Software Agents
A Contribution to Agents Specification

Vojislav Stojkovic [1]
William Lupton [2]
Computer Science Department
Morgan State University
Baltimore, MD 21251, USA

Abstract

This article presents informal and formal specifications of some basic concepts (terms) and properties of agent theory, the design and imperative and recursive implementations of intelligent agents and supports agent approach in computer science.

Keywords: Agent, agent model, intelligent agent, agents theory, software agent, agent program.

1. INTRODUCTION

We regard an agent theory as a specification for an agent. Agent theorists develop formalisms for representing the properties of agents, and using these formalisms, try to develop theories that capture desirable properties of agents.

The concepts (terms) of "agent," "agent model," intelligent agent," "agents theory," "software agent," "agent-oriented programming language (AOP language)," "agent-oriented programming system (AOP system)," "agent-oriented programming (AOP)," "agent program," etc. are central concepts of agents theory. They are also one of the most important concepts in some other areas of artificial intelligence, communications, robotics, and user interfaces. Unfortunately some of these concepts are still "noise" concepts, subjects to abuse, misuse, and confusion even in the artificial intelligence agent community.

In many textbooks and articles on Artificial Intelligence an agent is something that acts in the world, for example, a person, a robot, a dog or a cat.

The purpose of this paper is fourfold:

- Informally and formally define the basic concepts (terms) of the agent theory, such as “agent,” “agent model,” “intelligent agent,” “software agent,” and etc. (Still these basic concepts have not standard definitions!)
- Informally and formally define some basic properties of agents and their environment (world).
- Discuss design and imperative and recursive implementations of intelligent agents.
- Support (by our opinion very important) agent-approach in computer science and artificial intelligence.

2. AGENTS

An agent is a (simple) computer system that is capable of autonomous actions in some environment in order to satisfy its design objectives.

The autonomy means:

- Agents operate (reason, make decisions, and solve problems) without the direct intervention of others (agents and non-agents)
- Agents have some kind of control over their actions and internal states.

3. AGENT MODEL

An agent model is characterized by the fact that it is possible to develop (write, construct, make, build, etc.) independent agents (units, routines, functions, modules, pieces of code, systems, machines, etc.) to do something with some purposes.

[1] Stojkovi@Morgan.edu
[2] Lupton@Morgan.edu

This approach asserts that agents are self-contained, though they may contain references to other agents.

The agent model provides a framework for exchange of materials between agents. Agents must be represented and implemented in independent ways.

The main benefit provided by the agent model is reuse. An agent might be used in several different agent systems.

4. INTELLIGENT AGENTS

An intelligent agent is a (complex) computer system that is capable of flexible autonomous actions in some environment in order to satisfy its design objectives.

The flexibility means:

- Reactivity: agents perceive their environment through sensors and act upon that environment through effectors.
- Pro-activeness: agents do not simply act in respond to their environment, they are able to exhibit goal-directed behavior by taking the initiative.
- Social ability: agents communicate (request and/or responds) with others (agents and non-agents) via agent language.

For some researches intelligent agents have stronger and more specific properties than the properties identified above.

Intelligent agents are (complex) computer systems that have the properties identified above plus the properties conceptualized and/or implemented using concepts such as knowledge, belief, choice, decision, capability, intention, obligation, commitment, etc.

Some researches go further and consider emotional concepts.

Intelligent agents, sometimes, have properties, such as:

- Mobility: agents move around a network.
- Veracity: agents believe in the truth of their information.
- Benevolence: agents do not have conflict goals and always try to do what is asked of it.
- Rationality: agents act in order to achieve its goals and do not act in a way as to prevent its goals being achieved.

Intelligent agents with the properties usually characteristic for human beings are computer surrogate for persons and/or processes that fulfills stated needs or activities.

An ideal agent is an intelligent agent with behavior that maximized its performance measures, on the basis of experience and built-in knowledge.

An autonomous ideal agent is an ideal agent that takes the actions based completely on its experience. It operates successfully in a wide variety of environments, given sufficient time to adapt.

A non-autonomous ideal agent is an ideal agent that takes the actions based completely on its built-in knowledge. It operates successfully while the build-in knowledge - assumptions hold.

It is reasonable to provide an ideal agent with some initial knowledge as well as an ability to learn.

Information retrieval (network) agent is a very popular form of intelligent agents whose main task (job) is to find and gather relevant information.

5. FORMAL SPECIFICATIONS OF AGENTS

The agent's environment is a collection of resources available to the agent. The environment can be represented as a set S = {s1, s2, …} of environment states. At any given instant, the environment is assumed to be in one of these states.

The agent's effectoric capability can be represented as a set A = {a1, a2, …} of actions.

An agent can be represented as a function

action: S* -> A

which maps sequences of environment states to actions.

For most agents, a function action can be specified by a table. For most agents, such a table is infinite unless the number and length of the sequences of environment states and actions are bound.

Environment might be:

- accessible (vs. inaccessible)
- deterministic (vs. nondeterministic)
- episodic (vs. nonepisodic)
- static (vs. dynamic)
- discrete (vs. continuous).

An environment is accessible to an agent, if an agent's sensors give agent access to the complete state of the environment.

An environment is deterministic, if the next state of the environment is determined by the current state and the action selected by the agent.

An agent decides what action to perform on the basis of its history – its experiences to date.

Experience (experience of an agent, agent experience) is

a collection of evidences provided to the agent by history.

An experience (history) can be represented as
a sequence of environment states – those that the agent has thus far encountered.

experience = {s1', s2', …}

The non-deterministic behavior of an environment can be represented as a function

environment: S X A -> environment-states(S)

which takes the current state of the environment s in S and an action a in A (performed by the agent), and maps them to a set of environment states environment(s,a) – those that could result from performing action a in state s.

If all the sets in the range of environment are all singletons, then the environment is deterministic, and its behavior can be accurately predicted.

An environment is episodic, if the agent's experience is divided into "episodes." An episode consists of the agent perceiving and then acting.

An environment is static, if the environment cannot be changed while an agent is deliberating.

An environment is discrete, if there are a limited number of distinct, clearly defined percepts and actions.

Some environments are more demanding than others. Environments that are inaccessible, nondeterministic, nonepisodic, dynamic, and continuous are the most challenging.

6. SOME PROPERTIES OF AGENTS

A history h is a sequence:

h: s0(a0) -> s1(a1) -> s2(a2) -> … -> sn(an) …

s0 is the initial environment state (i.e., its state when the agent starts executing).

si, i=1, 2, …, n, … is the n'th environment state (which is one of the possible results of executing action an-1 in state sn-1).

an, n=0, 1, 2, … is the n'th action that the agent chose to perform.

If

- action: S* -> A is an agent,
- environment: S X X -> environment-states(S) is an environment, and
- s0 is the initial state of the environment,

then the sequence

h: s0(a0) -> s1(a1) -> s2(a2) -> … -> sn(an) -> …

represents a possible history of the agent in the environment iff the following two conditions hold:

- for all u in N, a(u) = action((s0, s1, …, sn)) and
- for all u in N, such that u > 0,
sn in environment(s(n-1), a(n-1)).

Behavior (behavior of an agent, agent behavior) is the action of an agent caused by its history.
Agent behavior depends only on its history.

The characteristic behavior of an agent

action: S* -> A in an environment
environment: S X A -> environment-states(S)

is the set of all the histories that satisfy these properties.

If some property Fi holds of all these histories, this property can be regarded as an invariant property of the agent in the environment.

History(agent, environment) is the set of all histories of agent.

Agents agent1 and agent2 are:

- behaviorally equivalent with respect to environment env

iff history(agent1, env) = history(agent2, env)

- simply behaviorally equivalent

iff they are behaviorally equivalent with respect to all environments.

In general, agents whose interaction with their environment does not end, i.e., they are non-terminating are very important.

The right (intelligent) action (behavior) is the best possible action (behavior) of an agent in a given situation. It causes the agent to be most successful.

Goal (performance measure) the agent is supposed to achieve is a standard that defines degree of success of an agent in an environment.

Example

The performance measures of vacuum cleaner agents is a function of:

- the amount of cleaned up dirt
- the spent time
- the amount of consumed electricity
- the amount of generated noise

The time of performance evaluation is very important.

Example

Vacuum cleaner agents clean up more dirt in the first half of the work than in the second half of the work. The performance measure and the ways of use it is imposed by some authority.

7. TYPES OF AGENTS

There are four types of agents:

- A simple reflex agent, which responds to what it senses from its environment
- A keeping-track agent, which keeps track of the world
- A goal-based agent, which acts to achieve its goals
- A utility-based agent, which acts to maximizes own internal metrics.

8. AGENTS THEORY

Agent(s) theory (theory of agent(s), agency(ies) theory, theory of agency(ies)) is a collection of formalisms focussed on the aspects of agents.

Agents theory studies the relationships between the agents' attributes.

Example

Agents theory studies:

- How an agent's information and pro-attitudes are related
- How an agent's cognitive state changes over time
- How the environment affects an agent's cognitive state
- How an agent's information and pro-attitudes lead it to perform actions.

9. AGENTS VERSUS NON AGENTS

The world is not strictly divided into agents and non agents.

Example

A clock can be thought of as:
- An inanimate object or
- A simple, degenerate, semi-intelligent agent.

A clock is a degenerate agent because its percept sequence is empty. No matter what happens outside, the clock's action should be unaffected.

A clock is an intelligent agent because it always does the right action - moving its hands or displays its digits in the proper fashion.

A clock is a semi-intelligent agent because if it is moved from one time zone to another time zone, the right thing for the clock is to update its time, but it cannot do by itself.

10. SOFTWARE AGENTS

A software agent is a software process that exhibits the properties listed above.

Examples of Software Agents

- Software daemons, (such as background processes in the UNIX operating system), which monitor a software environment and perform actions to modify it are software agents.
- Agents for electronic mail handling.
- Agents for meeting scheduling.
- Agents for electronic news filtering.
- Agents for recommending books, journals, etc.

Software agents must contain the following most important information:

- Owner: user name, parent process name, or master agent name.
People, processes, and/or other software agents can spawn agents.
- Author: development owner, maintenance.
People or processes may create software agents.
- Creation date: date of request.
- Account: billing information, addresses, and etc.
- Goal: goal statement(s) and measures for success.
Statements of successful agent task completion and metrics for determining the task’s point of completion and the value of the return are necessary.
- Subject description: topic name and topic description attributes.
The subject description details the goal’s attributes.
The goal’s attributes provide the boundaries of the agent task.
- Duration: response requested by a certain date.
- Background: supporting information.

Software agent may and probably will contain much other useful information.

One of the most difficult aspects of software agent design is to define specific tasks that are both feasible using current technology, and are truly useful to the everyday user.

11. AGENT-ORIENTED PROGRAMMING LANGUAGES (AOP LANGUAGES)

An agent-oriented programming language (AOP language) is a programming language that allows programming in the terms of the concepts developed in agent theory.

An AOP language includes the structures and attributes and notions corresponding to agents.

Concurrent object programming languages (Actor, ABCL, and etc.) are in many respects the ancestors of the AOP languages. A self-contained, concurrently executing object, with a internal state that is not directly accessible to the outside world, and responding to messages from other such objects, is very close to the concepts of an agent.

The well-known AOP languages are:

- AGENT0
- PLACA (PLAnning Communicating Agents)
- Concurrent MetateM
- APRIL
- MAIL
- ABLE (Agent Behavior LanguagE)
- ACL (Agent Communication Language)
- KQML (Knowledge Query and Manipulation Language)
- KIF (Knowledge Interchange Format)

AOP language AGENT0 is a protype AOP language.

12. AGENT-ORIENTED PROGRAMMING SYSTEM (AOP SYSTEM)

An agent-oriented programming system (AOP system) has three primary components:

- A formal language for specifying agents
- A programming language for programming (instructing and defining) agents
- An agentifier for converting agent programs into executable programming.

AOP-system=(Formal-L,Programming-L,Agentifier)

13. AGENT-ORIENTED PROGRAMMING (AOP)

Agent-oriented programming (AOP) is a programming style that uses agents as basic units. AOP proposes agent programming in terms of mentalistic notions such as belief, desire, and intention. The intent of AOP is to provide an environment in which agents can interact with one another. Such an infrastructure greatly simplifies the creation of agents and makes it possible to study the properties of agents and their interactions.

14. AGENT-ORIENTED PROGRAMMING (AOP) VERSUS OBJECT-ORIENTED PROGRAMMING (OOP)

Agent-Oriented Programming (AOP) is a special case of Object-Oriented Programming (OOP).

The following table summarizes the relation between AOP and OOP.

	OOP	**AOP**
Basic unit	object	agent
Parameters defining state of basic unit	unconstrained	belief, decision, capability, obligation, choice, commitment, ...
Process of computation	message passing and response methods	message passing and response methods
Types of message	unconstrained	inform, request, offer, promise, accept, reject, assist, ...
Constraints on methods	none	honesty, consistency, ...

15. IMPLEMENTATION OF AGENTS

An agent can be implemented as:

- An agent architecture or
- An agent function.

Agent architecture is a classical approach to building agents viewing them as a particular type of knowledge based system. Typically it includes definitions of software data structures and operations on these structures.

An agent function maps a percept from an environment to an action. It uses some internal data structures updated as a new percept arrives. These data structures are operated on by the agent's decision-making procedures to generate an action choice, which is then passed to the architecture to be executed.

Example of a skeleton agent function

Imperative Solution

```
action_type function skeleton_agent
                              (percept_type percept)
  {
     local
        memory_type memory;
        action_type action;

     memory = update_memory(memory, percept);
     action = choose_best_action(memory);
     memory = update_memory(memory, action);

     return action;
  }
```

Example of a lookup table driven agent function

Imperative Solution

```
global
    table_type table;

action_type function
lookup_table_driven_agent(percept_type percept)
    {
        local
            percept_sequence_type percepts = empty;
            action_type action;

        percepts = append(percept, percepts);
        action = lookup(percepts, table);
        return action;
    }
```

Applicative (Functional) Solution

```
global
    table_type table;

action_type function
lookup_table_driven_agent(percept_type percept)
    {
        local
            percept_sequence_type percepts = empty;

        return lookup(
                        append(percept,percepts),   table
                        );
    }
```

Function lookup_table_driven_agent operates by keeping in memory entire percept sequence and using it to index into table, which contains the appropriate action for all possible percept sequences.

16. AGENT PROGRAM

Agent program is a collection of agent functions.

Agent program runs on some sort of computing device, called architecture.

The architecture might be:

- A plain computer,
- A special-purpose hardware for certain tasks, or
- A software that provides a degree of insulation between the raw computer and the agent program.

The architecture:

- Makes the percepts from the sensors available to the agent program,
- Runs the agent program, and
- Feeds the agent program's action choices to the effectors as they are generated.

The relationship among agent, architecture, and agent program can be expressed as:

agent = architecture + agent program

17. INFORMATION AGENTS

An information agent is an information retrieval (network) agent program to roam a network, interact with its host, gather information, and come home. They have access to multiple, potentially heterogeneous and geographially distributed information sources.

One of the main tasks of the information agents is an active search for relavant information. This includes retrieving, analyzing, manipulating, and integrating information available from different information sources.

Information agents must be able to negotiate with the network and ask for permission for everything they do. They must be ready to prove who they are and show that they have the authority to act for their masters. Upon arrival, they must be ready to negotiate credit for the host's services and, in the event that credit is not available, the network agent programs must be able to pay up front.

Some elements that make up an information agent are:

- state
- initialization functions
- main function
- cleanup functions
- authorization
- return address
- spending limits
- cost support

Current network transactions are too basic to be efficient. To do anything beyond making a simple request of a distant file server, several rounds of information must be exchanged before the goal is achieved. These exchanges take time and consume network resources. In many cases, a request might spend more than 90% of its time communicating.

Acknowledgements

This research was supported by:

- Advanced Telecommunication & Information Distribution Research Program (ATIRP), 1997-98.
- The MSU HUD/EDI Special Projects Grant, 1999.

References

1. Lupton, William and Stojkovic, Vojislav, 1998, "Solving Incomplete and Incorrect Information Problems using Conditional Planning, Execution Monitoring, and Situated Planning Agents." Twelfth Annual ASEET Symposium, The Naval Postgraduate School, Monterey, California.

2. Lynch, D.C. and Rose, M.T., 1993. Internet System Handbook, Addison-WesleyPublishing Company, Inc.

3. Russell, S. and Norvig, P., 1995, Artificial Intelligence: A Modern Approach, Prentice Hall Series in Artificial Intelligence.

4. Shoham, Y., 1993. Agent-oriented programming. Artificial Intelligence, 60, 51-92.

5. Stojkovic, Vojislav, Nnadi, Nkechi, and Jordan, Gareth, 1999, "Experiments with Concurrent Programming in Java." Neat Networks'99 Conference, The University of Texas at El Paso, El Paso, Texas.

6. Stojkovic, V. and Lupton, W., 1996, "Solving the Nine Tiles Problem Using the Genetic Algorithm Implemented in the Maple Programming Language." Intelligent Systems: A Semiotic Perspective, International Multidisciplinary Conference, Gaithersburg, Maryland.

7. Stojkovic, Vojislav and Tannouri, Samir, 1995, "C++ Object-Oriented Programming Implementation of Two Dimensional Binary Image Compression Using Hierarchical Coding Technique." Twenty-Ninth Annual Conference on Information Sciences and Systems, Department of Electrical and Computer Engineering, The Johns Hopkins University, Baltimore, Maryland.

8. Stojkovic, Vojislav and Tannouri, Samir, 1994, "Proving the Classical Theorems of the Predicate Calculus Using the PROVER." The American Society for Engineering Education (ASEE) Annual Conference, Edmonton, Canada.

9. Stojkovic, Vojislav and Tannouri, Samir, 1993, "Programming in Mathematica Using Accumulate and Reduce Functionals." The Sixth Annual International Conference on Technology in Collegiate Mathematics; Parsippany, New Jersey.

10. Stojkovic, Vojislav and Tannouri, Samir, 1993, "EXPET: A Microcomputer Based Expert System for Immigrant Petition for Alien Worker." Third Annual Conference of the Urban Business Association, Baltimore, Maryland.

11. Stojkovic, Vojislav, 1992, "An Overview of Parallel Computing Techniques." Second Annual Users Working Group Conference, Goddard Space Flight Center, Greenbelt, Maryland.

12. Stojkovic, Vojislav, 1991, "The Synthesis of a Predicate from Input/Output." Transactions of the DECUS.

13. Stojkovic, Vojislav, 1991, "An Implementation of Wang's Algorithm in COMMON LISP." Transactions of the DECUS.

14. Tannouri, Samir and Stojkovic, Vojislav, 1993, "Data Compression Algorithm for Network Communication." MU-SPIN Third Annual Conference, Greenbelt, Maryland.

15. Tosic, Dusan and Stojkovic, Vojislav, 1999, "An Implementation of the Distance Learning Based on the Profound Using of Java-Applets." Neat Networks'99 Conference, The University of Texas at El Paso, El Paso, Texas.

16. Watson Mark, 1996, AI Agents in Virtual Reality Worlds, Programming Intelligent VR in C++, John Wiley & Sons, Inc.

17. Watson, Mark, 1996, Programming Intelligent Agents for the Internet, McGraw-Hill.

18. Wayner, Peter, 1995, Agents Unleashed, A Public Domain Look at Agent Technology, AP Professional.