

**Business Technology  
Trends & Impacts**

**Vol. 8, No. 1**

**CUTTER  
CONSORTIUM**

# Agents: A Necessary Ingredient in Today's Highly Collaborative World

by James Odell, Senior Consultant, Cutter Consortium

Agent technology is now necessary to reduce costs; to improve efficiency and effectiveness; and to support the requirements of individuals, groups, companies, and universities as they collaborate globally. More importantly, it will enable us to create and support a whole class of IT applications and approaches that we previously could not have developed.

CONSORTIUM

Opinion

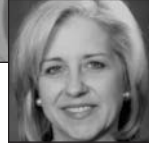
# Cutter Business Technology Council



Rob Austin



Tom DeMarco



Christine Davis



Lynne Ellyn



Jim Highsmith



Tim Lister



Ken Orr



Lou Mazzucchelli



Ed Yourdon



## About Cutter Consortium

Cutter Consortium is a unique IT advisory firm, comprising a group of more than 150 internationally recognized experts who have come together to offer content, consulting, and training to our clients. These experts are committed to delivering top-level, critical, and objective advice. They have done, and are doing, groundbreaking work in organizations worldwide, helping companies deal with issues in the core areas of software development and agile project management, enterprise architecture, business technology trends and strategies, enterprise risk management, business intelligence, metrics, and sourcing.

Cutter delivers what no other IT research firm can: We give you Access to the Experts. You get practitioners' points of view, derived from hands-on experience with the same critical issues you are facing, not the perspective of a desk-bound analyst who can only make predictions and observations on what's happening in the marketplace. With Cutter Consortium, you get the best practices and lessons learned from the world's leading experts, experts who are implementing these techniques at companies like yours right now.

Cutter's clients are able to tap into its expertise in a variety of formats including print and online advisory services and journals, mentoring, workshops, training, and consulting. And by customizing our information products and training/consulting services, you get the solutions you need, while staying within your budget.

Cutter Consortium's philosophy is that there is no single right solution for all enterprises, or all departments within one enterprise, or even all projects within a department. Cutter believes that the complexity of the business technology issues confronting corporations today demands multiple detailed perspectives from which a company can view its opportunities and risks in order to make the right strategic and tactical decisions. The simplistic pronouncements other analyst firms make do not take into account the unique situation of each organization. This is another reason to present the several sides to each issue: to enable clients to determine the course of action that best fits their unique situation.

**For more information, contact Cutter Consortium at +1 781 648 8700 or [sales@cutter.com](mailto:sales@cutter.com).**

# Agents: A Necessary Ingredient in Today's Highly Collaborative World

## BUSINESS TECHNOLOGY TRENDS & IMPACTS ADVISORY SERVICE

Council Opinion, Vol. 8, No. 1

---

### DOMAIN:

IT technology

### ASSERTION 157:

**Agent technology is now necessary to reduce costs; to improve efficiency and effectiveness; and to support the requirements of individuals, groups, companies, and universities as they collaborate globally. More importantly, it will enable us to create and support a whole class of IT applications and approaches that we previously could not have developed.**

### SYLLABUS

We are no longer in the era of mainframe computing, when both companies and applications were typically command-and-control-oriented and organized in vertical silos. With the combination of the Internet, fiber optics, and PCs, the business and technology playing field has been flattened. No longer primarily top-down, it has changed more to side-by-side — as individuals, small groups, and organizations interact around the world. As a result, organizations are now demanding and cultivating new business practices that encourage less command and control and more horizontal connecting, collaborating, and competing. Agent technology is a primary enabler to support this new era. In fact, *without* agent technology, our current technology will not scale to support the ever-increasing global interaction.



### OPINION BY JAMES ODELL

By 2000, people realized that:

... they were in touch with people who they'd never been in touch with before, were being challenged by people who have never challenged them before, were competing with people with whom they had never competed before, were collaborating with people with whom they had never collaborated before, and were doing things as individuals they had never dreamt of doing before. [1]

---

This is what Pulitzer Prize–winner Thomas Friedman refers to as the “flat-world platform”; this phenomenon is causing, enabling, and empowering small groups and individuals to collaborate and compete globally.

As a result, we are moving from a vertical chain of command to a more horizontal chain of collaboration in both commercial and noncommercial activities. However, we need to adjust to make the transition from vertical to horizontal thinking. Vertical thinking typically involves asking who or what commands and controls the system; horizontal involves a more peer-to-peer way of thinking.

### Global Collaboration Requires New IT Approaches — Using Agents

This talk of “tearing down the walls and silos” is not new, and it has been a hot topic particularly in recent years. So what is different now? Increasingly, we find that we must pull together services from other organizations to form a collaborative team for just one consumer. This solution will create value for the consumer that does not exist in any one application or service or from any one organization. The underlying effort must support fine-grained, individualized resources and be able to address the personal level as well as organizational. For this to scale adequately, it must be automated. How can this be accomplished?

One such approach is suggested in both the W3C Web Services Architecture [3] and the OASIS Service Oriented Architecture (SOA) Reference Model.<sup>1</sup> Here, a service provides some functionality on behalf of its owner, which is a person or an organization. The *provider* entity is the person or organization that supplies an appropriate agent to implement a particular service. A *requester* entity is a person or organization that wishes to make use of a provider entity’s service. It will use a requester agent to exchange messages with the provider entity’s provider agent. Figure 1 illustrates this further. In step 3, the service description and semantics are realized by the requester and provider agents; and in step 4, the requester and provider agents exchange messages, resulting in some task being performed on behalf of the requester by one or more provider entities. This reduces the information and processing overload on human operators.

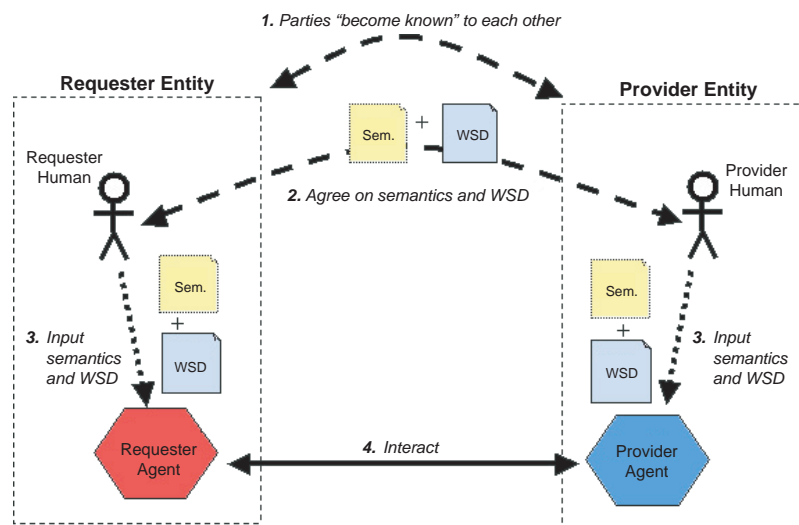


Figure 1 — Service architectures enhanced using agents for interaction. (Source: [3].)

<sup>1</sup>For more information, see OASIS SOA Reference Model ([www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=soa-rm](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm)).

## WHAT IS AN AGENT?

Imagine sitting in the park on a nice summer day, and a flock of birds sweeps the sky. One moment they are 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. Each bird follows a simple set of rules that it uses to react to birds nearby. When Craig Reynolds of DreamWorks developed his simulation,<sup>1</sup> each bird behaved basically according to the following rules:

- Maintain a minimum distance from other objects, including other birds.
- Be sociable (i.e., try to match velocities with other birds if they are nearby, and move toward the perceived center of their group).

The flock is organized without an organizer, coordinated without a coordinator. 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 a centralized authority. For IT applications, this can include order processing, supply chain, shop-floor control, inventory management, message routing, multiple database management, operating systems, and self-healing middleware. In other words, a decentralized approach is particularly suitable for agents and should be considered where local components have some degree of control (instead of limiting your approach solely to the centrally organized one traditionally employed by IT). After all, if New York City can maintain a two-week supply of food with only locally made decisions, why can't a supply chain system perform in a similar manner? Agents *can* be designed to behave in a hierarchical manner. However, the strength of agents is that they can be used in both hierarchical and collaborative ways — whereas conventional approaches lack such flexibility and scalability.

Conventional objects can be thought of as passive, because they wait for a message before performing an operation. Once invoked, they execute their method and go back to “sleep” until the next message. A current trend in many systems is to design objects that both react to events in their environment and are proactive. In UML 2.0, these are known as active objects; in the agent community, they are known as agents. Whether they are called active objects or agents, this new direction is going to radically change how we design systems. It is a way of providing very fine-grained distributed processing, where each processing unit, or agent, can interact with others. Here, entire “social” systems, based on these autonomous and interactive processing units, are possible.

The basic dictionary definition of agent is “something that acts — sometimes on behalf of another.” However, for developing business and IT systems, such a definition is too general. While an industry-standard definition of agent has not yet emerged, a basic working definition is “an autonomous entity that can adapt to and interact with its environment.”

Agents, then, can be software agents, hardware agents, firmware agents, robotic agents, human agents, and so on. While software developers naturally think of IT systems as being constructed of only software agents, a combination of agent mechanisms might in fact be used from shop-floor manufacturing to warfare systems. That is why the definition is so general.

Whether or not you restrict your view of agents to the software variety, most agree that agents deployed for IT systems are not useful without the following three important properties:

1. **Autonomy.** The agent is capable acting without direct external intervention. Agents have some degree of control over their internal state and can act based on their own experiences. They can also possess their own set of internal responsibilities and processing that enable them to act without any external choreography. When an agent acts on behalf of (or as a proxy for) some person or thing, its autonomy is expected to embody the goals and policies of the entity that it represents.
2. **Interaction.** The agent communicates with the environment and other agents. Agents are interactive entities because they are capable of exchanging rich forms of messages with other entities in their environment. These messages can support requests for services and other kinds of resources, as well as event detection and notification. They can be synchronous or asynchronous. The interaction can also be conversational in nature, such as negotiating contracts, marketplace-style bidding, or simply making a query.
3. **Adaptivity.** The agent is capable of responding to other agents and/or its environment. Agents can react to messages and events and then respond appropriately. Agents can be designed to make difficult decisions and even modify their behavior based on their experiences. They can learn and evolve.

<sup>1</sup>See [www.red3d.com/cwr/boids](http://www.red3d.com/cwr/boids).

When the links between requesters and providers are few and change is not an issue, agents are not needed and conventional IT techniques can be employed. However, when the links require complex, dynamic binding and are subject to rapid change, agent-based approaches should be considered. Prior to the demand for global collaboration, we developed a centralized controller to ensure that all interactions would be appropriately managed. Now this is no longer possible. The world is too big, and a central bottleneck would paralyze the effort. Furthermore, the access techniques now require a more horizontal style of interaction rather than one that is centrally administered.

Both collaboration and competition are supported by an agent-based world. Among the techniques using agents are auctions, which link a requester's service requirement to a rich world of service providers (e.g., British auction, Dutch auction, eBay auction, or contract net). In the contract net protocol, for example, requester agents send out requests for quotes (RFQs) and award the winning provider(s) with a "contract" to provide a resource. For instance, an energy company like DTE has a finite number of resources (e.g., vehicles and crew members) to handle all the energy-related services that are required for a power outage system. Under normal conditions, such planning and scheduling could be handled adequately by human schedulers and dispatchers with automated support. However, an ice storm knocking out the power at 100,000 locations will tax even the best human schedulers. Here, agents can be employed to *automate* the process by bidding on behalf of the various resources, including those from power companies in neighboring states and countries.

Cape Canaveral is now installing a similar system for its launch control services and resources using an agent-based approach developed by Intelligent Automation, Inc. (IAI).<sup>2</sup> NASA is investing using this technique for planning and scheduling sensor-based services and resources. These systems would use agents to identify, assemble, and execute the services that would form a virtual service for the end consumer. Other agents plan and schedule those resources required.

Managing services and resources in this manner enables and promotes global interaction and collaboration. At the heart of all these systems, agents digitize and decompose the supply chain and move the work around to anywhere in the world.

#### ***For Example, How Can Agents Enable SOA to Work in a Complex World?***

Agent technology is an appropriate supplement to many technologies. Service-oriented architecture is just one. SOAs have the flexibility and responsiveness to enable organizational priorities to finally drive technology decisions. Many companies are also expanding SOA beyond IT resources to include hardware (such as sensors, electronic equipment, and robotics, as well as human resources). By adding agent-based functionality to the SOA, even greater flexibility and agility are possible [2]. In particular, agent technology can be used to enable the following features:

- **Resource provision and request.** In both the W3C and OASIS approaches to Web services architecture, the actual service requester and provider entities will agree on the service description and semantics that will govern the interaction between the requester and provider agents. The agents will then automate performing services on behalf of the actual requesters and providers. The agents can choose the smartest and most efficient way to accomplish a task by connecting different nodes in a network — whether horizontally or vertically.

---

<sup>2</sup>For more information, see IAI ([www.i-a-i.com/view.asp?tid=21](http://www.i-a-i.com/view.asp?tid=21)).

- **Processing support.** For complex processes, agents can be used to orchestrate and “team” software processes to solve problems collaboratively or compete intelligently. Services result from self-assembly. The provider agents determine the value proposition for the consumer agents and then identify and assemble the services that will form a virtual service for the end consumer.
- **Resource discovery.** Dynamic service selection is increasingly common as organizations recognize the benefits of its flexibility. If an agent has flexibility in picking its business partners, it can select them to optimize any kind of quality-of-service (QoS) criteria, including performance, availability, reliability, and trustworthiness. A requester agent can be chartered to do this, but specialized service-discovery agents are sometimes useful for facilitating this.
- **Middleware support.** Within SOA middleware, agents should be used for functions that require flexibility, autonomy, and robustness. For example, managing application-level fault tolerance, security, performance, and QoS are common uses for agents. In addition, agents can be used as a more “intelligent” way of handling and propagating changes in ontology and dealing with incompatibilities in vocabularies, semantics, and pragmatics among service providers, brokers, and requesters.

### ***Making Things Possible with Agents — Not Just Faster, Better, and Cheaper***

Using agents has produced applications faster and cheaper than other approaches. For example, DHL found that its pricing and tracking applications could be developed 50%-80% faster using agent development tools like those available from OSLO<sup>3</sup> or Agentis.<sup>4</sup> Time to market was reduced and ROI increased. This doesn't mean that agent development tools will always reduce function points, but it does for some classes of application.

The most compelling case for using agent technology is when agents can enable solutions that were either impossible or very difficult using conventional mechanisms. At manufacturing companies like Deere & Company, several hours each night are typically dedicated to computing optimized production schedules for the shop floor's use the next day. When an organization has the time to use techniques like these, agent technology is probably unnecessary. However, what if John Deere's IT department delivered its shop-floor schedule at the beginning of a day when several key employees were out sick or a vital piece of machinery in the assembly line broke down? If they needed to stop production for several hours to recreate a new schedule, this solution clearly is not the answer. Under these circumstances, an agent-based approach should be considered. Instead of starting over to reoptimize the schedule for all the resources, agents can be used to produce a *local* optimum. If one resource is no longer available, agents can look for a replacement and “contract” with alternative resources. If the resources are going to be delayed, agents in a supply chain can determine which way to solve the delay with the least impact — using automated speeds.

Even techniques such as linear programming, neural networks, and genetic algorithms are sometimes used for similar situations. Again, when these provide accurate results in a timely matter, agents are not necessary. However, for large complex systems, such approaches can sometimes become costly to configure or reconfigure and involve scalability problems. In these situations, they are often run offline and do not respond rapidly to dynamic changes in environment. When this is the case, the horizontal approach of agent technology can significantly outperform a centralized scheduling approach. Companies like IAI have developed

---

<sup>3</sup>For more information, see OSLO Software ([www.oslo-software.com](http://www.oslo-software.com)).

<sup>4</sup>For more information, see Agentis Software ([www.agentissoftware.com](http://www.agentissoftware.com)).

systems involving over 100,000 agents that can handle the resource rescheduling in a manner of *seconds* — instead of hours. General-purpose agent platforms (like IAI's) can be used, or a more specialized package can be chosen. For example, agent-based logistics software is available from Cougaar<sup>5</sup> and Whitestein<sup>6</sup> and event-driven sense-and-respond solutions from Rhysome.<sup>7</sup> Many other large software companies already employ agents inside their offerings, but do not call out the technology by name. Agent technology makes it possible to do things that previously either were impossible or could not be accomplished in an acceptable time frame.

### **Where to Go from Here**

Adopting any new technology is disruptive, and agent technology is no exception. Early-adopter companies are currently choosing agents for one or more of the following benefits:

- Faster return on investment (ROI)
- Lower maintenance
- Higher productivity
- Leverage of existing infrastructure
- Reuse of processes and services
- Foundation for future projects
- Reduction of time to market
- Increased agility to respond to business needs

Is an agent-based approach useful for every application and usage? Does it always provide the benefits listed above? Certainly not. If your business is predictable and stable, and your processes are centralized and scalable for the foreseeable future, adopting an agent-based approach is not necessary. However, for those applications that must support complexity and change in a scalable and timely manner, agents will likely be a necessary technology. The typical organization probably has both of these situations in one form or another, and the savvy organization will have a mixture of technologies: object-oriented (OO), relational, *and* agent-based. OO and relational technologies enable a top-down and centralized solution to business applications. Agents provide one more tool that conventional IT shops do not have: a bottom-up and distributed approach. The real benefit, then, comes when an organization can choose the appropriate mix of technologies for a given application — providing a balance of both centralized and distributed approaches. (For a more expansive discussion of how agent technology is being applied, see my paper “Why Should We Care about Agents?”)<sup>8</sup>

---

<sup>5</sup>For more information, see Cougaar Software, Inc. (<http://cougaarsoftware.com>).

<sup>6</sup>For more information, see Whitestein Technologies ([www.whitestein.com](http://www.whitestein.com)).

<sup>7</sup>For more information, see Rhysome ([www.rhysome.com](http://www.rhysome.com)).

<sup>8</sup>See [www.jamesodell.com/WhyShouldWeCareAboutAgents.pdf](http://www.jamesodell.com/WhyShouldWeCareAboutAgents.pdf).

If you think agent technology might be appropriate for your organization, start by reading some books on agents (such as [2]). Then, talk to companies that provide or use agent technology (such as those listed in this article). In doing so, you will get an idea of how and where to use agents. Remember, though, that the biggest breakthrough with agents is that they are an *evolution* of existing technologies. What is *revolutionary* is the way we think about and use agents to design IT systems.

### Conclusion

As Friedman said so eloquently, we need:

... a global, Web-enabled platform for multiple forms of collaboration. This platform enables individuals, groups, companies, and universities anywhere in the world to collaborate — for the purposes of innovation, production, education, research, entertainment, and, alas, war-making — like no creative platform before. This platform now operates without regard to geography, distance, time, and, in the near future, even language. Going forward, this platform is going to be at the center of everything. [1]

This emergence of new business processes is resulting in new IT practices and approaches — where the two mutually reinforce each other. Now that individuals, groups, companies, and universities are interacting on a global scale, IT must not forget its supporting role. Agents are a key ingredient in IT globalizing itself.

### References

1. Friedman, Thomas L. *The World Is Flat*. Farrar, Straus and Giroux, 2006.
2. Singh, Munindar P., and Michael N. Huhns. *Service-Oriented Computing: Semantics, Processes, Agents*. Wiley, 2005.
3. W3C, Web Services Architecture, W3C Working Group Note, 11 February 2004 ([www.w3.org/TR/2004/NOTE-ws-arch-20040211/](http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/)).



### CONCURRENCE BY TOM DEMARCO

For the past 15 years, the desktop and laptop Macs that I use have had two or three, sometimes even five processors on board. More recently, even Windows machines have multiple processors.

Multiprocessing (parallel processing, array processing), which seemed like mere esoterica a while back, has come mainstream. More importantly, we're beginning to see more and more multiprocessed applications, something that would have made us all quake in our boots only a decade ago.

The agent technology phenomenon is a child of this trend toward increasingly asynchronous computing. While agents are a relatively new trend for most mainstream organizations, objects are not. When we use an object approach to any application, we're imagining that the components that make up the whole of a system are implemented as asynchronous processes. In fact they usually aren't — their asynchronous nature is merely simulated in a pure von Neumann

machine — but that's not the point. The point is that objects fit naturally in the human mind because we think of them as independent actors.

And designing systems by assigning work to independent actors uses extremely old mental muscles. It's the way we designed manual systems long before computers came along. It's the way the Romans ran their armies and conquered most of the ancient world.

### ***An Extension of the Object Approach Steers Us Toward Agents***

Agents are self-contained objects that do and must act asynchronously. I predict that this new direction will be no flash in the pan. It will involve us all for the rest of our lives in computing. The von Neumann mindset is inherently limiting, but the agent model can be used to build largely perfectable systems that advance orders of magnitude beyond the largest and most ambitious that we see today.

So far we've been experimenting with agents that interact in small numbers. In the very near future, I believe we'll see networks of thousands of interacting agents, and from there an explosion into a vastly new theory of computing with uncountable numbers of agents interacting in ways that none of us and none of them will be able to understand completely. The practice of computing will become more and more like the practice of medicine, more holistic. The reductionist model of computing will be gone.

We programmers are control freaks, and the idea of building a system as part of a network that extends out into the messy world can be upsetting, even more upsetting when we realize that some of its interfaces are to components whose very existence may be unknown to us. Our systems will no longer be hammered into proper function as in the past. Rather they will converge — Wikipedia style — on usability, and as near as we're willing to pay for, to perfection.

System development will be more and more uncontrolled, flirting with uncontrollable. Scary but wonderful. This is what the future of computing is about.



### **CONCURRENCE BY LYNNE ELLYN**

Agent technology holds tremendous promise for banishing the ennui that has set into the IT industry. For most of the last 30 years, information technology has pushed the automation boundary further and further into the realm of human activity — computers are now performing tasks that previously only humans could perform. We are *not* amazed today that computers can process payroll, communicate orders, or control machinery. All of these activities were, at one time, work that only humans could perform. The first automated payroll, the first EDI application, and the first process control computers were amazing when they were introduced because they were examples of the expanding automation boundary. But we have grown so accustomed to computers doing or assisting in medicine, education, and business that we hardly take note of new applications these days. Agent technology will shake up that complacency, as agents emerge with increasingly sophisticated abilities and become the equivalent of a corporate avatar.

Jim points out that the Internet has changed both society and business — a move from vertical structures to horizontal structures. In reality, the structure of business has become much like the Web — a network of connections that defy fixed structures — vertical or horizontal.

Businesses are realizing that markets, suppliers, and even employees are available in cyberspace. The physical location of a person or an asset has less importance than the timing, availability, and quality of those people or assets. The biggest barrier to acquiring or coordinating dispersed people or assets is the linear nature of human interaction. People can only be in one place at one time; they can only analyze one situation at a time. To determine the best price available on any given commodity can take hours or even months. The usual RFI/RFQ/RFP process takes several weeks to several months, and contractual commitments to acquire or deliver goods bind both seller and buyer into projected demand. With such a system, buyers and sellers forfeit flexibility and agility.

As agent technology advances, it is clear that a virtual world of buyer and seller agents could emerge that truly increases flexibility and agility for all. For many products, a software agent could negotiate with another software agent to transact in a virtual supply chain. Rudimentary agents already exist that do some of this today. On eBay, you can name a maximum price and allow eBay to bid for you in an auction. While this is a pretty simple example of an agent, it certainly suggests that software “agency” is a nascent trend.

The real potential of agent technology will be realized as specialized commerce environments for agents emerge. The gaming industry provides some clues for how this might occur. EverQuest, Second Life, SimCity, and other virtual worlds are models that businesses might investigate when looking for leading-edge commercial ideas. These games are spawning real micro-economies as participants buy virtual assets from one another in game money that is purchased with real currency. In a similar way, commercial buyers and sellers could be “certified” and acquire credentials allowing them to deploy agents in a virtual marketplace. Over time, software agents could become more capable through the efforts of software developers employed by the participating companies or through crowdsourcing or open source development. Software agents would transact business in an agreed-upon virtual space — but a space that could be distributed across the globe. Payment and security arrangements could be handled through third-party companies like PayPal that could also verify delivery and receipt of the appropriate merchandise. All of this could be accomplished with software agents that were empowered to act on behalf of a company or person. As the sophistication grows, people would seldom act or transact but would more likely consult to or advise a software agent regarding buying limits or the urgency of a transaction.

Agent technology is already arriving in selected areas of business. When agent *environments* emerge that allow businesses to transact real commerce in a defined virtual world the tipping point will be quickly reached that will surely blow the automation boundary beyond the confines of B2B models. Exciting times ahead!



#### **DISSENT BY LOU MAZZUCHELLI**

Red flags go up for me whenever I see statements that a new technology is necessary, especially when followed by unsubstantiated claims for improved efficiency and effectiveness. These claims are often followed by comments like “well, the first few times, you’ll actually spend more and accomplish less because you’re learning the new [whatever].” And the reality is that many followers will jump to the next new thing without ever gaining mastery of the current, if it is even to be had.

On the other hand, I have been a believer in distributed computing for my entire career, and I see what is now being called “agent” technology as an evolutionary progression of this

architectural trend, driven by substantial improvements in foundation technologies. That this progression will result in whole new classes of IT applications remains to be seen; certainly the discussion will be spirited.

One of my favorite metaphors for an agent is a home thermostat. It is autonomous: you set it and walk away, confident that it will act based on changes to its environment without you having to check in on it. You don't particularly care how it accomplishes its task, and you are certain that it will not exceed the bounds you have put on its behavior.

One might imagine that such a simple agent could not cause a lot of mischief. I held this belief as well, until I moved into a new condo one day and could not for the life of me understand why my simple little temperature agent would not regulate my heat. It was only when I ventured into the basement that I discovered my thermostat was working perfectly — but was wired to the HVAC for the condo next door. My neighbor and I had been living examples of an experiment in positive feedback — his condo would get cold, so his thermostat would turn up my heat, causing my thermostat to turn down his heat, etc. We had a good laugh and fixed the problem with a screwdriver, and I learned some interesting lessons about systems architecture and testing.

The following questions have proven to be useful for me when evaluating autonomous distributed computing systems:

- **For each autonomous task, is bounds-checking constantly performed on all inputs, outputs, and processing results?** These bounds-checking policies should be part of the system specification, as they define part of global policy.
- **For any task deemed to be “adaptive,” what are the limits to its adaptation — in other words, how much will we let the boundary conditions vary and at what rate?** This is also a necessary part of the system specification.
- **What is the system policy if an agent attempts to adapt outside its boundary conditions?** This could range from unregulated self-modification to human-moderated exception handling — items that should be explicit, not discovered later.
- **What is the default behavior of an autonomous task in the absence of input?** This is an important specification item.

As one who has seen mainstream ideas of system specification go from Victorian novel to picture book to sandbox, I am bemused by those who gushingly embrace the concept of systems exhibiting emergent behavior — this is the equivalent of a specification that says “surprise me.” Those of us with teenage children witness emergent behavior daily — while there are moments of great joy, these are not achieved without pain.

As system developers, our mission is to provide solutions that are at worst pain-free. This has implications for system specification, verification, and validation, which are only compounded by the complexities of distributed computing.

Therefore, it is logical to imagine a hierarchy of system architectures, based on complexity, and a systems development strategy that begins with a low-complexity architecture and considers

the next-higher complexity architecture if and only if the system requirements over a given time frame cannot be met with the lower-complexity architecture.

I imagine that an architecture based on a network of autonomous agents would not be a “simple” extreme of such a hierarchy. Therefore, in a logical application of the above development strategy, we would not expect the majority of systems to be based on such an architecture.

However, with a nod to the social realities of the field, I expect that we will see a great many agent-based systems implemented by influential champions of the “new” trend, followed by a generation of systems pain illuminated with the occasional flash of cyber brilliance.



#### **PARTIAL CONCURRENCE BY KEN ORR**

I think that there is definitely something significant in the development of systems from communicating components, whether you call those components “agents” or “actors” (or as they were called in a methodology called MASCOT, simply “systems” or “processes”). Indeed, I was set to write a “So what is new here?” response to Jim’s opinion, until I spoke to a friend of mine who runs a company that does a lot of high-tech consulting.

My friend had just returned from a demo his organization had created for a client, and he was pumped up. He and his colleagues had picked up the client and drove him around town; the client sat in the backseat with a laptop with wireless Internet and GPS capability, while an agent-based system tracked their GPS location and produced real-time graphics about whatever the client wanted to look for: stores, restaurants, traffic, etc. The secret to this application, my friend said, was the agents that they were able hook up with that were available on open source.

So I’m pretty sure that agent technology is real. To second Tom’s comments, the development of agents that can send and receive messages using common interfaces makes possible a great many interesting opportunities for collaborative systems. But the reason that mine is not a full concurrence with Jim’s opinion has to do with quality control. It is one thing to build something out of dozens (or hundreds) of easily available agents from the Web; it is quite another to bet someone’s life on them. Agents and agent communication are elegantly simple, and asynchronous to boot, but that is only part of the equation. How do you know if what they are delivering is correct, and how do you know that the contributor of this particular agent won’t change it for the worse at some particularly inauspicious moment?

In this latter concern, I am closer, I suspect, to Lou Mazz’s dissent than Jim’s opinion. While I am a believer in collaboration and agile development, I try not to let those beliefs override my equally strong belief in elegant design and testing. Mashups and hacks are wildly popular, and that’s cool, but there is still no magic. We are constantly reminded that the Web is all about small pieces, loosely coupled, but that doesn’t obviate the need to design, test, document, and support each of those small pieces carefully.

## ABOUT THE CONTRIBUTING AUTHOR

James Odell is a Senior Consultant with Cutter Consortium's Enterprise Architecture practice and a regular contributor to the Enterprise Architecture advisory service. He is a consultant, writer, and educator in the areas of agent-based and object-oriented (OO) systems. Throughout most of his 35-year career, Mr. Odell has been heavily involved in developing better methods to understand, communicate, and manage system requirements. He was one of the early innovators of information engineering methodologies.

Mr. Odell is one of the first practical implementers of OO analysis and design. Working with the OMG and other major methodologists, he continues to innovate and improve OO methods and techniques. He participated in the development of UML, and is the cochair of the OMG's Analysis and Design Task Force as well as the Agents Special Interest Group.

Mr. Odell conducts international seminars and workshops and provides consulting to major companies worldwide. Formerly, Mr. Odell was the principal consultant for KnowledgeWare, Inc., where he pioneered and taught the concepts of data modeling, information strategy planning, and CASE technology application. He now works with Intelligent Automation, Inc., to advance the state of agent technology within the OMG and the computer industry.

Mr. Odell has coauthored, with James Martin, several books including *Object-Oriented Analysis and Design*, *Object-Oriented Methods: Pragmatic Considerations*, and most recently, *Object-Oriented Methods: A Foundation, UML Edition*. He also has published three books on agent-oriented software engineering.

Mr. Odell's clients include Amazon.com, Netscape, Chrysler, Capgemini, DHL, NASA, and other major companies across various business sectors in 19 different countries. He can be reached at [jodell@cutter.com](mailto:jodell@cutter.com).

---

The *Business Technology Trends & Impacts Advisory Service Council Opinion* is published by Cutter Consortium, 37 Broadway, Suite 1, Arlington, MA 02474-5552, USA. Tel: +1 781 641 9876 or, within North America, +1 800 492 1650; Fax: +1 781 648 1950 or, within North America, +1 800 888 1816; E-mail: [service@cutter.com](mailto:service@cutter.com); Web site: [www.cutter.com](http://www.cutter.com). Group Publisher: Kara Letourneau, E-mail: [kletourneau@cutter.com](mailto:kletourneau@cutter.com). Managing Editor: Cindy Swain, E-mail: [cswain@cutter.com](mailto:cswain@cutter.com). Production Editor: Pamela B. Ardila; E-mail: [pardila@cutter.com](mailto:pardila@cutter.com). ISSN: 1529-4870. ©2007 by Cutter Consortium. All rights reserved. Unauthorized reproduction in any form, including photocopying, faxing, image scanning, and downloading electronic copies, is against the law. Reprints make an excellent training tool. For information about reprints and/or back issues of Cutter Consortium publications, call +1 781 648 8700 or e-mail [service@cutter.com](mailto:service@cutter.com).