Towards a Role-Based Agent Development Environment for Open Multi-Agent Software Systems

Prof. Haiping Xu
Concurrent Software Systems Laboratory
Computer and Information Science Department
University of Massachusetts Dartmouth
North Dartmouth, MA 02747
Email: hxu@umassd.edu
URL: http://www.cis.umassd.edu/~hxu/

Acknowledgment

- **Prof. Xiaoqin Zhang**
  Neural and Intelligent Systems Laboratory
  Computer and Information Science Department
  University of Massachusetts Dartmouth
- **Rinkesh Patel**
  Concurrent Software Systems Laboratory
  Computer and Information Science Department
  University of Massachusetts Dartmouth

Outline

- Part 1: A Formal Framework for Role-Based Agent Modeling
  - Background and Motivation
  - An Organizational Approach
- Part 2: Development of Role-Based Open MAS
  - Three Layered Model-Driven Development Model
  - Case Study: Organizing a Conference
  - Role-Based Agent Development Environment (RADE)
- Conclusions and Future Work

Intelligent Agent – An AI Perspective

- From AI perspective, an agent is a computer system situated in some environment, that is capable of flexible, autonomous actions in order to meet its design objectives
- Agent properties include
  - Situatedness (reactiveness)
  - Autonomy (proactiveness)
  - Sociability (responsibility, communication capability, organization capability, etc.)

Software Agent – A Software Engineering Perspective

- From software engineering perspective, an agent can be considered as an *active* object, i.e., an object with a mental state
- A software agent is a program that acts on behalf of (human) user
- Example: air ticket seller agent and air ticket buyer agent in e-commerce

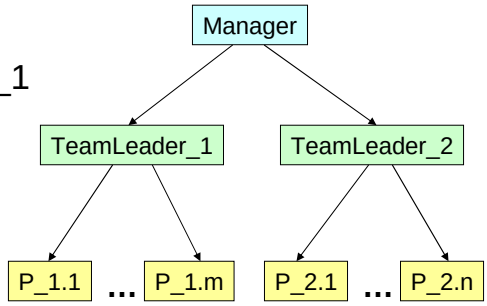
Open Multi-Agent System

- Multi-agent system (MAS) is a concurrent software system with more than one agent
- A traditional MAS consists of a fixed number of software agents
- In an open MAS, agents can
  - join or leave an agent society at will
  - take or release roles dynamically

An Example (Simulating a Company)

- Employee 1:
  - TeamLeader_1
  - P_1.1
- Employee 2:
  - P_1.m
  - P_2.2
- Employee X:

...



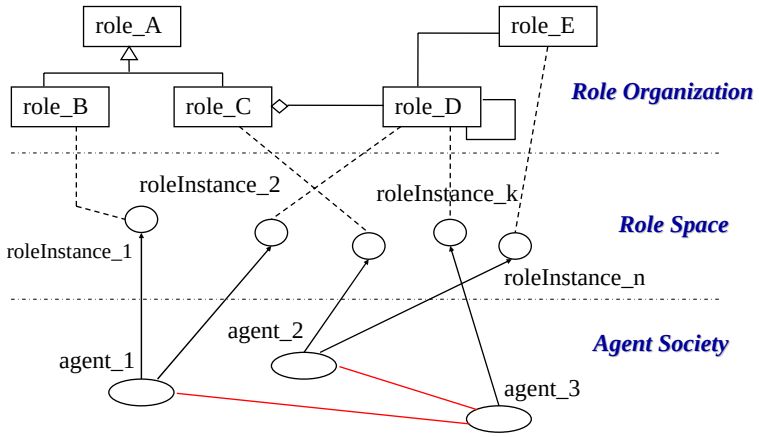
Role-Based Modeling

- Role-based modeling is one of the most effective methodologies for agent-based system analysis and design
- In most of the existing work
  - abstract constructs used to conceptualize and understand the system
  - no realizations in the implemented system
  - Suitable for closed multi-agent systems
- We propose a methodology for role-based modeling of open multi-agent systems

An Organizational Approach

- Separate the concepts of role organization and role space
  - role organization contains conceptual roles
  - role space contains role instances
- At the third layer, we define an agent society that consists of agent instances
- The agent society can be designed independently of the role organization and role space

A Generic Model of Role-Based Open MAS



Formal Specifications - Role and Role Organization

Role

$attributes : \mathbb{P}\ Attribute$
$goals : \mathbb{P}\ Goal$
$plans : \mathbb{P}\ Plan$
$actions : \mathbb{P}\ Action$
$permissions : \mathbb{P}\ Permission$
$protocols : \mathbb{P}\ Protocol$
$beTaken : \mathbb{B}$

Init
$permissions = \varnothing$
$protocols = \varnothing$
$beTaken = false$

setPermission
$\Delta permissions$
$perm? : Permission$
$permissions' = permissions \cup \{perm?\}$

addProtocol
$\Delta protocols$
$prot? : Protocol$
$procotols' = protocols \cup \{prot?\}$

RoleOrganization

$roles : \mathbb{P} \downarrow RoleMetaClass$
$relationship :$
$\quad \downarrow RoleMetaClass \times \downarrow RoleMetaClass \nrightarrow Relationship$
$\forall r1, r2 \in roles, r1 \neq r2 \bullet (r1, r2) \in \mathrm{dom}\ relationship$

Init
$roles = \{Role\}$

addRole
$\Delta roles, relationship$
$role? : \downarrow RoleMetaClass$
$role? \notin roles \ \land \ roles' = roles \cup \{role?\}$
$\exists r \in roles \bullet relationship' =$
$\quad relationship \cup \{(r, role?) \mapsto inheritance\}$

setRelationship
$\Delta relationship$
$r1?, r2? : \downarrow Role$
$rela? : Relationship$
$r1?, r2? \in roles \land rela? \neq inheritance \land$
$\quad relationship' = relationships \cup \{(r1?, r2?) \mapsto rela?\}$

Formal Specifications - Role Space and Agent

RoleSpace

$roleOrganization : RoleOrganization$
$\#roleOrganization.roles > 1$

$roleInstances : \mathbb{P} \downarrow Role$
$\forall ri \in roleInstances \bullet ri.getClass \in roleOrganization.roles$

Init
$roleInstances = \varnothing$

createRoleInstance
$\Delta roleInstances$
$ri? : \downarrow Role$
$ri?.getClass \in roleOrganization.roles$
$ri? \notin roleInstances \land roleInstances' = roleInstances \cup \{ri?\}$

deleteRoleInstance
$\Delta roleInstances$
$ri? : \downarrow Role$
$ri? \in roleInstances \land roleInstances' = roleInstances - \{ri?\}$

findRoleInstance
$\Xi roleInstances$
$ra? : Role.Attribute$
$ri! : \downarrow Role$
$(NotFound \land ri! = null) \lor$
$\quad \exists ri \in roleInstances \bullet ri.attributes = ra? \ \land \ ri! = ri$

Agent

$attributes : \mathbb{P}\ Attribute$
$motivations : \mathbb{P}\ Motivation$
$sensor : Environment \nrightarrow SensorData$
$reasoningMechanism :$
$\quad \mathbb{P}\ SensorData \times \mathbb{P}\ Motivation \rightarrow \mathbb{P}\ Goal \rightarrow \mathbb{P} \downarrow Role$
$rolesTaken : \mathbb{P} \downarrow Role$

Init
$rolesTaken = \varnothing$

takeRole
$\Delta rolesTaken$
$ri? : \downarrow Role$
$ri?.beTaken = false \land ri?.beTaken' = true$
$ri? \notin rolesTaken \land rolesTaken' = rolesTaken \cup \{ri?\}$

releaseRole
$\Delta roleTaken$
$ri? : \downarrow Role$
$ri?.beTaken = true \land ri?.beTaken' = false$
$ri? \in rolesTaken \land rolesTaken' = rolesTaken - \{ri?\}$

Role-Based MAS Design

- Design *Role* classes and their relationships
  - inheritance relationship
  - aggregation relationship
  - association relationship
  - incompatibility relationship
- To ease software engineer's effort, we propose a design process for MAS development

A Generic Procedure to Design Open MAS

1. Design the set of *Role* classes Ω and their relationship $\Pi 1$: Ω X $\Omega \rightarrow$ [IH | AG], where IH and AG represent the relationship types of *inheritance* and *aggregation*, respectively.
2. Design the role organization Φ according to the class schema *RoleOrganization*, and define any *association* relationships and *incompatibility* relationships between classes, i.e., $\Pi 2$: Ω X $\Omega \rightarrow$ [AS | IC], where AS and IC represent the relationship types of *association* and *incompatibility*, respectively.
3. Design the role space Γ according to the class schema *RoleSpace*. The role space Γ should support creating, advertising, and searching for role instances. It may use existing middleware, e.g., Sun Jini, for its purpose.
4. Refine the *Agent* class with a set of sensors and a set of appropriate reasoning mechanisms. This step may overlap with Step 1-3.
5. Design agent society Θ according to the class schema *AgentSociety*. The agent society Θ contains a set of agent instances of type *Agent*, and it corresponds to the role organization Φ with the same organization/society design purpose.

Open Role Space and Open Agent Society

- Open role space
  - role instances can be added into or deleted from a role space dynamically
- Open agent society
  - agents can join or leave the system at will
  - agents can take or release role instances in a role space dynamically

A-R Mapping

- *A-R mapping* is a process for agents from an agent society Θ to take or release role instances in a role space Γ.
- Both of agent society Θ and role space Γ are defined upon the role organization Φ.
- Formally, the *A-R mapping* is defined as follows

$$A\text{-}R\ mapping \mathrel{\widehat{=}} f : Agent \nrightarrow \mathbb{P} \downarrow Role$$

where *f* is a partial function mapping from an agent instance to a set of role instances.

The Process of A-R Mapping

1. *Initialization:* The agent society Θ makes a request to the role space Γ to instantiate the major *LeadingRole* class defined in the role organization Φ, and create a role instance for it.
2. *Role assignment*: for each agent α in the agent society Θ, do the following:
   a. When agent α receives any sensor data from its environment, it may decide to generate some new goals or subgoals based on the sensor data and agent α's motivations.
   b. With its reasoning mechanisms, agent α further deduce a set Ω of needed roles of types defined in the role organization Φ. If none of the roles in set Ω is of type *LeadingRole*, go to step 2.d.
   c. If any role in role set Ω is a leading role of type *LeadingRole*, agent α takes the corresponding role instance from the role space Γ, if available, updates the hiring number of other roles as needed, and makes requests to the role space Γ to create role instances for those roles under hiring.
   d. Repeat the following for a period of time Υ: Search the role space Γ for any role instances that match roles in role set Ω. If there is a match, agent α takes that role instance. If all roles in role set Ω have been matched with some role instances in the role space Γ, go to Step 3.
   e. If any role in the role set Ω cannot be matched with a role instance in the role space Γ, agent α may decide to release all role instances or keep its current occupations.
3. *Marking role incompatibility:* for each agent α, mark its role incompatibility as the following: for any role instances *r1*, *r2* $\in \alpha.$*rolesTaken*, if $\Phi.$*relationship(r1.getClass, r2.getClass)* == *incompatibility*, mark agent α as potential *role incompatibility* with a self-loop.
4. *Setting up interaction relationships:* for each agent α, set up the interaction relationship between agent α and other agents from the same agent society Θ as the following : for any agent instance $\beta \in \Theta.$*agentInstances*, where $\alpha \neq \beta$, if $\exists$ *r1* $\in \alpha.$*rolesTaken*, *r2* $\in \beta.$*rolesTaken* such that $\Phi.$*relationship(r1.getClass, r2.getClass)* == *association*, then $(\alpha, \beta) \in$ dom $\Theta.$*interaction*.

Outline

- Part 1: A Formal Framework for Role-Based Agent Modeling
  - Background and Motivation
  - An Organizational Approach
- Part 2: Development of Role-Based Open MAS
  - Three Layered Model-Driven Development Model
  - Case Study: Organizing a Conference
  - Role-Based Agent Development Environment (RADE)
- Conclusions and Future Work

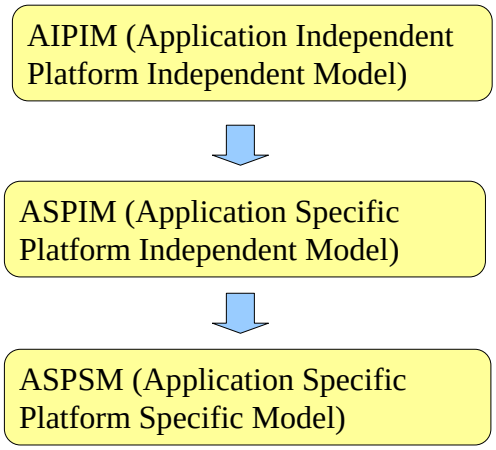
Development of Open MAS

- "The sooner you start, the longer it takes." by Fred Brook
  - Need to spend time on requirements capture
  - Need to spend time on software design
- Propose our model-driven development of open MAS
- Develop a Role-based Agent Development Environment (RADE) to support rapid application development (RAD).

Model-Driven Development of Role-Based Open MAS

- Inspired by the Model-Driven Architecture (MDA), proposed by OMG
- We propose a three layered development model
  - Separation of concerns: architecture domain, application domain, solution domain
  - Support automatic or semi-automatic rapid application development (RAD) of open MAS

Three Layered Development Model



Application Independent Platform Independent Model (AIPIM)

- Defines a high level of abstraction that is independent of any particular applications and any implementation technology
- An AIPIM is typically suitable for a set of applications
  - Mobile agent model
  - Multi-agent system model
  - Role-based agent model

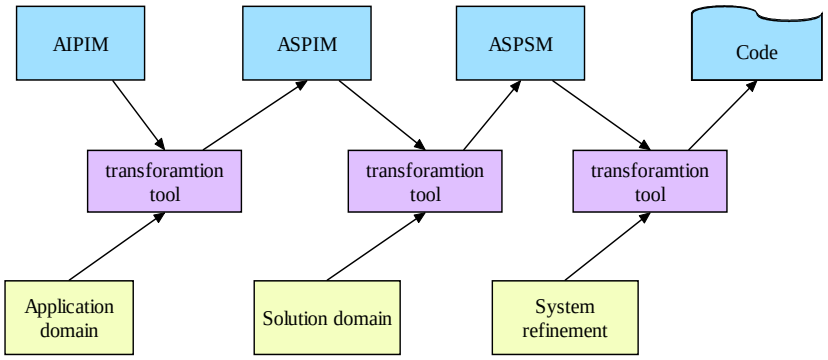
Application Specific Platform Independent Model (ASPIM)

- Defines a high level of abstraction
  - Specific to a particular application
  - Independent of any implementation technology
- Needs application domain knowledge
- Describes a software system that supports some business logic.

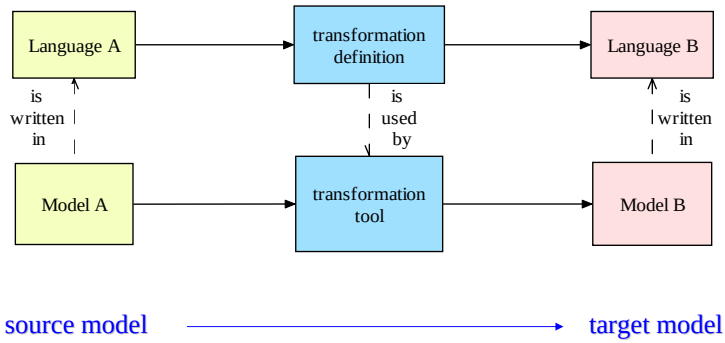
Application Specific Platform Specific Model (ASPSM)

- Defines an abstraction of the software system
  - Specific to a particular application
  - Specific to an implementation technology
- Specifies the software system in terms of some specific implementation technology
  - J2EE, EJB, Java Servlets
  - Microsoft .Net, C#
  - IBM Websphere, web services technology

Model-Driven Development Process



Transformation between Two Models



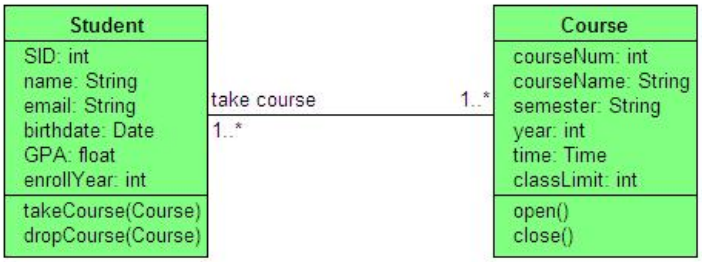
Example: Transformation Definition

- Suppose model A is written in UML and model B is a relational database model
- A transformation definition that translates an association in UML into a foreign key relation may look as follows

A Transformation Rule

```
if the association A to B is adorned by an association class
   or the multiplicity at both ends is more-than-one
then create a table representing the association class or the
     association
     and create foreign keys in both the table representing A
     and the table representing B referring this new table
else if the multiplicity at one end is zero-to-one
   then create a foreign key in the table representing the
        class at that end, referencing the other end
   else // the multiplicity of the association is one-to-one
        create a foreign key in one of the tables,
        referencing the other end
   end if
end if
```

Association Relationship



Relational Database Schema

STUDENT

SID	NAME	EMAIL	BDATE	GPA	EYEAR

COURSE

CNUM	CNAME	SEMEST	YEAR	TIME	CLIMIT

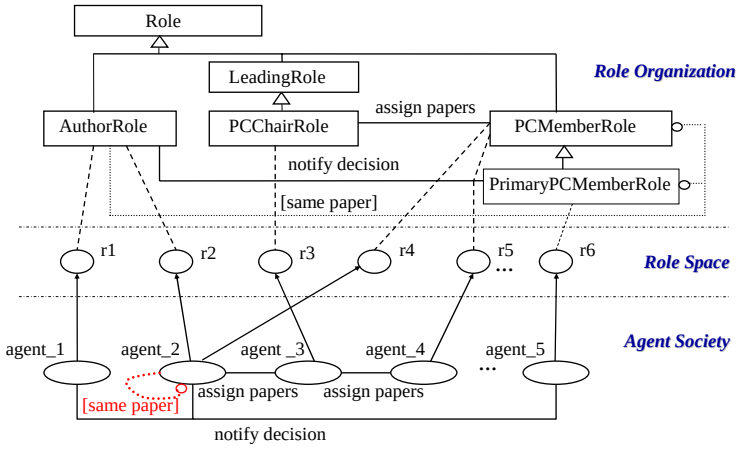
COURSETAKEN

SID	CNUM	SEMEST	YEAR

Case Study: Organizing a Conference

- Involves the following major processes
  - Submit papers by authors
  - Assign program committee members
  - Assign primary program committee members
  - Assign papers to (primary) program committee members
- Goal: Automate (or semi-automate) the above processes using agent technology

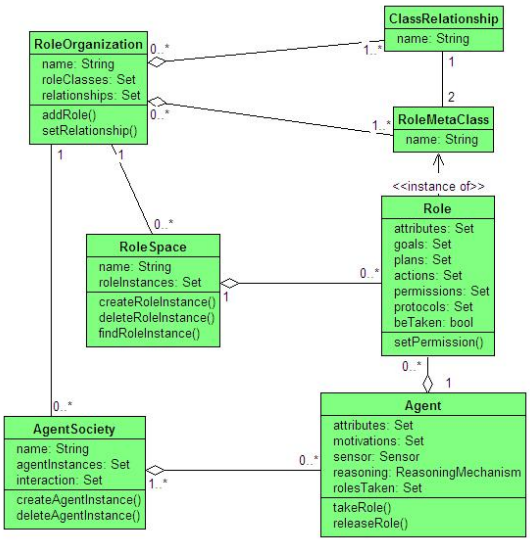
An Open MAS for Organizing a Conference



Design of the AIPI Model

- Is based on the role-based agent formal framework
- Is independent of the application of organizing a conference
- Can be reused for development of other role-based open MAS applications

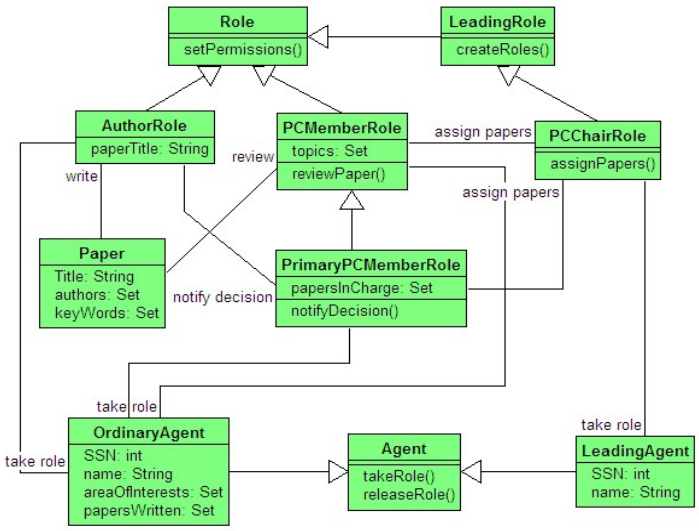
AIPIM



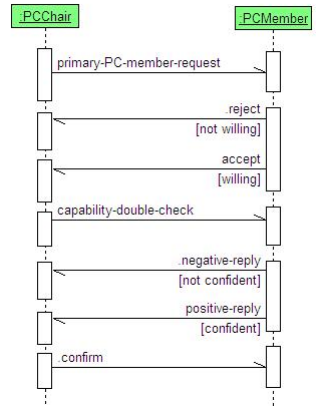
Design of the ASPI Model

- Define the *LeadingRole* class
  - Each role organization defines a leading role
  - Is responsible for creating other role instances
- Define role classes in role organization
  - *PCChairRole* as a subclass of *LeadingRole*
  - *PCMemberRole*, *PrimaryPCMemberRole*, and *AuthorRole* as subclasses of the *Role* class
- Define *LeadingAgent* and *OrdinaryAgent*

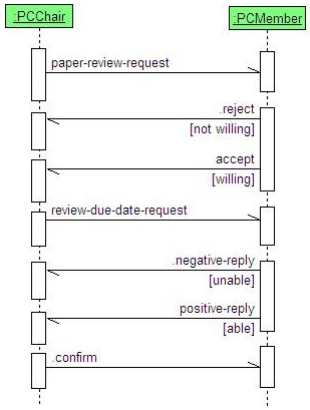
ASPIM



Examples of Agent Interaction Protocol (AIP)



Request for Primary PC Member

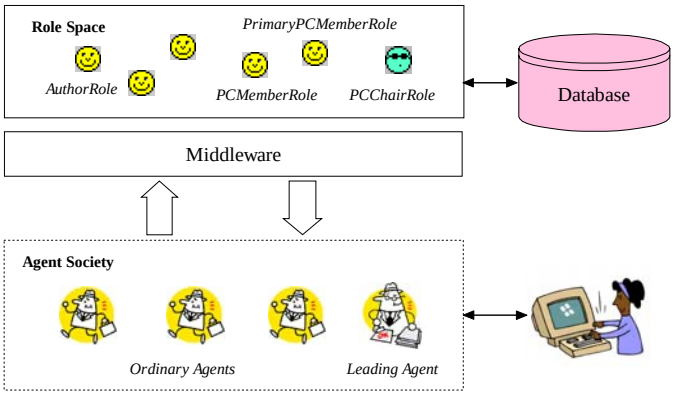


Request for Paper Reviewer

Design of the ASPS Model

- Design the role space as a server
  - Contains role instances
  - Provides two interfaces: Interface for the leading agent and interface for ordinary agents
- The database server is behind the role space
- Each agent can run on a different machine
  - An agent communicate with the role space through middleware
  - An agent society is chaired by the leading agent
  - Agents communicate with each other using AIP

Architecture of Agent-Based Conference Organizer



Design of Database Schema

PCMEMBER

MID	ATTR	GOAL	PLAN	ACTION	AID

AGENT

AID	ATTR	MOTI	SENS	REASM

PAPER

PID	TITLE	KEYWDS	PAGES

AUTHOR

TID	ATTR	MOTI	SENS	REASM	AID

WRITEPAPER

AID	PID

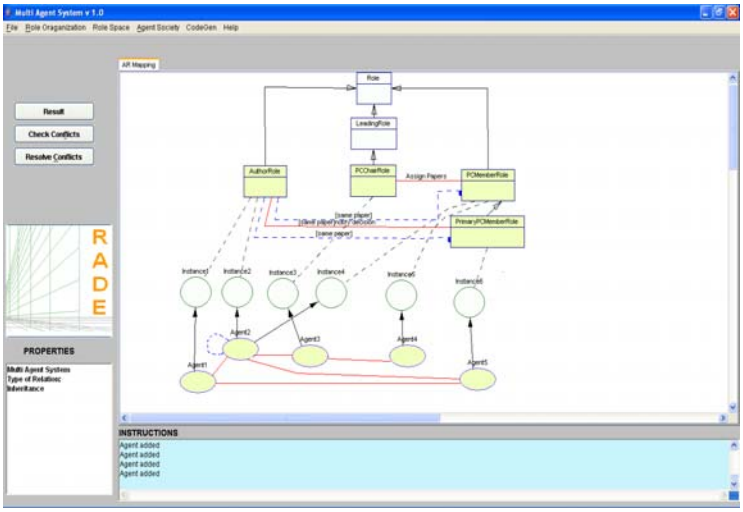
Choosing the Right Middleware

- Communication support
  - between agents and the role space
  - between the leading agent and ordinary agents
- Middleware functionalities
  - Service provider *vs.* service consumer
  - Service publisher *vs.* service subscriber
  - Synchronous vs. asynchronous message passing
  - Security issues
- Middleware options: RMI, CORBA, Sun Jini, Web Services, etc.

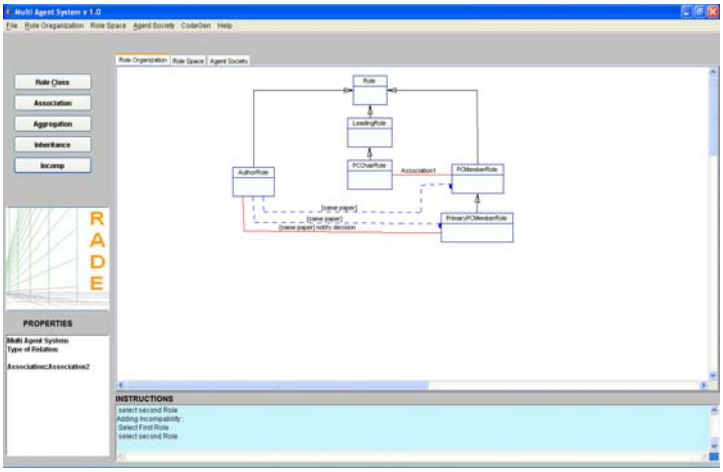
Role-Based Agent Development Environment (RADE)

- To support rapid application development (RAD) of open MAS
- To provide a friendly graphical user interface for software development
- To provide automated or semi-automated tools for model transformation
- To automatically generate code based on ASPSM

A Prototype of RADE

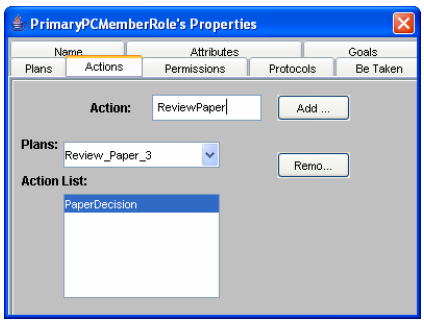


Graphical Editor for Role Organization

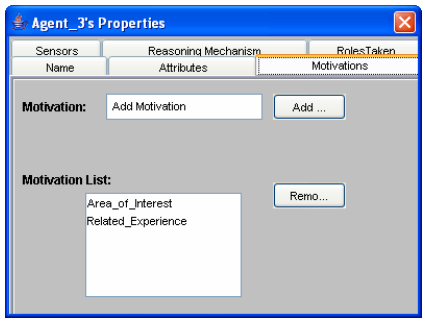


Some User Interfaces

Role Properties



Agent Properties

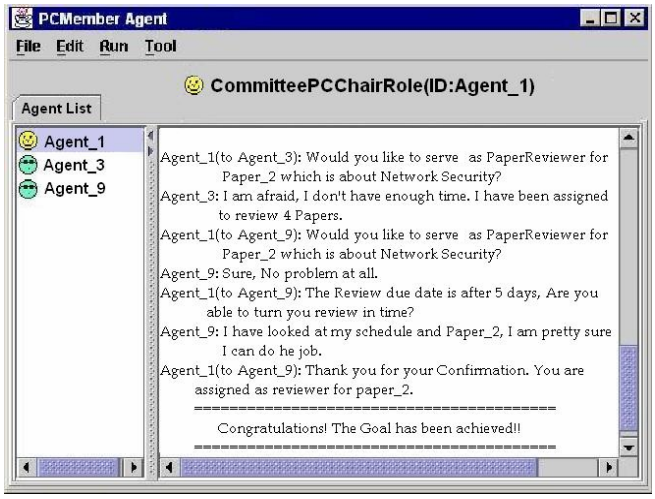


Role Assignment and Paper Assignment

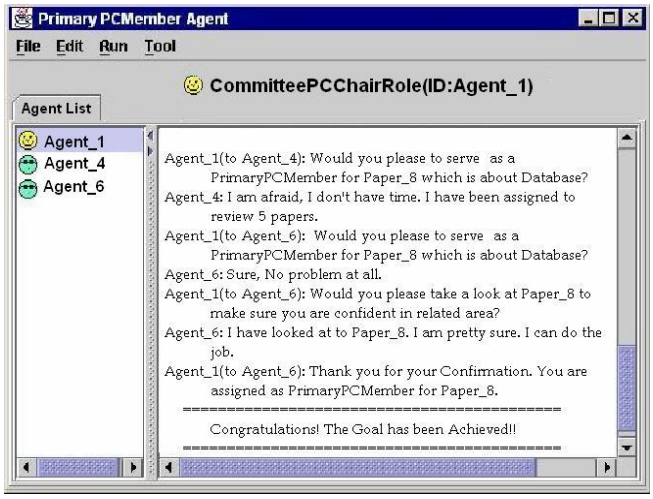
Simulation Result:: Reviewer

Paper Decision	PCMemberRole	PCMemberRole	PCMembeRole	PCMemberRole	PCMemberRole	PCMemberRole	PCMemberRole
Paper_4	Agent_6	Agent_4	Agent_3	Agent_7	Agent_9	Agent_12	Agent_5
Paper_8	Agent_6	Agent_8	None	None	None	None	None
Paper_3	Agent_12	Agent_3	Agent_5	Agent_8	Agent_10	Agent_13	None
Paper_1	Agent_5	Agent_1	Agent_7	Agent_3	None	None	None
Paper_5	Agent_8	Agent_9	Agent_10	Agent_11	Agent_14	None	none
Paper_7	Agent_3	Agent_1	Agent_10	Agent_5	Agent_2	Agent_7	Agent_12
Paper_2	Agent_1	Agent_2	None	None	None	None	None
Paper_9	Agent_10	Agent_5	Agent_7	Agent_8	Agent_11	None	None
Paper_6	Agent_7	Agent_10	Agent_13	None	None	None	None
Paper_10	Agent_1	Agent_12	None	None	None	None	None
Paper_11	Agent_7	Agent_4	None	None	None	None	None
Paper_12	Agent_4	None	None	None	None	None	None
Paper_13	Agent_5	Agent_6	Agent_8	Agent_12	None	None	None

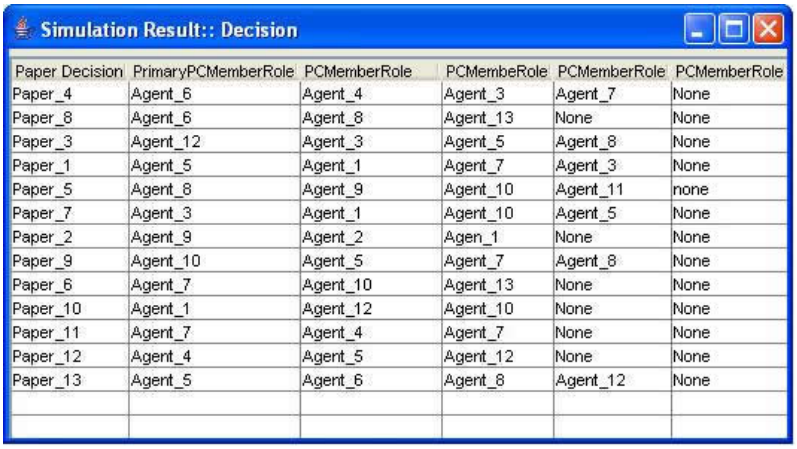
Conversation Example - 1



Conversation Example - 2



EXPECTED Final Schedule



Paper Decision	PrimaryPCMemberRole	PCMemberRole	PCMembeRole	PCMemberRole	PCMemberRole
Paper_4	Agent_6	Agent_4	Agent_3	Agent_7	None
Paper_8	Agent_6	Agent_8	Agent_13	None	None
Paper_3	Agent_12	Agent_3	Agent_5	Agent_8	None
Paper_1	Agent_5	Agent_1	Agent_7	Agent_3	None
Paper_5	Agent_8	Agent_9	Agent_10	Agent_11	none
Paper_7	Agent_3	Agent_1	Agent_10	Agent_5	None
Paper_2	Agent_9	Agent_2	Agen_1	None	None
Paper_9	Agent_10	Agent_5	Agent_7	Agent_8	None
Paper_6	Agent_7	Agent_10	Agent_13	None	None
Paper_10	Agent_1	Agent_12	Agent_10	None	None
Paper_11	Agent_7	Agent_4	Agent_7	None	None
Paper_12	Agent_4	Agent_5	Agent_12	None	None
Paper_13	Agent_5	Agent_6	Agent_8	Agent_12	None

Conclusions

- A role-based methodology has been proposed for development of open MAS
- The design of roles and agents can be separated, which simplifies agent development
- A three-layered agent development model is proposed
- A prototype of RADE is built to show the feasibility of our approach (in progress)

Future Work

- Develop and demonstrate the RADE prototype with case studies
- Design automatic model transformation tools
- Incorporate agent negotiation mechanisms for agent communications
- Develop the Role-based Agent Development Environment (RADE) for rapid application development of open MAS

Thank you for your attention!

The slides for this talk can be downloaded from

http://www.cis.umassd.edu/~hxu