



**Oak Ridge National Laboratory  
Oak Ridge, Tennessee**

## **A Methodology to Evaluate Agent-Oriented Software Engineering Techniques**

**40<sup>th</sup> Hawaii International Conference on  
System Sciences (HICCS -40)  
Session DT 11 Sat. 8:00-9:30 Waikola 3**

**Hilton Waikoloa Village Resort  
Waikola, Big Island, Hawaii  
January 6, 2007**

**Paul Lin, Ph.D.**

**Krishna M. Kavi, Ph.D.**

**Dept. of CSE, Univ. North Texas**

**Frederick T. Sheldon, Ph.D.**

**Robert K. Abercrombie, Ph.D.**

**OAK RIDGE NATIONAL LABORATORY  
U. S. DEPARTMENT OF ENERGY**

Managed by UT-Battelle, LLC,  
for the US DOE under Contract  
No. DE-AC05-00OR22725

## **Agenda**

- **What is an agent**
- **Historical Perspective**
- **Application Perspective**
- **AOSE Design Frameworks Overview**
- **AOSE Evaluation Methodology**
- **Observations and Results**
- **Conclusions**

**OAK RIDGE NATIONAL LABORATORY  
U. S. DEPARTMENT OF ENERGY**



Page 2

## What is an agent

- No single definition
- A software component (object) that performs one or more tasks in some predefined manner.
- Agent properties include:
  - Deliberative vs. Reactive
  - Mobility
  - Autonomy
  - Learning
  - Cooperation

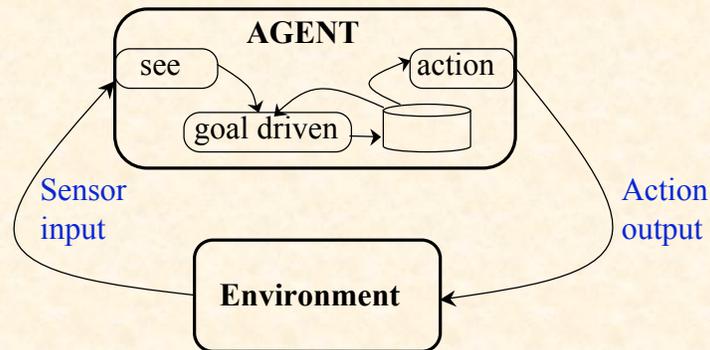
## What is an Agent (cont.)

- Agents are computer software systems (i.e., programs) situated in some environment
  - Capable of autonomous actions within that environment *as necessary to meet their design objectives.*



## What is an Agent (cont.)

- Proactive Agents



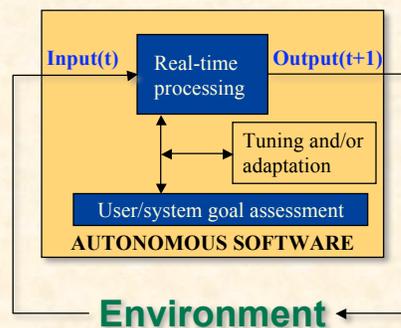
Output may be different at different times  
Even for the same input (based on goals)

## What is an Agent (cont.)

- An agent is a software component (or system) that is:

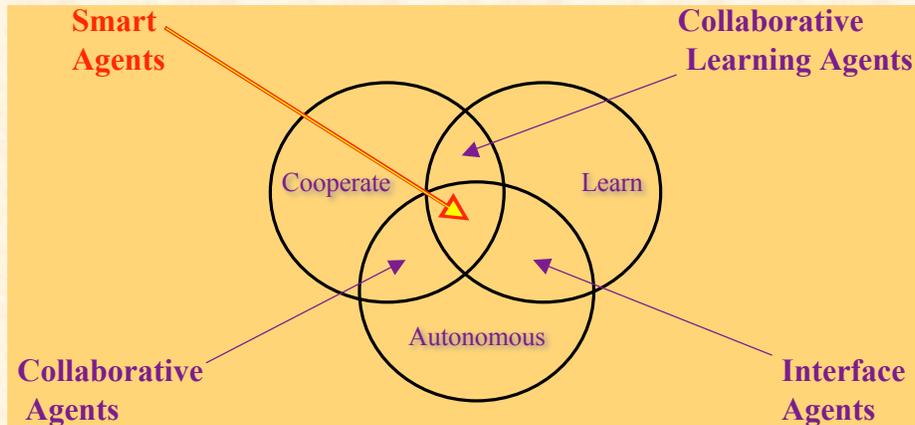
- Embedded in, and “aware” of, an environment
- **Dynamic** in its behaviors (not single I/O mapping)
- User enabled/steered, but “empowered” to act on behalf of the user
- Able to **improve** its behavior over time

Complexity



## What is an Agent (cont.)

- Intelligent Agents



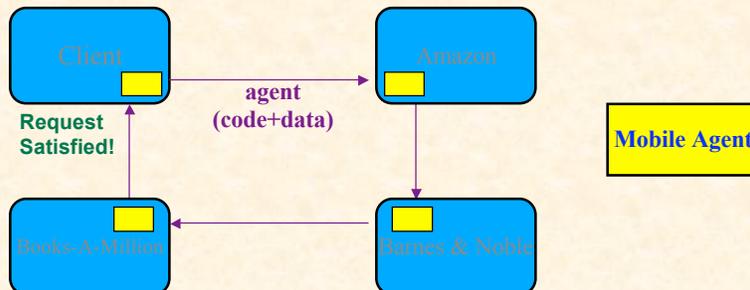
OAK RIDGE NATIONAL LABORATORY  
U. S. DEPARTMENT OF ENERGY

UT-BATTELLE  
Page 7

## What is an Agent (cont.)

- Mobile Agents

- An object capable of autonomously migrating from host to host performing actions on behalf of its creator.



OAK RIDGE NATIONAL LABORATORY  
U. S. DEPARTMENT OF ENERGY

UT-BATTELLE  
Page 8

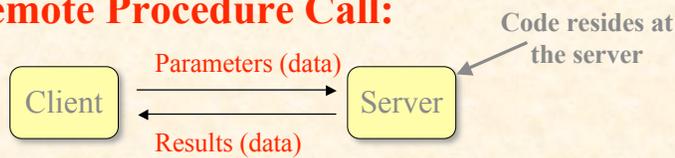
## Agenda

- What is an agent
- Historical Perspective
- Application Perspective
- AOSE Design Frameworks Overview
- AOSE Evaluation Methodology
- Observations and Results
- Conclusions

## Historical Perspective

- Evolution

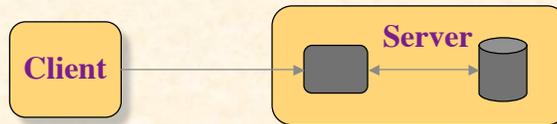
### Remote Procedure Call:



- Courier at Xerox PARC in 1980
- Sun RPC 1984
- DCE, CORBA late 1980's

## Historical Perspective (cont.)

- **Client Server Model:**



- **Server provides services and resources (and code to access resources)**
- **Client can request services by name**
  - Name service provided from a variety of means, including CORBA

## Historical Perspective (cont.)

- **Process Migration**

- Allows a partially executed process to be relocated to another node.
  - **Execution state** of the process is migrated.
    - *Stack, memory, program counter, state of open files.*
  - Mainly used for load balancing.
- **Mid 1980s**, several mechanisms were investigated and supported in LAN environments.
  - Locus (UCLA)
  - Sprite (UC Berkeley)
  - Condor (Wisconsin)

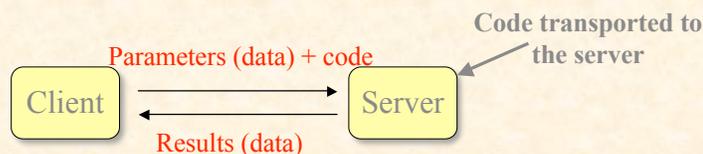
## Historical Perspective (cont.)

- **Object Migration**

- **Allows** objects to be moved across address spaces at different nodes.
  - Requires mobility of object's **code and data**
- **Emerald** supported object mobility under **program control** (Univ. of Wash., **1986**)
- **Chorus distributed system** (**1988**) supported object mobility with **autonomous control**.
- **Most supported migration in homogeneous environment**

## Historical Perspective (cont.)

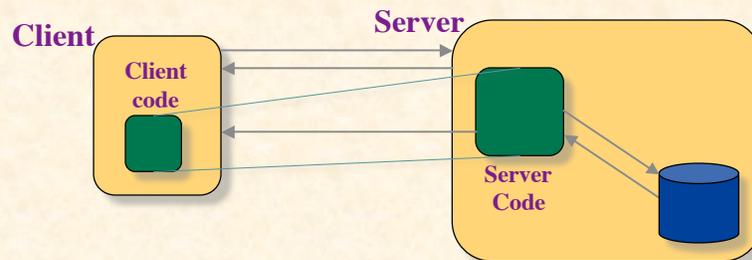
- **Remote Programming and Code Mobility:**



- **Remote Evaluation model by Stamos and Gifford (MIT, 1990).**
- **Java - Sun Microsystems (1995)** allows code migration across **heterogeneous platforms**.

## Historical Perspective (cont.)

- Code on Demand



- Client requests services from server. Server may obtain code from Client to satisfy the request.
- Java Applets

## Historical Perspective (cont.) Artificial Intelligence Roots

- Learning technologies
  - Expert systems
  - Stochastic learning methods
  - Other AI algorithms
- Modeled... after human societies and organizations

## Historical Perspective (cont.)

### Agent Communication Languages

- Involves “**structured messages**”, based on **speech acts** known as Performatives<sup>1</sup>.
- A set number of performative types are used to describe the act
  - ask () -- asking for some information
  - set () -- set a data or state variable
  - do () -- invoke a method or function
- Standards like FIFA and languages like **KQML**

<sup>1</sup>Designing a Message Handling Assistant Using the BDI Theory and Speech Act Theory  
Dissertation by Insu Song 2003 (Griffith Univ. Australia)

## Historical Perspective (cont.)

### Blackboards for communication

- Agents can communicate using a *shared message board* called blackboard
- The concept has roots in AI and systems (Linda tuple space)
  - read () read a matching tuple but not remove
  - in () read a matching tuple and remove
  - out () place a new tuple
- Powerful communication/ coordination system

## Historical Perspective (cont.)

**Recent initiatives** where ABC will likely play an important role...

- IBM multi-million dollar initiative called **autonomic computing**
- NSF initiative to find the “**science of software design**”
- DOE research funding in **network environment research**
- NASA’s interest in developing **adaptive software systems** for space missions
- Berkeley’s David Patterson’s concept called **Recovery Oriented Computing (ROC)**

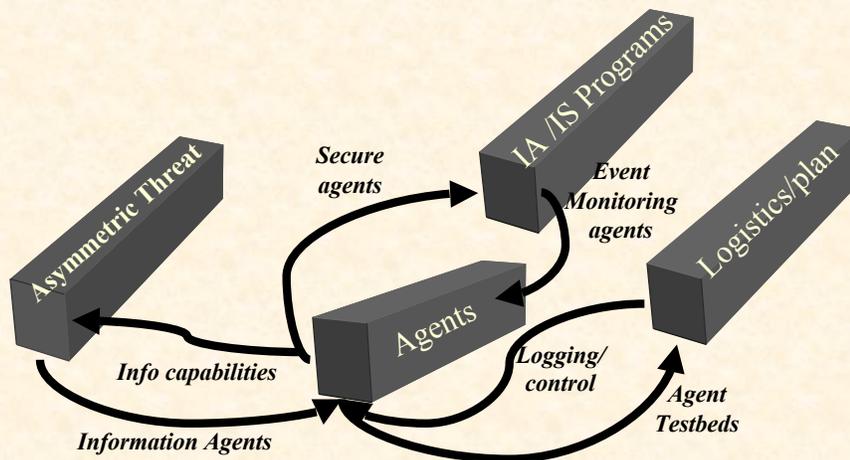
## Agenda

- What is an agent
- Historical Perspective
- **Application Perspective**
- AOSE Design Frameworks Overview
- AOSE Evaluation Methodology
- Observations and Results
- Conclusions

## Application Perspective Using Intelligent Agents

- Military Applications
- Industrial Process Control
- Information Management
- Air Traffic Control
- Electronic Commerce
- Personal agents

## Application Perspective (cont.) Agents are Crucial to Military Applications

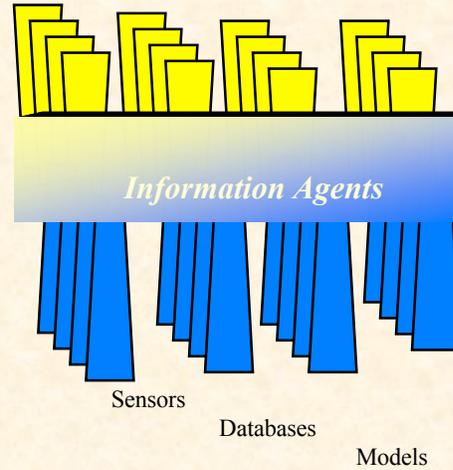


**Agent-Based computing is mandatory for delivering on the key Military goals.**

## Application Perspective (cont.)

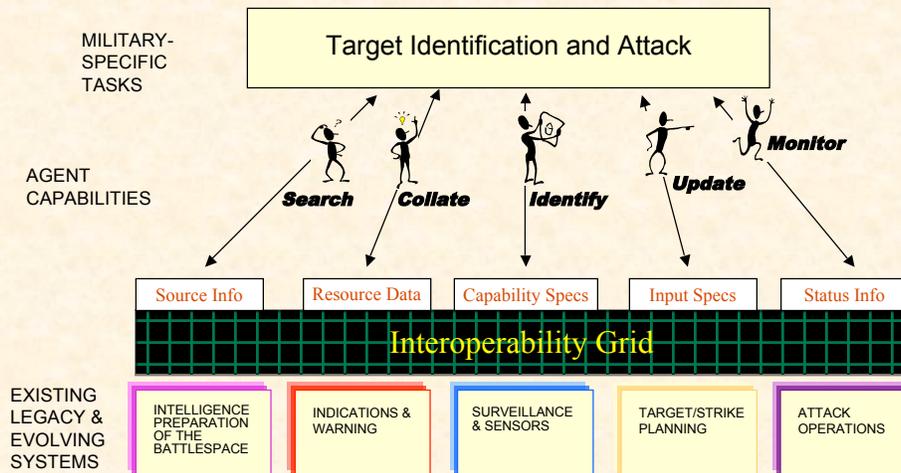
### How Agents Help: Breaking down the stovepipes

- Provide tools to help the run-time coupling of decision-making and analysis tools with appropriate data and sensing resources
- Development is focused on shared information needs



## Application Perspective (cont.)

### Information Agents



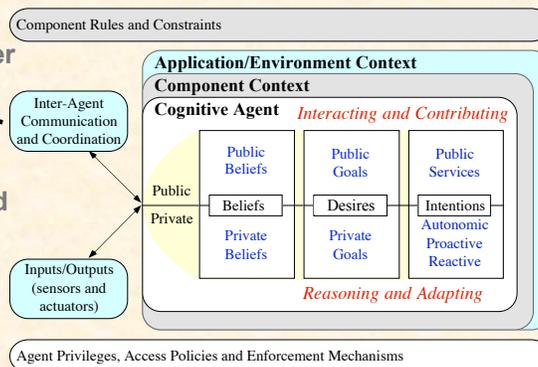
## Application Perspective (cont.) Conceptual Design Models

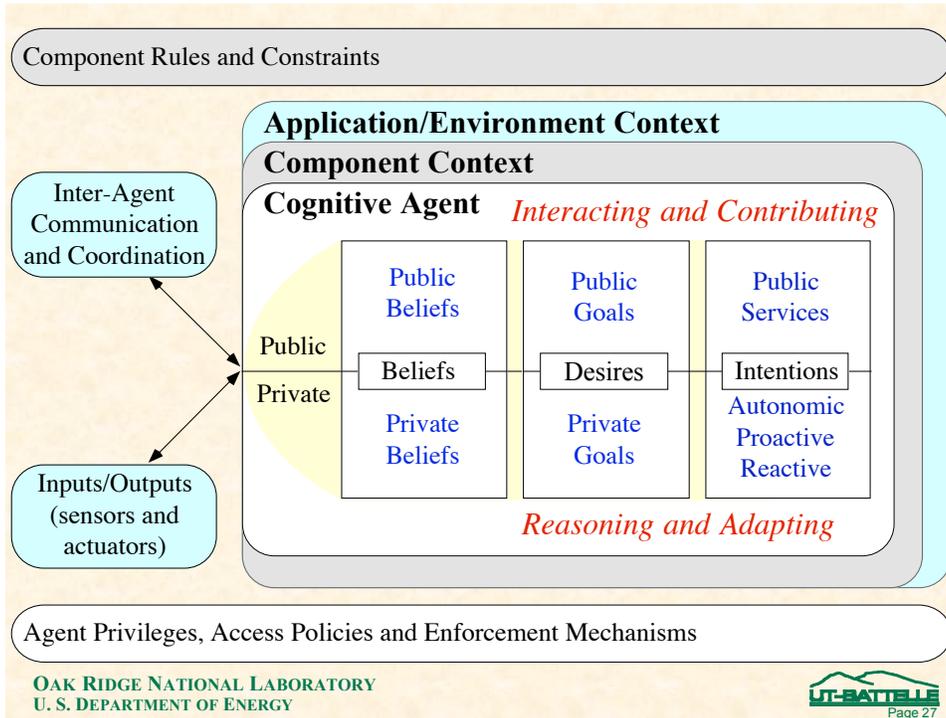
- Some preferred agent models used for development include:
  - BDI
    - Belief–Desire–Intention (BDI) or
    - Belief–Goal–Plan model
  - Societal Models

## Application Perspective (cont.) BDI Model

### Belief -Desire-Intention (BDI) or Belief-Goal-Plan model

- Agents have beliefs
  - About environment, other agents or itself
- Agents have Desires or Goals
  - May be both reactive and proactive
- Agents use Intentions (or Plans) to achieve Goals
- Agents are autonomous

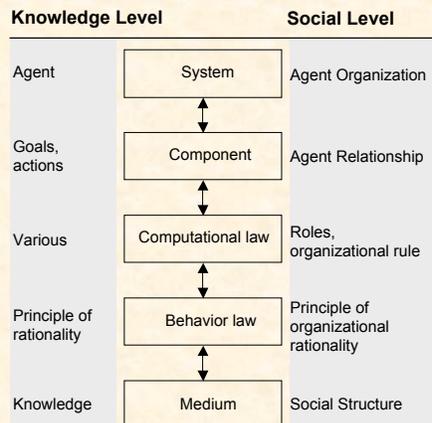




## Application Perspective (cont.) Societal Model for Agent

### • Agents Play Roles

- Roles have responsibilities (can be viewed as goals)
- Responsibilities need permissions or resources
- Activities are needed to implement roles
- Protocols are needed to define communication



## Agenda

- What is an agent
- Historical Perspective
- Application Perspective
- **AOSE Design Frameworks Overview**
- AOSE Evaluation Methodology
- Observations and Results
- Conclusions

## Agent-Oriented Software Engineering (AOSE) Methodologies Overview

- Tropos
- Gaia
- MaSE
- A framework extending UML
- Object-oriented framework for agents

## AOSE Methodologies - Tropos

- Notations used:

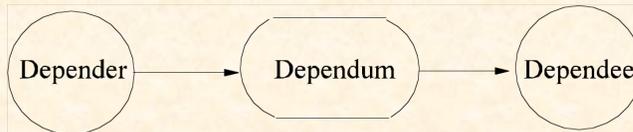


- Process

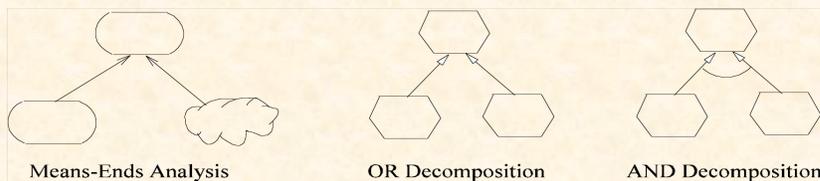
- 1. Early requirement analysis
- 2. Late requirement analysis
- 3. Architecture design
- 4. Detailed design
- 5. Implementation

## AOSE Methodologies – Tropos (cont.)

- Strategic dependency (SD)

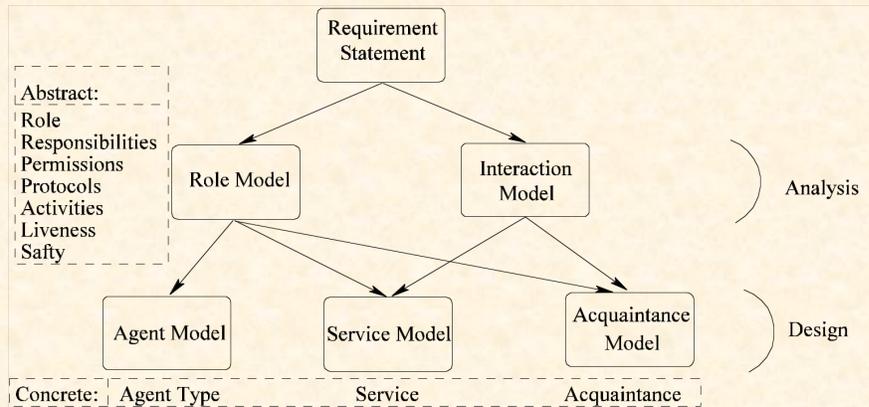


- Strategic rationale (SR)



# AOSE Methodologies - Gaia

- **Process Model**

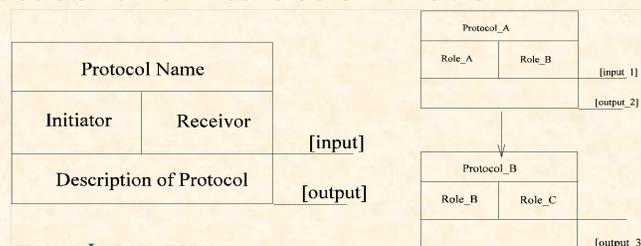


# AOSE Methodologies – Gaia (cont.)

- **Role template**

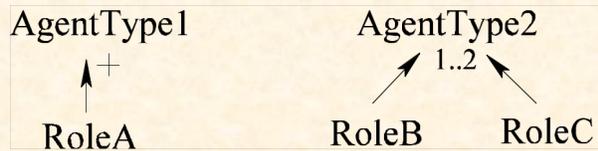
Role Schema:	name of role
Description	short English description of the role
Protocols and Activities	protocols and activities the role plays a part
Permissions	right associated with the role
Responsibilities	
Liveness	liveness responsibilities
Safety	safety responsibilities

- **Protocol and interaction model**

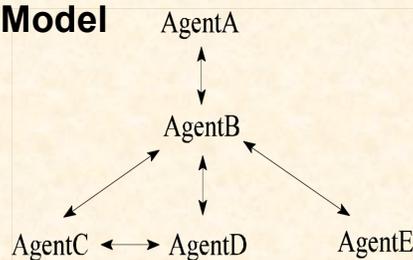


## AOSE Methodologies – Gaia (cont.)

- Agent Model

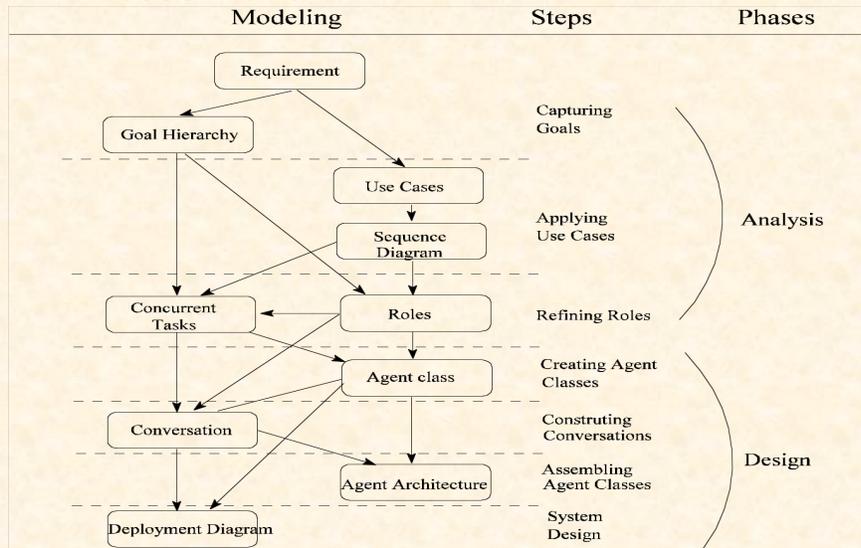


- Acquaintance Model



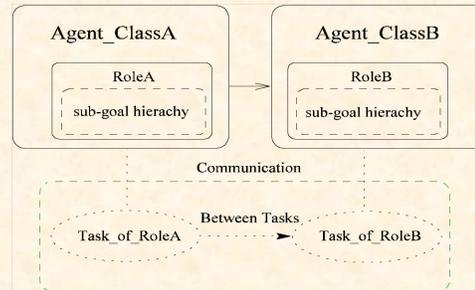
## AOSE Methodologies - MaSE

- Process Model

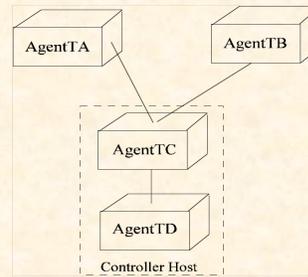


## AOSE Methodologies – MaSE (cont.)

- **Class Model**



- **Deployment Model**



OAK RIDGE NATIONAL LABORATORY  
U. S. DEPARTMENT OF ENERGY

UT-BATTELLE  
Page 37

## AOSE Methodologies – Extending UML

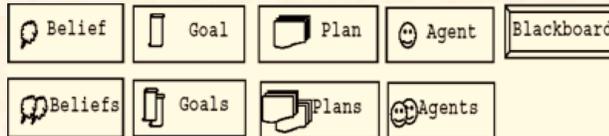
- **BDI architecture (Belief, Goal, Plan)**
- **Communication mechanisms (FIPA, KQML, Blackboard)**
- **New Diagrams:**
  - Agent Goal Diagram (AGD),
  - Use Case Goal Diagram (UCGD),
  - Agent Domain Model (ADM),
  - Agent Sequence Diagram (ASD),
  - Agent Design Diagram (ADD),
  - Agent Activity Diagram (AAD),
  - Agent State Chart Diagram (ASCD)

OAK RIDGE NATIONAL LABORATORY  
U. S. DEPARTMENT OF ENERGY

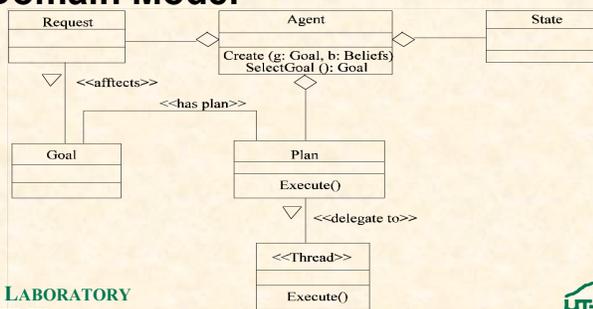
UT-BATTELLE  
Page 38

# AOSE Methodologies – Extending UML (cont.)

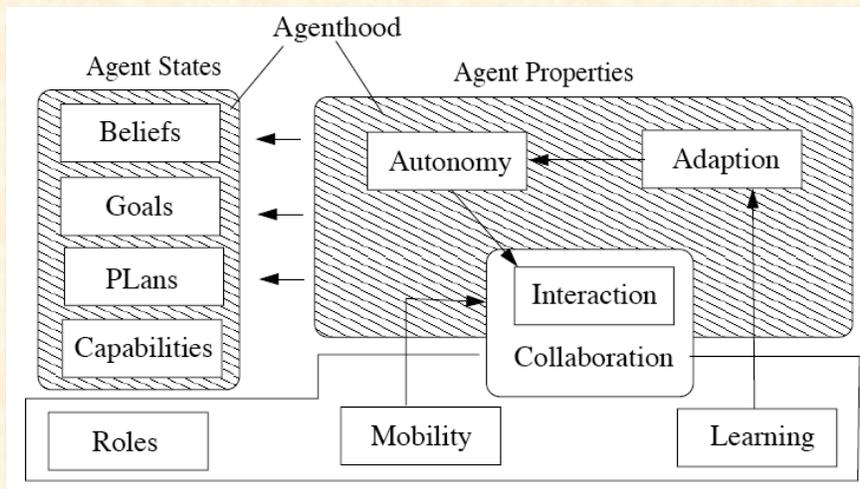
- Notations



- Agent Domain Model

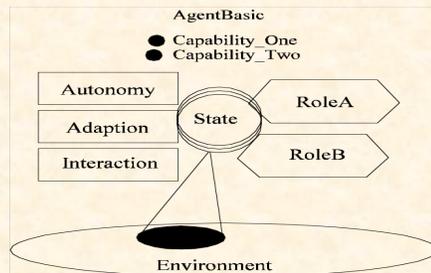


# AOSE Methodologies – Object-Oriented Framework with Aspect

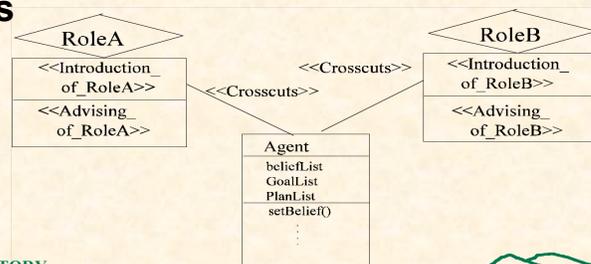


## AOSE Methodologies – Object-oriented framework with Aspect (Cont.)

- Agent model



- Role aspects



## Agenda

- What is an agent
- Historical Perspective
- Application Perspective
- AOSE Design Frameworks Overview
- AOSE Evaluation Methodology
- Observations and Results
- Conclusions

## What AOSE Method Works Best?

- Agent technology is often the preferred architectural framework for many distributed software systems.
- Agent-based systems (ABS) are often endowed with:
  - intelligence,
  - autonomy, and
  - reasoning (alluring to both legacy and new systems).
- Agents are attribute building blocks composed to form software entities.

## What AOSE Method Works Best? (Cont.)

- The more complex an ABS, the more sophisticated the methodology needs to be.
- At present **there are no consensus standards on how to create Agents or model them** in the development process.
- Current methods for creating ABS were reviewed to gain insight on what attributes are most useful in leading to better design methodologies ...

## **AOSE Evaluation Methodology Comparison Framework Establishment**

- **Objective: evaluate criteria concerning components from both:**
  - formal software engineering process definitions
  - agent-oriented characteristics
- **Framework adopts 4 major divisions:**  
(proposed by A. Sturm and O. Shehory)
  - (a) Concepts and Properties
  - (b) Notations and Modeling Technique
  - (c) Software Engineering Process
  - (d) Pragmatics

## **AOSE Evaluation Methodology (Cont.) Comparison Levels**

- **Overview Level:**
  - Derived from general discussions of
    - agent definitions,
    - AOSE methods, and
    - software engineering methods
  - ... to derive criteria within each of the 4 divisions
- **Detailed Level:**
  - Proposed questions which address the logical relationships among criteria and provide a basis for comparison.

## AOSE Evaluation Methodology (Cont.) Comparison Criteria

- A set of questions were developed for each division and applied to each method
- ... resulting in a matrix that aligns each criteria next to each method, one matrix for each division:
  - (a) Concepts and Properties
  - (b) Notations + Modeling Techniques
  - (c) SE Process
  - (d) Pragmatics

## AOSE Evaluation Methodology (Cont.) A. Concepts and Properties Criteria

Criteria	Description
Autonomy	Agents can make decisions on their own based on inner states without external supervision.
Mental Mechanism	An Agent has mechanisms to realize its intentions and achieve goals.
Adaptation	An Agent is flexible enough to adjust its activities according to a dynamically changing environment.
Concurrency	An Agent may need to perform multiple tasks concurrently.
Communication	There are protocols or mechanisms defined for Agent interactions.
Collaboration	An Agent has methods to cooperate with other Agents to achieve goals.
Agent Abstraction	Methodology provides theory/facilities for describing Agents using high level abstractions.
Agent-oriented	Methodology originates from the consideration of Agent-oriented approaches primarily addressing Agent-based features during the analysis and design phases.

## AOSE Evaluation Methodology (Cont.) Questions for Concepts and Properties Comparison (Section A)

- A1. What concepts are at the root of the methodology and what are the advantages?
- A2. How is an Agent created within the methodology?
- A3. How well does a design deal with the Agent's mental mechanism?
- A4. How well does a design deal with an Agent's environmental perception and its ability to react based on the perception?
- A5. How efficient are Agents in achieving their goals?

## AOSE Evaluation Methodology (Cont.) Comparison Using Concepts and Properties

Concepts + Properties (A)	Tropos	Gaia	MaSE	Extending UML (ExtUML)	OO-Framework (OOF)
Autonomy	Yes	Yes	Yes	Yes	Yes
Mental mechanism	Goal, soft goal tasks	No	Goal, tasks	BDI	BDI
Adaptation	Yes	Yes	No	Yes	Yes
Concurrency	Yes	Yes	Yes	Yes	Yes
Communication	Yes	No details	No details	Yes	Yes
Collaboration	Yes	Yes	Yes	Yes	Yes
Agent Abstraction	Social actors	Organizational roles	Roles	Mental entity	Agenthood
Agent-oriented	Yes	Yes	Yes	Yes	Yes

## AOSE Evaluation Methodology (Cont.)

### B. Notations + Modeling Technique Criteria

Criteria	Description
Expressiveness	Notations are used in the methodology to help design process.
Complexity management	There are abstraction levels from high to low available to tackle complexity in the problem domain.
Modularity	Support is available for using components or modules in to promote modeling in an incremental manner.
Executable	Models used in a methodology are capable of simulation or generating prototypes for some aspect(s) of the specification.
Refinement	Modeling technique can refine factors into simpler entities advantageously (i.e., enable reification).
Traceability	Within the methodology, it is possible to track dependencies between models.

## AOSE Evaluation Methodology (Cont.)

### Questions for Notations & Modeling Techniques Comparison (Sect. B)

- **B1. How well are notations and models formed to address Agent-based system scenarios?**
- **B2. How consistent and unambiguous are models throughout the process?**
- **B3. How well does the modeling technique address traceability and reuse?**
- **B4. How well does the modeling technique represent Agents?**

## AOSE Evaluation Methodology (Cont.) Comparison Using Notations and Modeling Techniques

Notations + Modeling (B)	Tropos	Gaia	MaSE	Extending UML (ExtUML)	OO-Framework (OOF)
Expressiveness	Yes	Yes	Yes	Yes	Yes
Complexity management	Decomposition of goals and tasks	Role	Goal, role refine	Goal refine	Property aspects
Modularity	Yes	Yes	Yes	Yes	Yes
Executable	No	No	No	Yes	No
Refinement	Yes	No	Yes	Yes	No
Traceability	Yes	Yes	Yes	Yes	Yes

## AOSE Evaluation Methodology (Cont.) C. Process Criteria

Criteria	Description
Specification	The methodology provides guidance on how to form a system specification from scratch.
Life-cycle coverage	The methodology covers steps from analysis, design, to implementation and testing through out the system development process.
Architecture Design	The methodology provides a mechanism to facilitate design by using patterns or templates.
Implementation Toolkits	The methodology provides guidance on how to implement agents.
Deployment	Concerns about practical deployment of agents are addressed.

## AOSE Evaluation Methodology (Cont.) Questions for Process Comparison (Sect. C)

- C1. How well does the methodology define the application domain and system context?
- C2. How well does the process cover the whole lifecycle?
- C3. How well do the transitions between phases preserve goals?

## AOSE Evaluation Methodology (Cont.) Comparison of Process

Process (C)	Tropos	Gaia	MaSE	Extending UML (ExtUML)	OO-Framework (OOF)
System Specification	Stakeholders analysis	Role analysis	Use-cases goal and role analysis	Use-cases and goal analysis	Aspects analysis
Life-cycle coverage	Yes	Yes	Yes	Yes	No
Architecture Design	Yes	No	Yes	Yes	Yes
Implementation Toolkits	Yes	No	No	Yes	Yes
Deployment	No	Yes	Yes	No	No

## AOSE Evaluation Methodology (Cont.)

### D. Pragmatics Criteria

Criteria	Description
Tools available	There are resources and tools readily available for using the methodology.
Required expertise	Requires background of knowledge necessary to apply the methodology.
Modeling suitability	The methodology is based on a specific architecture.
Domain applicability	The methodology is suitable to a specific application domains
Scalability	The methodology is able to handle a reasonable number of agents within a given application

## AOSE Evaluation Methodology (Cont.)

### Questions for Pragmatics Comparison (Sect. D)

- D1. Is the methodology easy to use?
- D2. Do Agent concepts and properties evolve easily?
- D3. Is the Agent-oriented methodology flexible enough in re-engineering?
- D4. Are paradigm/architectures suitable in the general case?

## AOSE Evaluation Methodology (Cont.) Comparison of Pragmatics

Pragmatics (D)	Tropos	Gaia	MaSE	Extending UML (ExtUML)	OO-Framework (OOF)
Tools available	No	No	Yes	Yes	Yes
Required Expertise <sup>1</sup>	No	No	No	No	Yes
Modeling Suitability	BDI	No	No	BDI	Agenthood
Domain applicability	Yes	Yes	Yes	Yes	Yes
Scalability	Yes <sup>2</sup>	Yes <sup>3</sup>	Yes	Yes	Yes

<sup>1</sup>See Table 4.9 pg 78 of Lin Dissertation

<sup>3</sup>Requires fewer than 100 agent types

<sup>2</sup>Too many stakeholders is problematic

## Summary of Observations AOSE methodology should have ...

- **Good mental mechanisms** that support autonomy, adaptation, and collaboration
- **Communication protocols** crucial to empowering Agents to complete their tasks
- **Goal-oriented focus** including goal management
- **Practical conceptual modeling facilities** to deal with complexity

## Summary of Observations (cont.) AOSE methodology should have ...

- **Notational constructs** for clearly and concisely expressing key processes and properties
- **Concrete and complete software life-cycle process** guidance
- **Tools and modeling facilities** availability
- **Abstraction and refinement capabilities** for analyzing and integrating application (system) elements.

## What is needed ...

- Infrastructure/ idioms/ models for specifying “learning”, “fault-tolerance”, “adaptivity”, among others...
- Develop “monitoring” aspects to ensure agent design objectives and performance are satisfied
- Explore and develop “probabilistic” and “stochastic” V&V techniques and infrastructure
- More tools that are seamless throughout the full life cycle
- What about mobility and security issues?
  - Most experience is ad hoc and antidotal ...
  - Not codified inside AOSE methods ...

## Future work

- **More rigor for modeling throughout the AOSE development process (from specification to goal-oriented proofs)**
- **Software engineering process guidance facilitated with unambiguous and full life-cycle coverage**
- **Method to assign overall score(s) (e.g., A-F in all four divisions) for the evaluation framework**
- **Case studies that apply and validate the evaluation framework**

## Contact me

Frederick T. Sheldon, Ph.D.  
Computational Sciences and Engineering Division  
Oak Ridge National Laboratory  
PO Box 2008, MS 6418, 1 Bethel Valley Rd  
Oak Ridge, TN 37831-6418

---

<http://www.ioc.ornl.gov>  
<http://www.csm.ornl.gov/~sheldon>  
865/576-1339 Voice  
865/576-5943 Fax