1997

The Agent Test

Michael N. Huhns
University of South Carolina - Columbia, huhns@sc.edu

Munindar P. Singh

Publication Info

Published in *IEEE Internet Computing*, Volume 1, Issue 5, 1997, pages 78-79.

AGENTS ON THE WEB

Michael N. Huhns • *University of South Carolina* • *huhns@sc.edu*
Munindar P. Singh • *North Carolina State University* • *singh@ncsu.edu*

THE AGENT TEST

A lot has been written on agents, and most of what has been written includes—implicitly or explicitly—a definition of an agent. But definitions lead to seemingly endless debate, and there are always counterexamples.

Recall the definition of an elephant in Ronald Brachman's "I Lied about the Trees."[1]

> An elephant is a large (what about a baby elephant), gray (oops, what about an albino), four-legged (what if it lost a leg in an accident), trunked (ouch!, another accident),. . .

A DEFINITION YET TO EMERGE

For reviews and comparisons of several definitions of agents, you might look at the papers by Petrie2 and Franklin and Graesser.3 Our own proposed definition appears in our column "Agents Are Everywhere!"[4]

However, we believe that definitions emerge over time and represent unstated consensus, such as answers to the questions "What is physics?" and "What is chemistry?" which were debated at the beginning of this century.

Ultimately an agreed-upon definition of an agent will emerge—but we don't want to wait that long.

IN THE MEANTIME, A TEST

Rather than wait for a definition, we propose the Huhns-Singh Test for Agenthood (cf. the Turing Test for intelligence):

> A system containing one or more reputed agents should change substantively if another reputed agent is added to the system.

This requires some elaboration:

- By substantively, we do not mean the system will simply slow down because another software process is running, but that the reputed agents will somehow be aware of each other and adjust their behavior accordingly.
- The added agent must be equivalent to the reputed agents. That is, it should have the same architecture and functionality, although it might have different goals, knowledge, or beliefs.
- The existing and added agents do not necessarily have to communicate. Nor do they have to be autonomous, persistent, or intelligent, although these qualities would help.

A Mental Experiment

When someone with a piece of software (call it X) tells you that X is an agent, perform the following mental experiment: Imagine a second X is added to the system. We claim that if X is an agent, its behavior in the presence of another X will differ from its behavior when alone. If X doesn't behave any differently, then it is not an agent.

For example, if you add another sensing agent in Distributed Vehicle Monitoring Testbed (DVMT),[5] the new agent will help confirm or eliminate vehicle tracks proposed by the other agents, thus affecting their behavior. These are agents.

In the blackboard-based distributed Hearsay-II,[6] if you add another lexical or signal-processing agent to the system, it changes what gets put on or taken off the blackboard, and affects the other agents. These too are agents.

With some information-retrieval "agents," which find Web documents for a user, if you add a second, it will do exactly what the first "agent" is doing, and the user will end up with two copies of each document. This is because the two reputed agents are completely unaware of each other, and cannot take advantage of each other's activities. According to our test, these are not agents.

A Filter, Not a Prescription

How an agent achieves its change in behavior, such as whether or not it communicates with the new agent, is not important. A test is not a prescription so much as a filter. And whereas definitions in the classical Socratic sense are expected to be necessary and sufficient conditions, a test may be one or the other. Also, a definition would typically identify the essence of its subject, whereas a test—such as the one we propose—has only to identify properties that can be observed and tested. In some sense, this test represents a performance criterion, which is both more powerful than a definition and a lot easier to manage.

Our agent test would pass the DVMT and distributed sensor net agents, pass the agents in WARREN,[7] fail mail daemons, fail spreadsheets (and other software programs that just act on behalf of a user, which is a common definition for an agent), and fail simple Java applets. The test is independent of the mobility of the agents.

Must one examine a system's interior workings to determine if it is agent based? Perhaps the creators of the system do not wish to give away their secrets and perhaps you don't want to know them. Our test tries to identify a

property that can be verified or disproved without knowing the details of the agents' construction.

Our test (this should come as no surprise to regular readers of this column) is based on the sociability of agents.

THE SOCIABILITY FACTOR

Assume we are given an environment and some purported agent—an executing program—that exists and functions in that environment. What would happen if more programs of the same type were introduced into the environment? Obviously there would be some change in the behavior of the program already there, regardless of whether the new program was an agent. For the program to pass our agent test, however, the changes must be appropriate, or agent-like, in the following way.

An appropriate change relates to agents at the knowledge level, not to the infrastructure. For example, as the number of agents inhabiting an environment increases, so do the chances that there will be resource conflicts among them. If the programs share a CPU or network, they may run slower; they may livelock or deadlock in trying to access the underlying files. These are not in themselves agent-specific interactions; they can also occur solely as a result of the environment. To be agents, they should behave as if they recognize when other agents are being introduced.

AUTOMATING THE TEST

More formally, we consider the state of an environment, which evolves as agents act and spontaneous events occur in the environment.

- The *state* of an information environment consists of the open files, visited Web sites, accessible Web sites, and so on. An agent's actions include locking files, or reading or writing to a database or index. Spontaneous events are typically negative, and include occurrences such as a file being locked or a CPU being overloaded because of some underlying tasks. An environment that spontaneously reduces disorder would be desirable but unrealistic!
- A *sequence* of environment states is a history of what has transpired—what the state was at a designated initial time, and how it has evolved through agent actions and environmental events.
- A *configuration* is an environment along with some set of programs executing in the environment.

Imagine we can define the entire set of sequences of environment states for a given configuration. This will tell us all that can happen in that configuration. Suppose, then, additional programs are added. Because the configuration has changed, a different set of sequences of environmental states will result.

Thus an automated test for agenthood could rely on the characteristic that adding more of the purported agents causes some change in the possible sequences. To make sure that the changes are not gratuitous, we must determine that the changes could not be caused solely by adjusting the environment.

When we add this refinement, the test becomes more robust. If the program design in question has anything to do with agency, its instances will be smart enough to recognize and interact with each other.

If they are not, they can still have some interference, of course. But interference is inherently a kind of interaction that goes through the infrastructure, and can occur readily in the environment without intervention by an agent.

There will be an agreed-upon definition for an agent someday. Better yet, there will be a prescription for one, which could be used to build an agent.

But until then, we'll continue to apply our agent test. ■

SYSTEM OF THE BIMONTH

This month we feature the work of Victor Lesser and colleagues on cooperative information gathering.[8] Their system highlights agents that interact with one another in locating and retrieving documents on the World Wide Web.

REFERENCES

1. R.J. Brachman, "I Lied about the Trees," *AI Magazine*, Vol. 6, No. 3, 1985.
2. C. Petrie, Jr., "Agent-Based Engineering, the Web, and Intelligence," *IEEE Expert*, Dec. 1996.
3. S. Franklin and A. Graesser, "Is it an Agent or Just a Program?: A Taxonomy for Autonomous Agents," *Intelligent Agents III: Proc. Int'l Workshop Agent Theories, Architectures, and Languages*, Springer-Verlag, 1997, pp. 21–35.
4. M.N. Huhns and M.P. Singh, "Agents on the Web: Agents Are Everywhere!" *IEEE Internet Computing*, Vol. 1, No. 1, 1997, p. 87.
5. V.R. Lesser and D.D. Corkill, "The Distributed Vehicle Monitoring Testbed: A Tool for Investigating Distributed Problem Solving Networks," *AI Magazine*, Vol. 4, No. 3, Fall 1983, pp. 15–33.
6. L. Erman et al., "The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty," *Computing Surveys*, Vol. 12, No. 2, June 1980, pp. 213–253.
7. "WARREN: Intelligent Agents for Financial Portfolio Management," http://www.cs.cmu.edu/~softagents/warren/.
8. K. Decker et al., "MACRON: An Architecture for Multi-agent Cooperative Information Gathering," *Proc. CIKM Workshop Intelligent Information Agents*, Dec., 1995. Available at http://dis.cs.umass.edu/research/cig.html.