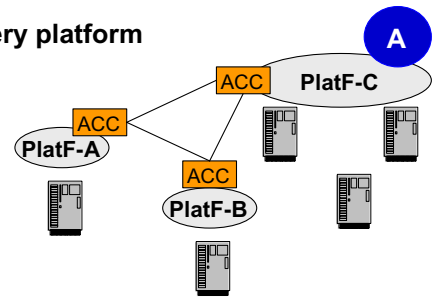
FIPA Agent Platform

- Software that implements the set of FIPA specifications.
- **FIPA-compliant** ⇔ implements at least the **Agent Management** and **Agent Communication Language** specifications.
- **A**gent **M**anagement **S**yst.
  - Authentication, Resources
  - White pages (naming)
- **D**irectory **F**acilitator
  - Directory (yellow pages)
- **A**gent **C**omm. **C**hannel
  - Message transport

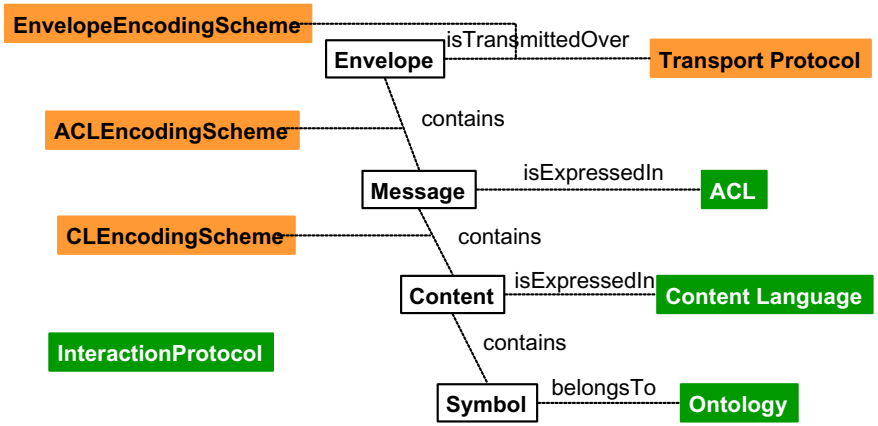


FIPA Platforms Inter-operability

- **Communication between agents can be**
  - Platform Internal – non-standard technologies
  - Platform-Platforms – uses the ACC and standard FIPA Message transport protocols.
- **Agent Environment on every platform**
  - Different languages
  - Different APIs
  - Different support features
  - Different agent architectures
- **Same**
  - Base services
  - Same transports
  - Same languages



FIPA : conceptual model of agent communication



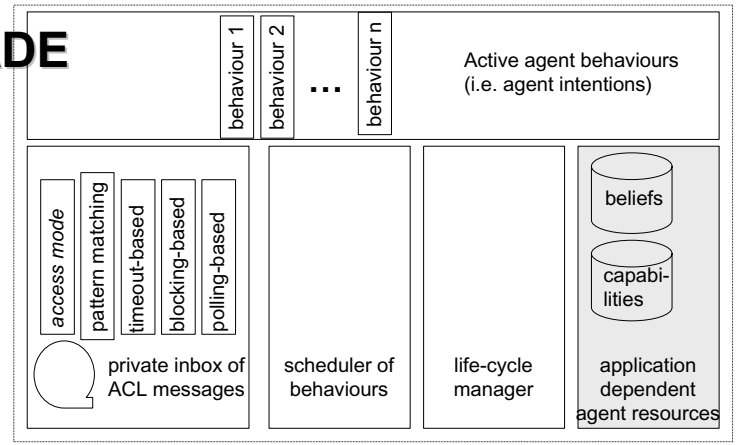
JADE : introduction (1)

- **J**ava **A**gent **D**evelopment **F**ramework (http://jade.tilab.com/)
- Developed by CSELT (It),
- Open Source GPL, Written in Java
- Developing multi-agent systems and applications conforming to FIPA standards for intelligent agents.
- Includes two main products:
  - a FIPA-compliant agent platform
  - a package to develop Java agents.
- Several extensions to run on small devices (PDA, MP, …) → LEAP Project

JADE : introduction (2)

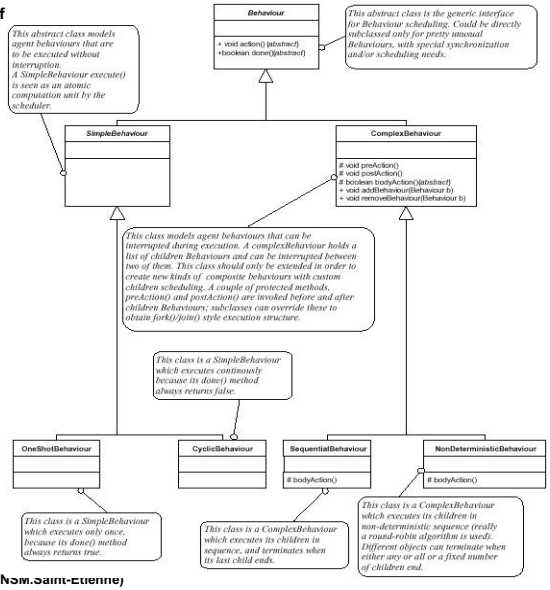
- **Development :** a JADE agent is simply an instance of a user defined Java class that extends the base **Agent** class.
- **Deployment :** some utilities are provided :
  - *Agent Management System* (AMS), *Directory Facilitator (DF), Remote Monitoring Agent* (RMA) allows controlling the life cycle of the agent platform and of all the registered agents.
  - *DummyAgent tool* allows users to interact with JADE agents in a custom way.
  - *Sniffer Agent* is basically a Fipa-compliant Agent with sniffing features.
  - FIPA-compliant IIOP to connect to different AP-s
  - GUI to manage several agents and AP-s

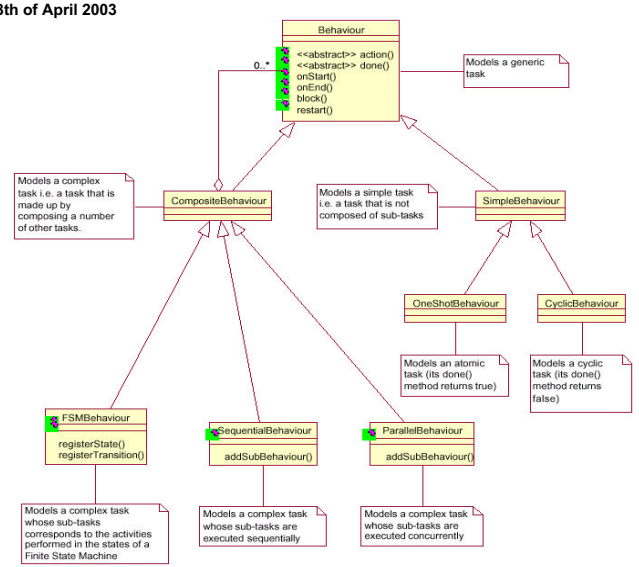
JADE



The JADE framework includes a library of interaction protocols and generic agent behaviours, that must be customized for the specific application needs in order to create the agent capabilities

JADE library of interaction protocols and of generic agent behaviours



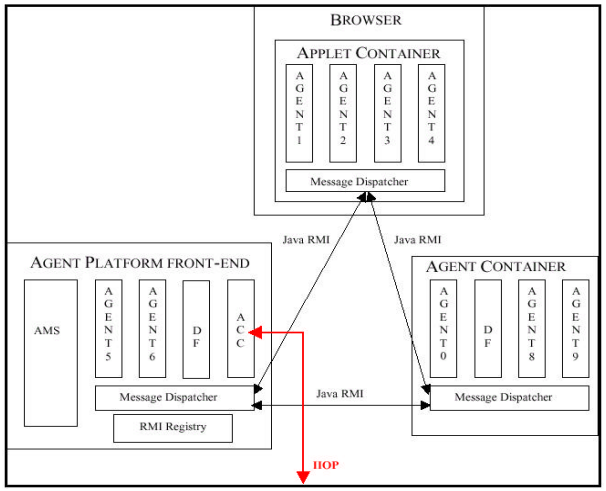


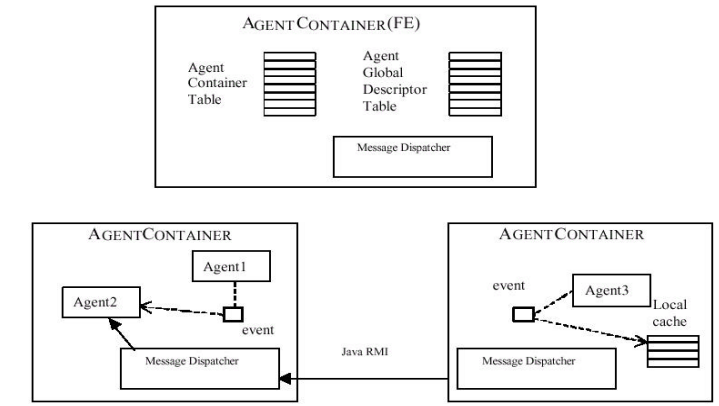
JADE : Agent Communication Model

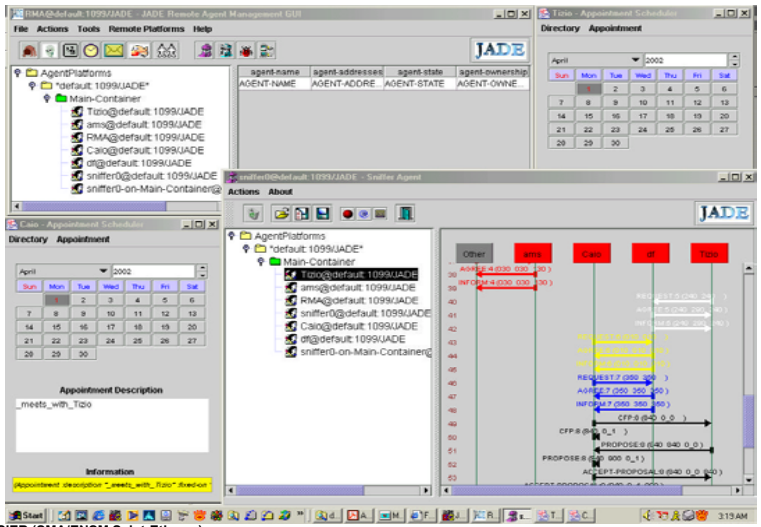
- Agent sends/receives Java objects, that represent ACL messages within the scope of interaction protocols
- Jade hides all message coding (encoding/parsing)
  - Envelope level (string-based, XML-based)
  - Agent Communication Language level (string-based, XML-based, bit-efficient)
  - Content Language Level (FIPA SL-0 + API to registed user-defined content language, support for Base64-encoded direct Java object serialization)
  - Ontology level (FIPA-Agent Management, JADE Agent Management, API to register user-defined content languages)
  - the framework can be extended by users
- JADE provides a library of common interaction protocols
  - users just need to implement the handle methods
  - users can compose agents tasks like super-states of FSM

JADE : Containers

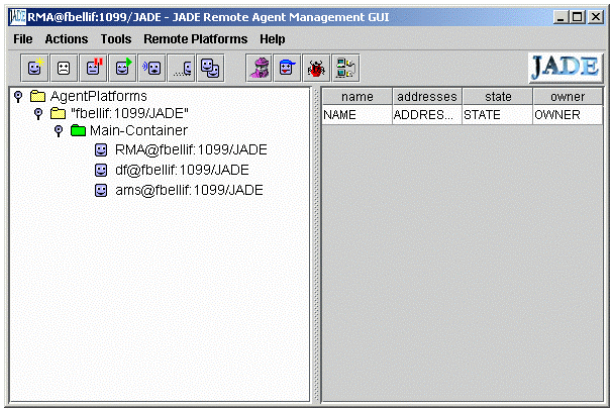
- Each agent lives inside a container
- A container is a JVM, provides a complete runtime environment for agent execution, allows several agents to run concurrently, controls the life-cycle of agents (create, suspend, resume, kill), deals with communication (dispatches and routes messages)
- Light-weight container provided for agent execution within a web browser
- Special Container - the front-end (FE) container runs the management agents, represents the whole platform to the outside world
- The GUI itself is implemented as an agent - Remote Management Agent (RMA)



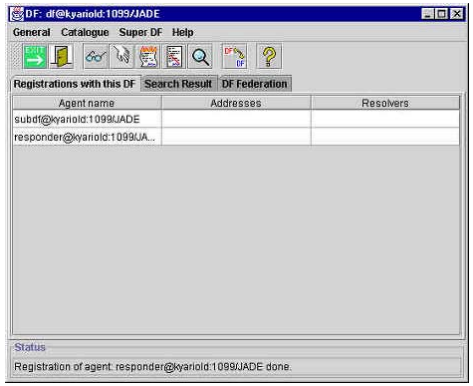




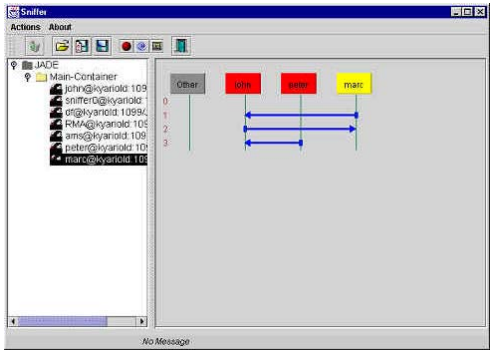
Remote Agent Management GUI



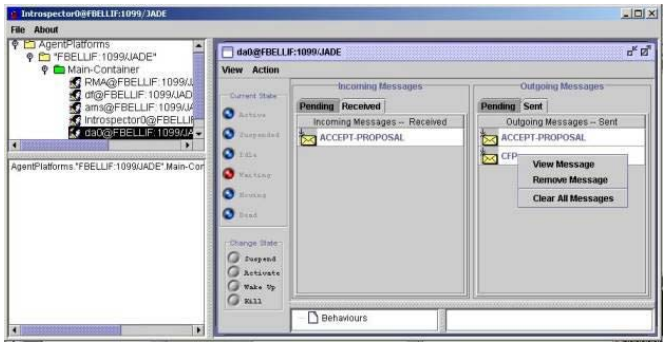
Directory Facilitator Agent



Sniffer Agent



Introspector Agent



Outline

- ✓ Introduction
- ✓ MAS Middleware
- ✓ Standards in MAS : FIPA
- ➢ **Multi-Agent Oriented Software Engineering**
- Some Multi-Agent Platforms

Why do we need MAOSE ?

- Need to support complex system development by
  - Providing techniques for handling complexity
    - Decomposition
    - Abstraction
    - Organization
  - Allowing for the characteristics of complex software
    - Hierarchical . composed of subsystems
    - Interactions between/within subsystems
    - Component decomposition is designer.s choice
    - Evolve more quickly than standard systems

MAOSE Questions

- A MAOSE methodology must address two levels : agent and multi-agent levels
- Multi-Agent Questions
  - How many agents are there?
  - What is the common medium (Environment) shared by the agents ?
  - What communication mechanisms will the agents use?
  - What communication protocols will the agents use?
  - How will relationships between agents be established?
  - How will agents coordinate their actions?
- Agent Questions
  - What in the system should become an agent?
  - How will each agent model the environment?
  - How will agents be structured internally?

Conceptual Step : Requirements

- Provided by the user
  - Often in terms of desired scenarios or undesired scenarios
  - Often ambiguous and contradictory
  - Generally insufficient basis for system design

Conceptual Step : Analysis

- Analyze requirements to determine
  - What is the overall desired behavior of the system
    - Main goals of the system
    - What roles must be played to solve the problem
    - Resources available to solve the problem
    - Human interface requirements
  - What can be varied to achieve this behavior
  - What cannot be varied

Logical Step : Design

- Transform high-level goals and roles into concrete agent types
  - What agents will perform which roles
  - Define number and types of agents
- Transform high-level interactions into specific interaction protocols
  - Derived from high-level role Interactions
- Define low-level functions of each agent type to carry out high-level behaviors
  - Define ontology used by each agent
- Define low-level functions of each agent type to carry out high-level behaviors
  - Define ontology used by each agent

Physical Step : Implementation

- If not part of requirements, decide on
  - environment implementation
  - communication mechanism
  - communication languages
  - content languages
  - implementation language
- Transform protocols and low-level behavior into code

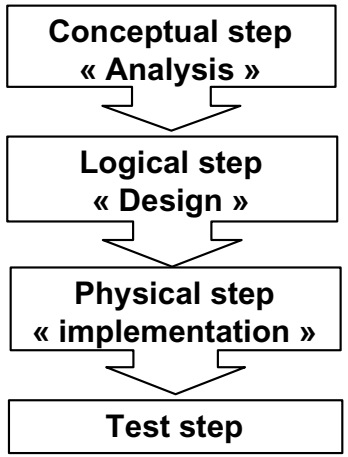
MAS "Life Cycle" [Occello 01]

"Agentification" of needs

Identification and Instantiation of MAS models

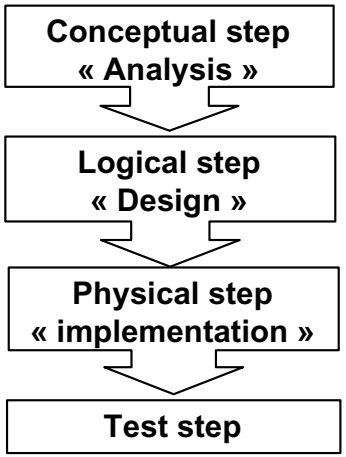
Implementation and deployment of MAS

Test and debugging, validation, simulation

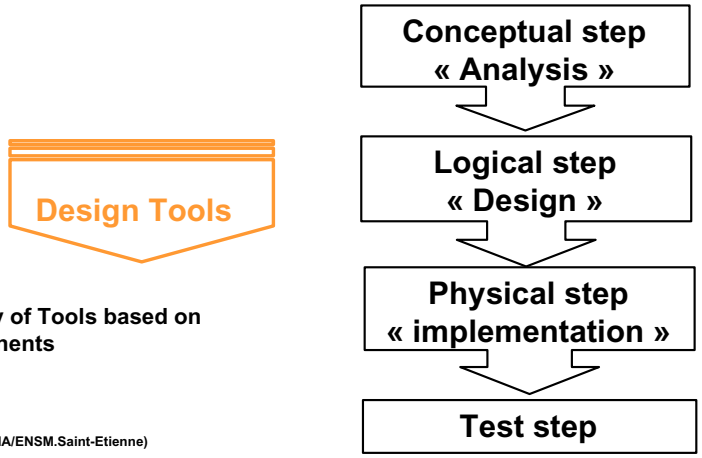


CASE Tools / MAS Life cycle (1)

Agent Architectures, Agent Communication Language + Agent Middleware



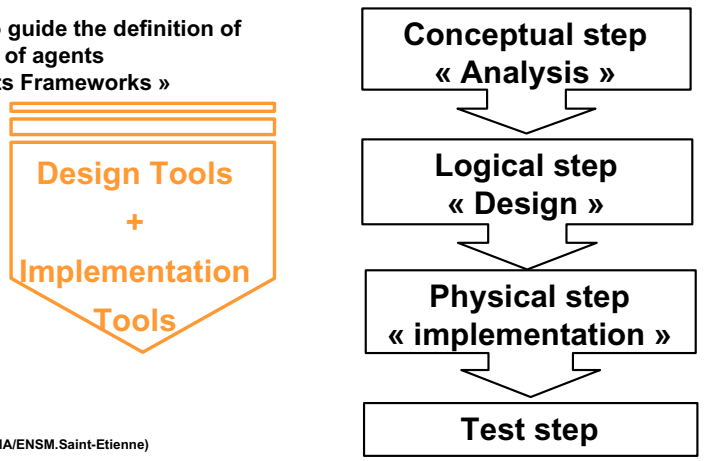
CASE Tools / MAS Life cycle (2)



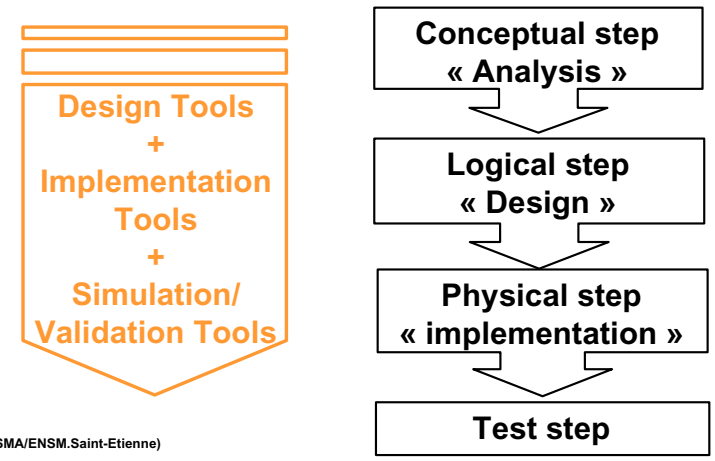
Majority of Tools based on Components

CASE Tools / MAS Life cycle (3)

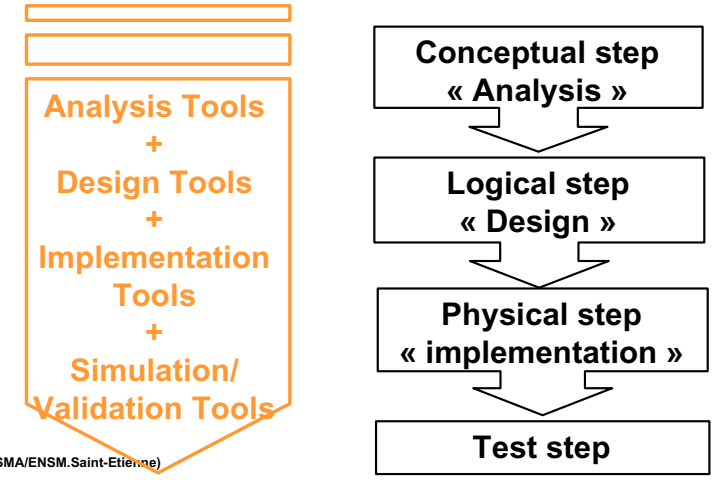
Used to guide the definition of Groups of agents
« Agents Frameworks »



CASE Tools / MAS Life cycle (4)



CASE Tools / MAS Life cycle (4)



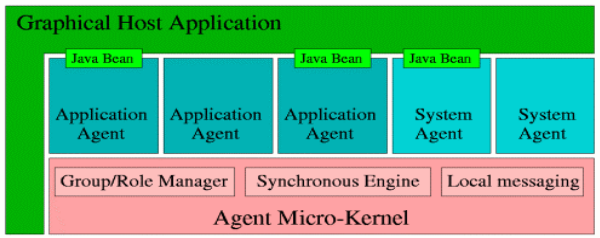
Outline

- ✓ Introduction
- ✓ MAS Middleware
- ✓ Standards in MAS : FIPA
- ✓ Multi-Agent Oriented Software Engineering
- ➢ **Some Multi-Agent Platforms**

MADKIT : introduction

- ➢ Multi-Agent Development Kit (www.madkit.org, community.madkit.org)
- ➢ Free for educational use.
- ➢ Developed by LIRMM lab.
- ➢ Java multi-agent platform based on the Aalaadin model (organizational model)
- ➢ MAS runtime engine using an agent micro-kernel

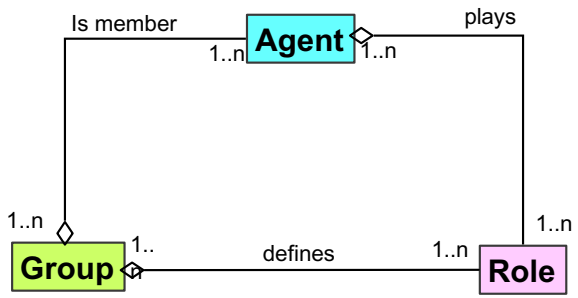
MADKIT : micro kernel



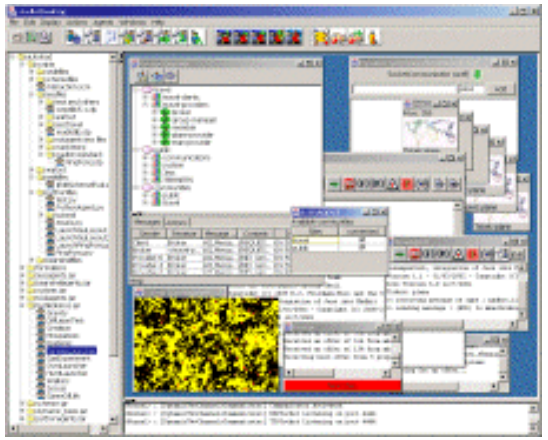
MADKIT / CASE Tools

- ➢ **Analysis** : no specific analysis method (functional, dependency, group context, coordination mechanisms)
- ➢ **Design** : definition of the organizational model (groups, roles), interaction model (protocols, messages), tasks, goals, … No software tools.
- ➢ **Development** : No agent model (to be implemented in java from scratch).
- ➢ **Deployment** : use of the G-box (eq. Sandbox)

MADKIT : AALAADIN



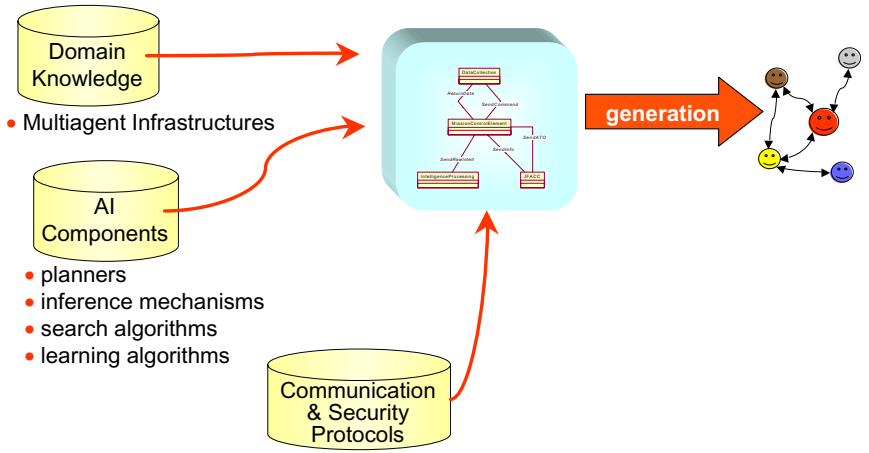
MADKIT : GUI



AgentTool : Introduction

- http://www.cis.ksu.edu/~sdeloach/ai/download-agentool.htm
- Free for research and academic use
- Developed at Kansas State University
- Written in Java 1.2, Runs on any Java 1.2 compliant platform
  - Conversation verification requires some extra installation
- Diagrams include
  - Agent Class, Conversation, Internal Agent
- 90% Code generation for agentMom

AgentTool



AgentTool / CASE Tools

- **Analysis** : analysis method
- **Design** : definition of the conversations, agents
- **Development** : Automatic code generation
- **Deployment** : ???

MaSE

- **Multiagent Systems Engineering Methodology**
- **Motivation behind MaSE**
  - **Lack of proven methodologies for agent-systems**
  - **Lack of industrial-strength software tools**
- **Similar to Gaia with respect to**
  - **Generality**
  - **Application Domain (closed systems)**
- **MaSE's differences from Gaia**
  - **Supports automatic code creation**

MaSE : Process

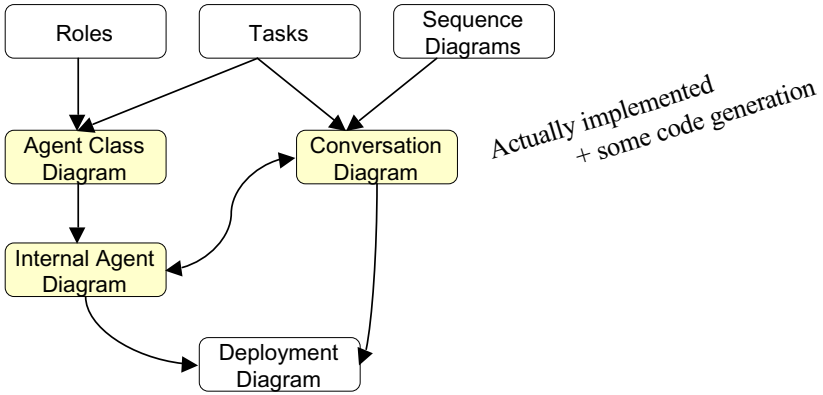
1. **Capturing Goals**
   - **initial system specification ⇒ struct. hierarchy of goals**
   - **i.e. similar to requirement specification**
2. **Applying Use Cases (i.e. UML)**
   - **Use cases and sequence diagrams based on spec.**
   - **Use cases – represent logical interaction path**
   - **Sequence diagrams – number of messages needed**
3. **Refining Roles**
   - **Creates roles corresponding to the goals (or a set of goals)**
   - **Creates tasks – how to solve goals related to the role**

MaSE : Process (suite)

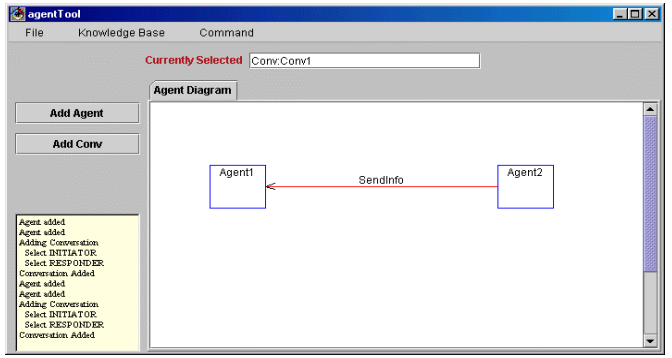
4. **Creating Agent Classes**
   - **Maps roles to agent classes in an agent class diagram**
   - **Resemble object class diagrams, but semantics is high-level conversation versus inheritance (and encapsulation)**
5. **Constructing Conversations**
   - **Defines coordination protocols for interaction with state diagrams**
6. **Assembling Agent Classes**
   - **Internal functionalities of classes created**
   - **Based on either BDI, reactive, planning, knowledge-based and user-defined architecture.**
7. **System Design**
   - **Create instances of the agent classes presented in a deployment diagram**

- **Future vision of MaSE**
  - **Support automatic code generation based on deployment diagram**

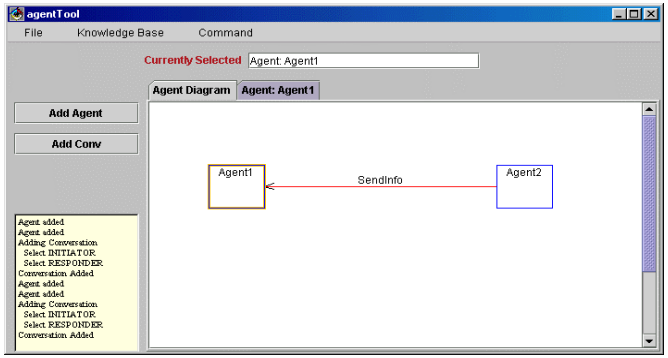
MaSE in agentTool



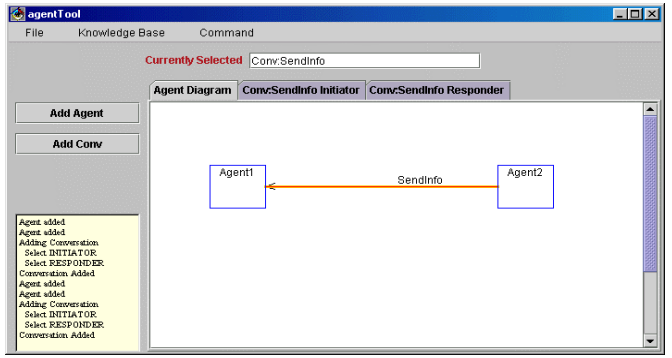
AgentTool : Agent Diagram



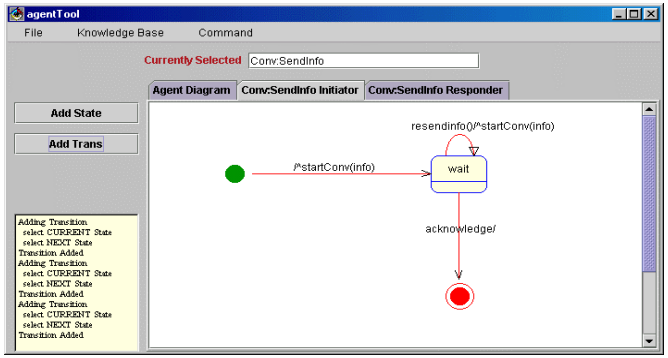
AgentTool : Selecting an Agent Class



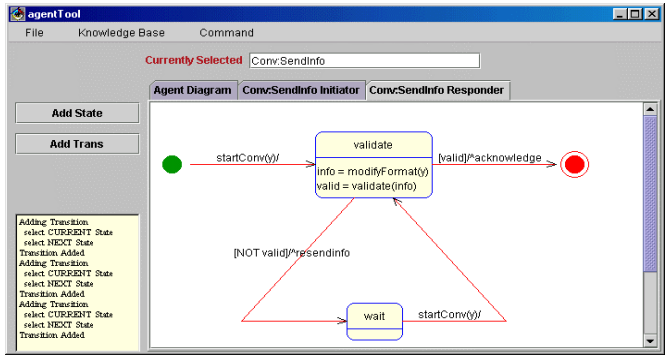
AgentTool : Selecting a Conversation



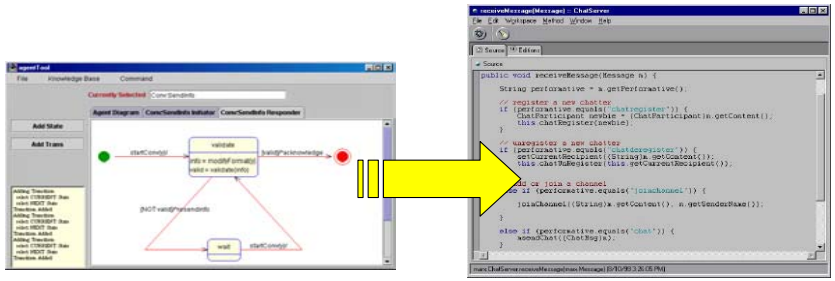
AgentTool : Conversation Diagram
(half a conversation)



AgentTool : Conversation Diagram
(the other half)



AgentTool : Code Generation



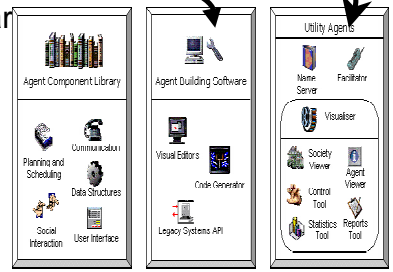
- Automatic from Agent and Conversation Diagrams
- Select platform-dependent components such as a messaging system
- Currently focused on agentMom

ZEUS : introduction

- Agent Building Toolkit (www.labs.bt.com/projects/agents/zeus)
- Open Source Licence (Mozilla)
- Written in Java
- Developed by Agent Research Programme of BT Intelligent Research Lab.
- Integrated environment for the rapid building of collaborative agent applications.
- Strong emphasis on the importance of methodology

ZEUS : software architecture

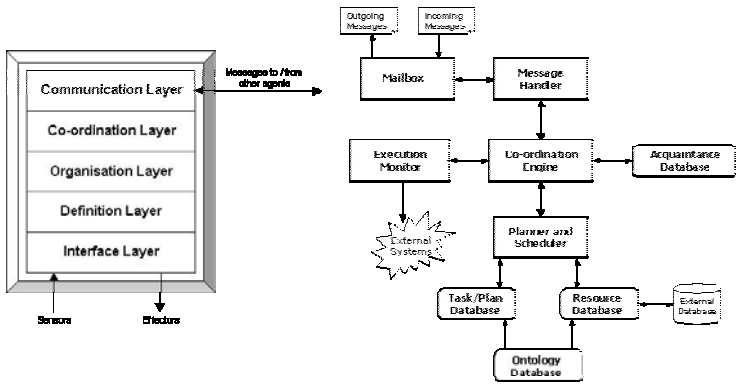
- Three libraries :
  - Utility Agents
  - Agent Building Tool
  - Agent Component Librar



ZEUS / CASE Tools

- **Analysis** : role modelling. No software tools. Role modelling with UML class diagrams and patterns.
- **Design** : Use of a software tool. A Zeus agent model is available.
- **Development** : *Application Realisation Guide*, graphical tools (ontology, agent, utility agent, task agent config., Agent implementation).
- **Deployment** : *Runtime Guide*, visualiser and monitoring tools

ZEUS : Agent's Architecture



ZEUS : Development step

- **Ontology Creation :** the declarative knowledge that represents the significant concepts within the application domain (ZEUS Ontology Editor).
- **Agent Creation :** the generic ZEUS agent is configured to fulfil its application-specific responsibilities [Task agents] (ZEUS Agent Editor) [Agent Definition, Task Description, Agent Organisation, Agent Co-ordination]
- **Utility Agent Configuration :** the attributes of the utility agents [agent platform] (Code Generation Editor)
- **Task Agent Configuration :** the runtime parameters of the task agents.
- **Agent Implementation :** the Code Generator can be invoked and agent source code automatically generated.

ZEUS : Utility Agents

- Nameserver and Facilitator agents : facilitate information discovery
- Visualiser agent : visualising or debugging societies of ZEUS agents.

➔ any number of these utility agents, with at least one nameserver agent in the system.

➔ constructed using the basic components of the Agent Component Library.

ZEUS : Visualiser Agents

- Society Viewer : all the agents in a society, organisational inter-relationships, messages exchanged.
- Reports Tool : decomposition/distribution of active tasks and the execution states of the various tasks.
- Agent Viewer : observes and monitors the internal states of agents.
- Control Tool : to remotely review and/or modify the internal states of individual agents.
- Statistics Tool : displays individual agent and society-wide statistics in a variety of formats.

ZEUS : Communication

- Available in the Agent Component Library :
  - a performative-based agent communication language [FIPA 97]
  - an asynchronous socket-based message passing system;
  - an editor for describing domain-specific ontologies and the domain concepts that are defined using the ontology editor are used as part of the content language within the ACL;
  - a frame-based knowledge representation language for representing domain concepts.

Other MAS Middleware and MAS CASE Tools

- See Review of Software Products for MAS, AgentLink, June 2002
- FIPA Compliant Platforms
  - FIPA-OS http://fipa-os.sourceforge.net/
  - ZEUS www.labs.bt.com/projects/agents/zeus
- SACI http://www.lti.pcs.usp.br/saci/
  - Simple Agent Communication Infrastructure
- JACK http://www.agent-software.co.uk (cf. PRS)
  - Runtime environment, Compiler, BDI Agent Model
- AgentBuilder http://www.agentbuilder.com/ (cf. AOP)
- JAFMAS/JIVE http://www.ececs.uc.edu/~abaker/JAFMAS/
  - Speech act based communication, agent architecture
- **…** DIMA, GEAMAS, MAGIQUE, MAST, OSACA.…

Conclusion

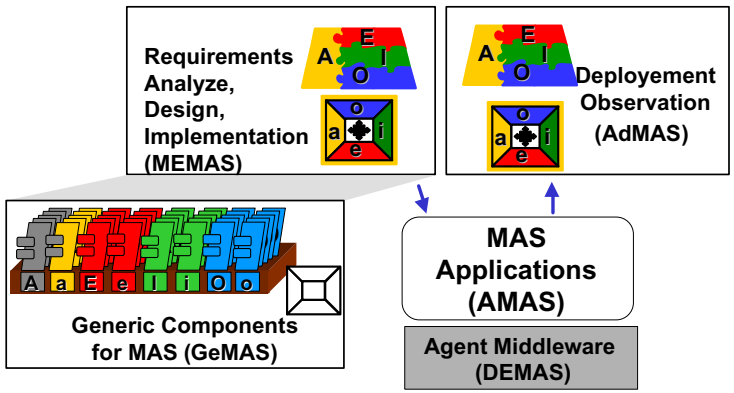
- Agent-based systems have a vital role to play in the immediate development of applications and services across the distributed and increasingly pervasive computing fabric of our everyday environments (stressed by the convergence of distributed computing and object-oriented development)
- Two conditions must hold :
  - Mainstream technologies must be used for infrastructural underpinning of agent applications to enable accessibility, further development, and, importantly, *integration*.
  - Second, the kinds of applications that we build must be constructed in ways that facilitate flexibility, evaluation, and the potential for secondary capabilities (that are still critical for many applications and environments) like mobility.
- Agreement on common infrastructure would enable that integration (use of other standards, …)

Conclusion

- Not only one MAOSE method BUT several
- Link between MAOSE Method and CASE Tool supporting this method
- BUT in MAS, due to the different models that are used, exist several ways to do MAOSE :
  - Agent Oriented Software Engineering
  - Environment Oriented Software Engineering
  - Interaction Oriented Software Engineering
  - Organization Oriented Software Engineering
- → Need to support Multi-Agent Oriented Development

Multi-Agent Oriented Development

Multi-Agent System Toolkit [SMA/SIMMO/ENSMSE]



MAST (Multi-Agent System Toolkit)

AgentCities

www.agentcities.org

- International Deployement of MAS Platforms (> 100)
  - Permanently accessible via Internet
  - Openness
  - FIPA Compliant
- Hosting multiple agent “services”
  - Interoperability between agent services
  - Experiment of composition of services / with added value
  - Experience on complex models and semantic descriptions

Bibliography

- L. Gasser, MAS Infrastructure, Definitions, Needs and Prospects, Autonomous Agent Workshop on Infrastructures for Agents and MAS, 00
- P.M. Ricordel, Y. Demazeau, From Analysis fo Deployment : a Multi-Agent Platform survey, Engineering Societies in the gents' World, (ESAW) 00
- R. Ashri, M. Luck, Towards a layered approach for agent infrastructure: the right tools for the right job, Autonomous Agent Workshop on Infrastructures for Agents and MAS, 00
- Review of different Agent tools : http://www.agentbuilder.com/AgentTools/index.html
- AgentBuilder : www.agentbuilder.com
- AgentTool : http://www.cis.ksu.edu/~sdeloach/ai/download-agentool.htm
- Jack : http://www.agent-software.co.uk
- Jade :
- Madkit : www.madkit.org, community.madkit.org
- Zeus : www.labs.bt.com/projects/agents/zeus