Chapter 4: Applications of Distributed Artificial Intelligence in Industry

Dr. H. Van Dyke PARUNAK
Industrial Technology Institute
PO Box 1485
Ann Arbor, MI 48106 USA

Abstract

In many industrial applications, large centralized software systems are not as effective as distributed networks of relatively simpler computerized agents. For example, to compete effectively in today's markets, manufacturers must be able to design, implement, reconfigure, resize, and maintain manufacturing facilities rapidly and inexpensively. Because modern manufacturing depends heavily on computer systems, these same requirements apply to manufacturing control software, and are more easily satisfied by small modules than by large monolithic systems.

This paper reviews industrial needs for Distributed Artificial Intelligence (DAI),[1] giving special attention to systems for manufacturing scheduling and control. It describes a taxonomy of such systems, gives case studies of several advanced research applications and actual industrial installations, and identifies steps that need to be taken to deploy these technologies more broadly.

1. The Demand for Multi-Agent Systems

This section describes the vision of agile manufacturing, outlines challenges that conventional CIM systems face in reaching toward this vision, and suggests why systems of autonomous agents may be particularly well suited to overcoming these challenges.

1.1. The Challenge of Agility[2]

Agility, the ability to thrive in an environment of continuous and unanticipated change, provides competitiveness in global markets. Manufacturers with shorter product cycles can track customer desires more closely and thus capture market share. Firms that can change volume of production rapidly can avoid sinking large amounts of capital in excess inventory, while still maximizing returns in periods of rapidly increased demand. With the demise of the cold war, agility is also the foundation for dual-use strategies that permit high levels of preparedness without diverting industrial investment into dedicated defense facilities.

Table 4.1 compares agility with other manufacturing objectives along four dimensions: cost, time, quality, and scope.

	Cost	Response Time	Quality	Scope	Relation to Agility
Agile	Low	Quick	Self Healing	Limitless	
Flexible	Low	Fast	Strong	Limited	Superseded
Lean	High	Slow	Fragile	Little	Complemented

Table 4.1: *Characteristics of Agility*

[1]In the sense used here, DAI systems may include simple reactive modules that would not individually be considered artificially intelligent, as well as humans integrated electronically into the overall system.

[2]This section relies heavily on (Dove 92).

Agility requires systems that are low in cost, quick to respond, able to correct themselves in the presence of faults, and unlimited in scope. The first two characteristics have been explored in the context of *flexible* systems (systems engineered to switch rapidly among predefined members of a family of products), but only within predefined limits, and quality has been guaranteed by brute strength rather than the ability for self-correction. *Lean* manufacturing invests heavily, both in time and money, to obtain a facility that is optimized for a particular task but deteriorates rapidly outside the design envelope.

Agility impacts the entire manufacturing organization, including product design, customer relations, and logistics, as well as production. For the sake of concreteness, this discussion focuses on production systems, but occasionally mentions applications of multi-agent systems to other manufacturing functions as well.

1.2. Problems with Conventional CIM Systems.

Conventional systems for Computer-Integrated Manufacturing (CIM) are often highly centralized. At the least, a central DB provides a consistent global view of the state of the enterprise, both for internal reference and as an interface to the rest of the organization. Typically, a central machine also computes schedules for the facility, dispatches actions throughout the factory in keeping with the schedule, monitors any deviations from plan, and dispatches corrective action.

This approach has three characteristics that impede agility. It is centralized; it relies on global plans; and it precedes execution with planning and scheduling. These characteristics aggravate four operational challenges to manufacturing scheduling and control (MSC): stochasticity (the impact of noise and randomness, such as machine breakdown), tractability (the combinatoric impact of large numbers of interacting entities), decidability (the operational implications of the factory's formal equivalence to a Turing machine, making most of its interesting properties undecidable), and formal chaos (sensitivity to initial conditions resulting from nonlinearities) (Parunak 91), as outlined in Table 4.2.

	Centralized	Planned	Sequential
Stochasticity		**X**	**X**
Tractability	**X**	**X**	**X**
Decidability		**X**	
Formal Chaos		**X**	

Table 4.2: *Challenges to Traditional MSC*

A *centralized* MSC approach is especially susceptible to problems of tractability, because the number of interacting entities that must be managed together is large and leads to combinatorial explosion.[3] All four challenges to MSC make it difficult to predict what will happen, and thus to form a global detailed *plan* that can be executed after it is computed. The need for *sequential* planning and execution is particularly difficult in the face of stochasticity (which changes the state of the world in ways unanticipated in the plan and thus makes the plan invalid) and tractability (which increases the computational resources needed to derive a schedule, to the extent that there may not be enough time available to complete a schedule before it is needed).

In addition to these operational challenges, it is difficult to implement agile systems with conventional technology in the first place. Centralized software is large, complex, and expensive. It is difficult to bring on-line. The resulting credo, "If it ain't broke, don't fix it!", is a strong disincentive to change. Because it requires extensive customization for each installation, it is difficult to realize economies of scale that would make new installations less expensive and easier to bring on-line.

1.3. Conventional MSC Approaches

The major technologies used today for MSC, particularly for scheduling and planning, are heuristic dispatching verified by simulation, numerical programming, traditional AI, and advanced heuristics.

---

[3]Some multi-agent systems rely on some sort of central control to coordinate their agents, often through a NASREM-type hierarchy (Albus et al. 87). While this approach lends itself well to conventional control techniques, the hierarchy induces a strong binding among agents that makes agile reconfiguration difficult.

1.3.1. Heuristic Dispatching

For many practical applications, shop floor control is dominated by heuristic dispatching, in which a simple decision rule determines the next job to be processed at a given workstation. Typical rules are "select the job with the shortest remaining processing time," "select the job with the earliest due date," and "process jobs in the order received.". Dozens of simulation studies have examined the performance of hundreds of such rules with respect to parameters such as total cost per job, number of late jobs, machine and labor utilization, and mean and variance of job residency in the shop (Pawalkar & Iskandar 77).

The simplicity and robustness of dispatching rules makes them attractive in the real world. However, there are regions of the solution space of schedules that cannot be generated by applying a single dispatching rule to every workstation in the plant (Blackstone et al. 82).

1.3.2. Numerical Programming

Techniques such as linear and integer programming offer clean analytical formulations of scheduling problems, and the combination of new algorithms and more powerful hardware make them feasible for solving fairly substantial problems. Their great advantage is that they can find true optima. The disadvantage is that they are combinatorially explosive and computationally intensive, and so require either a large investment in supercomputers or an environment that can wait for slow answers. Thus their main value is in relatively static situations such as long production runs. Even there, plans can be rendered useless by equipment breakdowns or the arrival of a priority order.

Because of the time required to schedule a large facility completely by numerical programming methods, two techniques have been developed to permit partial rescheduling: DEDS (Ho & Cao 83, Ho & Cassandras 83) and turnpike theory (Bean & Birge 85). DEDS (Discrete Event Dynamic Systems) adopts the techniques of perturbation analysis from continuous systems to the requirements of a discrete environment. Once a nominal trajectory of the system has been obtained (by simulation or exhaustive scheduling), limited deviations from the trajectory can be analyzed without complete recomputation. Thus predictions can be corrected within certain limits without recomputation. Turnpike theory derives its name from the problem of a traveler who has detoured off the highway before nearing the destination. Under certain conditions, it is easier and cheaper to find a way back to the highway than it is to replan the entire trip from the current location to the destination. Applied to scheduling, the challenge is to find ways to return to a previously computed schedule.

DEDS and turnpiking can help reduce rescheduling for a large facility that drifts off schedule, but are of relatively little use in the kinds of reconfiguration and capacity shifts anticipated in agile manufacturing. Agility requires machine configurations that can be changed daily and batch sizes approaching unity, not just small incremental shifts from present practice. The need to tailor the program to the facility being scheduled, as well as the time-consuming scheduling process, both make numerical programming less than ideal for an agile environment.

1.3.3. Traditional AI

Traditional AI has devoted considerable attention to problems of manufacturing scheduling and control (Smith 91). By taking into account semantic information about the domain that does not lend itself to numerical computation; by applying heuristics judiciously and selectively (rather than globally as with dispatch rules), and by adopting a "satisficing" approach that does not insist on a theoretically perfect optimum, symbolic computation has led to systems that are somewhat faster than numerical programming and are more flexible and able to accommodate richer constraints, while yielding results superior to dispatch rules. However, these systems still tend to be large, complex, and specific to a particular installation, thus making them expensive to construct and difficult to maintain and reconfigure. Furthermore, while they are faster than some numerical programming codes, they are not fast enough for a facility whose configuration and load changes daily.

1.3.4. Advanced Heuristics

Recent research in operations research (Morton & Pentico 93) combines heuristics, simulation techniques, and mathematical optimization theory in various ways to address scheduling problems. Such techniques overcome the restrictions of simple heuristics, and have many characteristics in common with AI approaches.

Implementations with an industrial track record are successful in some but not all contexts (for example, OPT (Meleton 1986, Lundrigan 1986, Vollman 1986) has difficulty with shifting bottlenecks), and some of the newer theories are still untested in industrial conditions.

1.4. How Can Multi-Agent Systems Help?

This section briefly reviews the characteristics of multi-agent systems and, based on the business context outlined above, suggests the kinds of applications where they are most likely to be valuable.

1.4.1. Characteristics of Multi-Agent Systems

Multi-agent systems offer a way to relax the constraints of centralized, planned, sequential control, though not every multi-agent system takes full advantage of this potential. They offer production systems that are decentralized rather than centralized, emergent rather than planned, and concurrent rather than sequential.

The autonomous agent approach replaces a centralized database and control computer with a network of agents, each endowed with a local view of its environment and the ability and authority to respond locally to that environment. The overall system performance is not globally planned, but emerges through the dynamic interaction of the agents in real-time. Thus the system does not alternate between cycles of scheduling and execution. Rather, the schedule emerges from the concurrent independent decisions of the local agents.

Autonomous agent systems are inspired by models from biology (ecosystems) and economics (markets), in contrast with the military patterns of hierarchical organization manifested by traditional approaches. Table 4.3 contrasts some of the advantages and disadvantages of the two philosophies.

Issue	Autonomous Agents	Conventional
Model	Economics, Biology	Military
Issues favoring conventional systems		
Theoretical optima?	No	Yes
Level of prediction	Aggregate	Individual
Computational Stability	Low	High
Issues favoring autonomous agents		
Match to reality	High	Low
Requires central data?	No	Yes
Response to change	Robust	Fragile
System reconfigurability	Easy	Hard
Nature of software	Short, simple	Lengthy, complex
Time required to schedule	Real time	Slow

Table 4.3: *Agent-Based vs. Conventional Technologies*

On the one hand, the autonomous agent approach may face some disadvantages. Theoretical optima cannot be guaranteed. Predictions for autonomous systems can usually be made only at the aggregate level. In principle, systems of autonomous agents can become computationally unstable. The degree of seriousness of these challenges needs to be assessed empirically: the optima computed by conventional systems may not be realizable in practice, and the more detailed predictions that conventional approaches permit are often invalidated by the real world.

On the other hand, an autonomous approach appears to offer some significant advantages over conventional systems. Because each agent is close to the point of contact with the real world, the system's computational state tracks the state of the world very closely, without need for a centralized database. Because overall system behavior emerges from local decisions, the system readjusts itself automatically to environmental noise or the removal or addition of agents. The software for each agent is much shorter and simpler than would be required for a centralized approach, and as a result is easier to write, debug, and maintain. Because the system schedules itself as it runs, there is no separate scheduling phase of operation, and thus no need to wait for the scheduler to complete.

1.4.2. When should Multi-Agent Systems be used?

Agent-based systems offer the greatest promise in applications with several characteristics.

Distribution.--Agents are an inherently distributed mechanism, and are a promising strategy when other constraints make a centralized architecture undesirable. Thus they are the technology of choice for interaction among widely distributed shops, or when for other reasons a centralized control computer is undesirable in a single shop (for example, when the single point of failure it presents is not acceptable).

Factorization.--Much of the power of an agent comes from its identification with some entity (such as an information source, a machine, a part, or a tool) that makes sense in the application domain. When the problem is easily conceived in terms of such naturally-occurring entities, agents can be applied fairly easily. However, factorizations that are suggested by traditional analysis but do not correspond to naturally occurring entities (such as the hierarchical decomposition of a factory) can lead to very inefficient agent architectures.

Variability.--A system whose parameters do not vary widely can be optimized through a carefully planned traditional scheduling and control system. A population of agents can reconfigure itself as it runs, an important advantage for systems that must respond to a wide range of different conditions. For example, a classical approach may be competitive in a paint shop for a factory that makes only one color of car, but a case to be described below shows the benefits of an agent architecture when many colors must be supported.

Change.--Using conventional techniques, the most expensive part of a manufacturing system is not the machinery, tooling, or energy to operate it, but software creation and maintenance. When a system has a long lifetime, this expense can be amortized over many years. When systems are expected to change frequently, the agent approach shifts the software effort from integrated systems that depend on a particular configuration to the individual agents that can be swapped in and out as the entities they represent are shuffled around.

2. A Taxonomy of Multi-Agent Systems

Several taxonomies of multi-agent systems have been published (Bond and Gasser 1988, Decker et al. 1989, Demazeau & Mueller 1990, Durfee et al. 1989, Grant 1992, Huhns 1988). The taxonomy presented here has been developed from these, giving special attention to features that are most relevant from an application perspective, based on an extensive survey of applied research and development in industrial applications of MAS. (This paper does not cite all of the cases considered, but only representative examples.) Three groups of perspectives are important: the manufacturing function to which MAS technology is applied, the architecture of an individual agent, and the architecture of the system within which agents interact.

2.1. Application Function

In manufacturing, most MASs are applied to two functions: production and design.

By far the most common application of MASs to manufacturing is in production, mostly in scheduling and control and to a lesser extent in monitoring and diagnosis. The fit of such systems to production lies in their ability to represent the distributed entities on the shop floor and endow each with some level of local intelligence.

Several systems (e.g., (Klein 1991)) support the design function. Here, multiple agents help designers to interact with one another and with their production counterparts to avoid design conflicts and ensure manufacturability.

A few systems (e.g., ARCHON/GRATE (Wittig 1992, Cockburn & Jennings 1994)) deal with other manufacturing functions, such as power distribution , information integration, and logistics.

2.2. Individual Agent Architecture

In considering individual agents, we consider how similar they must be with one another into order to interact, and how sophisticated the reasoning is within each agent.

2.2.1. How diverse are agents from each other?

If agents are to interact with one another, they must have something in common. In most research systems, agents have differing functions, but share a common architecture. Sometimes this commonality extends to every aspect of an agent except for a few parameters that it manipulates (Morley & Schelberg 1993). At the other extreme are "body-head" architectures, where agents' bodies can have radically different architectures, even being different off-the-shelf products, but with a common "head" to permit communications among agents (Jennings et al. 1992). (In the simplest case the "head" takes the form of a standard language, such as SQL.) "Body-head" architectures are a common mechanism for incorporating humans as peer agents in a system. Systems whose agents differ from one another by more than parametric variation may be termed "heterogeneous."

While communications usually takes the form of digital messages over a network, this is not the only possible coordination mechanisms. Agents may coordinate solely through their effects on a shared environment. In this case, the "head" of a "body-head" architecture consists of common sensor-effector modalities.

2.2.2. How sophisticated is an individual agent's reasoning?

Different levels of sophistication can be used by different agents, or even by the same agent at different times and under different circumstances. (Demazeau & Mueller 1991) offers a variety of studies along this important dimension.

Levels of Sophistication

In theory, an agent's sophistication can range from a simple *sensing* agent that reacts to its environment but has no memory and no model of other agents, all the way up to full *human* capabilities. In practice, actual implementations add at least *memory* to sensing, so that the agent maintains local state. The next level of sophistication is *self-consciousness*, in which each agent knows of the existence of other agents distinct from itself, and thus can carry on rudimentary communication. A *social* agent goes a step further and models other agents' states, goals, and plans. Higher capabilities yet include such functions as as making commitments to one another, planning tasks, and learning from experience (Shoham 1993).

Few industrial applications of artificial agents at present go beyond self-conscious agents. In industry, more complex functions are usually provided by using artificial agents as an interface for a human operator, to whom they furnish information and from whom they take commands. Thus the main point of the system is to augment, rather than replace, the human operator. From an industrial perspective, it is both expensive and technically complex to automate the controls on much existing production equipment to the point that a human operator could be replaced, and social and political concerns also challenge the wisdom of eliminating the human. Thus the most direct route for many firms to using DAI on the factory floor may be via products that are emerging under the rubric of "manufacturing execution systems" (AMR 1993, Gillespie 1992). These systems provide human staff throughout a manufacturing enterprise with electronic connectivity, data access, and decision support, and may be considered "groupware for the shop floor."

Combining Reaction and Planning

In some systems, the various levels of sophistication within each agent are separately accessible. For example, sensing capabilities provide a rapid reflex response, while higher level planning capabilities are invoked if there is time. The term "subsumption architecture" for this technique was introduced by (Brooks 1986), who eschews any symbolic representation within an agent (Brooks 1991), but the term can be extended to apply to architectures that include symbolic representations as well.

When we do add more sophisticated reasoning capabilities to a reactive agent, we would like to do so in a way that does not forfeit the benefits of simple reaction. Systems for reactive planning fall along a continuum from pure reaction (which is fast but inflexible) to pure planning (which is slow but highly adaptive) (Table 4.4).

Fastest

Pure reaction	Flavors (Morley & Schelberg 1993, Steels 1990, Brooks 1986)
Reaction modified by plan	RS (Lyons & Hendricks 1992)
Reaction overridden by plan	Phoenix (Cohen et al. 1989)
Pure planning	BOSS (Smith & Hynynen 1987)

Slowest

Table 4.4: *From Reaction to Planning*

The top three cases fall in the "reactive planning" category.

In *pure reaction* (Figure 4.1), there is no way to alter the behavior of the reactor. No planning takes place at run-time. The system is inflexible, but fast. (Brooks 1986, 1991) Such an architecture is most appropriate in tasks where the overall envelope of possible tasks is understood well in advance. Brooks applies the model to a mobile robot, and it would be appropriate for intelligent mobile vehicles used to transport materials around a factory.

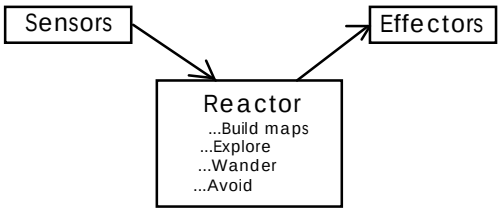

Figure 4.1: *Pure Reaction (e.g., Brooks)*

In *reaction overridden by plan* (Figure 4.2), the planner monitors the external world in parallel with the reactor, and when it disagrees with the reactor, pushes its own instructions out to the actuators, thus overriding the reactor for the nonce. However, the next time the situation arises, the reactor will still react the same way. (The planner may cache the preferred response so that the reactor can be overridden more efficiently the next time.) Because the preferred response comes via the planner rather than the reactor, it is slower. (Cohen et al. 1989) The developers of this model apply it to unmanned bulldozers used in fighting forest fires, emphasizing its suitability where the domain presents considerable uncertainty. It would be useful for agents that support interaction of human designers. Reactions can take care of anticipated requirements, while the system can plan around unanticipated situations without jeopardizing system security.

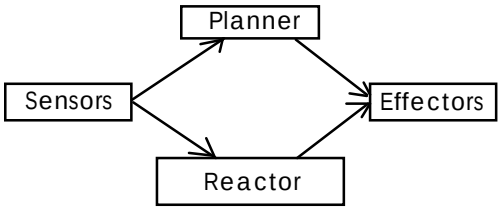

Figure 4.2: *Overridden Reaction (e.g., Cohen)*

In *reaction modified by plan* (Figure 4.3), the planner can rewire the reactor in real time so that in the future a new behavior will be performed at reaction speeds. This rewiring is in software, thus reactors of this class may be slower than hardware-wired reactors *a la* Brooks, but the response is still faster than invoking a planner. (Lyons & Hendriks 1992) This approach is useful for agents that must handle uncertainty in real-time, such as applying emergency corrections to system failures.

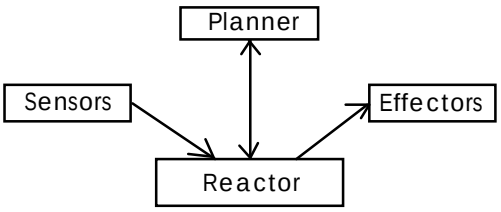

Figure 4.3: *Modified Reaction (e.g., Lyons' RS)*

2.3. System Architecture

Individual agents require the support of an overall system within which to interact. This system must provide a means for agents to communicate with one another, and define protocols for the resulting conversations. The special case when some agents are artificial and others are human merits special consideration.

2.3.1. How do agents communicate with each other?

The notion of a multi-agent system presumes that individual agents can change their shared environment, sense changes in this environment, and alter their behavior in response to these changes. These interactions via environmental change fall into two categories: *performance* and *convention.*

Performance interactions are those defined by the environmental changes that an agent must make in performing its domain function, whether or not there are other agents in the system. For example, a manufacturing cell can only run when its input buffers contain parts and there is room in its output buffer. As observers, we may interpret the action of placing a part in an input buffer or removing one from an output buffer as a "message" to the

cell to begin execution, but there need be no intentionality on the part of the suppliers and consumers of these parts to convey a message. The cell is simply performing its manufacturing function in response to changes in its environment. Usually, performance interactions take place through the physical environment rather than by way of digital communications.

Most applications in manufacturing production manipulate the physical environment (for example, changing the state of a part or the population of a buffer), and thus have performance interactions with each other.

Convention interactions rely on environmental changes that are independent of the problem domain, and whose meaning is assigned by convention. For example, the difference between "Start" and "Stop" messages to a workstation from an area controller is defined entirely by convention, not by the mechanics of the manufacturing domain. In one system, a '1' bit could mean "Start," while in another, the same '1' bit could mean "Stop." Usually, convention interactions take place over a digital communications medium.

"Communications" usually refers to convention interactions. Various communication strategies differ depending on whether a message is addressed to a specific agent or broadcast to the entire population, and whether messages persist after being sent. All combinations of these options have been explored (Table 4.5), and a single system will sometimes use more than one.

	Are messages addressed to specific agents?	
Do messages persist after being sent?	**Yes**	**No**
Yes	Interactive Transaction Net (Lee et al. 1993)	Blackboards (Nii 1986)
No	Directed	Broadcast

Table 4.5: *Communication Strategies*

2.3.2. What protocols govern the interaction?

A protocol defines the structure of the dialog among agents. Most commonly, agents *react* without question or discussion to messages or commands from one another. This simple technique can produce surprisingly complex behavior (Steels 1990). At the next level of sophistication (but one used only rarely in research systems), agents express their opinions to a central moderator, who counts or weighs their *votes* in order to select a course of action (Fordyce et al. 1992). The contract net (Davis & Smith 1983; Muller 1994) uses a *negotiation* among a limited set of participants and with a fixed protocol in order to select a course of action. *Constraint propagation* permits decisions to be made on the basis of multidirectional discussions among agents, without the rigid structure implicit in negotiation. A growing body of research permits agents to embody a model of *speech acts* that enables them to reason about messages in such categories as assertions, directives, declarations, and commitments (Shoham 1993, Jennings 1992, Jennings & Mamdani 1992). While the resulting discussions may be considered negotiation, the degree of complexity is much higher than in the cases labeled "Negotiation" or "Constraint Propagation."

2.3.3. Hybrid Systems: Communities of Humans and Machines

In some cases (Pan & Tenenbaum 1992) humans function as peer agents, rather than just as operators. That is, other agents do not distinguish between interactions with humans and interactions with artificial agents. All such hybrid systems are heterogeneous (as defined above), but there are heterogeneous systems that are not hybrid. This section describes hybrid systems as a natural outcome of the history of manufacturing technology, outlines two directions from which one can approach such an architecture, and raises some challenges for further research and development.

The Changing Roles of People and Machines

Throughout the history of manufacturing, the relative roles of people and their equipment has been changing. Multi-agent systems make possible yet another change in these roles (Kirn 1994; Hall 1994).

Before the industrial revolution, tools are mostly manual. People supply both the power to operate the tool and the intelligence to guide its application; the tool is both passive and ignorant. The role of the human is that of artisan, closely coupled to the material being transformed through relatively simple and inactive tools.

The industrial revolution moves from passive tools to active machines. New sources of energy supplement and even replace human strength, but human intelligence remains dominant. The strength and activity of the machine stands between the human and the material; the worker's interaction is more with the machine than with the material, and humans move from being artisans to operators.

With the advent of computers and the multi-agent paradigm, equipment in the factory gains intelligence in addition to strength, and machines become colleagues and partners to humans. People still remain in overall charge of the process, but the paradigm of emergent control presumes that equipment assumes much more responsibility than before. Hybrid architectures explicitly support this partnership of human and machine.

Two Approaches to Human-Machine Communities

Two areas of computer science have been developing techniques that can implement the partnership between people and machines toward which technical history is leading. Classical work in multi-agent systems builds systems of artificial agents, into which we can introduce people. Computer-supported collaborative work (CSCW) uses computer technology to integrate communities of humans, and opens the door to adding in artificial agents.

Both MAS and CSCW rely on the body-head architecture to integrate dissimilar agents (whether different people as in CSCW, or heterogeneous artificial agents in some MAS work). The head contains knowledge of the individual agent (which is customized for each class of agent), and knowledge of the rest of the community (common to all heads). It is responsible for translating between the individual agent's language and the *lingua franca* of the community.

While multi-agent systems open the door to a partnership between humans and machines, they do not guarantee it. Many multi-agent systems still model people as operators (for example, ARCHON (Wittig 1992) and PACT (Cutkosky et al. 1993)). But several others seek closer integration.

In theory, there are two ways to incorporate people into a network of agents. Either artificial agents can be made so intelligent that people come to view them as peers, or people can be represented in the network by artificial agents that make them look to other agents like computers. The first approach is the objective of classical AI's Turing Test, but most engineering applications of multi-agent systems take the second approach.

For example, the MKS system (Pan & Tenenbaum 1992) integrates a variety of human and artificial agents through a central model of the enterprise. Some of the agents implement conventional automation functions, such as monitoring and control, and their integration through the shared model bears a close architectural resemblance to conventional distributed systems. However, the system also assigns a computerized Personal Agent (PA) to each human in the network, and the other computerized agents see only this PA, not the human directly.

Traditional CSCW develops computer mechanisms (such as electronic mail, computer conferencing, interactive video conferencing, and shared databases) to help networks of humans interact more efficiently. In our terms, such systems provide each human with an electronic "head" that provides connectivity to other humans. Once such a network is in place, non-human bodies can be incorporated by giving them a head that is compatible with the others in the system.

For example, DCSS (Klein 1991) supports collaboration among designers of local area networks by providing them with a protocol for representing and justifying design decisions, mechanisms for detecting conflicts among proposals, and resources for suggesting alternative solutions. The system offers valuable support for purely human teams of designers, but once the protocols are defined, it can be augmented by the inclusion of automated design consultants as well.

3. Case Studies

Most of the cases considered in the survey that led to this taxonomy are laboratory prototypes that demonstrate feasibility but are not currently functioning in commercial environments. Five cases, described in this section, are either detailed simulations based on data from real applications (rather than toy problems) or in fact on-line systems in daily commercial use. These examples are restricted to those that have been discussed in the open literature. Other commercial applications exist, but cannot be described because of proprietary restrictions. In general, these systems rely much more on reaction than on sophisticated planning, and the contract net is the most

common protocol among agents. Shop-floor scheduling and control is the most common manufacturing application of distributed agents.

3.1. GE Power Generation Job Shop

(Baker 1991) describes a detailed simulation of a contract net against data from 35 workstations in a job shop that produces parts for steam turbines for the General Electric Power Generation business. The shop as a whole includes 115 machine tools and produces 350,000 parts of 25,000 different designs each year. The case study covered 491 orders of 184 unique products, with an average of 8.2 operations per job (and a maximum of 18 operations per job). Each workstation is represented by an agent in the system, whether it is automated or staffed by a human. Other agents represent the customer and the shop purchasing department.

The customer enters a request for bid for the final product. Agents that can deliver that product post requests for the input parts or assemblies that they need, and their suppliers repeat the process in turn with the next layer of supplying workstations, until the supply chains that emerge extend to the purchasing agent. Then bids percolate back to the customer. Bids are not single numbers, but functions that give cost per unit as a function of delivery time and lot size. The customer can then place a firm order for specific lots at specific delivery times.

Computationally, Baker's agents are identical with one another, all implementing the same algorithmic costing function. No sophisticated reasoning, modeling of other agents, or learning is involved. All messages are broadcast among all agents. Both humans and automatic machines are represented by agents, yielding a hybrid system. There is no central control, and the population of agents does not change as the system operates.

There are at least five different ways in which the agents in a contract net can be mapped onto the application domain. Baker's system illustrates one of them.

1. YAMS (Parunak 1987), perhaps the original implementation of multi-agent systems in manufacturing, assigns an agent to each node in a NASREM-type control hierarchy (factory, cell, workstation, machine) (Albus et al. 1987). An agent at one level uses negotiation to identify agents under its control at the next lower level to whom to assign tasks. This basic approach has been followed by numerous other researchers. Unfortunately, it inherits the overhead of the classical centralized hierarchical structure. Probably because of this centralization, YAMS experienced severe communications bottlenecks.
2. (Shaw & Whinston 1985) and (Baker 1991) describe schemes in which agents are workstations. In Shaw's scheme, workstations try to find consumers for their outputs, while in Baker's, they seek needed inputs. The CASCADE material handling architecture (Parunak 1990) models each segment of a transport system as an agent that can both push and pull.
3. (Maley 1988) and (Duffie et al. 1988) each build negotiating systems around smart parts that carry their process plans with them and at each stage negotiate with transport and processing equipment to get the next step done. Maley goes through a central bulletin board; Duffie simply broadcasts requests.
4. Mueller at Neuchatel is designing MARS, which models each behavior as an agent (Mueller 1992). This system focuses on assembly problems, where there may not be a distinguished part, and thus looks at the Operation as the agent.
5. Linguistic case theory (Parunak 1992) provides a theoretical integrating framework that permits workstations, parts, tools, operators, and operations to function as agents.

Compared with the actual performance of the shop as scheduled by classical techniques, the schedule that emerged in real time in Baker's simulation reduced work-in-process inventory (WIP) by 47%, shop residency time by 59%, and average job tardiness by 92%, and came within 0.2% of the theoretically optimal cost per job. Such improvements provide an important competitive edge to a job shop.

3.2. ARCHON

The ARCHON project (ESPRIT 2256) (Wittig 1992, Jennings 1992, Jennings et al. 1992, Jennings & Mamdani 1992, Cockburn & Jennings 1994) is developing an architecture for integrating multiple pre-existing expert systems. While the original application domain is electrical power distribution systems, the architecture has

also been applied to realistic prototype problems in robotics and the control in the domains of cement manufacture and particle accelerators.

ARCHON is a prototypical example of a heterogeneous body-head architecture. The head of each agent maintains models of its own state and that of other agents with which it is acquainted, and uses speech act theory to negotiate task assignments with other agents.

ARCHON addresses one of the major challenges in introducing DAI to industry, the challenge of "legacy systems," systems that were put in place before an agent-based architecture was adopted. Although in the long run the agent architecture may eliminate the need for some of these systems, it is usually necessary, both economically and politically, to introduce agents in a small, focused area, and then extend their scope. When preexisting systems can be cast in the role of information sources, ARCHON provides a mechanism for turning them into agents and incorporating them in a DAI paradigm.

3.3. The Flavors Paint Shop

The Flavors PIM (Parallel Inference Machine--a MISD supercomputer) has been used to control a paint shop in a truck plant (Morley & Schelberg 1993). Finished truck bodies enter the shop, each to be painted a specific color. There are only seven paint booths, fewer than the total number of colors that the factory can produce. A booth must be set up to spray a particular color. Thus it is sometimes necessary to change the color that a given booth is spraying. Changing color increases the idle time of the booth, wastes paint that must be purged from the plumbing, and increases environmental contamination. However, refusing to change color may delay the delivery of a truck whose color is not currently available.

Each booth is represented by its own processor in the PIM, and an additional processor represents in turn each truck that arrives at the paint shop. The implementation uses a negotiation protocol to assign trucks to booths.

- When a truck arrives at the broker, the broker announces to the paint booths that a truck is available.
- Each booth that has at least empty position in its input buffer responds to this announcement, reporting its current color and the number of trucks currently in its buffer.
- The broker assigns the new truck to a booth on the basis of three rules.
    1. If a booth of the required color has responded (and therefore has space), send the truck to that booth. (This rule avoids color changes.)
    2. Otherwise, if a booth reports that its input buffer is empty, send the truck there. (This rule avoids idle equipment.)
    3. Otherwise, send the truck to any responding booth. (This rule avoids holding up trucks.)

All booth agents run the same code. Booths do not model one another. While they communicate directly with the broker, their interaction with one another is indirect, through the effect of their actions on the incoming flow of trucks. Their reasoning is extremely simple, consisting of "If-Then" rules that read and write directly to the common environment. Both broadcast (for truck announcements) and directed messages (for bids) are used. The use of a broker imposes a primitive hierarchy on the system. Humans do not participate in the system. Booths can be added and removed as the system runs.

In spite of the extreme simplicity of this model, it cuts the number of paint changes in half in comparison with a centralized control code, resulting in annual savings of about $1M. The required code is less than one-tenth the length of the conventional program, resulting in easier maintenance and upgrading.

3.4. LMS

LMS (the Logistics Management System) (Fordyce et al. 1992) supports shop dispatching at a semiconductor fabrication facility operated by IBM. The fab produces 13,000 different kinds of circuits, each of which requires between 200 and 400 operations on a variety of processing equipment (including oxidation, photolithography, and diffusion/ion tools). The system handles some 240,000 transactions per day.

The LMS system as a whole is not agent oriented, but one component of the system uses a voting protocol among expert advisors. Each lot of wafers makes multiple passes through each kind of tool, and the specific routing within a tool group frequently depends on which pass is currently taking place. Several lots may be available when a tool group becomes available. The selection of the next lot depends on four goals:

1. completing each batch as close to its promised schedule as possible;
2. meeting daily production quotas for certain types of products;
3. satisfying the demands of downstream workstations;
4. reducing setup time and increasing the utilization of individual machines.

Each of these goals is represented by a separate agent that maintains the data and heuristic functions necessary to evalute a given lot from the perspective of its specific goal. When a tool group becomes available, each of the goal advocates issues a vote for each of the available lots. The votes range from 0 (vetoing the lot from the perspective of the goal) to 1 (indicating that the goal "must have" this lot next). Any lot vetoed by any advocate is dropped. If a single lot receives a 1 vote it is chosen. Otherwise the lot with the highest average vote is chosen.

The voting goal advocates in LMS are not homogeneous, since they implement different decisions functions, but their architecture is the same. Their reasoning includes simple heuristics, but they do not model their associates or learn over time. They communicate only with a centralized judge via directed messages, imposing a shallow hierarchy, and the decision protocol is voting. Humans are not incorporated directly in the architecture, although the entire LMS system provides sophisticated operator interfaces that can be used to override system decisions. The population of agents is fixed.

3.5. Hitachi ADS

The ADS (Autonomous Distributed Systems) architecture (Mori et al. 1986) is in operation at two steel coil processing lines at Kawasaki Steel (Mori et al. 1988) and the traffic control system for the 1070 km Shinkansen, or Bullet Train, which carries 400,000 passengers per day between Tokyo and Hakata, Japan (Ihara & Mori 1984).

ADS is a data-driven architecture for real-time control. Its design was motivated by the need to be able to modify specific software control modules without bringing down the entire system, and the functions it supports reflect this motivation. The system at Kawasaki Steel is reported to have been in nonstop operation since 1988.

Each agent ("atom" in ADS terminology) consists of a head with five standard functions, and a body containing one or more modules of application code. Messages are not addressed to a specific agent, but are broadcast into a "data field" and contain a content code that individual agents use to identify messages of interest to them. An agent's head includes five standard functions:

1. Data Field Management registers the content codes of interest to the applications currently resident in the agent, traps relevant messages from the data field and brings them into view of the applications, and passes messages from application modules back out to the data field.
2. Construction Management receives new or upgraded application modules through the data field and installs them in the agent's body.
3. Built-In Test generates test data and exercises resident applications. During testing, an application can see operational data, but its messages are not passed back onto the data field. Thus applications can be validated in the actual operating environment.
4. Execution Management authorizes the execution of specific application modules when all of their necessary data is available.
5. Data Consistency arbitrates between duplicate application modules when their results differ.

ADS supports heterogeneous agents through its body-head architecture. The available literature does not discuss the internal complexity of agents (and in particular, of application modules), but considering the languages in which they are implemented (Fortran at Kawasaki, Cobol at the Shinkansen), one may reasonably conclude that they are fairly traditional. Agents communicate through the global data field. There is no evidence of any

sophisticated negotiation among agents. The agents are embedded in a traditional centralized hierarchy which is scheduled each day in advance of operations. Humans interact with the system through operator consoles, but are not represented as separate agents.

4. Moving Multi-Agent Systems into Industrial Practice

The examples discussed in the last section show that simple techniques from MAS and DAI are being applied now in industrial settings. Nevertheless, many industrial firms find it difficult to justify the expense and risk of installing such systems. This section identifies ways that researchers, technology vendors, and end users can accelerate the deployment of multi-agent technologies into commercial use, and closes with some speculation about directions that the market will lead research in applied DAI.

4.1. Researcher Issues

Researchers can help by directing their attention to industrial problems. Much DAI research concentrates on elegant theories and structures that are applied only to highly abstract, toy problems. Such toy problems face one or more of four shortcomings: they may not address a realistic industrial problem; they may not scale up well, their scope may be inadequate, and they may not provide an adequate level of demonstration.

- Often DAI research problems sacrifice *realism* for research orientation. For example, the navigation of unmanned vehicles is a rich source of DAI problems, but a comparatively minor problem in manufacturing. While some plants do use automatic guided vehicles (AGV's), these vehicles operate in a structurally stable, engineered environment, and navigation and map-building are not their major challenges. Reliability and resource allocation are much more critical. While DAI could be applied to these problems, such application still does not offer the greatest industrial benefit, since by far most industrial material handling is with various forms of conveyor system. Far more promising applications for DAI, though less dramatic, are in the allocation and coordination of various pieces of shop-floor equipment.

- Problems of *Scale* arise when algorithms that work well for relatively few agents in a laboratory environment are transferred to realistic numbers of agents that must interact with the real world. While computer scientists have powerful tools for calculating the impact of combinatorics on algorithms, few researchers have the industrial experience to know the number of entities (machines, parts, workers) in a typical plant or the rates at which decisions must be made, and so cannot formulate realistic scenarios in which to apply these tools. In addition to combinatoric challenges, communities of interacting agents are susceptible to complex dynamic behavior that often is not adequately analyzed in the laboratory context, but that can lead to unexpected and unacceptable behavior in application.

- *Scope* issues result from the necessary process of abstracting from the real world to a research domain, and sometimes abstracting away necessary constraints. For example, the usefulness of vision in coordinating machining operations is severely limited in environments liable to contamination with cutting fluid (which can block camera lenses), and an algorithm that requires a Sun workstation will be useless if the computational resources available at a machine tool consist of an eight-bit processor. The research community should value the contribution of those researchers who are willing to grapple with the complexities of real problems.

- The level of *demonstration* provided in the research laboratory is usually inadequate to justify commercial investment for further development. Researchers should demand, and research funders and industry associations should provide, realistically detailed models of actual manufacturing problems, both static (detailed shop-floor performance data) and dynamic (sophisticated simulations), against which DAI research can be tested and exercised to evaluate its industrial potential.

4.2. Vendor Roles

Industry will not place a technology into regular commercial use without vendor support. In a few very large firms, the vendor may be an internal department that has implemented the technology itself, but in most cases, a

technology will not be broadly deployed until it can be purchased off-the-shelf from a recognized vendor with documentation and a hot-line to call with problems.

The vendor community is beginning to recognize the potential of multi-agent systems. The PIM parallel computer from Flavors Technology in Andover, New Hampshire is being marketed as a platform for MAS applications, and Hitachi's ADS architecture is available at least in Japan. Numerous hardware vendors, including Allen-Bradley, Honeywell, Mitsubishi, Echelon, and ZWorld, are developing small controllers and bus-like communication architectuers that are designed to turn small groups of sensors and actuators into autonomous agents. While some of these steps seem mundane from a research perspective, vendor support is critical to the deployment of this technology, and researchers who test their ideas in close partnership with a vendor and using industrial-strength platforms will find the transition path much shorter.

Perhaps the greatest challenge for the vendor community is managing the technology transition so that users can upgrade incrementally. Conventional manufacturing software tends to be structured according to functions (for example, scheduling, material management, resource planning) that span shop-floor entities such as people, parts, and machines. DAI techniques are most readily applied to these entities, from whose interactions the distributed functions will emerge dynamically. Systems vendors need to develop strategies for migrating from selling scheduling modules to providing intelligent parts.

4.3. User Challenges

Some of the greatest challenges faced by potential users of multi-agent technologies are social and organizational, not technical. Users need to address these issues up front if they are to succeed in implementing these technologies. For example:

- Multi-agent technologies can drastically change the roles and importance of such traditional manufacturing functions as scheduling, MIS, and systems engineering. Staff in these functions may perceive a threat to their jobs from the new technologies, and may seek to sabotage them in order to preserve their own security, unless their concerns are recognized and addressed up front.
- People can be notoriously difficult to "reprogram," and years in the "operator" role have conditioned most workers to think of the machine as a tool rather than a partner. Effective implementation of hybrid systems will require careful attention to human work customs.
- Assignment of credit and blame can be difficult enough in a purely human organization. When automatic partners are introduced, how does credit allocation work? In conventional automation, there is precedent for focusing attention on the programmer of the system. With relatively simple agents whose behavior is emergent, the situation is much more complex. The sociology of the workplace will change greatly as a result of such systems.
- Moving from a conventional centralized system to one that uses distributed agents poses a considerable risk, especially for "early adopters," those rare industrial users who are willing to be the first among their competitors in trying out a new technology. Special attention needs to be given to building the business case that justifies such risks. The potential impact on operations and the resulting financial benefits must be quantified in a persuasive manner if users are to be persuaded to make the shift.

4.4. Market and Research Directions

Agent research spans a broad spectrum in the granularity of agents, from complex agents that possess individual intelligence down to simple agents whose individual behavior is not intelligent but whose interactions yield overall system behavior that can be termed intelligent. Current industrial applications tend to favor the latter over the former approach. Usually, the nature of the decisions to be made by a given manufacturing entity (such as a part, a machine, or a worker) are fairly simple. Research in artificial life and emergent systems has shown that acceptable overall performance can be achieved from the local interactions of such simple agents, without the need or expense of more sophisticated reasoning and global data support. This tendency does not mean that more sophisticated inferencing will not find a place in industry. However, it is likely to be deployed as an enhancement of previously installed simple agents, rather than as the first step in an industry's adoption of agent technology.

One of the most challenging visions for capital-intensive industries is that of the self-configuring system. More than half of the cost associated with a piece of computer-controlled manufacturing equipment (such as a stamping press) is in the preparation and maintenance of its software. Manufacturing strategists anticipate "brilliant machines" that come from the vendor with an understanding of their own capabilities. Once such a machine is placed on the shop floor and plugged into a multiplexed source of power and communications, the only configuration necessary is to key in its physical location and orientation. By itself, it will identify the other machines in the shop, and together with them determine how to execute the various work orders that appear on the network.

A critical issue for agent research is the issue of aggregation: how simple individual agents team up to form more complex entities that then can act as a single more complex agent. Tomorrow's economy will favor teams of smaller firms over large monolithic companies (Nagle & Dove 1991), and there will be strong demand for agent architectures that can encompass the entire scope from machine components on the shop floor to aggregates of organizations that behave as though they were single entities.

While autonomous agents will continue to make steady inroads in traditional industry, their most dramatic deployment over the next decade will be in information networks. The emerging information economy has two advantages over many other industries that make it fertile ground for agents. First, it is a relatively new industry and so has little of the tradition and inertia associated with industries such as manufacturing, transportation, construction, or mining. Second, information applications are entirely computational in nature, unlike (say) manufacturing, in which digital manipulations must be coupled with material transformations through elaborate control schemes. The Digital Equipment Corporation is developing agent-based network management mechanisms, and a Silicon Valley startup called General Magic will soon begin selling agents that will roam through information networks to execute tasks specified by their masters (Markoff 1994). While all industrial applications of agents rely on network technology, those industries that live mostly on networks will see the most rapid deployment of DAI technology.

Conclusions

The increasing demands for agility in manufacturing are imposing limits on classical approaches to manufacturing scheduling and control. Multi-agent systems, particularly networks of autonomous agents with emergent rather than imposed behavior, offer significant promise in addressing these challenges. A few systems using these technologies have been fielded in industrial settings, but concerted effort on the part of researchers, vendors, and industrial users will be necessary to deploy DAI more broadly in industry.

References

Albus,J.S.; McCain,H.G. and Lumia,R. (1987), "NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)." NBS Technical Note 1235, 1987.

AMR (1993) *The AMR Report on Manufacturing Information Systems.* Boston: Advanced Manufacturing Research.

Baker,A. (1991), *Manufacturing Control with a Market-Driven Contract Net.* Ph.D. Dissertation, Dept. of Electrical Engineering, Rensselaer Polytechnic Institute.

Bean,J.C. and Birge,J.R. (1985), *Match-Up Real-Time Scheduling,* Tech. Report 85-22, Department of Industrial and Operations Engineering, University of Michigan, 1985

Blackstone,J.H.,Jr., Phillips,D.T. and Hogg,G.L. (1982), "A State-of-the-Art Survey of Dispatching Rules for Manufacturing Job Shop Operations." *International Journal of Production Research* 20:1 (1982), 27-45.

Bond,A.H. and Gasser,L. (1988), "An Analysis of Problems and Research in DAI," in A.H.Bond and L.Gasser, eds., *Readings in Distributed Artificial Intelligence,* Morgan Kaufmann, 3-36.

Brooks,R.A. (1986), "A Robust Layered Control System for a Mobile Robot." *IEEE Journal of Robotics and Automation* RA-2:1 (March), 14-23.

Brooks,R.A. (1991), "Intelligence without Representation." *Artificial Intelligence* 47, 139-59.

Cockburn & Jennings (1994), "ARCHON: A Distributed Artificial Intelligence System for Industrial Applications." O'Hare and Jennings, eds., *Foundations of Distributed Artificial Intelligence* (this volume).

Cohen,P.R., Greenberg,M.L., Hart,D.M., and Howe,A.E. (1989), "Trial by Fire: Understanding the Design Requirements for Agents in Complex Environments." *AI Magazine* 10:3 (Fall) 34-48.

Cutkosky,M., Engelmore,R., Fikes,R., Genesereth,M., Gruber,T., Mark,W., Tenenbaum,J. and Weber,J. (1993), "PACT: An Experiment in Integrating Concurrent Engineering Systems." *IEEE Computer* 26:1 (January), 28-37.

Davis,R. and Smith,R.G. (1983), "Negotiation as a Metaphor for Distributed Problem Solving." *Artificial Intelligence* 20 (1983), 63-109.

Decker,K.S., Durfee,E.H., and Lesser,V.R. (1989), "Evaluating Research in Cooperative Distributed Problem Solving." In L.Gasser and M.Huhns, eds., *Distributed Artificial Intelligence, Volume II*, Pitman, 487-519.

Demazeau,Y. and Mueller,J.P. (1990), "Decentralized Artificial Intelligence," in Y.Demazeau and J.P.Mueller, eds., *Decentralized AI*, North-Holland, 3-13.

Demazeau,Y. and Mueller,J.P., eds. (1991), *Decentralized AI 2*, North-Holland.

Dove,R. (1992), "Putting Agility in Perspective: A Profiling Tool." Presented at *AMEF Second Annual Conference*, Orlando, Dec. 15-18.

Duffie,N.A., Chitturi,R. and Mou,J.I. (1988), "Fault-tolerant Heterarchical Control of Heterogeneous Manufacturing System Entities." *Journal of Manufacturing Systems* 7:4, 315-28.

Durfee,E.H., Lesser,V.R. and Corkill,D. (1989), "Cooperative Distributed Problem Solving," in A.Barr, P.R.Cohen, and E.A.Feigenbaum, eds., *The Handbook of Artificial Intelligence, Volume IV*, Addison-Wesley, 83-147.

Fordyce,K., Dunki-Jacobs,R., Gerard,B., Sell,R. and Sullivan,G. (1992), "Logistics Management System: An Advanced Decision Support System for the Fourth Decision Tier Dispatch or Short-Interval Scheduling." *Production and Operations Management* 1:1, 70-86.

Gillespie,D.M. (1992), "The Architecture of an Execution System." *Autofact '92*, 18-7 to 18-20.

Grant,T. (1992), "Agents that Learn to Plan." *DECUS Newsletter*, Feb., AI-3-29.

Hall, L.E. (1994), "Interface Design Issues for Distributed Artificial Intelligence Systems." O'Hare and Jennings, eds., *Foundations of Distributed Artificial Intelligence* (this volume).

Ho,Y.C. and Cao,X. (1983), "Perturbation Analysis and Optimization of queueing Networks." *Journal of Optimization Theory and Applications.* 40:4 (Aug. 1983), 559-82.

Ho,Y.C. and Cassandras,C. (1983), "A New Approach to the Analysis of Discrete Event Dynamic Systems." *Automatica* 19:2 (1983), 149-67.

Huhns,M.N. (1988), "Foreword," in M.N.Huhns, ed., *Distributed Artificial Intelligence*, Pitman, v-ix.

Ihara,H. and Mori,K. (1984), "Autonomous Decentralized Computer Control Systems." *IEEE Computer* 17:8, 57-66.

Jennings,N.R. (1992), "Using Joint Intentions in Electricity Transport Management." *Proc. ECAI Workshop on Application Aspects of DAI.*

Jennings,N.R., Mamdani,E.H., Laresgoiti,I., Perez,J. and Corera,J. (1992), "GRATE: A General Framework for Co-operative Problem Solving." *IEE-BCS Journal of Intelligent Systems Engineering* 1:2 (Winter), 102-14.

Jennings,N.R. and Mamdani,E.H. (1992), "Using Joint Responsibility to Coordinate Collaborative Problem Solving in Dynamic Environments." *Proc. AAAI-92*, 269-75.

Kirn,S. (1994), "Organizational Intelligence and Distributed Artificial Intelligence." O'Hare and Jennings, eds., *Foundations of Distributed Artificial Intelligence* (this volume).

Klein,M. (1991), "Supporting Conflict Resolution in Cooperative Design Systems." *IEEE Trans. SMC* 21:6 (Nov-Dec), 1379-90.

Lee,C.K., Mansfield,W.H.,Jr., and Sheth,A.P. (1993), "A Framework for Controlling Cooperative Agents." *IEEE Computer* 26:7 (July), 8-16.

Lundrigan, R. (1986), "What is this Thing They Call OPT?" *Production and Inventory Management* 27 (Second Quarter), 2-12.

Lyons,D.M. and Hendriks,A.J. (1992), "Planning, Reactive." *Encyclopedia of Artificial Intelligence* (2nd ed.), John Wiley, 1171-81.

Maley,J. (1988), "Managing the Flow of Intelligent parts." *Robotics and Computer-Integrated Manufacturing* 4:3/4, 525-30.

Markoff,J. (1994), "Hopes and Fears on New Computer Organisms." *The New York Times*, Thursday 6 January, C1.

Meleton, M.P. (1986), "OPT--Fantasy or Breakthrough?" *Production and Inventory Management* 27 (Second Quarter), 13-21.

Mori,K., Ihara,H., Suzuki,Y., Kawano,K., Koizumi,M., Orimo,M., Nakai,K. and Nakanishi,H. (1986), "Autnomous Decentralized Software Structure and its Application." *Proc. Fall Joint Computer Conference*, Dallas, 1056-63.

Mori,J., Torikoshi,H., Nakai,K., Mori,K. and Masuda,T. (1988), "Computer Control System for Iron and Steel Plants." *Hitachi Review* 37:4, 251-8.

Morley,R.E. and Schelberg,C. "An Analysis of a Plant-Specific Dynamic Scheduler." *Proceedings of the NSF Workshop on Dynamic Scheduling*, Cocoa Beach, FL.

Morton,T.E. and Pentico,D.W. (1993), *Heuristic Scheduling Systems.* John Wiley.

Mueller,J.P. (1992), personal communication.

Muller,H.J. (1994), "Negotiation Principles." O'Hare and Jennings, eds., *Foundations of Distributed Artificial Intelligence* (this volume).

Nagle,R., and Dove,R. (1991). *21st Century Manufacturing Enterprise Strategy: An Industry-Led View*. Volume 1. Iacocca Institute, Lehigh University.

Nii,H.P. (1986), "Blackboard Systems." *AI Magazine* 7:3,4, 40-53, 82-107.

Parunak,H.V.D. (1987), "Manufacturing Experience with the Contract Net." In M.N.Huhns, ed., *Distributed Artificial Intelligence*, Pitman, 285-310.

Parunak,H.V.D. (1990), "Distributed AI and Manufacturing Control: Some Issues and Insights." In Y.Demazeau and J.-P.Mueller, eds., *Decentralized AI*, North-Holland, 81-104.

Parunak,H.V.D. (1991), "Characterizing the Manufacturing Scheduling Problem." *Journal of Manufacturing Systems* 10:3 (1991), 241-59.

Parunak,H.V.D. (1992), "How to Describe Behavior Space." *Working Papers of the Eleventh International Workshop on Distributed Artificial Intelligence*, 1992, 303-16.

Schelberg,C. (1992), "Parallel Scheduling of Random and of Chaotic Processes." In B.Soucek, ed., *Dynamic, Genetic, and Chaotic Programming*. New York: John Wiley, 1992, 535-49.

Shaw,M.J. and Whinston,A.B. (1985), "Task Bidding and Distributed Planning in Flexible Manufacturing." *Proc. IEEE Int. Conf. on AI Applications*, 184-89.

Shoham,Y. (1993), "Agent-oriented Programming." *Artificial Intelligence* 60:1 (March), 51-92.

Smith,S.F. and Hynynen,J.E. (1987), "Integrated Decentralization of Production Management: An Approach for Factory Scheduling." In C.R.Liu, A.Requicha, and S.Chandrasekar, eds., *Intelligent and Integrated Manufacturing Analysis and Synthesis*. New York: ASME, 427-39.

Smith,S.F. (1991), "Knowledge-Based Production Management: Approaches, Results and Prospects." CMU-RI-TR-91-21. Robotics Institute, Carnegie Mellon University, 1991.

Steels,L. (1990), "Cooperation between Distributed Agents through Self-Organization." In Y.Demazeau and J.-P.Mueller, eds., *Decentralized AI*, North-Holland, 175-196.

Vollman, T.E. (1986), "OPT as an Enhancement to MRP II." *Production and Inventory Management* 27 (Second Quarter), 38-47.

Wittig,T., ed.(1992), *ARCHON: An Architecture for Multi-Agent Systems*. New York: Ellis Horwood.

