Agent-Grid Intelligence Platform for Collaborative Working Environment

Zhongzhi Shi
shizz@ics.ict.ac.cn

Institute of Computing Technology
Chinese Academy of Sciences
Acknowledgement

- China Important Basic Research Programme 973
- China High-Tech Programme 863
- National Natural Science Foundation of China
- Ministry of Information Industry, China
- Knowledge Innovative Programme of CAS
Self-Introduction

- Zhongzhi Shi is Professor at the Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, IEEE Senior member.
- He is Chair of WG 12.2 of IFIP. He also serves as Vice President of Chinese Association for Artificial Intelligence.
- He received the 2nd Grade National Award of Science and Technology Progress in 2002. In 1998 and 2001 he received the 2nd Grade Award of Science and Technology Progress from the Chinese Academy of Sciences.
- His research interests include intelligence science, multiagent systems, Semantic Web, machine learning and data mining.
- He published 10 books, edited 11 books and more than 350 technical papers.
Outline

- Introduction
- Agent Model
- Multiagent Environment— MAGE
- Agent Collaboration
- Agent Grid Intelligence Platform
- Applications
- Conclusions
Agent computing is particularly well suited to the collaborative work. The agent-based computing paradigm has following features:

- **Autonomy** - agent operate without intervention;
- **Social ability** – agents interact each other using an agent communication language;
- **Goal driven** – agent exhibit goal-directed behavior;
- **Reactivity** – agents perceive and respond to their environment.
Semantic Web will provide well-defined meaning which better enabling computers and people to work in collaboration.
Grid Computing

- Flexible, secure, coordinated resource sharing among dynamic collections of individuals, institutions, and resource

From “The Anatomy of the Grid: Enabling Scalable Virtual Organizations”

- Grid Forum
  - www.gridforum.org
Brain Meet Brawn

Ian Foster, Nicolas R. Jennings, Carl Kasselman. Brain Meet Brawn: Why Grid and Agents need each others. AAMAS’04

The Grid community has historically focused on "brawn": infrastructure, tools, and applications for reliable and secure resource sharing within dynamic and geographically distributed virtual organizations.

The agents community has focused on "brain": autonomous problem solvers that can act flexibly in uncertain and dynamic environments.
Research on MAS

1994: Multiagent Processing Environment MAPE
1996: Agent-based CSCW
1998: Common Agent Request Broker Architecture CARBA, MAPE2
2000: Multiagent Environment MAGE
2002: Agent Grid Intelligence Platform AGrIP
2003: Dynamic Description Logic DDL
2004: Visual Agent Developing Tool VAStudio
2005: Ontology-based Knowledge Management KMSphere
BDI Model

Environment

Beliefs

FIPA

Goals

Plans

Beliefs

Goals
Description Logic

- Concepts and Role
- Tbox— Assertions
- Abox— Instance
- Reasoning mechanism in terms of Tbox and Abox
Reasoning in DL

1) Subsumption
2) consistency
3) satisfiability
4) instance checking
**TBox(Scheme)**

Man = Human ⊓ Male

Happy-father = Human ⊓ ∃ Has-child. Female ⊓ ...

**Abox(Data)**

John: Happy-father

<John, Mary>: Has-child
The primitive symbols

- Concept name: \( C_1, C_2, \ldots \);
- Role name: \( R_1, R_2, \ldots \);
- Individual constant: \( a, b, c, \ldots \);
- Individual variable: \( x, y, z, \ldots \);
- Concept operation: \( \neg, \cap, \cup, \exists, \forall \);
- Axiom operation: \( \neg, \land, \rightarrow \forall \);
- Action: \( A_1, A_2, \ldots \);
- Action construction: ; (composition), \( \cup \) (alternation), \( * \) (repeat), \( ? \) (test);
- Action variable: \( \alpha, \beta, \ldots \);
- Axiom variable: \( \varphi, \psi, \pi, \ldots \);
- State variable: \( u, v, w, \ldots \);
Concepts in DDL are defined as the following:

- (1) Primitive concept $P$, top $\top$ and bottom $\bot$ are concepts.
- (2) $\neg C$, $C \cap D$, $C \cup D$ are concepts.
- (3) $\exists R.C$, $\forall R.C$ are concepts.
An action description is the form of

\[ A(x_1, \ldots, x_n) = (P_A, E_A) \]

where

1. \( A \) is the action name.
2. \( x_1, \ldots, x_n \) are individual variables, which denote the objects the action operate on.
3. \( P_A \) is the set of preconditions, which must be satisfied before the action is executed.
4. \( E_A \) is the set of results, which denote the effects of the action.
DDL Semantics

- Actions in DDL are defined as the following:
  - Atom action $A(a_1, \ldots, a_n)$ is action.
  - If $\alpha$ and $\beta$ are actions, then $\alpha ; \beta$, $\alpha \cup \beta$, $\alpha^*$ are actions;
  - If $\varphi$ is an assertion formula, then $\varphi ?$ is action.
Metal State Model

Mental State: $\langle K, A, G, P, I \rangle$,

Where

- $K$ belief
- $A$ action
- $G$ goal
- $P$ plan
- $I$ intention
Belief

\[ K = <T, S, B> \]

**T**: Ontology

**S**: Constrains

**B**: Current belief
Belief Revision

\textbf{AddBelief}(F, B) \{ 

\begin{align*}
F' & \leftarrow \text{Extend}(F); \\
\text{Foreach} \ \varphi \in F' \ & \text{do} \\
\quad \text{If} \ \neg \varphi \in B & \text{ Then } B \leftarrow B - \{\neg \varphi\}; \\
B' & \leftarrow \text{Extend}(B \cup F); \\
\text{If} \ \text{Consistent}(B') & \text{ Then } \text{Return } B' \; ; \\
\text{Else}\{ \\
\quad \text{Let} \ \{\varphi, \neg \varphi\} = \text{ConflictSet}(B'); \\
\quad \text{If} \ \varphi \in B & \text{ Then Return } B' - \{\varphi\}; \\
\text{Else If} \ \neg \varphi \in B & \text{ Then Return } B' - \{\neg \varphi\}; \\
\quad \text{Else Return error}; \\
\}\}
\}

}\quad 2006-6-3 \quad \text{AGri P-SELMAS Zhongzhi Shi}
Goal

Let $A$ be a set of actions, $L$ be a set of assertions, $G$ can be defined recursively:

1. $A \subseteq G$, $A$ is basic action;
2. If $\phi \in L$, then $\text{achieve}(\phi) \in G$;
3. If $\phi \in L$, then $\phi? \in G$;
4. If $\delta_1, \delta_2 \in G$,
   then $\delta_1; \delta_2 \in G$, $\delta_1 \cup \delta_2 \in G$, $\delta_1^* \in G$;
Static Plan

\[ \delta_1 \leftarrow \varphi \mid \delta_2 \]

\[ \delta_1 \in \mathcal{G}, \quad \delta_2 \in \mathcal{G}, \quad \varphi \in \mathcal{L}. \]

\( \delta_1 \): Rule header
\( \delta_2 \): Rule body
\( \varphi \): Rule guard condition

\[ \text{go_floor}(x) \leftarrow \@\text{At} (\text{elevator}, y) \mid [y<x?; \text{up}(x) \cup y>x?; \text{down}(x)] \]
Dynamic Plan

\[ \text{Plan}(\delta, B) \]  // Goal $\delta$, $B$ Current belief set

\{ 

If perform action $\alpha B \vdash T^\delta$ Then
\{ Return $B$; Enqueue($\alpha, P$); \}

Else \{

look for subgoals to reach goal;
computing $E_\alpha$ and $\delta$ for each subgoal;
computing the order of subgoals
select optimal subgoal plan;
sorting subgoals and form sequence $\delta_1, ..., \delta_n$

For $\pi = \delta_1$ to $\delta_n$ do

$B = \text{Plan}(\pi, B)$
\}
Petri Net

OWL-S Interpreter → DDL

OWL-S Generator

Services

Incidences matrix + DDL
Emotion Agent

- Sensor
- Belief
- Effector
- Intention
- Emotion KB
- Emotion Inf.
- Emotion Inf. (Planning)
- Rational Inf.
- Desire
- Environment

[Diagram showing the relationships between the components of an emotion agent]
Agent Life Cycle

- Waiting
- Suspended
- Transit
- Initiated
- Active
- Wake Up
- Resume
- Move
- Execute
- Invoke
- Destroy
- Quit
- Create
Outline

- Introduction
- Agent Model
- Multiagent Environment— MAGE
- Agent Collaboration
- Agent Grid Intelligence Platform
- Applications
- Conclusions
Multiagent Environment MAGE

System Development

- Behaviour Library
- Agent Library
- Agent Society

System Deployment

AUMP

Requirement Analysis

System Design

VAStudio

MAGE Running Support

2006-6-3
AGriP-SELMAS Zhongzhi Shi
## Multiagent System Design Procedure

<table>
<thead>
<tr>
<th>Function Description</th>
<th>Use Cases Model</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Behavior Description</strong></td>
<td><strong>Agent Description</strong></td>
</tr>
<tr>
<td>Activity Model</td>
<td>State model</td>
</tr>
<tr>
<td>Reaction Rule</td>
<td>Inference Model</td>
</tr>
<tr>
<td>Inference Model</td>
<td></td>
</tr>
<tr>
<td>Plan Model</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2006-6-3  AGrI P-SELMAS Zhongzhi Shi  31
AUML—Use Cases

Actor

use case: goal

system boundary

association
generalization

<<extend>>
extend

<<include>>
include

System1

Use Case 3:
Goal3

Use Case 2:
Goal2

Use Case 1:
Goal1

Use Case 5:
Goal5

Use Case 4:
Goal4

Actor1

Actor2
AUML—Interactive Diagram

- Role
- Lifeline
- Timelimit split
- Thread
- And split
- Asynchronous message
- Or split
- Synchronous message
- Or combination
- Decision split
- Decision combination
- XOR split
- XOR combination
- Com-act: content
- Terminate of thread

Role1: QueryRef1, Cancel, InformRef4
Role2: QueryRef2, InformRef1
Role3: InformRef2
AUML—State Diagram

State diagram with states, events, and transitions.

- States:
  - State 1
  - Initial state
  - Final state

- Events/Actions:
  - Event 1/Action 1
  - Event 2/Action 2
  - Event 3/Action 3
  - Event 4/Action 4
### Event List (自定义事件表)

<table>
<thead>
<tr>
<th>Event Name</th>
<th>In Params</th>
<th>Out Params</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event1</td>
<td></td>
<td>int param1</td>
<td>Description of Event1 and param1.</td>
</tr>
</tbody>
</table>

### Action List (自定义动作表)

<table>
<thead>
<tr>
<th>Action Name</th>
<th>In Params</th>
<th>Out Params</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action1</td>
<td></td>
<td>int param1</td>
<td>Description of Action1 and param1.</td>
</tr>
</tbody>
</table>

### Reaction Rule List (反应规则表)

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>Event</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule1</td>
<td>Event1</td>
<td>Action1</td>
<td>Description of Rule1.</td>
</tr>
</tbody>
</table>

### Reaction Rule Lib List (反应规则库表)

<table>
<thead>
<tr>
<th>RuleLib Name</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReactionRuleLib1</td>
<td>Rule1, Rule2,</td>
</tr>
</tbody>
</table>
### Fact Predication List (事实谓词表)

<table>
<thead>
<tr>
<th>Predication</th>
<th>Meanings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predication1</td>
<td>Meanings of Predication1</td>
</tr>
</tbody>
</table>

### Rational Rule List (推理规则表)

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>Conditions</th>
<th>Conclusion</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule1</td>
<td>Condition1</td>
<td>Conclusion1</td>
<td>Description of Rule1.</td>
</tr>
</tbody>
</table>

### Rational Rule Lib List (推理规则库表)

<table>
<thead>
<tr>
<th>RuleLib Name</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>RuleLib1</td>
<td>Rule1, Rule2,</td>
</tr>
</tbody>
</table>
### AUML—Plan Model

<table>
<thead>
<tr>
<th>Environment Predication List (环境描述谓词表)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predication</td>
</tr>
<tr>
<td>Predication1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action Template List (动作模板表)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Action Template Name</strong></td>
</tr>
<tr>
<td>Action 1</td>
</tr>
</tbody>
</table>

### Action Lib (动作库)

<table>
<thead>
<tr>
<th>Action Template</th>
<th>Role</th>
<th>Agents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action 1</td>
<td>Role 1</td>
<td>Agent 1, Agent 2</td>
</tr>
<tr>
<td>Action 1</td>
<td>Role 2</td>
<td>Agent 1</td>
</tr>
</tbody>
</table>
AUML—Agent Class Diagram

agent class
goal
role: model
ontology-class
reaction-rule-lib
action-lib
rational-rule-lib

activity
:Goal1

role1, role2

agent class

Agent-class1
Goal1
Role1: Activity1
Ontology-class1

Activity1
:Goal1

Role1, Role2

Agent-class2
Goal2
Activity1: Role1, Role2
Activity2: Role1
Ontology-class1
Action-lib1
AUML—Organization Diagram

Organization

Agent 1
control
control
control
control
benevolence
benevolence
dependency
dependency
dependency

Agent 2
Agent 3
Agent 4
Agent 5
peer
benevolence
dependency
peer
benevolence
dependency
AUMP — Source Code

![Class diagram]

- Behaviour
  - action
  - done
  - onStart
  - onEnd
  - block
  - restart

- CompositeBehaviour
- SampleBehaviour
- OneShotBehaviour
- CyclicBehaviour
- SequentialBehaviour
- ParallelBehaviour
- FSMBehaviour
Compare AUML with UML

- **ModelElement**
- **Goal**

- **Feature**
  - **StructuralFeature**
    - **Attribute**
    - **Operation**
    - **Method**
  - **BehavioralFeature**

- **Class**
  - **Generalization**
  - **Association**

- **Agent Class**
  - **Goal**

- **Relationship**
  - **Organization**
  - **Dependency**
  - **End**
    - **Control**
    - **Peer**
    - **Beneficence**
    - **Dependency**
Compare AUML with UML

Function
- Use Cases Model

Behavior
- Activity Model
- State Model
- Interactive Model
- Reaction Model
- Inference Model
- Plan Model

Agent
- Agent Model
- Organization Model

Ontology
- Ontology Model

Platform
- Platform Model
Agent Unified Modeling Platform — AUMP
Computation Model for Agent

Agent
Client/Service

Agent
Client/Service

Soft Bus

Agent Service
Common Object Request Broker Architecture
Common Agent Request Broker Architecture

Application Pattern

Application Facilities

Agent Request Broker

Agent Services
VAStudio Architecture

Agent Society

Agents

Behaviours

Editor

Agent Library

Behaviour Library
VAStudio Architecture

**VAStudio Design Platform**
- Design GUI
  - Template Library
  - Composition Agent
  - Clone
  - FSM

**VAStudio Develop Platform**
- Develop GUI
  - Edit
  - Compile
  - Debug
  - Test

**VAStudio Run Platform**
- Run Monitor
  - Agent WorkList
  - ADL/BDL Loader
  - Reasoner Loader
  - Process Entity

**Library Interface**
- MAGE

**WS Interface**
- ADL/BDL Loader

**Ontology Interface**
- Reasoner Loader
Behavior Library

- Data Package
  package intsci.ace.data
- Neural Network Package
  package intsci.ace.neural
- Learning Package
  package intsci.ace.learning
- Data Mining Package
  package intsci.ace.mining
- Language Processing Package
  package intsci.ace.language
Behavior Library

- Graphic Package
  ```java
  package intsci.ace.graphics
  ```
- Image Package
  ```java
  package intsci.ace.image
  ```
- Search Package
  ```java
  package intsci.ace.search
  ```
- Expert System Package
  ```java
  package intsci.ace.expert
  ```
- Model Package
  ```java
  package intsci.ace.model
  ```
- Decision Making Package
  ```java
  package intsci.ace.decision
  ```
Screenshot of VAStudio
VAStudio: Visual Agent Studio

Select agent society
Select behavior from behavior lib.
Create Agent by VAStudio

Add behavior
Create Agent by VAStudio

Fill out behavior parameters
Create Agent by VAStudio

Build agent (code generation)
Build Multiagent System by VAStudio

Auction protocol using FSM
Build Multiagent System by VAStudio

Seller and buyer using clone
Build Multiagent System by VAStudio

Compile to generate the auction system
Build Multiagent System by VAStudio

Run the auction system in MAGE
Build Multiagent System by VAStudio

Display the agent running procedure
Build Multiagent System by VAStudio

Seller agent running
Run-Time Platform

MAGE

Directory Facilitator
Agent Management System
Agent
Message Transport System (MTS)

Software
Agent Library
Function Component

Other Agent Platforms
AgentCities

Beijing!
<table>
<thead>
<tr>
<th></th>
<th>AgentBuilder</th>
<th>Jack</th>
<th>Zeus</th>
<th>MAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Analysis</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Completeness</td>
<td>★★★</td>
<td>★★</td>
<td>★★★★</td>
<td>★★★★</td>
</tr>
<tr>
<td>Applicability</td>
<td>★★★</td>
<td>★★</td>
<td>★★★★</td>
<td>★★★★</td>
</tr>
<tr>
<td>Complexity</td>
<td>★★★★★</td>
<td>★★</td>
<td>★★★★</td>
<td>★★★</td>
</tr>
<tr>
<td>Reusability</td>
<td>★★★</td>
<td>★★</td>
<td>★★★★</td>
<td>★★★★</td>
</tr>
<tr>
<td><strong>Design</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Completeness</td>
<td>★★</td>
<td>★</td>
<td>★★★★</td>
<td>★★★★</td>
</tr>
<tr>
<td>Applicability</td>
<td>★★★</td>
<td>★</td>
<td>★★★★</td>
<td>★★★★</td>
</tr>
<tr>
<td>Complexity</td>
<td>★★★★★</td>
<td>★</td>
<td>★★★</td>
<td>★★★</td>
</tr>
<tr>
<td>Reusability</td>
<td>★★</td>
<td>★★</td>
<td>★★★★</td>
<td>★★★★</td>
</tr>
<tr>
<td><strong>Development</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Completeness</td>
<td>★★★★★★★</td>
<td>★★★★</td>
<td>★★★★</td>
<td>★★★★</td>
</tr>
<tr>
<td>Applicability</td>
<td>★★★</td>
<td>★★★★</td>
<td>★★★★</td>
<td>★★★★</td>
</tr>
<tr>
<td>Complexity</td>
<td>★★★★★</td>
<td>★★★★</td>
<td>★★★★</td>
<td>★★★★</td>
</tr>
<tr>
<td>Reusability</td>
<td>★★</td>
<td>★★★★</td>
<td>★★★★</td>
<td>★★★★</td>
</tr>
<tr>
<td><strong>Deployment</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Completeness</td>
<td>★★★</td>
<td>★★★★</td>
<td>★★★★</td>
<td>★★★★</td>
</tr>
<tr>
<td>Applicability</td>
<td>★★★</td>
<td>★★★★</td>
<td>★★★★</td>
<td>★★★★</td>
</tr>
<tr>
<td>Complexity</td>
<td>★★★★★</td>
<td>★★★★</td>
<td>★★★★</td>
<td>★★★★</td>
</tr>
<tr>
<td>Reusability</td>
<td>★★</td>
<td>★★★★</td>
<td>★★★★</td>
<td>★★★★</td>
</tr>
</tbody>
</table>
Outline

- Introduction
- Agent Model
- Multiagent Environment—MAGE
- Agent Collaboration
- Agent Grid Intelligence Platform
- Applications
- Conclusions
Agent Collaboration

- ACL
- Working Flow
- Ontology-based Knowledge Management
- Policy Driven
- Planning
Ontology

- In philosophy, an ontology is a theory about the nature of existence.
- An ontology is a document or file that formally defines the relations among terms.
- An ontology is a formal, explicit specification of a shared conceptualization.
- The most typical kind of ontology for the Web has a taxonomy and a set of inference rules.
Semantic Web Layer Cake

- 10 Feb 2004
- W3C Recommendation
- OWL

by Tim Berners-Lee
Collaborative Architecture

Agent A

Owner

Goal Instance

has(1,n)

uses(1,n)

Collaboration Service

Agent B

Owner

Goal Instance

has(1,n)

uses(1,n)

Collaboration Service

compatible goals = collaboration partners

automated collaboration execution

Goal Template Repository

Service Repository

owns

goal assignment

service discovery

KMSphere

Web Service

Web Service
KMSphere
KMSphere Demo

Create ontology by hand
Ontology acquisition from databases
Ontology acquisition from text
Edit ontology
KMSphere Demo

Ontology consistency check
RDQL (RDF Data Query Language)
Semantic Web Services

- Define exhaustive description frameworks for describing Web Services and related aspects (Web Service Description Ontologies)
- Support ontologies as underlying data model to allow machine supported data interpretation (Semantic Web aspect)
- Define semantically driven technologies for automation of the Web Service usage process (Web Service aspect)
OWL-S

Service

- Presents (What it does)
- ServiceProfile
- ServiceGrounding
- Support
  - How to access it
- ServiceModel

Described by

How it works
OWL-S Context

- Resource
  - Provides
  - R-Context
    - Show R-Context
- Service
  - Uses
  - Supports
  - Described by
  - Presents
  - Service Profile
    - Show S-Context
  - Service Grounding
    - Show U-Context
- User
- Service Model
  - S-Context
<asdl-descr>::=(ctype
  :service-name name
  :context context-name+
  :types  (type-name = <modifier> type)+
  :isa name+
  :inputs  (variable: <modifier> put-type-name)+
  :outputs (variable: <modifier> put-type-name)+
  :input-constraints (constraint)+
  :output-constraints (constraint)+
  :io-constrains (constraint)+
  :concept-description (ontology-name = ontology-body)+
  :state-language name
  :concept-language name
  :attributes  (attributes-name : attributes-value)+
  :text-description name
  )
Policy Driven

\[ p = \langle S_{\text{trigger}}, A, S_{\text{goal}}, U \rangle \]

- \( S_{\text{trigger}} \): set of states for policy execution
- \( A \): set of actions
- \( S_{\text{goal}} \): set of goals
- \( U \): set of utility functions
\[\text{Action Policy}\]

\[
\text{ActionPolicy} ::= \text{Name} \text{Parent} \text{Performative} \text{Type} \text{Subject} \\
\text{Object} \text{Action} \text{Precondition} \text{ConstraintLanguage} \\
\text{Duration} \text{Priority}
\]

- Access Control Policy
- Obligation Policy
**Action Policy**

**G:** IF \( RT_G > 100 \) msec
   THEN (Increase \( CPU_G \) by 5%)

**S:** IF \( RT_S > 200 \) msec
   THEN (Increase \( CPU_S \) by 5%)

Overlapping Action Policies
Conflict if CPU (almost) fully utilized!
<GoalPolicy>::= <Name><Performative><Subject><Object><Precondition><Postcondition><ConstraintLanguage><Duration><Priority>

( :name goalpolicy1
 :performative “Achieve”
 :subject ftpagent
 :precondition between(clienthost, 192.168.0.0, 192.168.0.255)
 :postcondition greater(bandwidth(clienthost), 1M) )
<UtilityPolicy>::=
    <Name><Performative><Subject><Object><Precondition><UtilityFunction><ConstraintLanguage><Duration><Priority>

( :name utilitypolicy1
  :performative “Optimize”
  :subject httpagent
  :utilityfunction \( f = w_1 \sum_i \text{bandwidth}(\text{client}_i) + w_2 \sum_i \text{resonsetime}(\text{client}_i) \) )
A plan $\Pi$ is a triple $<SO, OO, CS>$

$SO$: a set of action-steps,

$OO$: a set of ordering constraints on the actions in $SO$.

$CS$: a set of variable-binding constraints between the variables of the action-steps in $SO$ and other variables or constants.
Distributed Multiagent Plan Algorithm

\[ \text{Distributed} \rightarrow \text{MultiAgent} \rightarrow \text{Plan}(OPS, CaS, I, G) \]

Input: a set of action-templates \( OPS \),
\hspace{1cm} a set of capabilities of agents \( CaS \),
\hspace{1cm} initial state \( I \), and
\hspace{1cm} goal \( G \)

Output: Plan \( \Pi \)
Distributed Multiagent Plan Algorithm

Procedure:
1. Inform all agents that I will start planning, and wait for them finishing current actions and entering plan state.
2. Initialize: Build an empty plan \( \Psi = \langle SO, OO, CS, CLS \rangle \) where \( SO = \{Init, Goal\}, \ OO = \emptyset, \ CS = \emptyset, \ CLS = \emptyset \). Decompose goal \( G \) into a set of sub-goals \( GS \) through adding \( \langle g, Goal \rangle \) to \( GS \) according to every literal \( g \in G \).
3. Invoke recursive function \( \varphi = Generate\_Next\_Step(\Psi, GS) \)
4. Taking effect: For each action executed by other agent, request it to add the action. For each constraint related to the actions executed by other agents, request it them to add the constraints.
5. Inform all agents the planning finishes, thus they can continue their plans.
6. If the invoking succeeds, then return success, otherwise return fault.
## Agent Grid

### Applications Service
- IMS
- Power Supply
- Environment
- Biology
- E-Business

### Developing Toolkits
- Distributed Computing Toolkit
- Data-Intensive Applications Toolkit
- Collaborative Applications Toolkit
- Remote Visualization Applications Toolkit
- Problem Solving Applications Toolkit
- Remote Instrumentation Applications Toolkit

### Agent Environment
- Resource-independent and application-independent services
  - authentication, authorization, resource location, resource allocation, events, accounting, remote data access, information, policy, fault detection

### Common Resources
- Resource-specific implementations of basic services
  - E.g., Transport protocols, name servers, differentiated services, CPU schedulers, public key infrastructure, site accounting, directory service, OS bypass
Data Mining Platform MSMiner

MSMiner Architecture

MSDM

MSOlap

MSMetadata

MSETL

Data Mining

OLAP

Data Warehouse

Topic1 Topic2 Topicn

Extract Transform Load

Data resources

MSMiner Architecture

2006-6-3

AGrP-SELMAS Zhongzhi Shi
Outline

- Introduction
- Agent Model
- Multiagent Environment— MAGE
- Agent Collaboration
- Agent Grid Intelligence Platform
- Applications
- Conclusions
Emergency Interactive Systsem GEIS

Emergency Signal

Emergency Center

Case Base

Retrив

Emergency Case Base

Solutions

Emergency Execution

Decision Making in Terms of Reasoning
城市应急联动与社会综合服务系统

Intelligent Science Research Group, at Key Lab of ICT, CAS, China
Receive Crime Interface
Geographical Information System
CBR-Based Reserve Plan
Against Emergency

2006-6-3
AGriP-SELMAS Zhongzhi Shi

[Image of a computer interface with a webpage open, showing a form with various fields and options such as '事故类别', '事故地点', '伤亡人数', '救生车', '急救车', and '建议处警方法'.]
CBR-Based Reserve Plan Against Emergency
Fish Field Forecast in Eastern Sea of China
DSS for Oil Pipeline Design
Conclusions

- Proposed Agent Models
- MAGE Satisfies the Software Engineering for Large Multiagent Systems
- Collaborative Working Agents on Semantic Grid
- AGrIP is a Powerful Platform for Constructing Large Complex Systems
Published Paper List

- Zhongzhi Shi, Wenpin Jiao, Qiujuan Sheng. Agent-Oriented Software Methodology. CEEMAS2001, Cracow, Poland, 2001
- Zhongzhi Shi. Agent-based E-commerce. Invited Speaker, DS-9, Hong Kong, 2001
Published Paper List

• Zhongzhi Shi, Qijia Tian, Yunfeng Li. RAO Logic for Multiagent Framework. Journal of Computer Science and Technology, July, 1999
• Zhongzhi Shi, Hu Cao, Yunfeng Li, Wenjie Wang, Tao Jiang, A Building Tool for Multiagent Systems: AOSDE. IT \\& Knows, IFIP WCC '98, 1998
• Zhongzhi Shi, Qijia Tian, Yunfeng Li, RAO Logic for Multiagent Framework, DAIMAS'97, St. Petersburg, Russia, 1997.
THANK YOU!

Intelligence Science
http://www.intsci.ac.cn/