# 에이전트 시스템의 개발 현황과 응용 전망

이 재 호
서울시립대학교 전자전기공학부
전화: 02-2210-2629
Eamil: jaeho@ee.uos.ac.kr
URL: http://ee.uos.ac.kr/~jaeho

1999년 6월4일 (金)

한국 지능정보시스템 학회 '99 춘계 학술대회
한양대학교 백남학술정보관





## 에이전트 연구 배경

Raisin Bread

Intelligent agents are ninety-nine percent computer science and one percent AI (Etzioni 1996).

에이전트의 정의

An agent is a computer system, *situated* in some environment, that is capable of *flexible autonomous* action in order to meet its design objectives (Wooldridge and Jennings 1995).

- situatedness: the agent receives sensory input from its environment and can perform actions which change the environment in some way
- autonomy: the system is able to act without the direct intervention of humans and has control over its own actions and internal state.
- flexible: responsive, pro-active, social

## 에이전트 연구의 역사적 배경 (Jennings, Sycara, and Wooldridge 1998)

- Artificial Intelligence:
  - Planning systems: *first principles*
  - Behavioral AI (Reactive AI, Situated AI): *subsumption architecture*
  - Hybrid architecture: *deliberative + reactive*
  - Practical reasoning agents: belief-desire-intention (BDI) model
- Object and Concurrent Object Systems:
  - Objects can have control over their own internal state, but not it's behavior.
  - Agents supports the notion of flexible (reactive, pro-active, social) autonomous behavior
  - Agents are each considered to have their own thread of control
- Human-Computer Interfaces:
  - The view of computer programs as *cooperating* with a user to achieve a task, rather than acting simply as servants
  - *expert assistants*, *digital butlers*





## Multi-Agent Systems (Bond and Gasser 1988)

Distributed Problem Solving (DPS)

considers how a particular problem can be solved by a number of modules, which cooperate in dividing and sharing knowledge about the problem and its evolving solutions.

Multi-Agent Systems (MAS)

is concerned with the behavior of a collection of possibly pre-existing autonomous agents aiming at solving a given problem.

- each agent has incomplete information, or capabilities for solving the problem, thus each agent has a limited viewpoint
- there is no global system control
- data is decentralized
- computation is asynchronous

## Pitfalls of Agent-Oriented Development (Wooldridge and Jennings 1998)

- You oversell agents
- You don't know why you want agents
- You don't know what your agents are good for
- You believe that agents are a silver bullet
- You forget you are developing *(distributed) software*
- You decide you want your own agent architecture
- Your agents use too much AI
- Your agents have no intelligence
- You have too many (or too few) agents
- The *tabula rasa*
- You ignore *de facto* standards





## 에이전트 연구 분야

### Agent theory

- A specification for an agent.
- The construction of formalisms for reasoning about agents, and the properties of agents expressed in such formalisms.

### Agent architectures

- Software engineering models of agents
- The construction of computer systems that satisfy the properties specified by agent theorists.

### Language

- A system that allows one to program hardware or software computer systems in terms of some of the concepts developed by agent theorists.
- Software systems for programming and experimenting with agents.

## 학술 활동

**ICMAS'98:** Third International Conference on Multi-Agent Systems
(http://cosmos.imag.fr/MAGMA/ICMAS98/)

**Agents'99:** Third International Conference on Autonomous Agents
(http://www.cs.washington.edu/research/agents99/)

**ATAL-99:** Sixth International Workshop on Agent Theories, Architectures, and Languages
(http://www.elec.qmw.ac.uk/dai/atal/)

**PRIMA'99:** Second Pacific Rim International Workshop on Multi-Agents
(http://www.lab7.kuis.kyoto-u.ac.jp/prima99/)

**CIA-99:** Third International Workshop Cooperative Information Agents
(http://www.informatik.tu-chemnitz.de/ klusch/cia99.html)





## 에이전트 응용 분야

- Industrial Applications
  - Manufacturing
  - Process Control
  - Telecommunications
  - Air Traffic Control
  - Transportation Systems
- Medical Applications
  - Patient Monitoring
  - Health Care
- Commercial Applications
  - Information Management
  - Electronic Commerce
  - Business Process Management
- Entertainment Applications
  - Games
  - Interactive Theater and Cinema

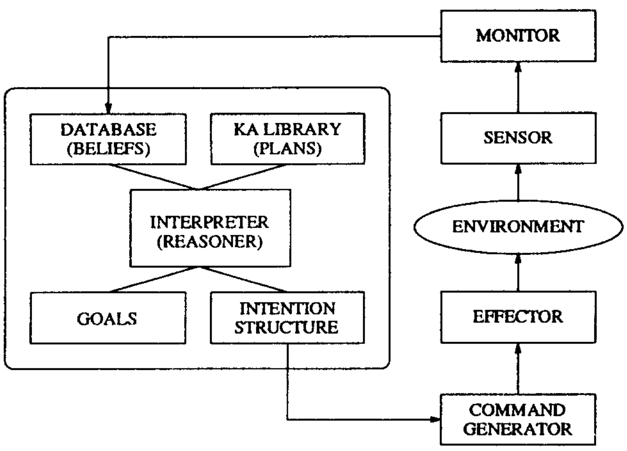
## 에이전트 구조에 관한 희망사항

- Real-time execution
- Interruptible execution
- Multiple foci of attention
- Hierarchical plan refinement and revision (mix and match strategies)
- Purposeful behavior (minimizing high-level plan revision)
- Adherence to predefined strategies
- Task migration
- Goal-driven and data-driven behavior
- Checkpointing and Mobility
- Explicit strategy articulation
- Situation summary and report
- Restrainable reactivity





## PRS System Structure (Ingrand, Georgeff, and Rao 1992)




- World Model (Database)
- Goals
- Knowledge Areas (Procedures)
- Intention Structure
- Interpreter

## Features of PRS for Reactive Systems

- The semantics of its plan (procedure) representation, which is important for verification and maintenance.
- Its ability to expand and act on partial plans.
- Its ability to pursue goal-directed tasks while being responsive to changing patterns of events in bounded time.
- Its facilities for managing multiple tasks in real time.
- Its default mechanisms for handling the environment's stringent real-time demands.
- Its metalevel (or reflexive) reasoning capabilities.





## UM-PRS
### University of Michigan Procedural Reasoning System
### (Lee, Huber, Durfee, and Kenny 1994)

- a C++ implementation of PRS.
- general reactive agent architecture.
- applied to both physical robots and software agents
- tested over years of applications

```
KA {
NAME:           "Example KA"
DOCUMENTATION:  "A nonsensical KA"
PURPOSE:        ACHIEVE goal_name $arg;
CONTEXT:        FACT task_complete "False";
BODY:
  FACT problem_solved $task $solved;
  OR
  {
    TEST (== $solved "YES");
    RETRACT working_on_problem "True";
  }
  {
    TEST (== $solved "NO");
    ACHIEVE problem_decomposed;
    ATOMIC
    {
        ASSERT working_on_problem "True";
        FAIL;
    };
    ASSIGN $result (* $arg $arg 5);
  };
  UPDATE (task_complete) (task_complete "True");

FAILURE:
  UPDATE (ka_example_failed) (ka_example_failed "True");
  EXECUTE print "Example failed.  Bailing out"
}
```

## UM-PRS Example

- World Model: a database of facts
- Goals: top-level goals, subgoals
- Knowledge Areas (KA)
  - Name
  - Purpose: goal
  - Context: condition
  - Body: procedure
  - Priority
  - Failure section
- Actions: achieve, execute, query, test, assert, retract, ···
- Intention Structures: runtime state of progress
- Interpreter





## UM-PRS Behaviors

- Incremental elaboration to identify appropriate actions.
- Recovery from a failed context.
- Suspension of one goal and pursuit of another.
- Interruption by new goals.
- Refocus due to a change of context.

## UM-PRS Applications

◇ Physical Agents

- indoor and outdoor mobile robots (UGV)

◇ Software Agents

- University of Michigan Digital Library (UMDL)
- Ship Systems Automation (SSA) — TAIPE project
- Intelligent, Coordinated Situation Assessment
- Multi-Agent Defensive Information Warfare
- Rescue Operations Planning





## TAIPE
**Tactical Assistant for Interaction Planning and Execution**
**(Durfee, Huber, Kurnow, and Lee 1997)**

- many specialized agents needed to be able to collectively form, reason about, and execute plans.
- plans needed to be generated, communicated, elaborated, visualized, analyzed, executed, revised, and so on.

*Planning Content Language* to support the development and execution of plans.

## JAM Agent Architecture (Huber 1999)

- Integrated, refined features of PRS, UM-PRS, and the Structured Circuit Semantics (Lee 1995) representation.
- Additional functionalities: observer, checkpointing, mobility





## Agent Goals

**ACHIEVE:** An achieve action causes the agent to establish a goal achievement subgoal for the currently executing plan.

**PERFORM:** The agent checks to see whether the subgoal has already been accomplished. Only if the goal has not been accomplished, the plan does subgoal.

If the agent detects (opportunistic) accomplishment of the goal (perhaps by another agent), it will consider the subgoal action successful and discontinue execution of the plan established to achieve the subgoal.

**MAINTAIN:** A maintain goal indicates that the specified goal must be reattained if it ever becomes unsatisfied.

**QUERY:** A query action is functionally identical to an achieve action. It is provided to allow the programmer to be more explicit about the semantics of the action's goal.

**WAIT:** The wait action causes plan execution to pause until the specified goal is achieved or the specified action returns successfully.

## Plans

**Plan Precondition:** specifies the initial conditions that must be met before the plan should be considered for execution.

**Plan Context:** specifies one or more expressions that describe the conditions under which the plan will be useful throughout the duration of plan execution.

**Plan Goal:** specifies goal-driven behavior. This field's contents specify the goal or activity that successful execution of the plan's procedural body will accomplish.

**Plan Conclude:** specifies data-driven behavior. This specifies a World Model relation that should be monitored for change.

**Plan Body:** describes the sequence of actions, a procedure, to be taken in order to accomplish a goal.

**Plan Effects:** specifies an atomic procedure that will be executed when the plan completes successfully.

**Plan Failure:** specifies an atomic procedure to be executed when the plan fails. If the plan fails, for example because the context fails, the agent interpreter will execute the actions found in the failure section before switching to other plans or goals.





## Observer

- The observer is a lightweight plan that the agent executes between plan steps in order to perform functionality outside of the scope of its normal goal/plan-based reasoning.
- The Observer's behavior is specified in a syntax identical to that of a plan body in one of the files parsed during initialization.
- This procedure may contain any plan action or construct that a normal plan body can contain except for subgoaling.

## Checkpointing and Mobility

functionality for capturing the runtime state of a Jam agent in the middle of execution and subsequently restoring that captured state to its execution state.

- to periodically save the agent's state so that it can be restored in case the agent fails unexpectedly.
- to implement agent mobility, where the agent migrates from one computer platform to another.
- to clone an agent by creating a checkpoint and restoring it execution state without terminating the original agent.





## UM-PRS와 JAM

UM-PRS

- satisfies most of the features requiring real-time interruptible execution, multiple foci of attention.
- naturally supports hierarchical plan refinement and revision, purposeful behavior, adherence to predefined strategies.
- demonstrated effective task migration capability.

JAM

- inherits all of the above capabilities and supports additional explicit data-driven behavior, checkpointing and mobility.
- the explicit strategy articulation capability is strengthened by incorporating the SCS execution semantics into Jam and is also supported partly by JAM's refined goal actions and plan conditions.

Other capabilities, situation summary/report and restrained reactivity are especially required for coordinated agent plan execution and are being studied in the context of explicit specification of execution semantics in the agent plan.





## JAM의 발전방향

Mobile Agent Architecture, JAM/Aglet

- Aglet's Mobility (Lange and Mitsuru Oshima 1997)
- JAM's Checkpointing and serialization

Applications of JAM/Aglet

- Believable agent
- Multimedia information gathering/filtering

## 참고문헌


Bond, A. H. and L. Gasser (Eds.) (1988). *Readings in Distributed Artificial Intelligence.* San Mateo, CA: Morgan Kaufmann Publishers.

Durfee, E. H., M. Huber, M. Kurnow, and J. Lee (1997, February). TAIPE: Tactical assistants for interaction planning and execution. In *Proceedings of the First International Conference on Autonomous Agents (Agents '97),* Marina del Rey, California, pp. 443–450.

Etzioni, O. (1996, August). Moving up the information food chain: Deploying softbots on the world-wide web. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence,* Portland, Oregon, pp. 1322–1326.

Huber, M. (1999, May). JAM: A BDI-theoretic mobile agent. In *Proceedings of the Third International Conference on Autonomous Agents (Agents '99),* Seattle, Washington.

Ingrand, F. F., M. P. Georgeff, and A. S. Rao (1992, December). An architecture for real-time reasoning and system control. *IEEE Expert 7*(6), 34–44.

Jennings, N. R., K. Sycara, and M. Wooldridge (1998). A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems 1,* 7–38.

Lange, D. B. and I. R. Mitsuru Oshima (1997). Programming mobile agents in java – with the java Aglet API. http://www.trl.ibm.co.jp/aglets/aglet-book/index.html.







Lee, J. (1995, March). On the design of structured circuit semantics. In *AAAI Spring Symposium on Lessons Learned from Implemented Software Architectures for Physical Agents,* pp. 127–134.

Lee, J., M. J. Huber, E. H. Durfee, and P. G. Kenny (1994, March). UM-PRS: an implementation of the procedural reasoning system for multirobot applications. In *Conference on Intelligent Robotics in Field, Factory, Service, and Space (CIRFFSS '94),* Houston, Texas, pp. 842–849.

Wooldridge, M. and N. R. Jennings (1995). Intelligent agents: Theory and practice. *The Knowledge Engineering Review 10*(2), 115–152.

Wooldridge, M. and N. R. Jennings (1998, May). Pitfalls of agent-oriented development. In K. P. Sycara and M. Wooldridge (Eds.), *Proceedings of the Second International Conference on Autonomous Agents (Agents'98),* pp. 385–391. ACM Press.