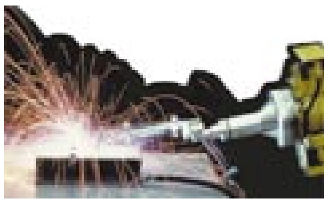
AGENTS AND THE INTERNET: Infrastructure for Mass Customization

Manufacturing is an inherently distributed process that requires effective management of physically constrained resources. The AARIA project explores how agent technology can combine this inherent distribution with the Internet's global communications infrastructure to make virtual manufacturing more cost-effective than existing, centrally managed operations.

Albert D. Baker
Enterprise Action Group
H. Van Dyke Parunak
ERIM/Center for Electronic Commerce
Kutluhan Erol
Intelligent Automation



Major market trends are driving the manufacturing complex from mass production, where the manufacturer tells the customer what to buy, to mass customization, where the customer tells the manufacturing complex what to make.[1] The Internet supports this transformation with global communication between customers and manufacturers. However, the physical realities of manufacturing impose requirements for more than just communication. In some sense, manufacturing enterprises must actually *exist* over the Internet as an efficiently managed distributed enterprise. Software agents offer a means to achieve this link and thus a reliable global infrastructure for mass customization.

The AARIA project (see the sidebar) provides a demonstration of how the manufacturing complex can move toward mass customization by using the Internet as a natural platform for managing distributed operations and by using autonomous agents as the tools for efficiently reconfiguring available productive resources. In this article, we begin by looking at the unique requirements manufacturing imposes on the infrastructure for virtual enterprises and describing the AARIA project components for meeting them. We then describe our scheduling technologies for efficient distributed resource management.

VIRTUAL MANUFACTURING ENTERPRISES

Most research related to electronic commerce is concerned with the nontrivial task of connecting corporations to each other. For example, the issues of discovering trading partners, negotiating terms and conditions, establishing secure communications, and agreeing on common product and process representations are widely addressed by the research community.

In nonmanufacturing virtual environments like digital libraries and financial services, the distinction between establishing formal relationships and operating the virtual enterprise is sometimes blurred, since both classes of transactions are fundamentally manipulations of information and do not concern themselves with matter.

Forming a virtual manufacturing enterprise is also an information processing exercise that differs little from forming (or running) any nonmanufacturing virtual enterprise. The operation of a virtual manufacturing enterprise, however, requires the manipulation of matter as well as information, and thus raises issues that do not appear in many other domains.[2]

There are two fundamental differences:

- *Speed*. The movement of information between members of a virtual enterprise is limited only by network technology. In most cases it already exceeds human timescales, and is increasing rapidly. Matter moves slowly enough to generate delays that inconvenience humans, and the technologies that limit its velocity are not improving nearly as quickly. Thus operation of an Internet-based manufacturing system must manage temporal delays that are long compared with the associated information movement, and whose length is often of more concern to human users of the system.
- *Conservation*. Information can be duplicated as often and as accurately as desired, at an ever-decreasing cost. Matter can be neither created nor destroyed, and the processes and equipment that can change its form are complex, expensive to build and operate, slow, and prone to malfunction.

Error correction is a good example of these fundamental differences. If a message is garbled during transmission, the defect is routinely discovered and corrected by retransmission. Usually neither the sender nor the recipient is aware of the slight delay or increased cost involved. However, if a physical part is damaged in production or shipping, the delay and cost of repair or replacement impose considerable burdens on both the supplier and customer.

AARIA SUPPORT FOR VIRTUAL MANUFACTURING

AARIA supports the hybrid "info-mechanical" domain of a virtual manufacturing operation through three components:

- an agent architecture that decomposes the system to address both information modularity and the physical realities of manufacturing;
- a simulator that allows agents to reason about the consequences of their actions in the physical world before actually executing them; and
- an infrastructure to support the implementation of these agents.

This section describes each component in more detail.

Agent Architecture

Traditional software engineering emphasizes functional decomposition as a guide to developing a new system architecture. Because people usually construct information artifacts to correspond to specific functions, the functional approach is often successful in purely informational applications.

In an info-mechanical system, however, functional decomposition can lead to agents that are a poor fit to the physical entities in the system. Our work shows that much more robust and reusable agents result from building agents around *things* rather than *functions*, wherever possible.[3] This stance does not exclude functional agents, but requires them to be subordinate to and consistent with the physical entities in a system.

Domain partitioning. Our design methodology uses a linguistic method based on case grammar to partition the problem domain.[4] Beginning with a narrative description of the domain, we focus on the linguistic case slots that relate the nouns to the verbs. Each type of case slot is a candidate for an agent class or species, and each noun is a candidate for an instance of that species. We sketch out the interactions among these agents by role-playing exercises.

Subsequent implementation and refinement of the agents takes place in a simulation environment that continually tests them against the constraints of the physical world.

THE AARIA PROJECT

The Autonomous Agents at Rock Island Arsenal (AARIA) project has been funded by the US DARPA Agile Manufacturing and SBIR programs to demonstrate a factory scheduler based on autonomous agents that actively represent each step of manufacturing a part. The demo system runs across a network of Pentium-based computers.

The University of Cincinnati maintains a home page for the project at http://www.aaria.uc.edu.

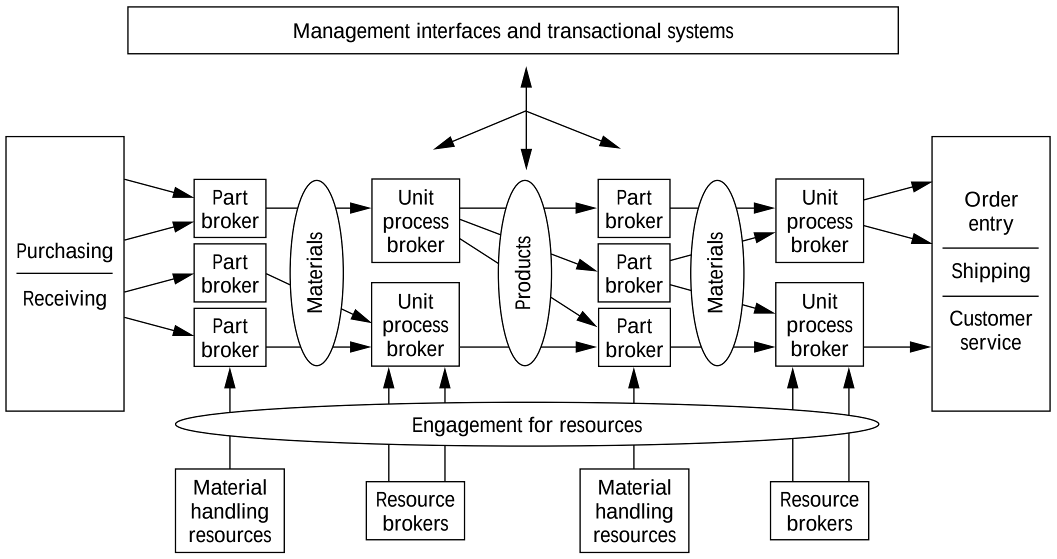


Figure 1. Collaboration diagram for the AARIA agent architecture.

Agent negotiations. Figure 1 is the skeleton of a collaboration diagram showing the major agent species in AARIA and their interactions:

- *resource broker agents* manage the capacity-constrained resources of the system, such as people, machines, and facilities;
- *part broker agents* manage material handling and inventory; and
- *unit-process broker agents* use their knowledge of how to combine resources and parts to make other parts.

These three agent types negotiate among themselves and with the customer along the axes of possible production (for example, price, quality, delivery time, product features, and speed of answers). The AARIA architecture is designed to support a dialog that iteratively reduces these axes until the manufacturing complex and customer come to agreement.[5]

In our current prototype, this dialog is only along the axes of price and delivery. We use a simple protocol whereby a customer requests a specific product, the manufacturing system responds with a bid of cost versus delivery time (as shown in the example output in Figure 2), and the customer chooses an acceptable delivery time and cost. The priority of delivery time in this dialog is a direct consequence of the info-mechanical nature of manufacturing. In purely information systems, delivery is virtually immediate. In info-mechanical systems, delivery can take days, weeks, months, or years; and expedited delivery can significantly increase cost.

Each agent in the internal supply chain that emerges makes and maintains a commitment to perform its part of the job. Agents modify their plans to meet their commitments in light of any problems such as machine failures, sickness, or emergency jobs. They also perform background computations attempting to reduce the costs of their commitments by moving jobs around or swapping resources.

System self-configuration. Later stages of production become customers to earlier stages. This structure provides a self-configuring manufacturing system where any capability can become another capability's customer or supplier. The final consumer is viewed as just another customer in this supply chain. Customers and suppliers can be from different companies, so long as they have an established business relationship.

When the customer requests a product from the manufacturing system, requests for bids propagate down the supply chain from part broker to unit-process broker to resources and other part brokers. Resource brokers and part brokers that sell raw material stocks issue bids (which are functions of cost versus time). These bids then propagate up the supply chain, getting folded together with production costs at each stage, until a final bid is presented to the customer.

When the customer chooses a cost and delivery time, purchase orders then propagate back down the supply chain, establishing commitments that are then maintained by each individual agent.

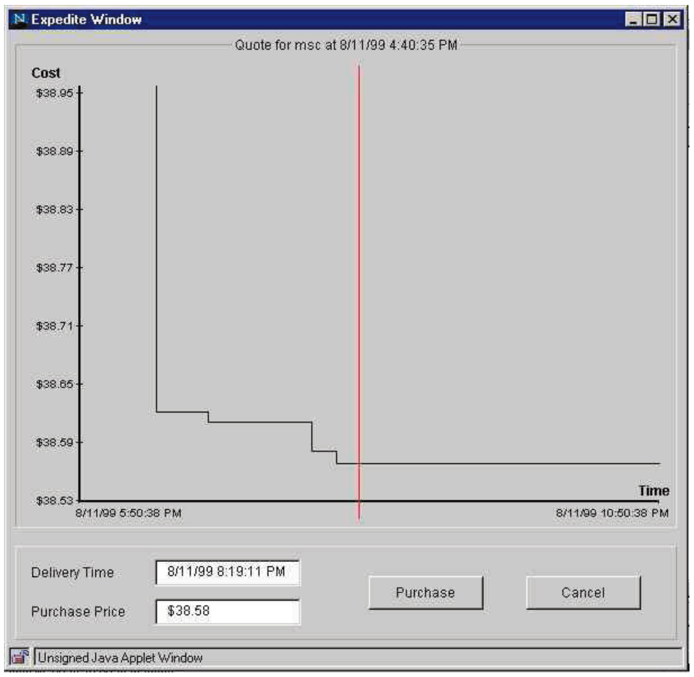
Persistent and transient agents. The agent architecture accounts for the slowness of material movement and change by using two different kinds of agents:

- *persistent agents* are resource, part, and unit-process brokers that do not leave or enter the system without human involvement; they change little over the time required for manufacturing a single part.
- *transient agents* represent entities whose rate of change is comparable to part processing times, for example, a specific material that is produced by the system and sold by a part broker; they go through a life cycle with the successive states of birth, inquiry, commitment, availability, activity, death, and archive.

Transient agents are created by relationships between persistent agents, and their life cycle reflects the development of those relationships. Early in the life cycle, they prepare for a piece of material or an active manufacturing process that does not yet exist. In later stages they represent the physical part while it is resident in the system. In the final stage they maintain an archival trace of the part.

By explicitly agenting these relationships between persistent agents, AARIA enables independent, autonomous action on the part of all the commitments in the supply chain all the way to the final consumer. This pervasive application of the agent model improves the reliability of these commitments by explicitly managing each one.

The result is an architecture that is distributed: The agents tie together resources and parts and process knowledge in a single location, in multiple locations, or in multiple corporations. The architecture is agile: the agents configure themselves to each new order placed on the system. Finally, it supports mass customization: Any capability needed for a custom product can be dynamically added into the supply chain.



Delivery Time: 8/11/99 8:19:11 PM
Purchase Price: $38.58

Figure 2. Example AARIA quote for a customized product along the possible axes of production for price and delivery. The stair-step nature of the quote shows the price varying according to when the customer wants the product delivered.

Simulation Support

Virtual enterprises must perform at least as well as their physical counterparts or there is little point in forming them. Our architecture supports fine-tuned performance through advanced simulation tools.

Again, the state of the art in manufacturing is based on functional decomposition, so factory scheduling, manufacturing execution control, and simulation are traditionally viewed as disparate information systems. As a result, scheduling systems often work off-line, without considering the dynamics of the factory floor. Existing simulation software packages have limited scripting and external integration capabilities, which make it difficult to implement complex control mechanisms. They often approximate the behavior of sophisticated factory control algorithms via simple dispatch rules and scripts, resulting in limited accuracy.

In contrast, AARIA uses a vertical decomposition based on physical and logical system entities, reminiscent of a subsumption architecture.[6] Scheduling, execution control, and simulation are integral parts of the system. An agent that represents a physical entity in the factory can communicate with that physical entity (directly or indirectly) or with another agent that emulates the physical entity's behavior and interactions. In the latter case, the system is being run in simulation mode, without any changes int the behavior of the agents representing physical entities. Thus, we have a much more realistic simulation of expected overall system behavior.

This approach has several advantages over conventional simulation. The AARIA simulator is inher-

INTERACTIVE LEAST-COST SCHEDULING

The current version of AARIA demonstrates the interactive insertion of new jobs into a distributed schedule. When a job arrives, the customer causes bid requests, bids, purchase orders, and commitments to propagate through the network. Through this process, the system finds the minimum-cost schedule for each new job, assuming already-scheduled jobs cannot be moved. (Such movement is saved for later background optimization.)

General Optimization

Given that

n_j = number of tasks in job j
p_k = processing time for task k ($k \in 1 \ldots n_j$),
m = number of machines available to perform all tasks, and
r = the cost to hold inventory (in \$/\$/time)

the general optimization problem can be stated as finding, for job j and every possible time d_j that job j can be delivered,

m_k = machine selected for task k ($m_k \in 1 \ldots m$, $k \in 1 \ldots n_j$) and
x_k = start time for task k ($x_k \in \mathrm{R}$, $k \in 1 \ldots n_j$)

which minimizes

$$\sum_{k=1}^{n_j} \$_{m_k}(x_k, p_k) \cdot \left\{1 + r \cdot \left(d_j + p_k - x_k\right)\right\} \qquad \textbf{(1)}$$

such that

$$x_k + p_k \le \begin{cases} x_{k+1} & \text{for } k = 1 \ldots n_j - 1 \\ d_j & \text{for } k = n_j \end{cases}$$

where $\$_{m_k}$ is the cost to use machine m_k from x_k to $x_k + p_k$.

Computational Complexity

This problem of scheduling a new job has both a routing decision—which of m machines will perform each of n tasks—and a sequencing decision at each machine—how long the new job's task will take fitting between any previously scheduled jobs' tasks. In the routing decision, there are m^n possible choices. In the sequencing decision, the number of possible start times is of order $|\mathrm{R}|$ or infinite (since the number of elements in the real numbers $|\mathrm{R}|$ is infinite). Yet the algorithms for finding the optimal machine sequence and start time for each task can be shown to be of order mn at each agent in the practical case where data structures remain bounded in length.[12]

Intuitively, though the number of possible routings grows exponentially with the number of tasks in the job, the number of bids sent to each machine's agent remains bounded by m, as shown in Figure A. The bids are piecewise continuous functions of cost and time. The size of a bid is bounded by the number of jobs already scheduled in the system. This places an upper bound on the amount of computation required to process a bid.

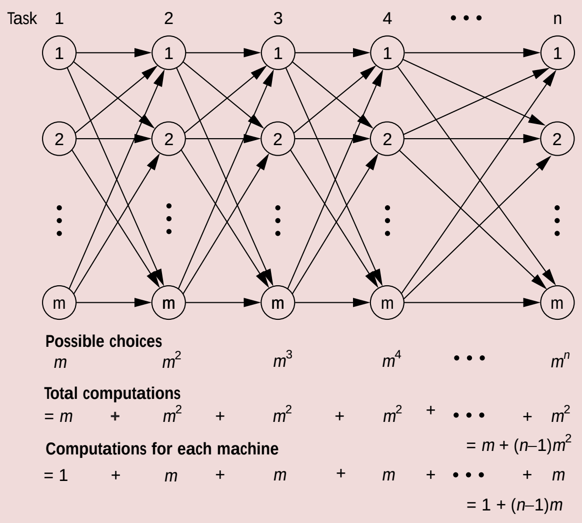


Figure A. Computations for finding minimum cost routing and start times.

Solving Equation 1 does not solve the NP-complete job-shop scheduling problem, which is left for background optimization. As customers interact with the system, the scheduling system assumes that all previous jobs have been optimally scheduled and cannot be moved (Equation 1), then finds the least-cost schedule for each new job. As a part of this process a due date d_j is determined through interaction with the customer. Then, as CPU cycles become available in the background, the issue of solving the NP-complete optimal scheduling of all jobs currently in the system is revisited.

Practical Significance

Because the algorithms are of order mn, the number of computations required to process a customer's new job will be bounded if the number of tasks in a job is bounded (a reasonable assumption) and the set of m machines is restricted to only those machines that have a chance of "winning" the task, as determined by an intelligent use of subject-based addressing (see main article section, "Agent Infrastructure").

In other words, these algorithms scale linearly as the system grows; and using subject-based addressing, they can be scaled sublinearly.

ently integrated to the actual scheduling and execution control system, rather than to a simplified control model. The simulator runs concurrently with the scheduling and execution control system. Because the execution control system models the duration of each process in proportion to actual time, this concurrency generates an accurate model of the effects of time and thus supports time-sensitive decision making.

Agent Infrastructure

Implementing multiagent software systems requires infrastructure services that are not readily available in standard computer platforms. Without services such as dynamic agent creation and interagent communication, multiagent software is prohibitively expensive to implement, in terms of both cost and time to develop.

There are several agent infrastructures available today, with varying maturity, capabilities, and target application areas. The AARIA project was implemented on the Cybele agent infrastructure. Cybele is particularly suitable for large-scale multiagent systems. It includes services for creating, migrating, and terminating agents at runtime. It supports multithreaded, shared-threaded, and nonthreaded agents, and both proactive and reactive behavior. It provides an activity model that allows an agent to concurrently handle multiple conversations in a simple, flexible way.

Cybele also includes load-balancing mechanisms that can migrate agents at runtime for improved computational resource utilization. It uses subject-based addressing to provide location-independent communication among agents, with efficient, dynamic multicasting.

Current communication services are based on TCP/IP and UDP. However, Cybele provides a flexible communication framework that can be configured at deployment time to utilize any service such as CORBA or DCOM. The Cybele communication services have been used to support agents resident in LANs and those distributed over the Internet. The AARIA implementation is object-oriented with a Java API. It runs on virtually all operating system platforms that support the Java virtual machine, including Windows NT, Windows 95, and Unix.

SCHEDULING COST-EFFECTIVE CYBERFACTORY OPERATIONS

Manufacturing is inherently distributed, yet existing manufacturing information systems are artificially centralized on large servers. Because the AARIA architecture is distributed, it offers an opportunity not only to be competitive with factories controlled by centralized computers, but even to outperform them.

The central problem is to solve the distributed scheduling problem in an agile manufacturing environment so that a "cyberfactory" can schedule its resources at least as well as a traditional factory.[7] The solution requires a scheduling system that is

- *modular*. As each agent is added to the system, the scheduling algorithms implemented by that agent must join the algorithms of other agents so the whole system produces a meaningful schedule in a reasonable amount of time.
- *interactive*. As each new customer interacts with the system over the Internet, the scheduler must make reliable commitments about when the product can be delivered.
- *data-driven*. As a new customized product is requested, the scheduling system must parse the product description, select an appropriate routing, and place the order into the schedule.
- *open*. Each individual agent must be able to participate in multiple virtual corporations at a single time.

The AARIA project developed a set of scheduling algorithms to meet these requirements (see the sidebar, "Interactive Least-Cost Scheduling").

Enhanced Forward/ Backward Scheduling

Our basic approach ensures performance comparable to traditional centralized scheduling systems by starting with standard factory scheduling technology. The most common algorithms used in industry are variations of forward/backward scheduling.[8] These algorithms form the backbone of all commercial manufacturing and materials resource planning (MRP) systems and their more recent enterprise resource planning (ERP) derivatives.[9] They are also commonly used in supply chain management products.

Such systems iteratively schedule orders forward from the time they are released into the factory or backward from their assigned due dates. State-of-the-art forward/backward scheduling systems can schedule by job, by task within the job, or by machine. Many have net-change scheduling methods that reschedule only those parts of the schedule affected by changes. Some can schedule both people and machines to perform a single task.

Many can schedule according to the most highly constrained resource to achieve Goldratt's concept of synchronous manufacturing.[10]

It is relatively straightforward to modularize and distribute forward/backward scheduling. For example, if we schedule forward one job at a time, then the first task in the job is first scheduled with the agents that represent the resources needed to perform that task. These resource agents then pass the completion time to the resources needed to perform the next task, and so on. Backward scheduling in an agent architecture is equally simple.[7]

Once the algorithms have been distributed in the multiagent framework, the computer power available over the network can enhance what is actually being done in the algorithms. Specifically, AARIA minimizes costs while scheduling over the continuum of forward and backward schedules. A possible schedule is developed for every possible delivery time, starting from the earliest possible time given in the forward schedule, and going out to any time in the future. After a delivery time is chosen and a tentative schedule is accepted, the available computational resources are then used to optimize this schedule in the background and to recover from any faults that occur in the manufacturing system.

The AARIA scheduling algorithms scale bilinearly in proportion to the number of resources participating in the scheduling decision and the number of tasks being scheduled (see the sidebar, "Interactive Least-Cost Scheduling").

The schedule system uses parameterized product descriptors,[11] which it parses dynamically to select an appropriate routing for a possibly custom product. This routing is independent of whether the resources needed are local or distributed over the Internet. The algorithms allow each agent to participate in multiple virtual corporations, so long as each such corporation uses a similar scheduling technique.

Performance

The enhanced forward/backward scheduling algorithms can optimally allocate unscheduled resources to each new order as it arrives to the system. 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.[12]

The AARIA scheduling algorithms facilitate the Internet-based manufacturing vision described in this article, but they could also be used in cases, for example, where the Internet is being used to coordinate the use of any distributed, capacity-constraining resources.

CONCLUSION

The AARIA project has demonstrated how agents can be placed on every productive resource, linked through the Internet, and scheduled to run an effective virtual manufacturing enterprise that is self-configuring according to each new demand placed on it.

Given these technologies and the current pressures on the manufacturing complex toward distribution, agility, and mass customization, we see the manufacturing complex moving to become a network of flexibly interconnected manufacturing capabilities. The result would be a more dynamic and fuller utilization of existing manufacturing resources and thus, better service and lower costs.

The Internet offers a backbone for this vision and a medium for customer interaction with the factory floor. Together, the Internet and related agent technologies can substantially reduce the economies of scale and scope that have historically consolidated manufacturing into large mega-corporations.

Just as trucking changed our highways, a significant manufacturing presence may also change aspects of the Internet:

- Increased use of specialized WAN technologies that accommodate the sale of time-constrained manufacturing capabilities may affect the development of Internet quality-of-service protocols.
- Use of different bands and channels for sale of different capabilities that are geographically specific might generate things like Internet subchannels that cover some specified distance from any specific production site (assuming distance, and therefore transportation costs, are major determinants of whether a resource will be competitive).
- New types of routers could evolve for making these distributed and possibly localized spot markets efficient and responsive.
- New standards and protocols could evolve for describing different products and manufacturing capabilities to find each other in cyberspace, even to the point that established protocols like TCP/IP will no longer dominate.

The AARIA project has demonstrated the feasibility of agile manufacturing enterprises through a working software prototype. Work is continuing to develop and benchmark the scheduling technology. Future work is anticipated in developing market mechanisms for efficient pricing and efficient computational resource usage. The Enterprise Action Group is currently commercializing the core technology under the brand name eFactory. ■

ACKNOWLEDGMENTS

We wish to thank the other members of the AARIA team including Len Haynes, Renato Levy, Egeman Tanin, and Jun Lang (Intelligent Automation); Steve Clark and Jorge Goic (ERIM/CEC); Karen Davis, Bradley Matthews, Brian Birtle, Ben Moore, and Veena Pandiri (University of Cincinnati); Howell Mitchell and Doug Currie (Flavors Technology); Greg Peters, Don Tice, and Steve Bruen (Rock Island Army Arsenal); and Art Davies and Alan Galan (Enterprise Actin Group). The AARIA project has been funded under the DARPA Agile Manufacturing Program BAA 94–31 under contract F33615–95–C–5524, administered by the Air Force ManTech Directorate; and under the DARPA SBIR Program under contract DAAH01–97–C–R255, administered by the Army Ballistic Missile Defense Command. Other studies on agent infrastructures are in part funded by the Department of Commerce SBIR program under the contract 50–DKNB–6–90112.

REFERENCES

1. W. M. Cox and R. Alm, "The Right Stuff: America's Move to Mass Customization," Federal Reserve Bank of Dallas, Annual Report, 1998; available online at http://www.dallasfed.org/htm/pubs/pdfs/anreport/arpt98.pdf.
2. H.V.D. Parunak, A.D. Baker, and S.J. Clark, "The AARIA Agent Architecture: An Example of Requirements-Driven Agent-Based System Design," in *Proc. First Int'l Conf. on Autonomous Agents (ICAA)*, 1997.
3. H.V.D.Parunak, "'Go to the Ant': Engineering Principles from Natural Agent Systems," *Annals of Operations Research*, Vol. 75, 1997, pp. 69-101.
4. H.V.D. Parunak, J. Sauter, and S.J. Clark, "Toward the Specification and Design of Industrial Synthetic Ecosystems," in *Intelligent Agents IV: Agent Theories, Architectures, and Languages, Lecture Notes in Artificial Intelligence* , 1365, M.P. Singh, A. Rao, and M.J. Wooldridge, eds., Springer, Berlin, pp. 45-59.
5. H.V.D. Parunak, A.D. Baker, and S.J. Clark, "The AARIA Agent Architecture: From Manufacturing Requirements to Agent-Based System Design," to be published in *Integrated Computer-Aided Engineering*, 1999.
6. R.A. Brooks, "A Robust Layered Control System for a Mobile Robot," *IEEE J. Robotics and Automation*, Vol. 2, No. 1, 1986.
7. A.D. Baker, "Which Factory Control Algorithms Can Be Implemented in an Agent Architecture: Dispatching, Scheduling or Pull," *J. Manufacturing Systems*, Vol. 17, No. 4, 1998, pp. 297-320.
8. A.B. Wright, "Basic Scheduling Techniques," in *Proc. Am. Production and Inventory Control Soc.*, 1983, pp. 200-204.
9. T.J. Greene et al., "Scheduling and Loading Techniques," in *Production & Inventory Control Handbook*, J.H. Greene, ed., 2nd edition, McGraw-Hill, New York, 1987, pp. 17.1-17.61.
10. E.M. Goldratt, *Theory of Constraints*, North River Press, Croton-on-Hudson, N.Y., 1990.
11. J. Ring, "Current Enterprise Models are in Conflict with Mass Customization," available from ; http://www.parshift.com/Speakers/Speak018.htm, July 1999.
12. A.D. Baker, "Metaphor or Reality: A Case Study Where Agents Bid with Actual Costs to Schedule a Factory," in *Market-Based Control: A Paradigm for Distributed Resource Allocation*, S.H. Clearwater, ed., World Scientific Publishing, River Edge, N.J., 1996, pp. 184-223.

Albert D. Baker is the president of the Enterprise Action Group, which develops Internet-based manufacturing solutions based on software-agent technology. Baker has a BSEE from Rice University, an MSME from the University of Michigan, an MBA from Harvard University, and a PhD from Rensselaer Polytechnic Institute.

H. Van Dyke Parunak is a scientific fellow at the Environmental Research Institute of Michigan's Center for Electronic Commerce, where he directs research and development in applications of complex adaptive systems to industrial applications. Parunak has an AB from Princeton University, an MS from University of Michigan, and an MA and a PhD from Harvard University.

Kutluhan Erol is the director of distributed information systems and technologies group at Intelligent Automation, which develops software design and implementation tools for multiagent systems. Erol received a PhD degree in computer science from the University of Maryland.

Readers may contact the authors at adb@enterpriseaction.com, vparunak@erim.org, and kutluhan@i-a-i.com.