University of South Carolina
Scholar Commons

Faculty Publications | Computer Science and Engineering, Department of

2000

Sensors + Agents + Networks = Aware Agents

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

Sreenath Seshadri

Follow this and additional works at: https://scholarcommons.sc.edu/csce_facpub

 Part of the Computer Engineering Commons

Publication Info

Published in *IEEE Internet Computing*, Volume 4, Issue 3, 2000, pages 84-86.
http://ieeexplore.ieee.org/servlet/opac?punumber=4236
© 2000 by the Institute of Electrical and Electronics Engineers (IEEE)

This Article is brought to you by the Computer Science and Engineering, Department of at Scholar Commons. It has been accepted for inclusion in Faculty Publications by an authorized administrator of Scholar Commons. For more information, please contact digres@mailbox.sc.edu.





Sensors + Agents + Networks = Aware Agents

Michael N. Huhns • *University of South Carolina* • *huhns@sc.edu*
Sreenath Seshadri • *University of South Carolina* • *sree@sc.edu*

Software agents are being deployed in increasing numbers to help users find and manage information, particularly in open environments such as the Internet. For the most part, they operate independently and are typically designed to be aware only of their users and the environment in which they perform their tasks. Thus, they fail to take advantage of each other's abilities or results.

For example, a shopping agent might periodically access several online databases to find the best price for a music CD and then purchase it if the price falls below its user's threshold. Other agents might be tracking prices for the same CD, duplicating each other's work. Similarly, if your agent and an agent for the person in the next cubicle are both browsing the same Web site, two identical data streams arrive on your LAN, using twice the bandwidth actually needed.

To be more effective, agents must be aware of each other; therefore, they must acquire models of each other. One way to do this is by exchanging messages. ("Hi, agent 25 knows about CD prices.") A second form of awareness involves the state of the agent's own environment, including characteristics of the computer on which it is executing and its network connection. ("How many bytes can I send in one second?") A third involves self-awareness—knowing its name, age, ontology, goals, areas of expertise and ignorance, and reasoning abilities. Finally, the agent should be aware of its physical environment.

A Physical Agent Scenario

Let's consider a vacuum-cleaning agent in a room containing some furniture and assume at first that the agent has no sensing or modeling capabilities. Thus, its best strategy is to wander continuously throughout the room, sucking up dirt wherever it finds it. To an observer the agent would appear to be very single-minded, pursuing just one goal. It might occasionally crash into the furniture or the walls, but if its wandering were truly random, it would not get stuck and would eventually clean all accessible parts of the room. Because it might clean some areas more than others, it would not be efficient or optimum.

Now let's begin increasing the agent's capabilities and see what ensues. First, we add a sensor (possibly ultrasonic) so that the agent can tell when it is about to bump into something. This will save wear and tear on the agent and the furniture, but it won't make the agent any better at cleaning up dirt.

Next, we add the ability to construct a model of the room based on the agent's encounters with walls, furniture, and dirt. As it wanders, it refines the model until it includes all the accessible areas. This won't help the agent in any way, however, unless we also add an ability to *use* the model for planning its movements. With such a planning ability, the agent can figure out how to clean the room methodically and can potentially minimize the time needed for the task.

At this point, the agent is maximally effective: adding more sensors or planning abilities will not improve its cleaning performance (although better sensing and planning might enable the agent to construct a model or plan more quickly). Now let's introduce a second mobile agent into the room, as shown in Figure 1, without telling the first agent about it.

Effect on the Environment Model

The second agent will affect both the cleaning agent's model of the room and the execution of its plan. Wherever the cleaning agent encounters the mobile agent, it will incorporate an obstacle at that location into its model. Depending on whether the model is deterministic or nondeterministic, it will have either inaccurate or uncertain elements, owing to the presence of the second agent. In either case, the agent can improve its model by exploring its environment and using what it discovers to adjust the probabilities or verify the locations of obstacles. However, as long as the model is not perfect—incorporating only fixed obstacles—the agent will miss cleaning part of the room.

Effect on Plan Execution

As before, using its model of the room, the cleaning agent can con-

struct a plan for cleaning the room and devise a strategy that is methodical and minimizes the time required. However, as it follows this strategy, it will occasionally and unexpectedly encounter the mobile agent, causing it to deviate from its plan and thus from optimality.

How can the cleaning agent improve its performance? Besides spending even more time exploring its environment, it can construct a model of the mobile agent and then incorporate this model into its plans.[1]

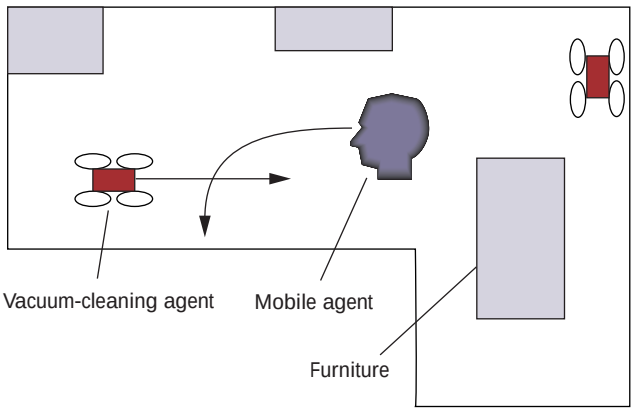


Figure 1. The vacuum-cleaning agent and its working environment.

System of the Bimonth



Taalee (http://www.taalee.com) uses an interesting system of information agents to automatically extract information from a variety of sources and produce the components needed for a corporate Web portal. The agents both use and extend an ontology during their interactions and enable the portal to support third-generation semantic-based search.

Check it out!

Modeling Other Agents

A model of an agent can have many possible forms and amounts of detail, and it can be used in many ways—for example, to plan a route that minimizes encounters with the other agent or to decide on a course of action once the other agent is actually encountered. In the latter case, if the second agent is modeled as being slow, the cleaning agent can plan a path around it, confident that it cannot interfere. If the second agent is thought to be fast, the cleaning agent can simply resume its current plan after a short pause that allows the other agent to move on.

These example strategies are appropriate if the other agent is treated as merely an obstacle in the room. But what if the other agent is discovered to be the source of the dirt? Then the cleaning agent could adopt a strategy of simply following the mobile agent everywhere and cleaning up after it—there would be no need to clean anywhere else—and the cleaning agent would again be maximally effective.

What if the other agent rearranges the furniture? Knowing when or how this happens can help the cleaning agent decide when and how to reexplore or replan.

What if more agents enter the room—too many for the cleaning agent to model during planning and too many to account for during cleaning? The cleaning agent might discover and then act in accordance with *social conventions*, learning, perhaps, that "agents always pass each other on the right." Another possibility, if the other agents are organized, is for the cleaning agent to learn their organization. This could simplify its modeling task because it would need to model only a subset of the agents—those with which it interacts and those that are representative of a particular agent role. Moreover, it would know a priori something about the agents on the basis of their role in the organization. For example, it would expect subordinate agents to try to satisfy its requests and commands, and it would expect to receive commands from managers or superiors.

Representations of Other Agents

How should one agent represent another, and how should it acquire the information it needs to construct a model in that representation? This has, I believe, a simple and elegant answer: *The agent should presume that unknown agents are like itself, and it should choose to represent them as it does itself.* As it learns more about them, it has only to encode any differences it discovers. This can make the resultant representation concise and efficient. Here are some other advantages:

- An agent has a head start in constructing a model for an unknown or just-encountered agent.
- An agent has to manage only one kind of model and one kind of representation.
- The same inference mechanisms used to reason about its own behavior can reason about the behavior of other agents. An agent trying to predict what another will do has only to imagine what it would do itself in a similar situation.

One ramification of such representation is that an agent constructed with a belief-desire-intention (BDI) architecture would attribute intentions to other agents even if they lacked a BDI architecture or any explicit intentions at all.

ANTS

An agent's awareness of its physical environment is the domain of the

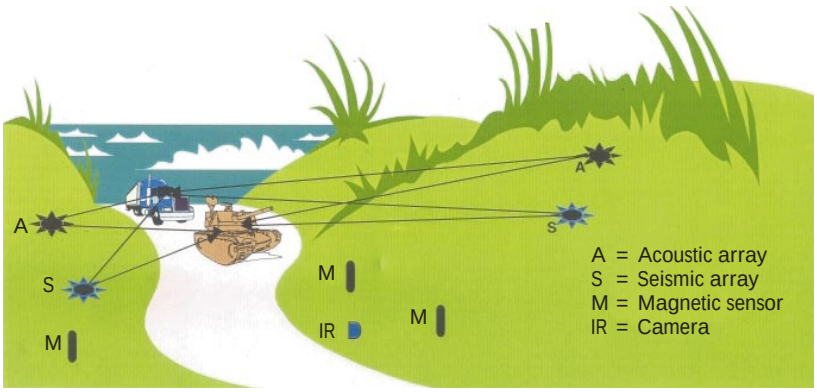


Figure 2. Sensors coordinating to detect vehicles.

Autonomous Networked Tactical Sentries (ANTS) project at Raytheon TI Systems and the University of South Carolina.[2] The project's purpose is to combine miniature sensors and software agents. The sensors can listen, sniff, see in the infrared and visible spectrums, and feel seismic vibrations. The resultant sensors and agents will be deployed on a battlefield to detect the position and movement of enemy forces with minimal risk to friendly forces. Detection results would be communicated to appropriate personnel, who would make strategic and tactical battlefield decisions.

The project uses Java as the high-level development language for signal processing, command, and control, with agent behaviors encoded as Java threads. Figure 2 visualizes an instance of these networked sensors at work.

How Does Agent Awareness Relate to ANTS?

Time-critical battlefield decisions require conclusive evidence about targets, which can be provided only by a variety of sensors. Every sensor has a software agent that gathers data about happenings in the physical environment. It interprets the data and sends strategic information to a controller. If the agents know not only about their surroundings but also about themselves and other agents, entire system effectiveness multiplies.

Transforming agent-gathered data into useful information requires awareness beyond one's current environment or state. Awareness in ANTS agents involves, first, awareness of self. This means knowing, for example, how much computing power it has left, how long it would take to complete a task, and what resources are needed to do so. Second, it involves awareness of other agents so that it can answer questions such as

- Where is the nearest agent that could provide additional information about the target vehicle?
- Which agent can garner details about the target that I am missing?
- Which agent has enough battery power left to transmit what I have discovered to the controller?

Agents that know their own capabilities and those of their peers stand a better chance of acquiring additional details about the target than unaware agents. Studies indicate that people who are more successful have faster networks of more capable experts,[3] so we shouldn't be surprised that the principle also applies to agents.

What else can agents do by knowing the capabilities of their neighbors? Well, given a target's characteristics, they can focus their collective efforts on locating it. Aware agents can federate and track down specific targets, just as certain carnivores benefit from pack hunting. As part of a federation, an agent can obtain information about a target even though the target may be well out of its sensing range. Thus, a sensor's capabilities are extended by the presence of software agents that can socialize with other agents.

Resource Management through Awareness

Since these networked sensors are to be deployed in battlefields, battery power is a scarce resource to be used with discretion. Here again, agents have an advantage. Since they know each other's physical location, they can predict the path of a vehicle from the information they have and then alert other agents of the vehicle's speed and coordinates. For example, a seismic sensor could pick up vibrations, calculate the coordinates of the source, predict the probable path of the object, and notify an infrared camera along the path. This would help the camera compute when it should turn itself on, saving valuable battery power. In another scenario, two agents at a specific location running low on battery power might take turns performing their duties.

Whether providing critical tactical information for battlefield decision-making or vacuuming our floors without endangering our furniture (or each other), agents with awareness of self and others hold the key to accomplishing a wide range of tasks in our increasingly complex world. ■

REFERENCES

1. M. Tambe and P.S. Rosenbloom, "RESC: An Approach for Real-Time, Dynamic Agent Tracking," *Proc. Int'l Joint Conf. Artificial Intelligence*, Morgan Kaufmann, San Francisco, 1995, pp. 103-110.
2. M.N. Huhns, "Networking Embedded Agents," *IEEE Internet Computing*, vol. 3, no. 1, Jan./Feb. 1999, pp. 91-93.
3. R.E. Kelley, "How To Be a Star Engineer," *IEEE Spectrum*, Oct. 1999, pp. 51-58.

Michael N. Huhns is a professor of computer science and engineering at the University of South Carolina, where he also directs the Center for Information Technology.

Sreenath Seshadri is a graduate student of computer science and engineering at the University of South Carolina, where he is conducting research in embedded and distributed computing.