James Odell
jodell@compuserve.com

Agents and Beyond: A Flock Is Not a Bird

Imagine sitting in the park on a nice summer day as a flock of birds sweeps the sky. One moment they are a circling, another they dart to the left or drop to the ground. Each move is so beautiful that it appears choreographed. Furthermore, the movements of the flock seem smoother than those of any one bird in the flock. Yet, the flock has no high-level controller or even a lead bird. The phenomenon is a result of what is often called "self-organization."[1] Each bird follows a simple set of rules that it uses to react to birds nearby. In Craig Reynolds' simulation,[2] each bird behaved according to three simple rules:

1. Maintain a minimum distance from other objects, including other birds.
2. Try to match velocities with other birds.
3 Try to move toward the perceived center of the mass of birds in its neighborhood.

Orderly flocks emerge from simple rules such as these. No one bird has a sense of an overall flock. The "bird in front" is merely a position of a given bird. It just happens to be there—and it will be replaced by others in a matter of minutes. "The flock is organized without a organizer, coordinated without a coordinator."[1]

Flocks of birds are not the only things that work like this. Bee hives, ant colonies, freeway traffic, national and global economies, societies, and immune systems are all examples of patterns that are determined by local component interaction instead of centralized authority.

Agents

Another name for local component is agent. A basic dictionary definition of agent is "one who acts." Under such a broad definition, agents can have a host of properties. One way to think about these properties is as follows3 :

- ***Autonomous.*** Exercises control over its own behavior and state.
- ***Communicative.*** Socially able, communicates with other agents, such as humans, machines, and software agents.
- ***Mobile.*** Able to transport itself from one environment or platform to another.
- ***Reactive.*** Sensing and acting; responds in a timely fashion to changes in the environment.
- ***Temporally continuous.*** A continuously running process.
- ***Flexible.*** Actions are not scripted.
- ***Character.*** Believable "personality" and emotional state.
- ***Able to learn and evolve.*** Learning; changes its behavior based on its previous experience.

Most people who work with agents find this definition overly broad. They argue that an agent is not very useful without at least the first four of the above properties, so a typical definition for agent is "an autonomous entity that can interact with its environment."

Lately, we've been hearing a lot about agents in software.[4,5] While there are many definitions,[6] generally software agents are autonomous software entities that can interact with their environment. In other words, they are agents that are implemented using software. They are autonomous and can react with other entities, including humans, machines, and other software agents in various environments and platforms. The preceding example of birds was simulated using software agents. A similar set of agents was employed to animate the penguin sequence in a recent Batman movie.

Agents and Object Orientation

An agent-oriented approach is employed when a particular situation requires a decentralized and self-organized approach to processing instead of a centrally organized one. While a centrally organized program could have been written to handle the bird simulation, the system would have been far too cumbersome. It would have required a single set of top-level rules telling each bird precisely what to do in every conceivable situation. Not only would such an application be touchy and fragile, it would likely end up looking jerky and unnatural—more like an animated cartoon than animated life.[2]

Yet, most developers tend to build centrally organized applications. They are also biased toward object-oriented notions, such as class, association, and message. While these constructs are useful for a certain category of applications, they do not directly address the requirements of agents. As we have seen, agents have such characteristics as autonomy, mobility, and adaptability. Furthermore, business users like to express other concepts, such as rules, constraints, goals and objectives, and roles and responsibilities. In short, the agent-oriented approach distinguishes between autono-

James Odell is a consultant, educator, and author. He works with IntelliCorp, James Martin & Co., and Quoin, Inc.

mous, interactive, mobile objects (agents) and the passive objects of conventional object orientation. This does not mean that object orientation is dead; instead, it can be used to enable, rather than drive, agent-oriented technology.[4]

Adaptive Agents

A more advanced category of agents includes those that can learn and evolve. Such agents can change their behavior based on their experience. For example in Art Samuel's checkers game,[7] each agent could evaluate its move as well as learn to modify its evaluations over time based on playing experience. There are more established "intelligent" techniques based on such approaches as inference engines, neural nets, and genetic algorithms. So-called intelligent agents employ these techniques so that they can react, proact, or do both to their environment. Recently, adaptive agents have gone beyond the traditional approaches of intelligent agents. For example, the rule engines can now support contradictory rules as well as have the ability to learn.

Using a biological metaphor is also a common technique. Here, agents can mate, reproduce (either by giving birth or by cell division), exchange resources, absorb, kill, and die. Such a metaphor is appropriate for many life or life-like forms, such as cells, persons, work units, companies, societies, countries, or alliances among countries.8 Furthermore, such life-forms can evolve beyond predefined, close-ended possibilities. In Tom Ray's Tierra project,[9] the agents are capable of replication and open-ended evolution. In other words, new species of agents can emerge—species that were not planned or anticipated—solely through the workings of the evolutionary process itself.

Complex Adaptive Systems

We have thus far discussed agents only as individuals. Yet, what about the flocking phenomenon described at the beginning of this column? Individually, each bird followed three simple rules, yet a secondary effect, that is, the flock, was produced. While flocking was neither dictated nor expected from the rules, an entity with its own structure and behavior emerged. Tremendously interesting and beguilingly complex things can emerge from collections of extremely simple components.2 This concept of emergence is attributed to many real-life phenomena, such as molecules, amino acids, cells, immune systems, ant colonies, bee hives, ecosystems, societies, market economies, stock markets, organizations, supply chains, traffic jams, and the Internet.

In short, we are talking about a whole that is more than just the sum of its agents. Such systems are called complex adaptive systems (CAS)—sometimes called complex systems, for short. Complex systems theory is the study of systems where a great many independent agents interact with each other. It is a new way of thinking about the collective behavior of many basic but interacting units. These units can be simple, yet their interaction yields a complex result. This means that a flock is more than just a collection of birds, and a traffic jam is more than just a collection of cars. Complex systems theory, then, includes the study of transitions from one level of organization to a higher one, where the higher one can be a component in an even higher one, and so on. In the meantime, agents can learn and evolve, resulting in the birth, change, or extinction of individual agents as well as the complex systems in which they participated. 10

Implications for Software Development

The behavioral description of software agents needs to be expanded to include agent learning and evolution, as well as complex systems. For this to happen, software developers must be aware of the following issues11:

- ***Concurrency.*** In any complex domain, many things happen at once. A general framework must deal at its outset with concurrency and the autonomy of concurrent processes.
- ***Partial information.*** No part of the system can have information about any other part of the system unless specific interactions have occurred to transfer that information.
- ***Visibility and connection.*** For reasons of scalability, a given process can directly affect or query only a finite set of other processes.
- ***Creation and destruction.*** A process not only has limited knowledge of the other processes with which it can interact, but the collection of these processes is not fixed.
- ***History.*** The action of an encapsulated process can depend in general on any part of its history, including every interaction in which a process has participated since its creation.
- ***Reflection.*** The external behavior of an object can be a function not only of its history and an initial set of constitutive rules, but of internal processing that extends or evolves any initially predictable behavior.
- ***Emergence.*** The interaction of many individual agents can give rise to secondary effects where groups of agents behave as a single entity, or multiagent. While such occurrences are often unplanned and unexpected, they should be anticipated.

Next in this Column

The purpose of this month's column is to introduce the notion of agents. The phenomenon is not science fiction: many commercial IS tools that support agents, adaptive agents, and complex systems already exist (or are under development).

In future columns, I will explore in greater detail various areas of the agent technology, examining how it can be applied to business as well as software.

References

1. Resnick, M., *Termites, Turtles, and Traffic Jams: Explorations in Massively Parallel Microworlds,* Cambridge: MIT Press, 1997.

2. Waldrop, M. M., *Complexity: The Emerging Science at the Edge of Order and Chaos,* New York: Simon and Schuster, 1992.
3. Franklin, S., and Graesser, A., Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agent, *Proc. of Third International Workshop on Agent Theories, Architectures, and Languages,* Springer-Verlag, 1997, (www.msci.memphis.edu/~franklin/AgentProg.html)
4. Farhoodi, F., and Fingar, P., Intelligent Agents: Going Where No Objects Have Gone Before, *Distributed Object Computing,* 1:8, October 1997, home1.gte.net/pfingar
5. Farhoodi, F., and Fingar, P., The Next Big Thing: Developing Intelligent Enterprise Systems, *Distributed Computing,* 1:1, November 1997.
6. Graham, I., The Architecture of Agents, *Object Magazine,* 7:6, August 1997.
7. Samuel, A. L., Some Studies in Machine Learning Using the Game of Checkers, *Computers and Thought,* E. A. Feigenbaum and J. Feldman eds., New York: McGraw-Hill, 1963.
8. Axelrod, R., *The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration,* Princeton, NJ: Princeton University Press, 1997.
9. Ray, T., An Approach to the Synthetic Biology, *Artificial Life: An Overview,* C. Langton, Ed., Cambridge: MIT Press, 1997.
10. www.santafe.edu.
11. Burkhart, R., Effective Description of Enterprise Processes, position paper for the Workshop on Specification of Behavioral Semantics in Object-Oriented Information Modeling, OOPSLA '93, August 9, 1993. www.santafe.edu/~rmb/oopsla93.html