1995120958

N95-27379

S4-63
47254
p. 11

A User-System Interface Agent

Nagi T. Wakim
Sadanand Srivastava
Mehdi Bousaidi
Gin-Hua Goh

Center for Research in Distributed Computing
Department of Computer Science
Bowie State University
Bowie, MD 20715

ABSTRACT

Agent-based technologies answer to several challenges posed by additional information processing requirements in today's computing environments. In particular, (1) users desire interaction with computing devices in a mode which is similar to that is used between people, (2) the efficiency and successful completion of information processing tasks often require a high-level of expertise in complex and multiple domains, (3) information processing tasks often require handling of large volumes of data and, therefore, continuous and endless processing activities.

The concept of an agent is an attempt to address these new challenges by introducing information processing environments in which (1) users can communicate with a system in a natural way, (2) an agent is a specialist and a self-learner and, therefore, it qualifies to be trusted to perform tasks independent of the human user, and (3) an agent is an entity that is continuously active performing tasks that are either delegated to it or self-imposed.

The work described in this paper focuses on the development of an interface agent for users of a complex information processing environment (IPE). This activity is part of an on-going effort to build a model for developing agent-based information systems. Such systems will be highly applicable to environments which require a high-degree of automation, such as, flight control operations and/or processing of large volumes of data in complex domains, such as, the EOSDIS environment and other multi-disciplinary, scientific data systems.

The concept of an agent as an information processing entity is fully described with emphasis on characteristics of special interest to the User-System Interface Agent (USIA). Issues such as agent "existence" and "qualification" are discussed in this paper. Based on a definition of an agent and its main characteristics, we propose an architecture for the development of interface agents for users of an IPE that is agent-oriented and whose resources are likely to be distributed and heterogeneous in nature.

The architecture of USIA is outlined in two main components: (1) the user interface which is concerned with issues as user dialog and interaction, user modeling, and adaptation to user profile and (2) the system interface part which deals with identification of IPE capabilities, task understanding and feasibility assessment, and task delegation and coordination of assistant agents.

OVERVIEW OF AN AGENT-BASED MODEL

There are almost as many definitions of agents as there are researchers in this field. Traditionally agents have been defined according to their capabilities and architectures. For Miley, "intelligent agents" are nothing but programs "that learn the habits of a user, receive instructions, and then run off to receive or manipulate data" [Miley 1993]. Others, on the other hand, perceive agents as specialized action-oriented entities that can form a collaborative working-group and acquire their knowledge from past experiences available to each other as a they collectively attempt to solve a problem [Lashkari 1994]. While Shoham defines his agent as an entity that is perceived to have different states in line with mental components such as belief, capability, choices, and commitments [Shoham 1993]. Lastly, Ted Selker sees an agent as a program "that simulates a human relationship, by doing something that another person could do" [Selker 1994].

However, we define an agent as an entity that is capable of performing information processing tasks which are *delegated* to it with *incomplete specifications*. An agent may be represented by a processing element, hardware or software, which is *qualified* to perform tasks in a particular *domain*.

Agent Characteristics

An agent can best be described by the following main conceptual and operational characteristics:

- ***Existence***

An agent exists as a processing element. It is created either by initiation or through cloning. A cloned agent inherits the same capabilities and qualification as its parent. However, an agent that is initiated for the first time will evolve to qualify through training.

- ***Self-Determination***

An agent must be able to describe its capabilities to potential users/clients. This property, which is a type of reflection, is essential in order for an agent to determine whether or not to delegate a task to a particular agent.

- ***Delegation***

An agent must be able to accept delegated tasks as well as be able to delegate tasks to other agents. Therefore, an agent may play the role of a client or a server depending on its responsibility in performing a task. However, delegation should occur only after it has been determined that an agent is capable of performing a task.

- ***Operation***

There is a number of capabilities which define the operational aspects of an agent:

(1) *Concurrency*: being able to operate in parallel and, therefore, contributing to an improved system performance.

(2) *Autonomy*: requiring minimum intervention from other agents or users and, therefore, possessing a greater level of independence.

(3) *Cloning*: being able to reproduce itself with identical capabilities and, therefore, maximizing system reliability and performance through dynamic parallelism.

(4) *Migration*: being able to relocate from one node to another in a distributed system. This can lead to improved efficiency through balancing workload, minimizing network communications, and providing locality of service.

(5) *Persistence*: being able to try different possibilities in a solution space until a task is performed provided no time constraint is violated.

- ***Communication***

An agent can communicate with other agents in four different ways:

(1) Direct manipulation takes place when an agent directly instructs another agent to render a service. This is used in support of task delegation.

(2) *Confirmation* is a way for an agent to ask another to confirm an action, usually by responding with *yes* or *no*.

(3) *Feedback* is a way of providing positive or negative reinforcement after the completion of a task. This helps agents assess their own performance and learn from their own experience.

(4) *Negotiation* is a way for two agents to enter into a brief dialog in order to agree on some terms and/or constraints before a task is delegated.

The Qualification and Trust Factor

Since agents are intended to perform complex tasks, mostly independent of the human user, it is essential that an agent be qualified to perform tasks in a particular domain. Therefore, we extend our prior definition of an agent to include *qualification* while recommending that a computer process does not qualify to be an agent unless it meets the following:

- The process's program must be correct. That is, it must conform to design specifications and testing standards based on proven software engineering principles. However, a correct program does not imply that the corresponding agent will be able to perform all the tasks delegated to it.

- The process must have access to a knowledge base within a well defined domain. The knowledge base itself must be correct, that is, its facts and rules are consistent and it has been verified by a (human) domain expert. However, completeness is not a prerequisite to correctness.

It is important that all of agent's qualification standards be observed, for it to qualify to perform various tasks, and be trusted [Lashkari 1994] by the users or the agents it assists. All agents become qualified after a period of training through which the knowledge base itself is built. Once an agent becomes qualified, it is then ready to assume its responsibilities. Therefore, since an agent is a representative of a user (directly or indirectly) with an opportunity of being delegated tasks, it has to be *trusted* based on certain level of *confidence* which can only be determined through *qualification.*

MULTI-AGENT BASED SYSTEMS

A multi-agent based system is an environment in which a community of agents work collaboratively on solving problems with a common domain. However, each individual agent has a particular role to play which depends on the expertise and the specialization of the agent.

Since agents are highly specialized and are often distributed over a network of computers, it becomes more difficult to provide potential users with transparent access to system services. Therefore, we introduce an interface agent to facilitate such access.

THE USER-SYSTEM INTERFACE AGENT (USIA)

USIA is a special agent that may be thought of as a "middle-man" between human users and an information processing environment (IPE). An USIA may also be viewed as a front-end system which provides human users with a transparent interface to a community of agents of which each agent may have a different type of expertise and, hence, a special interface protocol. Therefore, without an USIA, a user who is in need of information processing services will need to first locate other agents in the IPE that are capable of performing its task and then learn to interact directly with each of them based on their interface protocols.

USIA offers an intuitive approach to the way a user can request services from an agent-based system by shifting the burden of locating and interacting with agents from the user to itself.

Main Responsibilities of USIA

USIA accommodates interaction with a whole spectrum of users ranging from novices to experts. In doing so, it performs a series of tasks:

- *User Dialog*: USIA provides its users with interaction capabilities through a graphical interface which offers two types of interaction media: (1) a taxonomy-based 'select-and-combine' type of interface that is dynamically derived from domain-specific services which are available in the IPE and (2) a restricted query language that is simple enough for novice users to state their fuzzy and often ambiguous requests, but is capable enough for expert users to state their specific and often complete requests.

- *User Adaptation*: The main advantage of USIA over a common interface system is that it is capable of monitoring user interaction with the IPE and, based on user modeling techniques, it is capable of adapting to changes in user profiles. The purpose of user modeling is to give USIA the ability to predict user behavior and, hence, assist the user more efficiently. USIA is also capable of gathering unobstructively usage patterns and offers facilities to automate them and build a knowledge base of user models. This knowledge base is then used in two ways: (1) to aid in resolving ambiguity in user requests and, hence, understanding them and (2) to predict possible next steps in user requests and, therefore, minimize interaction.

Task Understanding and Delegation: In order for USIA to handle high-level requests for processing information (e.g., data searches), it needs to complete a number of steps: (1) be aware of the capabilities of the IPE as reflected in a knowledge base and the services provided by the Agent Manager, (2) analyze and understand a request, (3) decompose a request into a set of tasks, (4) assess service feasibility based on the current state of IPE capabilities, i.e., availability of agents with the needed specialty, (5) delegate tasks to qualified agents, and (6) coordinate execution and assemble and communicate results back to the user.

A Real-World Analogy of USIA: The Hotel Concierge

One way to model USIA is to think of it as a concierge in a hotel environment whose main role is to assist hotel patrons in obtaining services which are in turn provided by various types of specialists. The pool of resources available to the concierge may include specialists such as car rental agents, travel agents, laundry cleaning agents, and taxi cab dispatcher agents.

As Figure 1 illustrates, a hotel concierge is an interface between a hotel guest (i.e., a user) and the specialist agents (i.e., the IPE). A hotel patron may ask for a variety of services from the concierge. Once the concierge accepts a service request, it then identifies the appropriate specialists and delegate responsibilities to them.

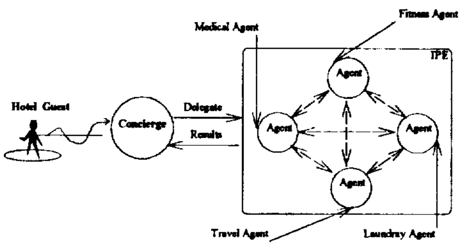


Figure 1. A Real-World Anaglogy of USIA.

Suppose, for example, that a guest desires to take a vacation somewhere, and would like the concierge to handle all the necessary arrangements, such as air travel, hotel accommodation, and tour guides. All the guest has to do is to present him/herself to the concierge and to ask for the services with the desired specifications, such as, intended date and time, travel destination, and cost range. The guest may also specify any special preferences that he/she might have concerning choice of an airline, the type of seat, and the type of meal.

The main point here is that the concierge, which is a special type of agent, must be able to provide different types of support and, therefore, handle the following different modes of interaction. However, in all cases, we assume that the service requester (i.e., a hotel patron) always has a goal or a purpose, such as, the intent to take a vacation.

1) The user knows the task (i.e., what to request, such as, arrange a vacation to Bermuda for two people during the month of January), knows the task is feasible, has the expertise (i.e., the 'know how'), but needs someone else to perform the task for him.

2) The user knows the task, knows the task is feasible, but does not have the expertise to perform the task.

3) The user knows the task but has no information on its feasibility.

4) The user only has a goal but has no knowledge of what to do, whether or not it is can be done, or how to do it.

Obviously, each type of user requires a different level of attention from the concierge and, therefore, the kind and length of dialog will vary with each type of user.

This example highlights an important role that a user interface agent can play in providing services transparently and efficiently to various types of users whose requests may require several sources of expertise in order to be serviced.

- ***Architectural Highlights***

As illustrated in Figure 2, the architecture of USIA is comprised of two main components: the User Interface (UI), and the System Interface (SI). The User Interface is responsible for facilitating the interaction with human users, monitoring their behavior in order to learn their habits, and be able to adapt in order to better serve their needs. In turn, UI interacts with the System Interface which is responsible for interacting with a community of specialty agents in the IPE in order to process service requests.

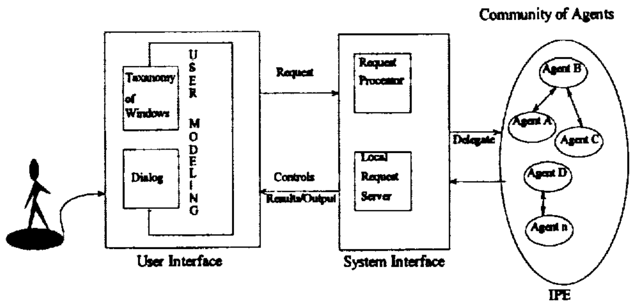


Figure 2. USIA's High Level Architecture.

Figure 3 outlines a detailed architecture of an USIA prototype system which has been developed as a front-end to an agent-based system, known as AFLOAT [Truszkowski 1993], for Report Generation in the Flight Operations domain at the NASA/Goddard Space Flight Center.

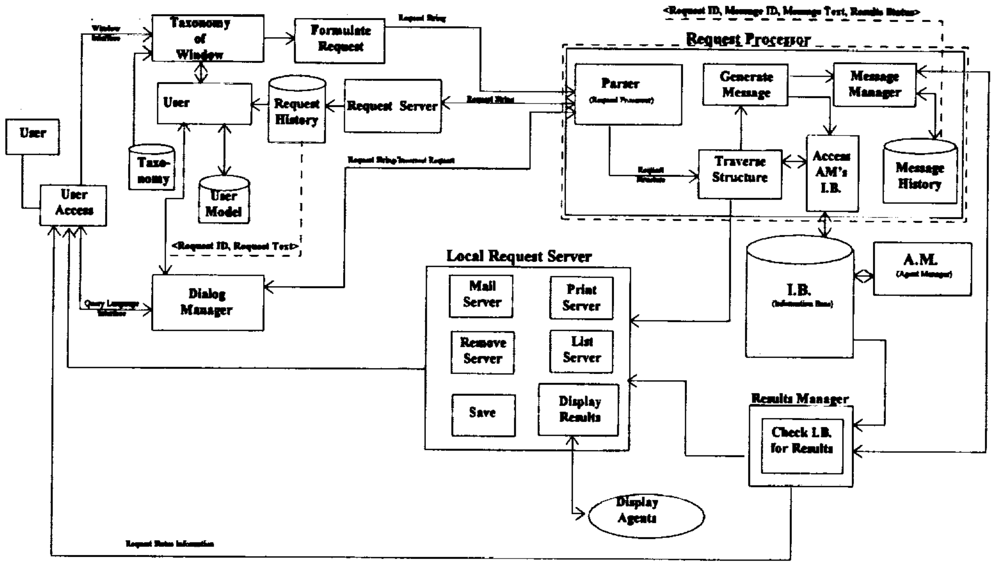


Figure 3. USIA Prototype Architecture.

- ***The User Interface Module***

The primary goal of the UI module is to formulate a user request for information processing services and pass it to the System Interface for processing. In the first phase of development, USIA employs two interaction mechanisms, as shown in Figure 4-a:

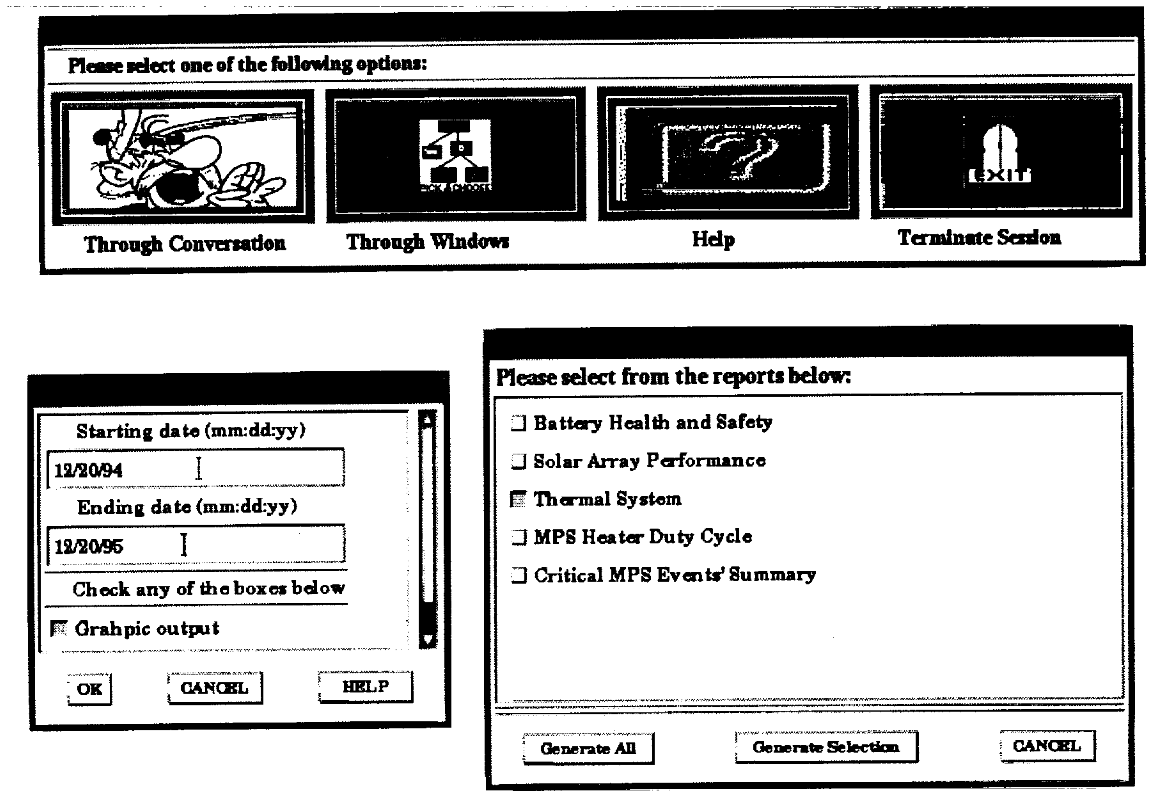
1. The user is presented with a dynamically generated, domain-driven taxonomy of windows from which the user 'selects-and-combines' services based on which a request statement is formulated by the system. Figure 4-b shows a snapshot of sample windows for the Flight Operations Report Generation domain.

2. The user types in a service request as a query statement chosen from a restricted, intuitive language which was developed for this domain. This language has capabilities which range from being able to show available services and generate reports to being able to display and mail reports. The following are some examples statements:

- ***show category** command and data handling subsystems*

- ***generate category** command and data handling subsystems **report** orbit decay **starting** 11/10/84 **ending** 12/13/84 **in** graphics*

- ***mail report** orbit-decay-1 **to** tom@internet, anne@internet*


Please select one of the following options:

Through Conversation | Through Windows | Help | Terminate Session

Starting date (mm:dd:yy)

12/20/94

Ending date (mm:dd:yy)

12/20/95

Check any of the boxes below

▨ Grahpic output

OK | CANCEL | HELP

Please select from the reports below:

❑ Battery Health and Safety
❑ Solar Array Performance
▨ Thermal System
❑ MPS Heater Duty Cycle
❑ Critical MPS Events' Summary

Generate All | Generate Selection | CANCEL

Figure 4-b. Graphical User Interface to a Report Generation and Management Application.

Once a request has been formulated, it is passed on to the System Interface for processing. Upon execution, results are then presented to the user through UI. Figure 5 illustrates the main steps of UI.

This version of USIA incorporates minimal user modeling techniques which include capturing user requests and logging them for comparative analysis in order to predict future user behavior in requesting services. It also allows for automating tasks based on users preferences for routine and off-line processing. However, efforts are underway in the second phase of USIA's development to employ a significant user modeling component which addresses issues such as: (a) modeling of individual users as well as classes of user populations, (b) a structure for user models, (c) techniques for identifying changes in users behavior and to reflect them in the corresponding models, and (d) methods for adapting to changes in user models in order to serve the end user more efficiently.

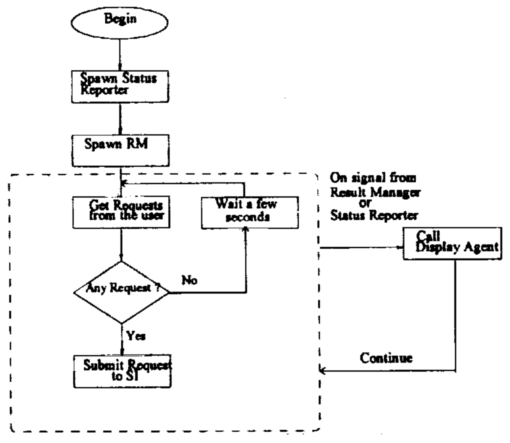


Figure 5. User Interface Flowchart.

- ***The System Interface Module***

The goal of the SI module is to process a service request which has been received from a user via the UI module. Each request is first parsed and analyzed for grammatical and semantically correctness. Upon detecting any errors (including ambiguity), USIA attempts to correct the request based on its knowledge of the domain and the user (through user modeling) and may enter in a dialog with the user for request clarification purposes if necessary.

In addition, SI is responsible for decomposing a request into tasks- a *task* is defined as one unit of work which can be delegated to a single agent at one time. Therefore, depending on the available pool of specialty agents, a service request may be decomposed into one or more tasks. Also, a request may be either *local* or *remote*. A local request is one which can be processed by USIA and need not be delegated to another agent, for example: a request to list reports which have been already generated and are saved in the user's work space.

However, a remote request is executed by delegating each of its corresponding tasks to an agent that is capable of performing it. In order for USIA to assess the feasibility of a request, it utilizes an Information Base (IB) which catalogs information on all which are available in the IPE at that particular time. For each agent, the IB stores a list of its skills which is used by USIA in order to determine which, if any, agent is capable of performing a particular task. Upon making such a determination, SI formulates a special message and delegates the task to the agent while assisted by an Agent Manager (AM), which is responsible for locating the agent and dispatching the message (i.e., the task) to it. In our present configuration, there is one AM for each node of a distributed IPE.

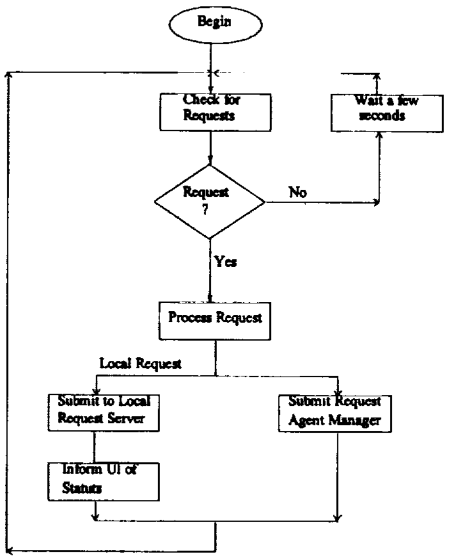


Figure 6. System Interface Flowchart.

Once a task is delegated and performed by an agent, its results are communicated to a Results Manager via the IB. The Results Manager is a special daemon which is responsible for assembling outcomes from processing a request (by executing one or more tasks) and for informing the UI module, which in turn notifies the user. Once results are ready, a user may choose to display them and/or save them. Special agents are utilized depending on the type and format of the result object.

Upon request by user, special agents are utilized to display the results depending on the format and type of a result object.

Integrating UI and SI Modules

Figure 7 illustrates the cyclic flow of high-level activities from the user through the different components of USIA and the interaction with the IPE, via the Agent Manager and the Information Base, and back to the user. We should note that the whole processing of a request is done in the background and transparently from the user.

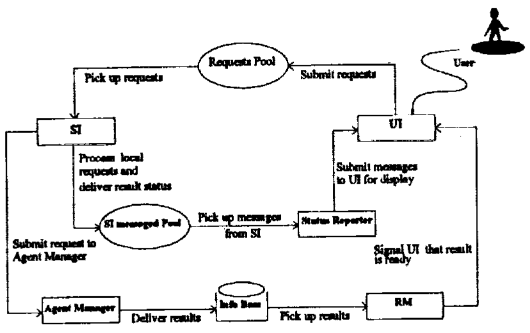


Figure 7: User Interface & System Interface Interaction

CONCLUSION

An Assessment

The first version of USIA demonstrated a few limitations at the User Interface level. Our form of domain restricted query language proved to be not as flexible as we had hoped, especially for novice users. The taxonomy of windows option was also a bit cumbersome to use, simply because the user had to go through several levels before a request could be formulated. Also, we noticed that, specially for requests that might require extended time to be serviced, there is a need to display status information during the different processing stages of a request and to allow the user to abort a request at any time after it has been delegated to USIA.

Further Work

The above limitations and other proposed features have posed several challenges in the USIA project. Work is already underway in the second phase of development to make progress in two main areas: *ease of use* and *intelligence*. To this end, the following issues and features are being addressed:

- Provide for a two-way *voice interface* for interaction between users and USIA.

- Provide for a full *natural language* processing capability for interface and request delegation purposes.

- Incorporate a capable *user modeling* subsystem which would support modeling of different user types in a multi-domain environment.

Our experience has been challenging but enjoyable. We believe that any progress in this field is bound to have a significant impact on the way people perceive and work with computers.

ACKNOWLEDGMENTS

We wish to acknowledge the participation of other members of the USIA project, specially Alexander Glockner, Amit Thukral, Gayathri Rao, Jerome Farley and Mark Flemming. We are thankful for their valuable contribution.

REFERENCES

1. Michael Miley (Agent Technology) MacWeek, Apr. 19, 1993.

2. Yezdi Lashkari, Metral Max, and Maes Pattie. 1994. Collaborative Interface Agents. In *proceedings of the National Conference on Artificial Intelligence,* 444-449, Seattle, Washington: AAAI Press.

3. Yoav Shoham. 1993. Agent-oriented programming. Artificial Intelligence, Vol. 60 (1), pp. 51-92, Elsevier.

4. Henry Kautz, Selman Bart, Coen, Michael, Ketchpel Steven, and Ramming Chris. 1994. An experiment in the Design of Software Agents. In *proceedings of the National Conference on Artificial Intelligence,* 438-443, Seattle, Washington: AAAI Press.

5. Norman Donald. 1994. How Might People Interact with Agents. Communications of the ACM, Vol. 37, N.7, pp. 68-71.

6. Etzioni Oren, Weld Daniel. 1994. A Softbot-Based Interface to the Internet. Communications of the ACM, Vol. 37, N. 7, pp. 72-76.

7. Ted Selker. 1994. "Coach: A Teaching Agent That Learns". Communications of the ACM, Vol. 37, N.7, pp. 92-99.

8. Truszkowski, Moore, Odubiyi. 1993. Working paper on "A Multi-Agent-Oriented Approach to Support Satellite Flight Operations Analysis. NASA/Goddard Space Flight Center. Greenbelt, MD.