



James Odell

jodell@compuserve.com

www.jamesodell.com

Considerations for Agent-Based Technology

WITHIN THE OMG, AN AGENT WORKING GROUP was established to provide a forum for building industry consensus and convergence around agent technology development. One of this group's first efforts is identifying the key features of agent technology. This column presents excerpts from the group's efforts—in particular, the areas which the work group considered vital to the success of developing and deploying agent-based systems.¹

Agent Communication

Currently, one of the most important areas for standardization is agent communication. If every designer developed a different means of communicating among agents, our agent systems would be worse than a tower of Babel. Not only would the content and meaning of a communication likely be different, but the means of communication could occur in a variety of ways.

- **Agent communication languages.** Messages must have a well defined semantics that is computational and visible. Thus, we need standardized agent communication languages (ACL), so that different parties can build their agents to interoperate. Furthermore, they must have a formal semantics so that different implementations preserve the essential features of the ACL. By specifying an ACL, we effectively codify the basic elements of interaction that can take place between agents.
- **Message transportation mechanism.** In agent environments, messages should be schedulable as well as event driven. They can be sent in synchronous or asynchronous modes. Furthermore, the transportation mechanism should support unique addressing as well as role-based addresses (i.e., “white page” versus “yellow page” addressing). Lastly, the transportation mechanism must support unicast, multicast, and broadcast modes and such services as broadcast behavior, non-repudiation of messages, and logging.
- **Ontology communication.** Agent communication implies that concepts will be part of a communication among agents. Furthermore, agents can have different terms for the same concept, identical terms for different concepts, and different class systems. A common ontology, then, is required for representing the knowledge from various domains of dis-

course. The two primary challenges here are building them and linking them.

- **Conversation policy.** Agents can interact in various patterns. Cooperative agents work toward a common goal. For example, to produce a coherent plan, agents must be able to recognize subgoal interactions and either avoid or resolve them.

Partial Global Planning does not assume any distribution of sub-problems but allows the agents to coordinate themselves. Joint intention frameworks require joint “commitment.” The *Shared Plan* model is based on the mental attitude—the intention to act. *Joint Responsibility* is based on a joint commitment to the team's goal and the recipe for attaining that goal. Self-interested multiagent interactions are usually centered around negotiation and contract nets.

Whichever kind of conversation is chosen, the pattern—or protocol—of that conversation must be understood by the participating agents. Standards and guidelines here would greatly accelerate our usage of agents, because they could then “converse” without requiring human intervention.

Agent Mindspace

As we design and build agents that are more intelligent, we need to consider aspects such as understanding, adaptation, beliefs, desires, intents, and knowledge.

- **Goal representation and manipulation.** Many theories of goals exist. In the early work on problem solving, a goal was a state to be achieved or a proposition to be made true. However, agent goals could be seen as divided into various classes: achievement goals (long-term goals), satisfaction goals (recurrent goals such as resource gathering), preservation goals (for preserving life and property), and delta goals (i.e., other state changes).

Also, in the BDI (beliefs, desires, intents) approach, *desires* describe the agent's goals (sometimes including a motivational aspect) and *intentions* characterize the goals (desires) that the agent has selected to work on. Goals can have sub-goals, particularly in problem solving. We need to address both the maintenance and the communication of goals.

- **Reactivity versus proactivity.** One of the minimum requirements for an agent is that it is *reactive*. Reactive agents can selectively sense/perceive events (state changes, messages) in their environment and respond to them. The syntax for reactive agents is typically in the form of WHEN event IF con-

James Odell is a consultant, educator, and author in the area of object-oriented and agent-based systems.

dition THEN action/assertion. As such, inference engines can be used by reactive agents.

In contrast, *proactive* agents do not simply react, they should be able to exhibit opportunistic, goal-directed behavior and where appropriate take the initiative. Proactive agents can choose to query the environment about its state rather than just wait until events arrive; that is, they are active rather than just receptive. Agents that continuously reason about future events are sometimes called *deliberative*.

- **Adaptive agents.** Adaptive agents can be simply reactive or they can learn or evolve. Agents that learn or evolve can change their behavior based on their experience with other agents and the environment. Here, learning and evolution is orthogonal to reaction/proaction.
- **Procedural versus declarative process specifications.** Procedural approaches specify how a computation should proceed; declarative approaches specify what an operation should do rather than how. Both approaches have their merits and therefore both should be considered.
- **Comprehension of an environment.** In order for an agent to recognize features and characteristics of an environment, interfaces are required through which an agent can obtain service information. The CORBA 2.2 specification provides the `get_service` interface (PIDL) at the level of the ORB. An equivalent interface is required to support the registration and retrieval of information concerning available facilities and applicable policies. Such interfaces should enable an agent both to evaluate an environment and to register itself with the environment as a potential services provider.

Life-Cycle Management

Agents will be software running in software environments. As such, they must have well understood mechanisms for such activities as starting, stopping, being managed, and being traced. Some agents are implemented as mobile code, which introduces additional lifecycle issues, such as permissions to run, permissions to perform certain tasks, and to have communication occur at different locations other than its original starting point. Finally agents can evolve and can possibly clone themselves, which introduces issues related to the delegation of responsibilities and permissions.

As we examine lifecycle management, we must also examine the software environments in which agents will run. These can be very small, intermittently connected devices like cell phones or Personal Digital Assistants, to very large clusters of servers capable of running huge numbers of agents. The requirements for each environment are quite different, and each must be accommodated.

- **Virtual existence or persistence.** Agents may logically exist for indeterminate periods of time during which they may display dormant behavior. Interfaces supporting agent lifecycles need to consider requirements for “logical” existence of possibly very large numbers of agents as opposed to in-memory physical existence. This has a number of implica-

tions around storage, messaging and communication, and management, particularly for agents that can be “put to sleep” and saved in some way. When the agent is persistent, the current state and data are preserved, and are restored when the agent is “awakened.” There are often other state transitions associated with “putting the agent to sleep” and “awakening” it, having to do with querying whether it is currently in a situation where this can successfully be performed, and on awaking, a notification that time has elapsed and it might want to reassess the environment.

- **History.** Mechanisms are required to provide a historical recording of the agent’s actions—so that agent behavior can be audited and that agents can evaluate prior actions. This could include a range of situations, from merely obtaining an agent’s state to providing a comprehensive log of actions.
- **Mobility.** Static agents exist as a single process on one host computer; mobile agents can pick up and move their code to a new host where they resume executing. From a conceptual standpoint, such mobile agents can also be regarded as itinerant, dynamic, wandering, roaming, or migrant. The rationale for mobility is the improved performance that can be achieved by moving the agent closer to the services available on the new host.

Mobile agents can also be part of an agent system that has static agents. For example, there may be a large “intelligent” agent coordinating a set of actions (such as a workflow, a negotiation, or data synthesis), which in turn sends out smaller mobile agents to perform a specific task, such as acquiring some data. Mobile agents create an additional set of requirements. For example, they require an agent server where they can run. They also introduce the complexities of management and security. Naming and identification become very important, as well.

- **Dynamic and multiple classification.** During an agent’s lifecycle, the interfaces exposed by the agent may be dynamic, reflecting changes in its state or environment. Mechanisms are required to support the existence of multiple interfaces relative to a single agent identity. Furthermore, control mechanisms must be established to enable and disable an agent’s features. Such mechanisms are particularly useful when multiple roles are being supported.

Depending on the situation, the roles or behavior of individual agents can change. For example in ant societies, workers can become foragers when food availability increases, and nest-maintenance workers can become patrollers when foreign ants intrude. However, once an ant is allocated to a task outside the nest, it never returns. In human societies we not only change our roles, we can assume many roles at the same time. For example, we can be parents, employees, and customers. In other words, an agent can both change its role (dynamic classification) and assume more than one role at any moment in time (multiple classification).

Agent is a Principal

A key attribute of an agent is that it be able to act autonomously. Agents can then take on a wide range of responsibilities on our behalf, including purchasing goods and services and entering into agreements. Furthermore, agents may have several identities which they can use while operating. Any robust system will consider these identities in its transactions with an agent.

- **Agents are operating on behalf of an entity.** An agent will often perform some tasks on behalf of another. For example, a software agent could perform a task on behalf of a person. It could also perform on behalf of another piece of software (another agent), an organization, or a particular role (manager, system administrator). This leads to a number of issues that need to be considered. For example, an agent needs to be able to identify on whose behalf it is running. Furthermore, if it requests an action from another entity, the other entity may want to assess whether it will permit the action, based on the proffered identity.

Subsets of permissions are another issue. In human interactions, we delegate portions of our decision making routinely as well as in an *ad hoc* manner. For example, we might ask one of our co-workers to schedule the weekly status meetings staff meeting, knowing that the person will pick a convenient time, include the right people, and schedule an appropriate room. To another co-worker, we may delegate signature authority to spend up to \$1000 in our absence. In both cases, co-workers have been delegated authority but given very different permissions. A similar set of constructs needs to be created for agents.

- **Agents can be software.** Agents can be software and as such may be more or less reliable. Various schemes permit unknown software to run in an environment. For example, code signing might be employed to determine whether the agent will be permitted to run in a given environment. Since an agent may in fact be composed of multiple objects, a model may be needed to assess the overall reliability of all components.

Agent modeling and specification

- **Agent modeling and specification.** We need to better understand how to develop applications using agent technology. Therefore, development tools and methods play an important part in agent-based systems. Here, we need to determine what kinds of modeling languages are required as well as the underlying metamodels needed to support agent specification and deployment. Furthermore, we need to ensure interoperability across the lifecycle of agent tools and their designs/work products.
- **Testing, debugging, validation, and simulation.** Concurrency and nondeterminism in execution make it very difficult to comprehend the activity of an agent system. We need visualization tools, debugging facilities, and simulation.
- **Agent system development methodologies.** We must deter-

mine the steps needed to guide the life cycle of agent system development—at both micro and macro levels. Existing methods can be used to some extent, but we need to identify how an agent-based approach will change them.

Agent enterprise architecture and services

For the most part, agents will be deployed within conventional enterprises and will draw on the enterprise for many services. CORBA provides a rich source of services and a proven architecture. This section provides a framework for considering how a system supporting agents might draw on CORBA services and facilities. The architectural basis for this discussion will be the FIPA architecture.² The FIPA Agent Platform provides a good construct from which to discuss the enterprise-related issues in agent deployment.

- **Agent Platform.** The key element to the enterprise architecture is the Agent Platform. An Agent Platform (AP) provides an infrastructure in which agents can be deployed. An agent must be registered on a platform in order to interact with other agents on that or other platforms. Minimally, an AP consists of three capability sets: an *Agent Communication Channel*, an *Agent Management System*, and a *Directory Facilitator*. The Internal Platform Message Transport (IPMT) is the local (possibly proprietary) means of exchanging messages within an AP.

FIPA does not specify the physical nature of a platform. However, two cases should be considered, that of a single host and that of multiple processors deployed as a “virtual” platform. If the Platform is virtual, having it fulfill several requirements would be wise. It should have high-speed communications, a single system manager, and a single security enclave.

These last two requirements make the agent system easier to use. The responsibility for humans managing the system is simplified. Also, we avoid the situation where control of the agent lifecycles on a platform is shared by several people. From the system perspective, the lifecycle of all agents in a given platform is controlled by the Agent Management System. From the perspective of a human (or agent proxy), the platform itself should also be controlled by a single entity.

- **Agent Management System.** The Agent Management System (AMS) is an agent that supervises access to and use of the AP. Only one AMS will exist in a single AP. The AMS maintains a directory of logical agent names and their associated transport addresses for an agent platform. The AMS offers “white pages” services to other agents and is responsible for managing the lifecycle of the agents on the platform.³ Its actions include authentication, registration, de-registration, modification, query platform profile, search, and control of agent lifecycle
- **Directory Facilitator.** The Directory Facilitator (DF) provides “yellow pages” services to other agents. The DF is a mandatory, normative agent. Agents may register their services

with the DF or query the DF to find out what services are offered by other agents.

- **Agent Platform Security Manager.** The Agent Platform Security Manager (APSM) is responsible for maintaining security policies for the platform and infrastructure. The APSM is responsible for run-time activities, such as communications, transport-level security, and audit trails. FIPA 98 security cannot be guaranteed unless, at a minimum, all communication between agents is carried out through the APSM.

The APSM is responsible for negotiating the requested inter- and intradomain security services with other APSMs in concert with the implemented distributed computing architecture, such as CORBA, COM, and DCE, on behalf of the agents in its domain. It is responsible for enforcing the security policy of its domain and can, at its discretion, upgrade the level of security requested by an agent. The APSM cannot downgrade the level of services requested by an agent but must inform the agent that the service level requested cannot be provided.

- **Agent Resource Broker.** An Agent Request Broker (ARB) is an agent that brokers a set of software/service descriptions to interested agents. Clients query it about what nonagent software services are available.
- **Wrapper Agent.** This agent allows an agent to connect to a nonagent software system/service uniquely identified by a software description. Client agents can relay commands to the wrapper agent and have them invoked on the underlying services. The role provided by the wrapper agent provides a single generic way for agents to interact with nonagent software systems.

The wrapper increases robustness by exposing an agent interface capable of exception and event handling for the requester. The use of an adapter also allows the agent manager to control platform access to the remote service.

- **Agent Communication Channel.** All agents have access to the Agent Communication Channel. It provides a path for basic interchange between agents, agent services, AMS, and other agent platforms. At least, it must support IIOP. Agents can reach agents on any number of other platforms through the Agent Communication Channel. Access to agents outside of the local namespace could be supported by the CORBA Trader Services.

Acknowledgments

I would like to acknowledge the following people who contributed to this Agent Work Group effort: Gregory Mack, Booz-Allen & Hamilton, Inc.; David Mattox, Mitre Corporation; Francis McCabe, Fujitsu Labs; Stephen McConnell, OSM sarl; Kate Stout, Sun Microsystems, Inc.; and Craig Thompson, Object Services and Consulting.

References

1. The Agents Work Group website is www.objs.com/isig/agents.html.
2. *Foundation for Intelligent Physical Agents FIPA98 Agent Management Specification*, Geneva, Switzerland, Oct. 1998.
3. *Foundation for Intelligent Physical Agents FIPA97 Agent Management Specification*, Geneva, Switzerland, Oct. 1997.