

Part 1

Why Should We Care about Agents?

Centralizing a corporation was once considered an efficient way to run an enterprise. Decisions and information processing occurred in an orderly, top-down, hierarchical manner. However, it is now clear that this type of system only works in a reasonably stable market. Globalization and changes in technology are causing today's market to be in a state of constant flux. Companies that cannot adapt fast enough to thrive in new markets will be left behind.

In response, many companies are now building agent-based systems. These systems employ agents that can distribute functionality across a vast computing network. Furthermore, agents can not only adapt to their environment but can also evolve by learning from the environment. In short, they are the ultimate in distributed computing. Such an approach prepares enterprises for an increasingly complex marketplace and enables them to respond rapidly to change.

However, agents and agent-based technology are an evolution, not a revolution. They are being built *from* today's technology and will work together *with* today's technology. While agents, objects, relational databases, legacy systems, service-oriented architectures, event-driven approaches, and so on each

have their own niche, together they can orchestrate rich systems that none of these technologies could provide alone.

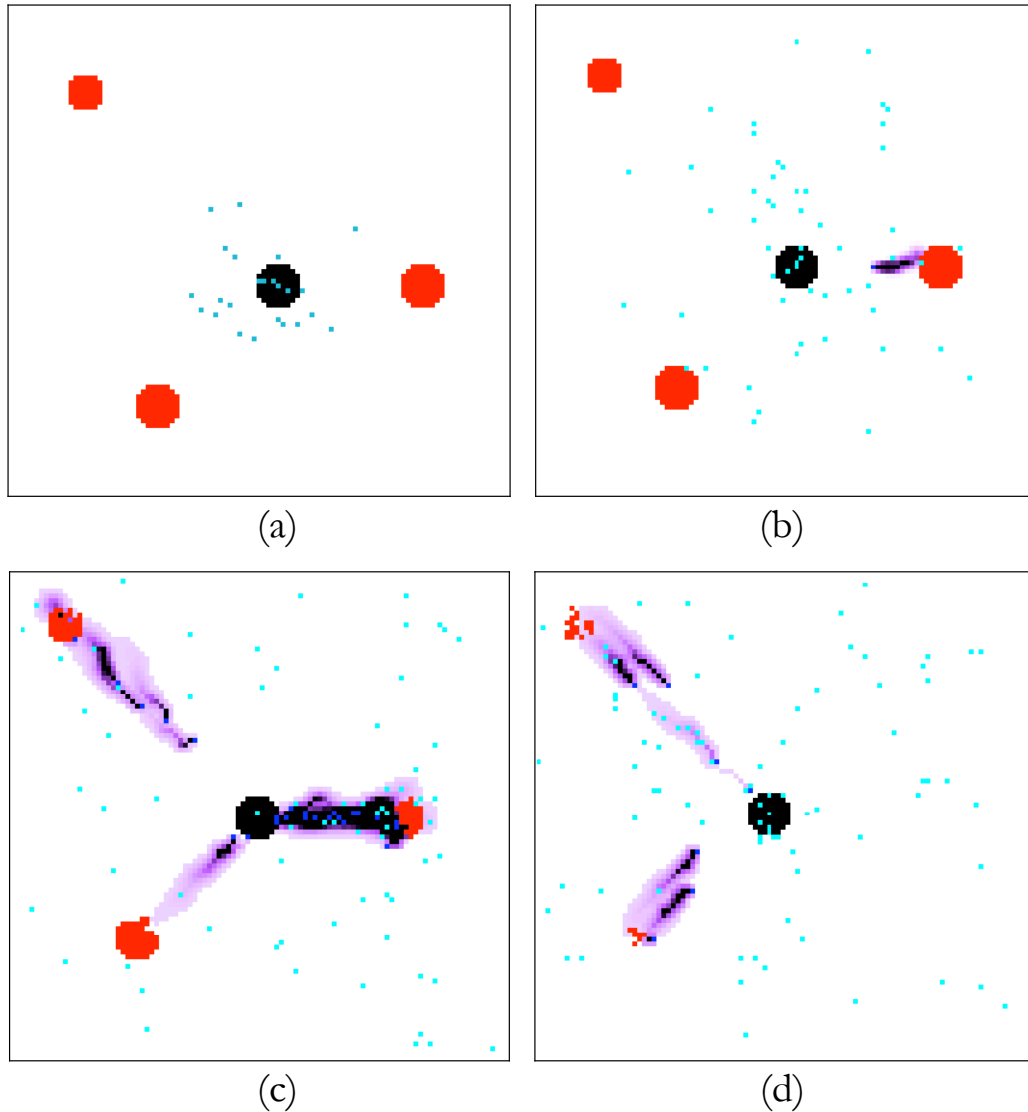


Figure 1.1 – Snapshots from an ant colony simulation.

Agents in everyday life

Biological agents can produce everyday phenomenon such as ant colonies, traffic jams, stock markets, forest ecosystems.

Software agents can enable supply chain systems, planning and scheduling applications, web services, and so on. Software agents need not be complicated. For example, in the ant colony simulation of StarLogo [1] illustrated in Figure 1.1, each ant has three simple rules:

1. Wander randomly.
2. If food is found, take a piece back to the colony and leave a trail of pheromones that evaporates over time; then go back to rule 1.
3. If a pheromone trail is found, follow it to the food and then go to rule 2.

In Figure 1.1(a), the ants are just emerging from the anthill to begin their random walk. Eventually, an ant discovers a food source and returns some to the colony, leaving a trail of evaporative pheromones (Figure 1.1(b)). Figure 1.1(c) shows the ant colony well underway in retrieving food. Lastly, Figure 1.1(d) depicts two very depleted food sources and one that is exhausted altogether.

Ant-like Agents at British Telecom

As it turns out, the foraging technique of the simple ant provides an inspiration to tackling some difficult technological problems. While the insect's totally random approach to locating food appears to be inefficient, at best, a new area of research known as *swarm intelligence* [2] has been established.

British Telecom (BT) Future Systems uses swarm intelligence for inspiration to make software systems more intelligent and robust, cheaper and simpler. The head of BT Future Systems, Chris Winter, was concerned that the trend of IT systems was toward central control. "Ants walk around at random and

there's no intelligent controller telling them what to do," said Winter. "When they find food, they simply run back to the nest laying a little pheromone trail that other ants can smell. The rest of the colony need only follow that path to fetch the remaining food." In other words, the processing is distributed, not centrally controlled

Winter used ants initially to weed out system bugs. In BT's development network, little ant-like programs run around finding problems. They leave pheromone-like timestamps which highlight the fault and draw help from other software. In other words, BT is using the ants as a first step in self-healing systems software. Many software companies are now embedding ant-like agents in systems software and middleware for similar reasons.

In another application, BT and Hewlett-Packard Laboratories created the world's first ant-based systems [3]. Both telephone networks and the Internet typically route connections through a number of intermediate switching stations. When the network is large, many possible routes exist for each connection. Using a centrally controlled routing approach, however, scales badly and can lead to failure of the entire system. A decentralized mechanism like the ant-based approach above was found both to scale well with the network size and to avoid the any possibility of a system-wide failure.

The approach here is to send ant-like agents periodically between nodes. At each node, the agent updates the node's *routing table* with information ("pheromone") indicating how long the journey took from its origin and which nodes the agent used along the way. The routing table contains a list of the node's immediate neighbors and probabilities associated with using that neighbor as the next step on the journey towards a target node in the network. The fastest ants will have a positive effect

on the probability scores of the nodes they used, while slow ants will have a negative effect.

Network congestion is another problem that was addressed by BT and HP. They considered congestion in a particular section of the network to be analogous to the depletion of a food source near an ant colony. Here, ant-like agents would search for new routes and dynamically update the virtual pheromone trail recorded in routing tables.

Other uses of ant-based systems

In ant-based systems, an ant colony of a finite size searches collectively for a good solution to a given optimization problem. The optimal solution can only be found through the global cooperation of all the colony's ants, where the ants only communicate indirectly by adding pheromones to the environment. The BT applications above are only two examples of where techniques start with models of ant behavior, then add things that are not present in the real world. However, there are many more [1,2], some of which include the:

- Traveling Salesman Problem – where a salesman must find the shortest route by which to visit a given number of cities, each city exactly once.
- Quadratic Assignment Problem – the problem of assigning facilities to locations so that the costs of the assignment are minimized.
- Job-Shop Scheduling Problem – for a set of machines and a set of jobs, operations must be assigned to time intervals in such a way that: (a) no two jobs are processed at the same time on the same machine and (b) the maximum of the completion times of all operations is minimized.

- Vehicle Routing Problem – finding minimum cost vehicle routes where (a) every customer is visited exactly once by exactly one vehicle, (b) for every vehicle the total demand does not exceed the vehicle capacity, (c) the total tour length of each vehicle does not exceed a given limit, and (d) every vehicle starts and ends its tour at the same position (the depot).

Beyond ants

Individual ants are not very sophisticated insects. They have limited memory and a largely random element. Acting as a collective, however, ants can perform complicated tasks with reliability and consistency. Ant colonies, however, are not the only things that work like this. Beehives, flocks of birds, freeway traffic, national and global economies, societies, and immune systems are all examples of patterns that are determined predominantly by local component interaction instead of centralized authority. For IT applications, this can include order processing, supply chain, shop floor control, inventory management, message routing, and management of multiple databases. In other words, a decentralized approach should be considered where local components also have control—instead of limiting system-design approaches *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.

Painting Trucks at General Motors

Traditionally, assembly line schedules are centrally developed and controlled. Any change in the schedule must be centrally reconfigured. When the line is small and has few un-

planned stoppages, centrally controlled schedules work well. However, scheduling for most real-world assembly lines can be a nightmare: work stations break down, personnel get sick, environmental conditions are not always within acceptable limits, products coming down the line have or acquire expected defects, and so on.

Dick Morley, a technology visionary and the father of the programmable controller, swept away old assembly line schedules and developed a better system for painting trucks at GM's trucks in Fort Wayne, Indiana. "How do I schedule the non-schedulable?" Morley wondered. "Trucks do not come down the line in order of their color and frequently no paint booth is available with the correct color." Morley also discovered that many of the paint booths were typically broken down or being repaired.

In his technique, the scheduling program interacts with the each paint booth. Instead of assigning unpainted trucks to booths, GM's solution was to have the booths bid on the paint jobs. [4] To accomplish this, each booth was equipped with a simple software agent that was programmed to keep its booth busy and bid on each paint job. The amount of the booth's bid was based on how busy the booth was at the moment of bidding, whether it had to change to a different paint, and whether the booth was functioning properly.

To coordinate the various bids for each paint job, a scheduler agent acts as a broker. For example, when a truck arrives to be painted, the scheduler agent tells the booths, "I have a truck that needs to be painted red. A vacant paint booth already loaded with red paint will bid very high. However, a vacant booth with different color would bid lower because of the extra labor and time to clean and reload the paint gun. A booth that has just started to paint a truck, broken down, or otherwise less

suited for the job would bid even lower. Based on the outcome of the bidding activity, the scheduler assigns the truck to the highest-bidding paint booth.

In a top-down planned "push-through" world, if one booth malfunctioned, a centrally controlled system would require immediate recomputing; with bottom-up "pull-through" paint booth agents, other booths were ready to pick up the bidding slack at a moment's notice. This new design saved a million dollars in nine months and reduced the lines of computer code from hundreds to four. Dick Morley and GM tackled a problem where centralized scheduling did not work efficiently by adopting an agent-based approach where each booth acts on its own behalf using a market-based bidding system. Even though the scheduler was a centralized element, it deferred to distributed booth agents. Agent-based solutions do not remove centralization; instead, they try to balance it with distributed solutions *wherever it makes sense*.

Dynamic Scheduling

Current market trends are driving organizations from mass production (where the supplier tells the customer what to buy) to mass customization (where the customer tells the supplier what to provide). For small supplier organizations where resources and requirements are reasonably stable, this does not present a problem. The larger and more complex the organization, however, the more difficult it is to support centrally managed operations—particularly, large suppliers with volatile and demanding conditions that include unscheduled resource failures, periodic surges in new orders, and changes in requirements and priorities. Here, using agents can change the perspective to a more *distributed* approach, making the solution

more scalable, adaptable, and robust. In fact, many organizations have already realized their limit and can no longer scale using a centralized approach. Organizations like DHL and Credit Suisse have found that an agent-based approach is their *only* option for successfully managing the previously unmanageable size and complexity within their businesses.

At the heart of the solution, two primary forms of agents were used to gain distributed control:¹

- *Process-based agents* – have the knowledge of how to combine resources and create products as part of a workflow in a supply chain.
- *Resource-based agents* – manage the capacity-constrained resources of the systems, such as people, machines, materials, and facilities.

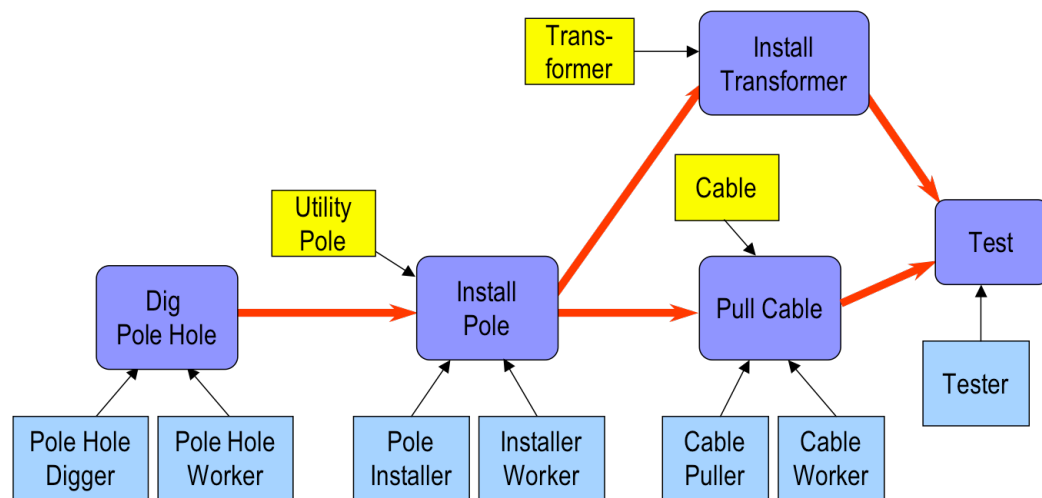


Figure 1.2 – Example of a business process where processes and resources are managed using agents.

¹ Several other types of agents were used, as well (e.g., order agents, dispatch agents, and supervisor agents). However, the process and resource agents are key to understanding the approach.

Business processes and resources using agents

Figure 1.2 contains an example of a workflow diagram for a process that installs utility poles for an electric company. The round-cornered rectangles represent processes and the square-cornered rectangles are those resources required by the processes.

In business process management (BPM) systems, such activities are usually centrally managed. Using agents, however, the process and resources can be managed in a distributed manner. For example, an Install Pole agent is a specialized entity that knows its Install Pole process needs a utility pole, pole-installer equipment, a person to operate the pole installer, and the prior completion of a Dig Pole Hole process.

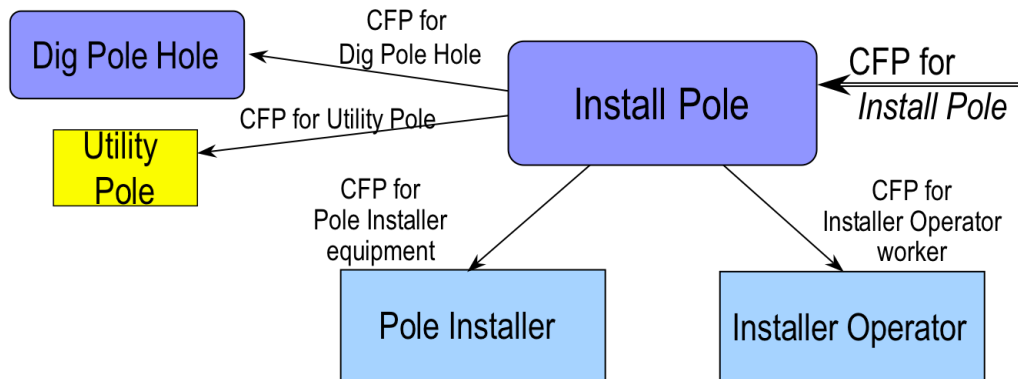


Figure 1.3 – A call for proposal (CFP) to the Install Pole agent results in a CFP to all of its resources and preceding operations.

Such knowledge can be used in several ways. For instance in Figure 1.3, a requester might be interested in an Install Pole service but would like to know its cost before the service is invoked. Here, a call for proposal (CFP) would be sent to an agent that provides such a service.² The agent would then send a CFP to those resources and preceding services that would be

² Agents are a useful way to extend service-oriented architectures (SOA).

required for the Install Pole process. Each resource agent would then calculate the charge for its usage based on the time and duration specified by the requester. If a resource is urgently needed and is in short supply, the cost might be high; if not, the price would be low. For example, if the requestor wanted a Utility Pole in two hours, but only an expensive provider has one available, the price would be higher than if the requester could wait for two days and obtain it from a cheaper supplier. In contrast, the agent representing pole installers might be able to provide a person immediately. In two days, however, they all might be busy on other assignments—but would do the requested job at double-time rates.

Once the resources had responded to the CFP with a bid, the Install Pole agent would tally up the cost for the service as a whole and send a consolidated bid to the requestor for the Install Pole service. In other words, the Install Pole agent acts like a broker on behalf of the process. If the requestor accepts the bid, the broker is notified of the award and will confirm (or decline) the award.

Agent negotiation

This interaction protocol just described is expressed using the sequence diagram depicted in Figure 1.4. The requester is playing the role of Customer and the provider is playing the role of Supplier. Interaction protocols are useful because they define the expected behavior between interacting agents—whether either can be a resource agent or a process agent.³

³ It should be noted that the notion of *requester* and *provider* agents are at the very heart of the W3C's Web Services Architecture study [6]

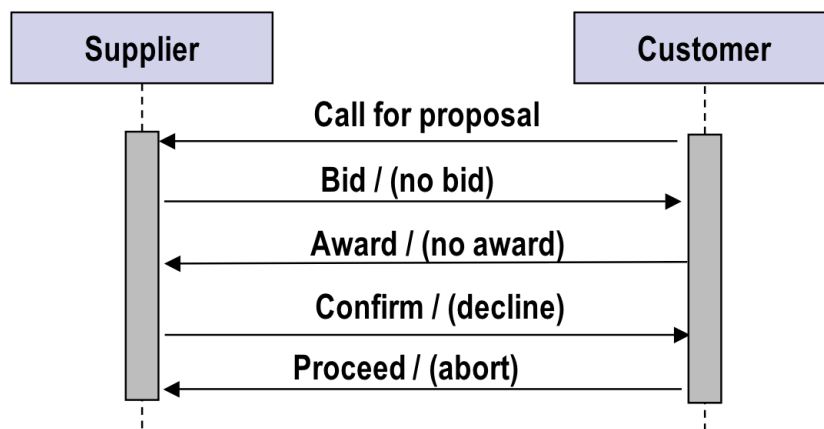


Figure 1.4 – An interaction protocol for making a contract.

Agent renegotiation

In the real world, suppliers can overcommit, have resource failures, and experience process delays. Using a top-down, centralized approach, the schedule changes tend to be re-optimized from a global level. Such a technique would work in a small operation but would not scale to a large one. However, using an agent-based approach—which is distributed by nature—enables more dynamic scheduling by adapting to these unexpected situations on an individual and local basis rather than a massive and global one.

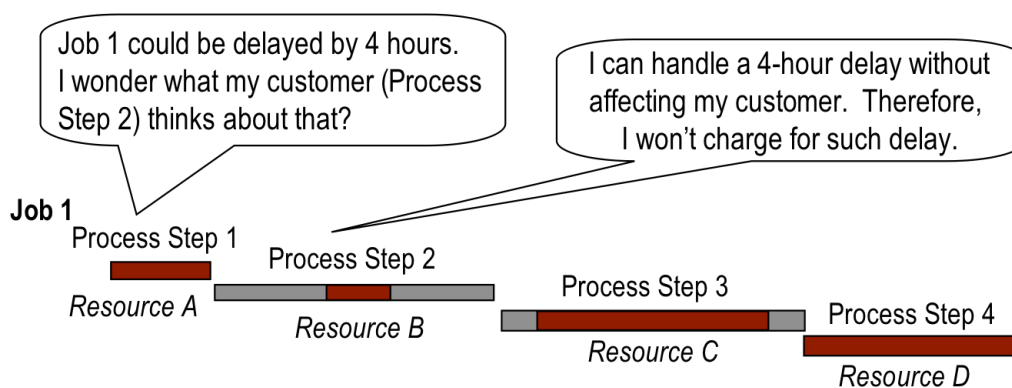


Figure 1.5 – Renegotiation example for Job 1 by process and resource agents. (based on [5])

For example, an important job request is received for Resource A, which is *already* allocated to Job 1 and Job 2. The agent representing Resource A needs to interact with the process agents for each job to determine whether the extra work can be accommodated. In Figure 1.5, the agent for Resource A asks the agent for the first step in Job 1 about any extra, or *slack*, time. It turns out that if Resource A takes on the new job, it will delay Step 1 by four hours. Now, Step 1's agent can ask Step 2 if a four-hour delay would affect its processing. Since Step 2 has lots of slack time, it can absorb the delay without affecting the overall schedule. (Note: the bar represents the total time allocated to perform the process; the segment within the bar indicates the *actual* amount of time needed by the process.)

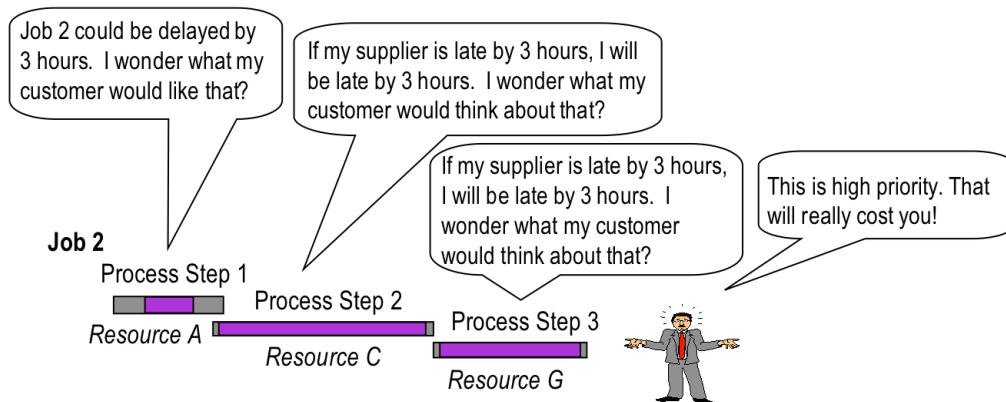


Figure 1.6 – Renegotiation example for Job 2 by process and resource agents. (based on [5])

Resource A's agent must now examine the remaining option of changing Job 2. (Figure 1.6) Step 1 would be delayed by three hours. Step 1's agent then asks Step 2 if a three-hour delay would impact its processing. Step 2's agent indicates that such a slippage would also cause a three-hour delay for it. Step 3 has a similar result and, therefore, its agent must ask the end customer if this slippage presents a problem. In this example, the

customer has a large penalty clause built into the contract for late shipments. In summary, Job 2 would result in a penalty if modified to handle the extra requirement for Resource A. However, adapting Job 1 would enable Resource A to be assigned to a new job without affecting the overall schedule. All decisions in this example were based on a local optimization rather than a global one—enabling decisions to be made in seconds instead of hours or days and reducing costs as well as time-to-market.

Agents and dynamic scheduling

The agent-based approach described above was developed with support from Rock Island Arsenal (RIA) as part of a DARPA contract to build an agent-based factory-scheduling prototype, named AARIA. In one benchmark test using these techniques, inventory costs were cut by 47 percent, lead times were cut 59 percent, and schedule reducible costs (such as overtime and inventory holding charges) were cut by 93 percent. [4]

Since that time, agent-based schedulers have been designed for suppliers in many industry sectors: manufacturing, finance, energy, and transportation. Altarum (www.altarum.com), Magenta Technology (www.magenta-technology.com), and Intelligent Automation, Inc. (www.i-a-i.com) have been active in this area.

Magenta Technology has developed an agent-based system that dynamically schedules the deployment of ships and cargo. Intelligent Automation is developing a "generic" scheduler that rapidly generates tailored and optimized scheduling engines. Using their software, an end user who has only domain expertise can define the:

- Components (e.g., resources, work center, parts, tasks/operation, and jobs) required for the scheduling application, using the GenSEB component libraries
- Protocols (interactions/constraints between components) using a standardized representation language called an Agent Interaction Protocol (AIP)
- Rules/Policies that define the order of interactions and the content of an interaction based on a user-defined scheduling algorithm

Other agent-based uses

Some common applications

Many companies have already tested agent-based applications in their research and development facilities. Successful applications are now used commercially throughout the world. A brief overview of some of them include the following:

- *Just-in-time delivery* – When operating in a highly competitive market, delivering just-in-time products and services is vital. SCA Packaging turned to an agent-based modeling solution that explored different strategies for reducing inventory levels without compromising its delivery commitments. An agent-based simulation developed by Eurobios (www.eurobios.com) enabled the company to reduce inventory levels by 35% while maintaining its delivery commitments. [7]
- *Insurance claims processing* – Acklin BV (www.acklin.nl) developed an agent-based, international, vehicle, claims-processing system for three European companies. The agent system ensures EU regulations for confidentiality and enables a robustness that can survive broader system shutdowns and

failures. As a result, the total time for client and claim identification is reduced from 6 months to 2 minutes. [7]

- *Distribution optimization* – Delivery of goods and services between site and customer is a problem for many industries. Companies such as British Telecom, DHL, and Air Liquide require transportation networks that can flexibly adapt in real-time to changing and unforeseen run-time conditions, fluctuations, and requirements. Not surprisingly, at least three companies provide agent-based solutions to this area: NuTech Solutions (www.nutech.com), Whitestein Technologies (www.whitestein.com), and Intelligent Automation (www.i-a-i.com).
- *Healthcare* – By modeling the stakeholders in primary-care systems as agents, Calico Jack (www.calicojack.co.uk) has developed a way to integrate health systems for the Scottish Executive Health Department.
- *Simulation* – Since multiagent systems involve many interacting agents, predicting the whole system's success or failure is difficult. Defining the behavior of each agent is one thing; knowing what will emerge from their joint interaction is another. Multiagent models can be used to simulate the behavior of complex computer systems, including multiagent computer systems. Such simulation models can assist designers and developers of complex application systems and provide guidance to software engineers. Agent-based simulation is also an important technique that offers strong models for representing complex and dynamic real-world environments. Agent systems simulating real-world domains may provide answers to certain physical or social problems that would be otherwise unobtainable due to their complexity. Agent-based simulation spans many areas. For example, it can aid social structures and institutions to: develop plausible explanations of observed phenomena, assist in designing organiza-

tional structures, and inform policy or managerial decisions. Agent-based simulation can help model physical systems, including intelligent buildings, traffic systems, biological populations, and software systems of all types, including eCommerce and information management systems. [7] The two most popular agent simulators are SWARM (www.swarm.org) and Cybele (www.cybelepro.com). Recently, Cybele was used to simulate air transportation systems involving flights, airports, terminal areas, controllers, and airline operation centers with varying levels of fidelity. Here, an individual simulation is comprised of tens of thousands of dynamically interacting agents configured to represent a concept/scenario.

- *Collaborative decision making and distributed control* – As systems become increasingly decentralized and autonomous, technology needs to support collaborative, team-based decision making and control. Agents can work individually and socially in teams to aid and accomplish this. In particular, Intelligent Automation has developed applications for team formation, cooperative sensing, tracking, monitoring, and traffic management.
- *Supply chains and logistics* – Today's supply networks need to be flexible, responsive, adaptive, and able to cope with the variability of demand. Traditional supply chain and logistics optimization systems are not designed to handle volatility and complexity or to function in a real-time environment. Using an agent-based approach is now a common solution for many companies. Software vendors include Magenta (www.magenta-technology.com), Cougaar Software (www.cougaarsoftware.com), and Intelligent Automation (www.i-a-i.com).

Architectures and ontologies — and agents

In addition to specific applications, the IT developer must address current architectural approaches. A brief overview of those particularly appropriate for agents include the following:

- *Ontologies* – Agents require a common vocabulary and set of concepts to communicate effectively with other agents. Cybele (www.cybelepro.com) and Cougaar (www.cougaar.org) have been adapted to use the semantic web language (OWL), the semantic web rule language (SWRL), and the services ontology (OWL-S) for the specification of agent systems. In addition, agents can be used to manage and maintain large distributed ontologies.
- *Peer-to-peer (P2P)* – P2P applications display agent-like characteristics that include both applying self-organization techniques that ensure continuous operation of the network and employing interaction protocol designs to enforce correct behavior among interacting nodes. For example, commercial e-marketplace systems like eBay include simple credit-reputation systems to reward socially beneficial behavior. As P2P systems become more complex, increasing the use of agent technologies will be appropriate. For example, the auction-based mechanisms and negotiation techniques used by agents could be used to enhance the level of automation of peers in popular applications. Since complex P2P applications are “social” in nature, they will require increasingly sophisticated approaches both to trust and reputation and to the application of social norms, rules, and structures. Social simulation would be particularly appropriate here to better understand the population dynamics of independent agents. [7]
- *Web Services (WS) and Service Oriented Architectures (SOA)* – Agents can be used to support WS and SOA in many places.

For example, Verizon chose a decentralized agent-based approach to SOA, rather than a centralized approach where all services are handled by a server or set of servers. Additionally, they employ agents to support the subscription, management, and dashboard layers of their SOA, using their IT Workbench software. By the end of 2004, Verizon was handling nearly three million service transactions per day.

Another goal of companies like Verizon is not only to reduce redundancies by building reusable services, but to put the services together to create more sophisticated and valuable services. *Service composition* aggregates services to create new functionality. Often, the composed functionality would itself be exposed as a new service with a standard interface. If this happens, a new service could be composed that would intelligently guide the service requester throughout the lifetime of a particular business process. Such an intelligent service could be implemented as an agent. The W3C Web Services Architecture study [6] recommends an agent-based approach to using service-oriented architectures—both on the service-requestor side and the service-provider side. Such an approach is not necessary or useful for simple services. However, an agent-based approach is useful when the services and their interactions require an active computational entity that: (i) has a persistent identity, (ii) can perceive, reason about, and initiate in its environment, and (iii) can communicate with other agents, including humans.

Agents act with varying levels of autonomy, depending on environmental constraints and their ongoing interactions. Because services are often best modeled as autonomous and heterogeneous, they can naturally be associated with agents. Agents make it possible to capture the interactions among services and the creation of new services as subtle compositions of others. Agents are not a panacea. However, applying

agents in the appropriate places within applications and systems enables us quite naturally to (i) capture deeper constraints on what services are willing to offer, thereby capturing richer requirements for service composition, (ii) discover trustworthy services, (iii) negotiate with external service providers, and evaluate the compliance of service providers within their contracts. They can also provide a more natural, human-like way of interacting with service requesters, as well as ensuring the entire process is carried out effectively and efficiently. [8]

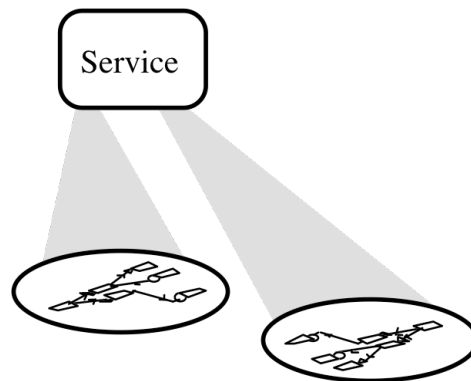


Figure 1.7 – Services can be carried out by many different business processes depending on business conditions and rules.

- *Business Process Management (BPM)* – BPM has come to mean an integrated collection of critical process technology necessary to support the business process life cycle. (Also called the *BPM suite*.) The actual business process logic can be centralized in one location, as opposed to being distributed across and embedded within multiple services. However, as illustrated in Figure 1.7, multiple business processes can be used to satisfy the service. For example, to request a product's price, the computation could vary depending: location of order, time and date ordered, quantity, customer, and other items in the same order.

The DHL pricing system had nearly 500 different processes for pricing an order because of the number and combination of variables. To handle this, DHL used agents to choose the appropriate process. In other words, each agent contained a process-execution engine that would select and execute the appropriate process for each pricing-query service request. Furthermore, if a problem arose in the middle of a process, the agent would select an alternate process. DHL found that using an agent-based approach not only executed faster but was easier to maintain. Changes to a process or process rule (such as adding a fuel surcharge) would take hours rather than weeks. The software they used from Agentis Software (www.agentissoftware.com) employed a graphic editor based on UML activity diagrams, so that even a user could change the process. Again, agents are not always appropriate for controlling business process logic. If the process is simple, straightforward, and does not require frequent changes and processing robustness, a language like BEL could be used. However, for complex processes that involve various operating conditions and require frequent and timely changes, agents should be used.

- *Event-driven architectures (EDA)* – Real-time response is often critical to customer satisfaction. Abnormal events or a combination of events can have a significant impact on an organization. More and more companies, e.g., Rhysome (www.rhysome.com) and Intelligent Automation, are using agents along with software and physical sensors to detect changes in the business environment. (Examples include RFID tags for retail supply-chain optimization, medical monitors, physical sensors that detect changes in air quality, and electronic data-capture tools for patient trials at pharmaceutical companies.) Companies that derive the greatest advantages from agent-based, event-driven architectures have

these characteristics: (i) large and heterogeneous environments, (ii) information that changes *constantly* in a variety of ways, (iii) complex exceptions and state changes in real-time, and (iv) the need to deliver and respond appropriately to that information.

- *SOA, BPM, and EDA* - These last three architectural approaches can be considered separately for including agent technology. Additionally, agent technology can be used to link business processes and services to facilitate a single architecture that integrates SOA, BPM, and EDA, as illustrated in Figure 1.8.



Figure 1.8 – Agents can be used to integrate SOA, BPM, and EDA.

Conclusion

The list above provides a sampling of uses for agent-based systems—happening now within the IT community. The underlying question, though, is why did these companies decide to use agent technology? Adopting any new technology is disruptive. To summarize, the reasons that agents were chosen by these early-adopter companies include 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
- Provides foundation for future projects
- Reduces 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.

In the typical organization, both of these situations probably exist in one form or another. As such, the savvy organization will have a mixture of technologies: object orientated (OO), relational, *and* agent-based. OO and relational technologies enable a top-down and centralized solution to business application. 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 the centralized and distributed approaches.

In the coming chapters, we wish to provide you with more insight about what agents are and how they can be used. It is one thing to say that that “agents can provide a helpful, dis-

tributive approach to software applications”; it is another to understand what that means and entails. The next chapter begins that journey by answering the questions: “What is an Agent.”

References

- [1] Resnick, Mitchell, *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds*, MIT Press, Cambridge, MA, 1994
- [2] Bonabeau, Eric, Marco Dorigo and Guy Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press, New York, 1999. (StarLogo software available from <http://education.mit.edu/starlogo/>)
- [3] Schoonderwoerd, R., Holland, O., Bruten, J. and Rothkrantz, L., 1996: “Ant-based Load Balancing in Telecommunications Networks,” *Adaptive Behavior*, vol.5, No.2, .
- [4] Christopher Meyer (ed.). *Embracing Complexity: Exploring the Application of Complex Adaptive Systems to Business*, 1996 Colloquium on the Business Application of Complexity Science, Boston, Ernest & Young Center for Business Innovation, July 17-19, 1996.
- [5] Baker, Albert, Parunak, H. Van Dyke, and Erol, Kutluhan, “Agents and the Internet: Infrastructure for Mass Customization,” *IEEE Internet Computing*, Vol. 3, No. 5, September, 1999.
- [6] Parunak, H. Van Dyke, presentation material from Altarum, 2000.
- [6] W3C, “Web Services Architecture,” W3C Working Group Note, 11 February, 2004. (Downloadable from W3C at <http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/>)

- [7] Luck, Michael, Peter McBurney, Onn Shehory and Steve Willmott, *Agent Technology Roadmap: A Roadmap for Agent-Based Computing*, report for AgentLink III, September, 2005.
- [8] Singh, Munindar P. and Michael N. Huhns, *Service-Oriented Computing: Semantics, Processes, Agents*, John Wiley, Chichester, UK, 2005.