NETWORKS

Software Agents Prepare to Sift the Riches of Cyberspace

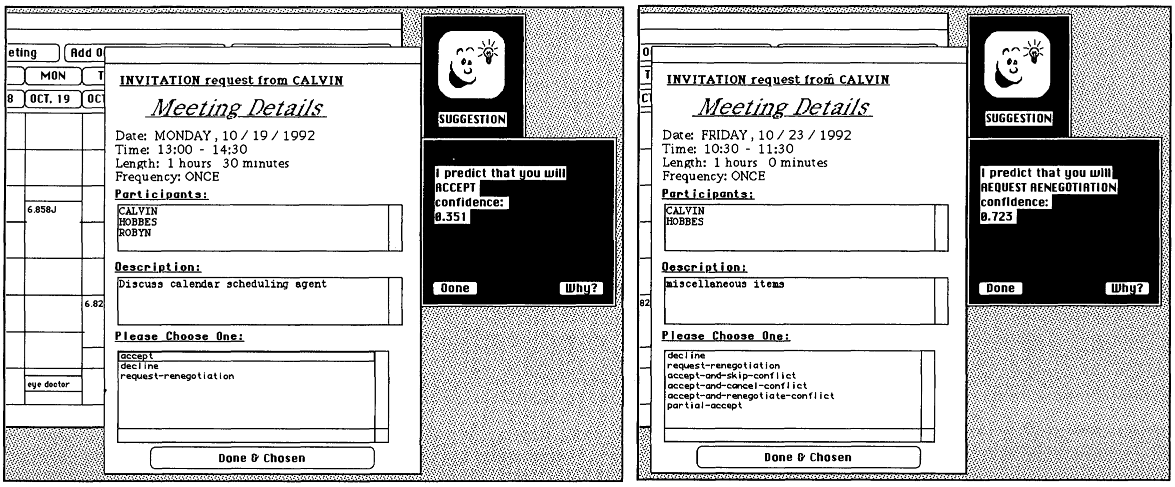
To the average Internet user, the power to go surfing through cyberspace at the click of a mouse seems magical enough. Information, graphics, on-line communities—it's all there for the asking. But veteran network engineer James White thinks that's just the beginning.

To White, now a vice president at the Silicon Valley start-up General Magic, it's time to stop thinking of the network as digital plumbing, suitable only for piping bits from one machine to another. Instead, he says, we should start thinking of the network as a digital universe with its own laws, its own structure—and ultimately even its own inhabitants. Those net-dwellers would be "agents," autonomous software robots that would be free to leave their home machine and fan out through cyberspace like productive versions of the sorcerer's apprentice.

Such agents might make airline reservations for you, schedule meetings, or search out information anywhere in the network. They could help scientists and lay users alike really *use* the network without drowning in an ocean of information. "After 25 years of working with networks," declares White, "I'm completely convinced that the mobile agent paradigm is the network of the future."

Of course, White's enthusiasm for the agent idea could be colored just a bit by the fact that he and General Magic have products to sell. This past January, with backing from AT&T, Apple, Matsushita, Motorola, Philips, and Sony, the firm first presented Magic Cap, a new graphical interface designed to help users interact with a network full of agents, and Telescript, a language devised by White to help developers and users create mobile agents of their own.

Nonetheless, White's enthusiasm for software agents does seem to be widely shared. General Magic may be mounting the best funded, best publicized effort to bring the concept to market, but it has plenty of company. Sun Microsystems, for example, recently hired John Ousterhout to develop his TCL scripting language, which he created in 1988 as a research project at the University of California, Berkeley, and which is similar in spirit to White's Telescript. And on the research side, the quest to build truly flexible and adaptable agents has proved an enticing challenge for artificial intelligence (AI) researchers and students of human-computer interaction alike. The July 1994 issue of *Communications of the Association for Computing Machinery*, a leading computer-science professional journal, had 15 articles devoted to the topic of "intelligent agents." And dozens of laboratories are now experimenting with prototype agents.



INVITATION request from CALVIN

Meeting Details

Date: MONDAY, 10 / 19 / 1992
Time: 13:00 - 14:30
Length: 1 hours 30 minutes
Frequency: ONCE

Participants:
CALVIN
HOBBES
ROBYN

Description:
Discuss calendar scheduling agent

Please Choose One:
accept
decline
request-renegotiation

I predict that you will
ACCEPT
confidence:
0.351

INVITATION request from CALVIN

Meeting Details

Date: FRIDAY, 10 / 23 / 1992
Time: 10:30 - 11:30
Length: 1 hours 0 minutes
Frequency: ONCE

Participants:
CALVIN
HOBBES

Description:
miscellaneous items

Please Choose One:
decline
request-renegotiation
accept-and-skip-conflict
accept-and-cancel-conflict
accept-and-renegotiate-conflict
partial-accept

I predict that you will
REQUEST RENEGOTIATION
confidence:
0.723

PATTIE MAES/MIT MEDIA LAB

At your service. A software agent filters meeting requests based on its knowledge of the user's preferences.

Indeed, people have been talking about agents since the late 1950s, when artificial intelligence researchers John McCarthy and Oliver Selfridge first used the word and sketched out the concept. An "agent," to take the broadest possible definition, is a program that can operate autonomously, carrying out tasks without direct human supervision. In one sense, such autonomy is already ubiquitous in software: Think of a communications program that automatically dials a remote bulletin board late at night and downloads a large file when long-distance charges are lowest.

In another sense, however, agents represent a fundamental change in human-computer interaction. In the standard "desktop" metaphor popularized by the Apple Macintosh and Microsoft's Windows program, the computer user is essentially a worker: He or she has to open each window, click on each icon, and browse through each menu personally. But as pointed out by Apple research fellow Alan Kay, who spearheaded the development of the desktop metaphor in the 1970s, the agent approach transforms the computer user from a worker into a manager: You delegate tasks to a set of agents who do things for you.

Help wanted. For example, a scientist who gets dozens or even hundreds of e-mail messages a day currently has to scan them all to find the ones that are worthwhile. Why not have an e-mail agent that could monitor what comes in, delete the junk, and put the urgent messages at the top of the pile? Meanwhile, all too many of us have spent hours searching through the vast reaches of the Internet for a certain piece of information. Why not have a mobile agent possessed of sufficient intelligence to take a general query and do the searching on its own?

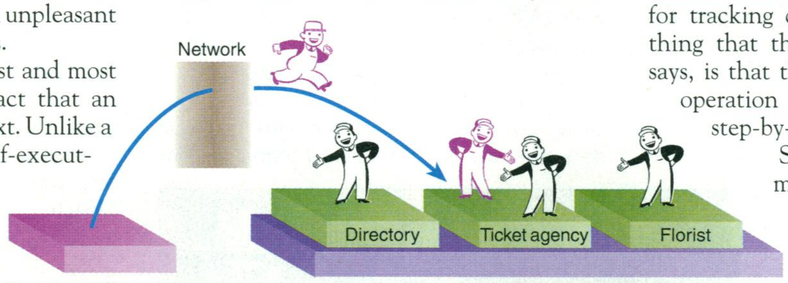
What's made this notion increasingly compelling is the convergence of necessity—the explosion of on-line information—with possibility. A proliferation of increasingly high-powered personal computers now allows for much more sophisticated, adaptable agents. And a proliferation of networks has opened the way for mobile agents that are no longer bound to any single computer.

It's the issue of mobility that is the key challenge for General Magic's White. Agent mobility isn't all that hard to achieve at a technical level, he explains. To create a mobile agent with Telescript, you just type out commands in much the same way as you might type out a program in BASIC. When you're done, you electronically feed the agent text into the Telescript "engine," a program that reads the written commands and carries them out step by step. To make the agent migrate through the network, says White, you simply include a line in the text with the command "Go." When the engine reaches that command, it dissolves the agent text into data packets and sends them out into the network as though they comprised a perfectly ordinary electronic message—which, in fact, they do. Then, once the text is reassembled in the target computer, a

Telescript engine in the new environment begins executing the agent's commands precisely where they left off.

Since movement through the network can be achieved so easily, says White, the real challenge isn't mobility per se, but trust: the fact that your agents can suddenly show up on my computer. "It's the overriding technical issue in any agent architecture," he says. "If you build a system where agents can move, what's to keep them from entering my machine and wreaking havoc?" Without safeguards, in fact, agents bear an unpleasant resemblance to computer viruses.

In Telescript, he says, the first and most fundamental safeguard is the fact that an agent is actually just a piece of text. Unlike a virus, which is made of active, self-executing code, the agent is nothing without the Telescript engine. Only the engine can pass the agent's requests to the host computer's hardware, says White. And, if need be, the engine can censor those requests. In effect, the engine serves as the digital equivalent of biological P4 containment.



Going downtown. A mobile agent leaves its home machine, traverses the network, and interacts with agents that reside on other machines.

SOURCE: GENERAL MAGIC

A second safeguard is the fact that every Telescript agent has to carry "credentials": an encrypted digital identification tag stating where it came from and to whom it belongs. A third is that each agent carries a set of "permits" spelling out what it can and cannot be allowed to do on the network, how long it will live, and how many resources it is permitted to consume. Finally, as Telescript is designed, an agent can't just come in and start rummaging around willy-nilly, says White. The only way it can make purchases or request data or computations is by interacting with another agent, resident on the host computer.

Computer users will have a chance to evaluate General Magic's agents later this year, when AT&T plans to launch the first large-scale implementation of a Telescript network. At first, says White, this network will be targeted toward handheld personal communicators and applications such as electronic mail, using agents to deliver it, filter it, and prioritize it. But ultimately, he says, "I'd love to see Telescript operating on the Internet"—and it's only a matter of time.

In the research community, veteran agent investigators generally give Telescript high marks for innovation and security. "General Magic hasn't released much about Telescript," says Sun Microsystems' Ousterhout, author of the competitor TCL language. "But my perception is that they've done a very, very good job of making agents very, very safe."

Social agents. Still, General Magic's agents are relatively simple compared to the intelligent software robots some researchers envision. Take agents that can learn. At the Massachusetts Institute of Technology's Media Lab, AI researcher Pattie Maes is studying how the ability to learn can not only help agents become more competent at their assigned tasks; it can also make their masters more comfortable about trusting them.

In principle, explains Maes, this shift from being a worker to being a manager of agents is supposed to make your life easier. But just as in real life, becoming a manager brings a whole new set of worries: Do you feel comfortable delegating tasks to a bunch of software robots? Do they really know what they're doing? "In some applications that's not such a big deal," says Maes. "But if your agent is handling your stocks or your disk files, it does matter."

And that's the virtue of learning, she argues: You don't have to worry about preprogrammed agents doing things you don't understand—or about programming the agent yourself and possibly making mistakes. "These are agents that look over your shoulder and notice patterns and regularities in your actions," she says. An e-mail agent, for example, might notice that you tend to read all messages from Person A immediately, while you tend to delete anything sent by Person B without looking at it. So over time, the agent will begin to organize your e-mail inbox with Person A at the top and with Person B marked for deletion.

"It doesn't require much knowledge on the part of the agent," says Maes. "We mostly use statistical techniques. On the other hand, you discover that people don't always realize how they actually act. For example, you may really be reading more gossip than hard, worthwhile news. So you have to be able to tell the agent, 'Don't pay attention to what I'm about to do!' Otherwise they're like children: They learn your bad habits as well as your good ones."

While Maes is studying the issue of agent-human interaction, University of Massachusetts AI researcher Victor Lesser is looking at an equally fundamental question: How should agents interact with each other? Ultimately, when swarms of agents are moving through the Internet, he explains, you will also have tremendous potential for digital teamwork. In effect, the agents will have the opportunity to say things to each other like, "Let's divvy up this job and get it done in parallel," or "Here's some information I found that you might be able to use."

"But you have to ask how this [agent interaction] will change the way you think about computation," says Lesser. Over the past two decades, he explains, he and his colleagues have studied how widely dispersed agents could cooperate to run, say, an air traffic control system or an airport system for tracking down lost baggage. And one thing that they've consistently found, he says, is that the real-world demands of cooperation strain the limits of standard, step-by-step algorithms.

Suppose you're a software agent monitoring an air traffic radar in one sector, for example, and the software agent in a neighboring sector tells you —mistakenly—that an airplane should be showing up on your screen any minute. It's been fooled by static. "That can send you off into a dead end, where it takes a long time to recover," says Lesser. You end up wasting a huge amount of your computational energy trying to interpret your own static as an airplane.

Within the confines of a single computer, says Lesser, designers go to great lengths to engineer such uncertainties out of existence, so that algorithms always have precisely known inputs and outputs. But out in the real world, says Lesser—or for that matter, out in the network—nobody has that kind of control. "You need processes that accommodate inconsistency, that negotiate with each other, that can come to some acceptable resolution in the face of uncertainty," Lesser argues.

To achieve these goals, which is really a problem in artificial intelligence, Lesser and others in the AI community have spent the past two decades developing methods for solving problems incrementally. An agent (or a team of agents) might keep several partial, tentative solutions around at all times. Think of a group of physicians trying to diagnose a mysterious illness and considering several possibilities, he says. The solutions can be mutually inconsistent. But as each new piece of information comes in, it will increase or decrease the likelihood of one or another solution until a winner emerges.

Of course, says Lesser, it's likely to be a long time before agents this sophisticated show up on your local computer screen. But that, he says, is precisely why he finds the appearance of commercial tools such as Telescript so exciting. With the tools in people's hands, he says, "a lot of the things we've been thinking about for the past 15 years could finally become real."

–M. Mitchell Waldrop