Autonomous Manufacturing: Automating The Job Shop

Article *in* Computer · October 1984

DOI: 10.1109/MC.1984.1659248 · Source: IEEE Xplore

2 authors:

David Bourne
Carnegie Mellon University
47 PUBLICATIONS **477** CITATIONS
SEE PROFILE

Mark Stephen Fox
University of Toronto
240 PUBLICATIONS **9,757** CITATIONS
SEE PROFILE

Planning and controlling "job-shop" activities is a complex problem. Constraint-directed shop-scheduling and language-oriented database systems offer some solutions.

Autonomous Manufacturing: Automating the Job-Shop

David A. Bourne and Mark S. Fox, Robotics Institute, Carnegie-Mellon University

The term "factory of the future" has lost much of its meaning because of excessive publicity heralding each new machine tool, robot, or computer-based controller. It has become increasingly difficult to differentiate between fact and fantasy. Our purpose in writing this article is to examine some of the issues involved in creating an "autonomous manufacturing environment" for discrete parts. We restrict the concept of autonomous manufacturing to only the activities performed on the shop floor. In particular, autonomous manufacturing pertains to the complete automation of *decision making* on the shop floor, whether or not the actual production is performed manually or automatically. While much of computer-aided manufacturing has been concerned with "flexible automation," we are concerned with the decision-making methodologies required for planning and control. The introduction of robotic and other flexible technologies into manufacturing increases the number of ways a product can be produced and decreases the production rate. Unfortunately, flexible technologies increase the complexity of operation and production scheduling and, because of the subsequent decrease in set-up times, there is less time for decision making. Today, decisions made manually in the shop are less than satisfactory, demonstrated by high in-process time of orders, low machine utilization, and high overheads. Such manual planning and control methods limit our ability to utilize the flexibility afforded by robotic technology.

We investigate the issues involved in constructing software systems for the planning and control of activities in the job-shop. Manufacturing is composed of many activities that can be monitored and controlled at different levels of abstraction. A shop floor can be viewed as a group of work centers, a work center as composed of manufacturing cells, and a manufacturing cell as composed of individual machines, robots, and tools (see Figure 1). Activity planning in such an environment is a complex problem in which activities must be selected and resources must be assigned and scheduled at each level of abstraction to meet production goals. While much of this can be performed before production begins, the dynamics of the manufacturing environment tend to quickly invalidate predictive planning, forcing the shop to adapt to changes. In our discussion, we assume the existence of a shop with the following characteristics:

- a set of predefined parts to be produced in small batches;
- one or more sequences of manufacturing operations defined for each part;
- one or more work centers in which an operation is performed; a work center may be either a machine, flexible manufacturing cell, or a manual station;
- orders and materials which enter the shop and a product which exits.

New product creation, advance planning, and other activities not performed on the shop floor are ignored.[1]

Furthermore, autonomy is discussed at two levels: the shop and cell level. Autonomy at the shop level involves the planning and scheduling of activities and the allocation of resources to support these activities. Activities are performed at work centers which contain one or more machines and a variety of support resources including personnel, tools, and raw materials. These resources may be shared among work centers or dedicated to a single center. The machine at a work center may be quite simple, such as a measuring device for quality control; or it may be complex, such as a transfer line or flexible manufacturing cell.

Software tools available today provide some level of decision support by means of a combination of capacity analysis and mathematical programming techniques. Except for some specialized systems, much of the decision support is predictive, which deals with only half of the problem. The shop floor is a dynamic environment where the unexpected continuously occurs, forcing changes to planned activities. Hence, the automation of decision making must not only predict shop behavior through planning, but also alter its plans based on more recent information.

Autonomy in a manufacturing cell confronts many of the same problems found on the shop floor. Scheduling

0018-9162/84/0900-0076$01.00 © 1984 IEEE

activities based on daily physical constraints are ubiquitous throughout the shop, and it is these physical constraints that change to suit the level of decision making. A factory manager is concerned with his sales to customers and with the capability of his people to produce the products within the time requirements of the customer. At the same time, a foreman is concerned with meeting his quota, set by the manager, and with maintaining the integrity of his physical charge so that he can meet the next day's needs. In the abstract, the structure of decision making is the same at each level of management, yet the particular knowledge and its application may be different at each level.

Process planning at the shop level

The first task in managing production at the shop level is to design the process by which a product is to be manufactured. Although initial process layout is performed by the planning department, many decisions related to planning are made on the factory floor by schedulers in response to unforseen occurrences such as machine malfunctions. Some level of process planning is required for true autonomy. The major components of process planning include process selection, elaboration, and sequencing.

The *selection* step determines the manufacturing subprocesses necessary to produce the part. Consider the fabrication of a cylindrical metal part with holes and planes cut into it. More than one process may be required. A forging process may be needed to transform a metal billet to approximate shape, a milling process to mill the billet to specification, and a drilling process to place the holes. The choice of process and machine may depend on the shape, size, and accuracy of the cuts required. There may exist alternative manufacturing processes to produce the same part, such as grinding versus milling or chemical processing versus machining. The complexity of subprocess selection grows with the set of alternatives.

The *elaboration* step generates the operation parameters for the chosen subprocesses. For example, tools must be chosen, Numerical Control, or NC, programs must be written, and cutting speeds must be determined.

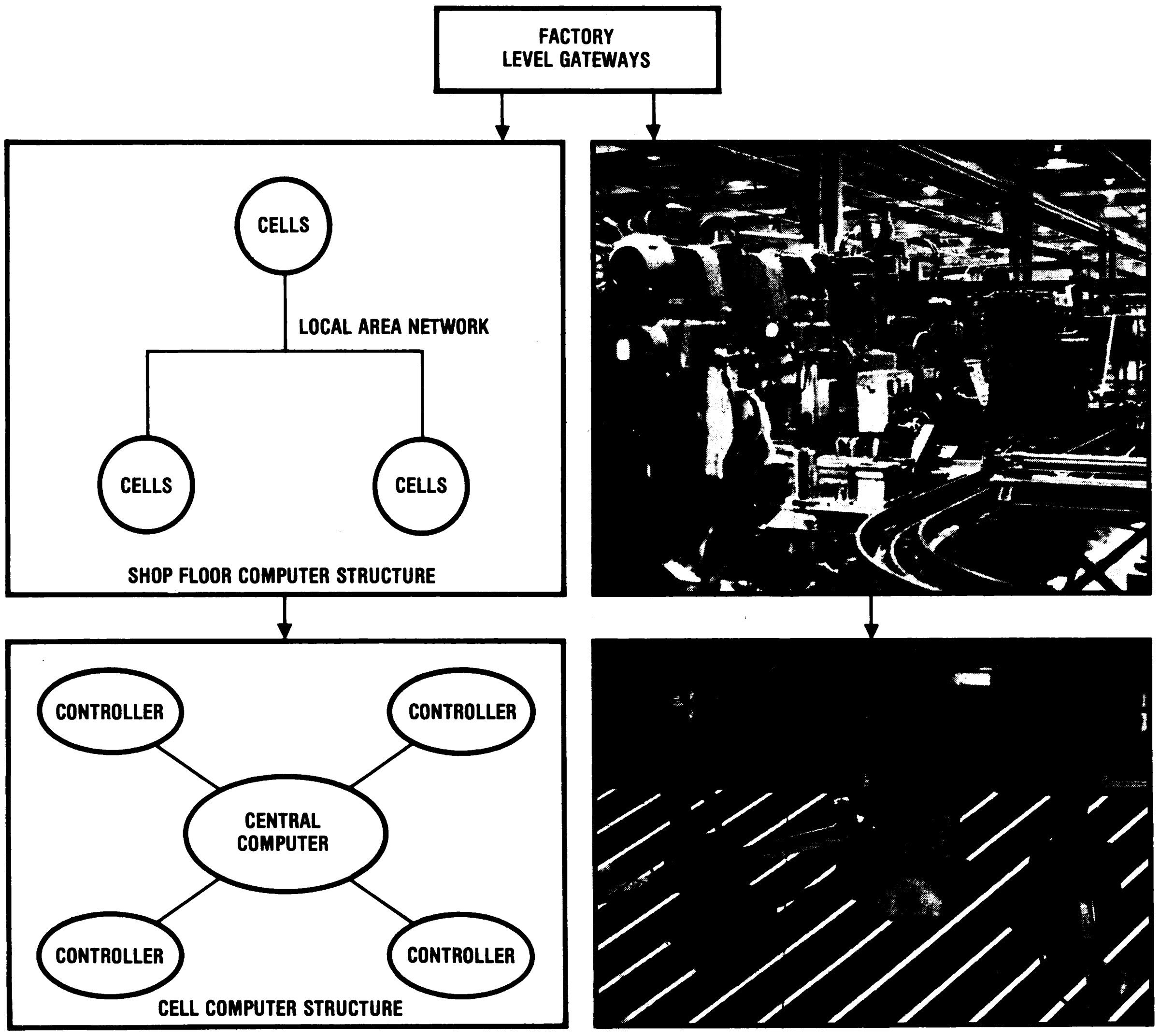


Figure 1. The autonomous manufacturing environment.

The *sequencing* step determines how to order the selected sub-processes. Consider the generation of a process plan for sheet-metal products of simple, noninterleaving structure. Subprocess selection may be simple if the product is described in terms of planes and their orientations. The language of planes and their angles of incidence is very close to the language describing the possible movements of a sheet-metal folding machine. But knowing the set of folds is not enough: the subprocesses may have to be sequenced to achieve the appropriate results. With complex parts, the possibility of a subprocess destroying the results of a previous subprocess becomes an issue (for example, to fold one part of the metal, a previous fold would have to be "undone").

From this analysis, we can see that generating a process plan requires two types of information: (1) a description of the part in terms of physical and functional characteristics; and (2) a description of the available manufacturing processes, including such descriptions as physical capabilities, quality, cost, and NC command set. Since there may be more than one process plan which can produce a part, these descriptions constrain the set of admissible process plans; that is, properties of the part to be produced and of machines which produce them restrict the selection of processes and their sequencing. For example, functional constraints (such as temperature and stress) may affect the choice of materials; physical/functional machine constraints (such as size of workbed) may restrict the size of parts; and properties of processes and materials may restrict sequencing (for example, a part must be heated before forging). Hence, the difficulty involved in process planning is dependent on product type and manufacturing technology.

A variety of computer-based approaches to process planning are being tested. The introduction of computer-aided drafting has reduced the design time but has had little impact on process planning. Interactive process planning systems exist in areas such as wire harness design and metal part production. These systems span a continuum of capabilities from simple menu selection of operations to automatic selection based on the part description. For example, an experimental system called the Automated Computer-Aided Planning Systems, or ACAPS,[2] uses part geometry, dimensions, and material specification to select appropriate operations and program them. Decisions made by ACAPS are suggestions; the user makes the final selection and performs the sequencing.

Further automation of process planning can be achieved. Selection may be further automated by means of rule-based systems[3] and constraint-directed reasoning techniques.[4] It appears that much of the knowledge relating part descriptions to processes may be represented as rules. Possible rules for process selection may include the following:

IF	the part possesses a standard cylindrical shape,
THEN	perform a milling operation on an NC milling machine.
IF	the part possesses a standard cylindrical shape and is less than 10 inches in length,
THEN	perform a milling operation on a "brand x" NC milling machine.

The first rule specifies a general "rule of thumb" that cylindrical shapes are to be milled on a milling machine. The second rule, an example of more specific knowledge, says small cylindrical shapes are to be milled on a "brand x" machine. The latter rule takes precedence over the former when both are applicable because it is more specific. These rules may be created by an expert process planner, or they may be constructed from what are called "group technology codings." Group technology endeavors to classify parts either according to common geometric descriptions or according to common manufacturing operations. In either case, the classification directly relates the part description to manufacturing processes.

The degree of difficulty in automating elaboration depends on the "distance" between the part description language and the machine programming language. ACAPS provides a facility for generating NC machine instructions.

The problem of sequencing falls into an area in artificial intelligence called planning. Planning research in AI began with the work of Newell and Simon[5] in the development of the General Problem Solver, or GPS, computer program. Planning research has evolved since GPS. Research has focused on robot planning applications, chemical experiment planning, project management, flight management, and job-shop scheduling. The possible impact of AI planning research on process planning is illustrated by the following example. Consider the manufacture of a steam turbine blade. Its initial state is a three-dimensional drawing. The goal state is the physical blade ready for shipment. Operators could be defined as heating, forging, milling, drilling, and any other operations which could be performed in the factory under consideration. A planning system would search for a sequence of these operations which would begin with raw materials and result in the production of a turbine blade. The simplicity of this approach is deceptive. Experience has shown that process planning is more complex, requiring a great deal of product and process knowledge. One attempt at constructing a a knowledge-based system to select and sequence process steps is the Gari system,[6] which utilizes the production rule formalism for the knowledge necessary to select and sequence operations.

At present, the automation of process planning exceeds the state of the art. Nevertheless, theories and techniques continue to evolve which may prove to bring its automation closer. In particular, the theories of constraint-directed reasoning appear promising. Constraint-directed reasoning is a heuristic search technique in which domain knowledge is represented as constraints that bound and guide search. Problems of conflict among constraints are handled by relaxation, a process in which acceptable alternatives to the constraint are investigated and possibly selected.

Scheduling at the shop level

Another aspect of shop-floor decision making is scheduling, which involves selecting a sequence of operations, called a process routing, that completes an order and assigns times (for example, start and end times) and resources to each operation. An operation is an activity which defines the resources required, machine setup and run times, and labor requirements.

Historically, the scheduling problem has been divided into two separate steps. Process routing selection is typically the product of a planning procedure, and the assignment of times and resources is typically the product of scheduling. In reality, the distinction between planning and scheduling is fuzzy. The choice of routing cannot be made without generating the accompanying schedule. The admissibility of a process routing is determined by the feasibility of each selected and scheduled operation. An operation is feasible when its resource requirements are satisfied during the scheduled time of the operation. Resource requirements for an operation are determined by the operation and, in turn, by the machines that may perform the operation.

An examination of the scheduling task in a turbine component plant has shown that orders are not scheduled uniformly.[7] Consider the activities required to produce a steam turbine blade. Each blade has one or more process routings containing many operations. Alternative process routings may be as simple as substituting a different machine, or as complex as changing the manufacturing process. Furthermore, the resources needed for an operation may be needed by other operations in the shop. Each scheduling decision entails side effects, the importance of which varies by order. One factor that continually appears is the sensitivity of human schedulers to information other than due dates, process routings, and machine availability. For example, as a schedule is distributed to persons in each department in the plant, each person on the distribution list can provide information which may alter the existing schedule. We found that the scheduler was spending 10 percent to 20 percent of his time scheduling and 80 percent to 90 percent of his time communicating with other employees to determine what additional "constraints" may affect an order's schedule. These constraints include:

- **organization goals**—due date requirements, work-in-process time requirements, cost restrictions, machine utilization goals;
- **physical limitations**—machine capabilities, product size, and quality limitations;
- **casual restrictions**—precedence of operations, resource requirements to perform an operation;
- **availability**—availability of resources (such as tools, fixtures, NC programs, and operators) to perform an operation;
- **preferences**—qualitative preferences for operations, machines, and other resources.

From this analysis, we conclude that the object of scheduling is not only to meet due dates, but to satisfy the many other constraints found in various parts of the plant. Scheduling is not a distinct function, separate from the rest of the plant; it is highly connected to and depends on decisions made elsewhere in the plant. The added complexity imposed by these constraints prevents human schedulers from producing efficient schedules, as evidenced by high work-in-process, tardiness, and low machine utilization. Hence, any solution to the job-shop scheduling problem must identify the set of scheduling constraints and their effect on the scheduling process.

Management science recognized early on that job-shop scheduling is an example of a constraint satisfaction problem which could be optimally solved using mathematical programming techniques. Unfortunately, such approaches, while theoretically valid, are useless on a practical level. Job-shop scheduling is a member of the class of problems described as nondeterministic polynominal time, or NP. Their optimal solutions are too complex to be tractable. For example, the sequencing of 10 orders through five operations has $(10!)^5$ possible schedules (without gaps or alternative routings). Adding more orders, operations, and resources to the selection process multiplies the complexity of the scheduling problem. This problem of algorithmic complexity has forced a bifurcation in research objectives.

One branch of research focuses on the attainment of optimal results, but algorithmic complexity has restricted these results to the one- and two-machine cases. Moreover, the achievement of these results requires the removal of most of the constraints, so as to focus on a single criterion for measuring the schedule's efficacy.

The second branch takes a heuristic approach realized by priority dispatch rules. A dispatch rule is a local decision rule which determines the next job to be processed on a machine from the set queued at the machine. Extensive simulation analyses have shown that the weighted shortest processing time rule, or WSPT, provides reasonable schedules with respect to a single criterion such as tardiness, but ignore many of the constraints found in a typical factory.[8]

The dispatch rule approach does not provide predictive information about future operations and their machine reservations; it does not consider alternative paths; and it does not incorporate other constraints. By contrast, the nonlinear programming approach is combinatorially explosive but can handle nonlinear constraints on variables.

An alternative approach to scheduling activities in a job-shop is to apply constraint-directed reasoning. By viewing the scheduling problem from a constraint-directed reasoning perspective, much of the knowledge can be viewed as constraints on the schedule generation and selection process. For example, the next step of an operation can be viewed as a precedence constraint, and the due date for the order as a goal constraint. Adopting a heuristic search paradigm, the former constraint can be viewed as an operator which elaborates the solution space of partial schedules, while the latter is used to rate schedules in that space. The problem of scheduling orders in a job-shop under these constraints raises a number of issues of interest to the artificial intelligence community, including the following:

- knowledge representation semantics for organization modeling;
- extending knowledge representation techniques to include the variety of constraints found in the scheduling domain;
- integrating constraints into the search process, particularly determining how to use constraints to bound the generation of possible solutions and focus selection among the alternatives;
- relaxing constraints when conflict occurs; and
- analyzing the interaction between constraints to diagnose poor solutions.

Monitoring and control at the shop level

The third part of shop-floor decision making is monitoring and control of work. Decision making can be divided into two modes: predictive and reactive. Planning and scheduling are *predictive* in that they define what should happen on the factory floor. Seldom is it the case that schedules are implemented unmodified. Consider the following excerpt from an actual production report:

> *Machine A lost three days because of a broken pinion gear. Machine B lost three shifts because of ball screw and way covers. Machine C lost three shifts because of a hydraulic leak in the spindle. The new machines had more downtime in May with machine D down 27.5 hours because of the main breaker tripping and machine E losing 23.5 hours due to lube blower fault.*

The monitoring and control of work is an extension of planning and scheduling because it alters plans and schedules in reaction to changing environments. This requires the acquisition of status information from the factory floor, comparing expectations with the actual, and determining the minimum change which keeps schedules close to original schedule to maintain production stability. Any solution to this problem must include methods for identifying deviant behavior on the shop floor and methods for removing or reducing the deviation.

Monitoring requires appropriate information gathering devices, whether automatic or manual, in order to update a model of shop status. With the appropriate information, detection of unpredicted events is quite easy since only the current state of the shop needs to be compared against the predicted state as defined in the schedule. However, determining the effects of an unpredicted event can be quite difficult. For example, a machine malfunction might require only rescheduling the affected orders, while a change in a production goal or critical facility may require a total rescheduling of the shop.

One technique for determining the effect of change focuses on goal-directed rule-based processing.[9] For each category of error, a set of rules is defined which specifies the procedure to be followed. Problems lying outside of the rule set's knowledge may be detected but not acted on correctly. Another approach embeds dependency links in the shop model which define how the information was derived and what would have to be done to re-derive it. The specification of the derivation procedure defines the work required to correct the situation. This is more difficult and requires further research.

Correction is not necessarily separate from planning and scheduling. When the effect of the unpredicted event is small, correction may be performed locally by the rule base. But in more difficult cases, correction requires looping back into the planning and scheduling steps in order to effect the appropriate change.

Managing the manufacturing cell

The knowledge engineering and model requirements for a flexible manufacturing cell present other challenging problems to an autonomous system. It is commonly understood that a "blue collar" worker has very different skills from his higher level management. Unfortunately, these skills are even harder to state than his manager's skills since they are based on a kind of sensory intuition. For example, a good machinist knows that his tool is starting to wear by the sound and vibration of his machine. To further complicate matters, fault characteristics change according to part materials and cutting speeds. The best strategy for managing a situation that is poorly understood is to collect a diverse group of sensor information and correlate it with modes of failure. Researchers are attempting to find a single piece of sensory information that accurately reflects the state of a machine[10] and its associated relationship to failures. An autonomous decision-maker doesn't have this privilege in every case, since its primary goal is not to perform research but to maintain the *status quo.* In short, an autonomous system must receive relevant and timely sensory information in order to make reliable judgements; without this influx of information, true autonomy is only a dream.

The expertise of a "blue collar" foreman doesn't stop with his sensory abilities, but also includes intimate knowledge of the workings of many different machines. This knowledge allows him to operate and even fix these machines so that they perform at their maximum capacity.

Managerial knowledge. Many different machines, part orders and priorities, tool availability, and other real-world constraints also force the flexible manufacturing system to have managerial skills. Even though these skills are also augmented by factory-wide systems, many managerial decisions must be made on site and quickly. For example, a manufacturing cell may have an inspection station that runs at half the speed of the overall cell. Now suppose there was a large order that had to be filled within the day and that this wouldn't leave enough time to inspect most of the pieces. Should the system generate the parts without an acceptable level of inspection, this might lead to a whole batch of bad parts while still missing the goal. Or should the system defer this question to higher level management system, this might not have an adequate amount of information to assess the actual risk. No matter what the conclusion, the autonomous system must take a major part in the decision making since it is the sole source of feasibility information. It is quite common in "people" situations for an experienced person to say that a goal with a marginal chance of success is impossible, while a less experienced person might try to impress his boss by trying to achieve the goal while taking unacceptable risks. An autonomous system should be the experienced expert.

Communication skills. An autonomous system managing a collection of random equipment must be an expert at communication. The system's model is a source of commands to controllers and machines, each of which is often controlled in its own "home brew" language. Therefore, one of the primary responsibilities of the model is to provide enough information to perform translations into these different control languages. In addition, responses from the controllers must also be understandable so that status and error reports can be recognized and saved in a uniform representation. Outgoing commands, on the other hand, must be translated from the internal representation to the appropriate command language.

This one feature of an autonomous system is the most critical feature of factory management, at every level. This skill has, for the most part, been side-stepped by commercial suppliers (such as Kearney & Trecker and Cincinnati Milicron) of flexible manufacturing systems, because, for marketing reasons, it is in their best interest to sell whole systems. This allows them to standardize around one machine controller that is capable of controlling machines built by the same supplier. In addition, commercial systems tend to have only a "weak" link between the central computer and controller. A fully functional link can at least download and upload part programs, determine the machine's status, remotely execute programs, and execute an emergency stop. However, many systems are preprogrammed by the supplier so that all that is expected from the communications system is a "go to the next program" command. These limitations seriously restrict the flexibility of these so-called "flexible" systems. For example, a new part style may require new robot programs; yet this is considered a change in the system specification which must be carried out by the supplier at his site rather than by the user.

Flexible manufacturing relies on input from factory workers as well as input from manufacturing equipment. This implies the commands and questions that originate from a person must also be translatable into an internal form and subsequently into the appropriate machine languages. Each person and each machine can thus work in a language most appropriate for their application.

A machine operator of a manufacturing cell has only a few needs. The language {Go, Stop, Style <integer>} may very well be adequate. So when the system speaks to the operator it uses this language and understands only this language. This avoids the problem of the operator trying to ask the system to perform tasks that the system doesn't fully understand and prevents the operator from abusing his right of control over the system. On the other hand, the maintenance engineer needs to be able to ask the system about the status of every component, and he may even want to ask statistically-oriented trend questions. The project engineer needs to be able to reprogram the sequence of operations, and a plant scheduler may want to ask *"what if"* questions. Each machine, each person and each autonomous system wants and needs to speak in at least a different dialect and probably a different language.

Mechanical knowledge. Once an autonomous system has detected a mechanical failure based on its sensory input, it has several options based on the scope of the problem and its own capabilities.

(1) **Getting help.** An autonomous system knows when a problem is critical and cannot be properly solved alone. Therefore, the system must be able to contact a maintenance crew capable of performing the appropriate corrections. This process is, of course, accompanied by a diagnosis of the situation and any suggestions that are obvious to the system.

(2) **Minor adjustments and changes.** Many problems are easily solved by minor machine adjustments. For example, tool wear can be compensated for by altering the cutting speed and, eventually, a tool change. These functional capabilities must be provided to the system along with the knowledge to carry them out.

Once the system has additional help from an outside source (such as a maintenance engineer or vision system), it must be able to quickly provide supporting information to the coworker so that he can effectively help. This process of communicating with the coworker may also involve updating the state of the system's database.

Reacting to errors in a manufacturing cell

Errors in the manufacturing process are not the exception but the rule. Machines change over their lifetime due to wear and, as a result, the programs that operate them must adjust to the new conditions. In the extreme, machines wear out and break, causing an interruption in service and possibly a dangerous interaction with another machine tool. These errors are common and must be accounted for directly and acted on in a way that causes the minimum amount of damage and downtime. The first step to dealing with manufacturing errors is to understand the range of errors that are likely to occur and what the reasonable corrective actions might be.

Programming errors. This kind of error is fairly new to manufacturing, but it too must be anticipated so that reasonable actions can be set up. Suppose a communication driver mulfunctioned because someone neglected to handle negative numbers coming back from a machine controller. At least the system should announce the problem to the cell operator and *stop* any other commands from being executed. An even more reasonable approach would be to automatically restart another image of the communication driver, get status from the machine controller and, if possible, send the machine to a safe home position. No systems have dealt with these issues in manufacturing. However, they have been considered in related systems such as the space shuttle and in computer operating systems which restart automatically on failure.

Communication errors. Most manufacturing environments are electrically noisy, and despite precautions, there often are many transmission errors to individual machines. The machine controllers should be smart enough to notice the error and not set out on a random task. Instead, information should be sent back to the host computer advising it of the error and requesting another transmission. Communication errors are not always really transmission errors, since they might have been caused by a programming mistake or a more serious hardware error. In any event, these errors should be logged so that they can be tracked to the cause.

Hardware errors. Hardware errors often are the most difficult errors to prevent since they can be very subtle and totally catastrophic. Imagine a robot gripper that fails to open after it has loaded a machine tool.* Once

*Gripper problems are all too common because grippers are usually built from a custom design. For example, one gripper was designed to have less strength opening than closing; it made the final gripper more compact. Unfortunately, the gripper was supposed to handle hot parts (2200°F) that are in fact quite sticky. The net result was that the gripper would often get stuck to the part and would fail to open; it needs more strength for opening than closing.

the first error has happened, it can easily escalate into more serious errors. For example, if the robot tries to pull away from the machine tool it could rip off the gripper. Since the machine tool has hold of the part, it might think that it could begin operation with the robot still in its midst. Obviously, a quick succession of catastrophic failures would follow. To avoid this error, the robot must first notice the error and abort further actions. The robot must "calmly" inform the host of the error and provide a reasonable description of its nature. Finally, the host could plan a way for the robot to extricate itself from the problem (retry gripper open), but it *must* tell the other machines to stop immediately. The first error is rarely catastrophic unless second and third errors allowed.

Manufacturing problems. Many manufacturing problems are not errors at all, but rather a natural part of the process. For example, the hammers in an open-die forge are in a constant state of wear, hitting the billets with 600-ton blows. There is really no excuse for letting the hammers get out of position and making bad parts just because of their wear. There are two general approaches to dealing with manufacturing errors. The part and hammers could be monitored constantly (online) and have the forge automatically screw down the hammers as their position drifts out of range. Alternatively, the measurement could be postponed until one part is made and then feed back corrective parameters to the forge's controller. The first approach is preferable since errors can be detected much faster. Unfortunately, it is often impossible because of the extreme conditions involved in the process.

Completely avoiding errors, or completely negating their effects, is rarely possible. However, there are general principles that can be followed to manage 99 percent of the errors.

(1) **Closed loops.** Each command should have a sensor which verifies that it has been completed successfully.
(2) **Centralized data.** A centralized database avoids the computational problems of maintaining a consistent distributed system.
(3) **Redundant emergency systems.** There should be mechanical systems that can back equipment out of catastrophic situations (i.e. the robot out of the furnace).

Of course, the best way to react to errors is to try and avoid them in the first place. One preventative measure is to schedule robot movements so that they are never on collision paths (see Figure 2). This is accomplished by defining a robot movement as a discrete interval (for example, moving from position A to position B) that is intersected with the *interval of conflict*; whenever the intersection is not nil, only one of the robots is allowed to proceed.

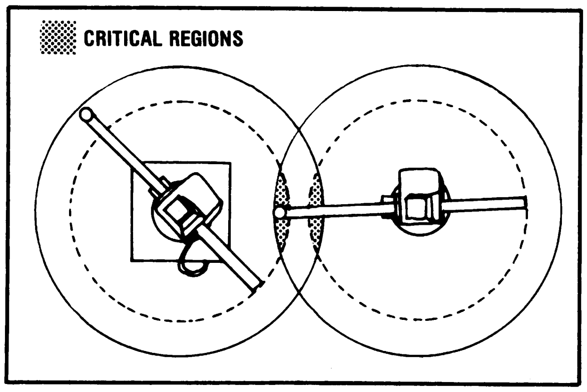


Figure 2. Two robots in conflict.

Systems for shop and cell level production management

In the preceding sections, issues of modeling, planning, scheduling, and reacting have been raised, and possible solutions have been described. In this section, two systems are described, Isis and Transcell, which embody some of these techniques in order to perform production management at the shop and cell level.

Isis: a production management system. The Isis system[4,7] is an artificial intelligence, constraint-directed reasoning system which addresses the problem of how to construct accurate, timely, realizable schedules and manage their use in job-shop environments.

Isis constructs schedules by performing a hierarchical, constraint-directed search in the space of alternative schedules (see Figure 3). The search is divided into four levels: order selection, capacity analysis, resource analysis, resource assignment. Each level is composed of three phases: a presearch analysis phase which constructs the problem, a search phase which solves the problem, and a post-search analysis phase which determines the acceptability of the solution. In each phase, Isis uses constraints to bound, guide, and analyze the search.

Level 1 selects an order to be scheduled according to a prioritization algorithm based on the category of the order and its due date. It outputs a prioritized list of orders to be scheduled.

Level 2 performs an analysis of the plant based on current capacity constraints. It determines the earliest start time and latest finish time for each operation of the selected order, as bounded by the order's start and due date. The times generated at this level are codified as operation time-bound constraints which control the start and end times of operations at the next level.

Level 3 selects the resources necessary to produce an order. Presearch analysis begins by examining the constraints associated with the order to determine the scheduling direction (forward from the start date versus backward from the due date) to determine whether any constraints are missing and should be created (for example, due dates or work-in-process) and to determine the set of search operators which will generate the search space. A beam search is then performed with the selected set of search operators. The search space is composed of states which represent partial schedules. The application of an operator to a state results in the creation of new states which are more complete. Starting with a null schedule, alternative partial schedules are generated either forward from the start date or backward from the due date. An operation operator generates alternative states which represent alternative operations in either the forward or backward direction.

Once the operation is known for a state, other operators extend the search by creating new states which bind the machine and/or the execution time of the operation. A variety of alternatives exist for each type of operator. For example, two operators have been tested for choosing the execution time of an operation. The "eager reserver" operator chooses the earliest possible reservation for the operation's required resources, and the "wait and see" operator tentatively reserves as much time as available, leaving the final decision to level 4. This enables the adjustment of reservations in order to reduce work-in-process time. Alternative resources are generated (such as tools or materials) by other operators. Each state in the search space is rated by the set of constraints found (resolved) to be relevant to the state and its ancestors. Constraints defined to be in the set are those which are attached to any resource (such as a machine, tool, or order) specified by the state. Each constraint assigns a utility between zero and two to a state; zero signifies that the state is not admissible, one signifies indifference, two maximal support. The rating of a state with multiple constraints is the average of the constituent constraints, each weighted by its importance. The importance of a constraint is defined statically or derived dynamically according to goal information.

Once a set of candidate schedules have been generated, a rule-based, post-search analysis examines the candidates to determine if one is acceptable. Currently, any schedule with a rating greater than one is accepted. If no acceptable orders are found, then diagnosis is performed. First, the schedules are examined to determine a type of scheduling error. The error is then fed back to preanalysis in order to select new operators which are used to reschedule the same order. The diagnosis of poor solutions caused by constraint satisfaction decisions made at another level can be performed by analyzing the interaction relations linking constraints. A poor constraint decision at a higher level can be determined by the utilities of constraints affected by it at a lower level, and an alternative value can be chosen.

Level 3 outputs reservation time bounds for each resource required for the operations in the chosen schedule. Level 4 (Figure 3) selects the actual reservations for the resources required by the selected operations which minimize the work-in-process time.

The scheduling of Isis is also reactive. The invalidation of reservations by actions, such as machine breakdowns or other orders taking too long on a machine, results in a minimal rescheduling of only the affected orders, while attempting to maintain previous reservations. Isis's scheduling is also suggestive. If constraints cannot be met, it attempts to generate a schedule which satisfies as many constraints as possible. For example, if the due date of an order cannot be met by backwards scheduling, it attempts to schedule in the forward direction and suggests an alternative due date.

The contribution Isis makes to the job-shop planning and scheduling problem is its focus on the representation, utilization, and relaxation of constraints in the scheduling process. Isis's knowledge representation language, SRL-2[11] can represent an extensive set of constraints and their relaxations. Categories of constraints which Isis covers include organizational goals (for example, due dates, cost, quality), preferences (for example, for machines), enabling states (for example, resources, previous operations), physical characteristics (for example, accuracy, size), and availability (for example, existing reservations for tools). Isis uses a constraint-directed search paradigm to solve the scheduling problem. Isis provides

- a knowledge representation language SRL for modeling organizations and their constraints;
- hierarchical, constraint-directed scheduling of orders, which includes constraint-directed bounding of the solution space, context-sensitive selection of constraints, and weighted interpretation of constraints;
- analytic and generative constraint relaxation; and
- techniques for the diagnosis of poor schedules.

Due to the conflicting nature of certain constraints (such as cost versus quality), there may not be a schedule which satisfies all of them. Hence, Isis considers relaxations of such constraints when generating and selecting schedules.

Isis performs a limited amount of process planning. The constraint set utilized in Isis includes constraints on the physical features of a product (for example, size and form) and the ability of machines to produce them. During the constraint-directed search, infeasible process routings are removed by these constraints.

As schedules are implemented on the shop floor, unpredicted events are detected by comparing actual to predicted job and resource status. When a deviation is detected, a complete rescheduling is not performed. Instead, only the affected jobs are rescheduled, with an attempt to minimize their variation from their earlier schedule. Isis is the first system to consider the myriad of constraints found in job shops and generate schedules which meet them or their relaxations in polynomial time.

Transcell: autonomy in heavy metal working. Imagine a factory filled with groups of machines built from different technologies and operating with different kinds of controllers: The result is chaos, the current state of manufacturing. A generic operating and database system that *reacts* to this overwhelming problem has been under development for several years.[12] Transcell (for Transportable Cell) is a system designed to manage a wide variety of machines, all of which communicate in different languages, and keep them coordinated and operating without error.

The operating system has three levels, each of which performs a different function (see Figure 4). The project engineer describes his manufacturing problem in a very high level, rule-based language (level 1). Each rule describes one set of actions that has a set of preconditions and can be executed independently from other rules. The system scans the list of rules for the set which is satisfied by the current state of the database model (level 2) and executes them, resulting in a change of state in the model. The database system's major responsibility is to maintain the consistency of the database. This often causes the system to generate a command that is sent (level 3) to the appropriate controller, which in turn forces the controller and its machine tool to change state. After the first round of instructions have been sent out to the machines on the floor, they perform their work and start to send back completion messages to the central machine. These new messages represent a state change in the cell that makes it possible for more rules to be activated. This flow of rule executions, output messages and completion results is a "dispatch" schedule of activities for machines (for example, when the machines are ready to do work, and the work is ready to be done, it is done). This kind of schedule is only optimal under very strong conditions; however, they are usually satisfied in a cell environment. A more complicated schedule would be very difficult to complete within the real-time constraints of cell work.

Suppose the project engineer wanted to lower a furnace's temperature during a long vacation so that the plant would conserve energy. The rule that would cause this to happen would *not* change the current temperature directly; it would change the "expected" temperature in the database. Consistency rules in the database are triggered as a result, and commands are automatically sent to the furnace only when the requested change is feasible. The discrepancy in the desire of the project engineer and the subsequent actions of the machines mandates the two levels of control, the third being the actual communication to the controllers.

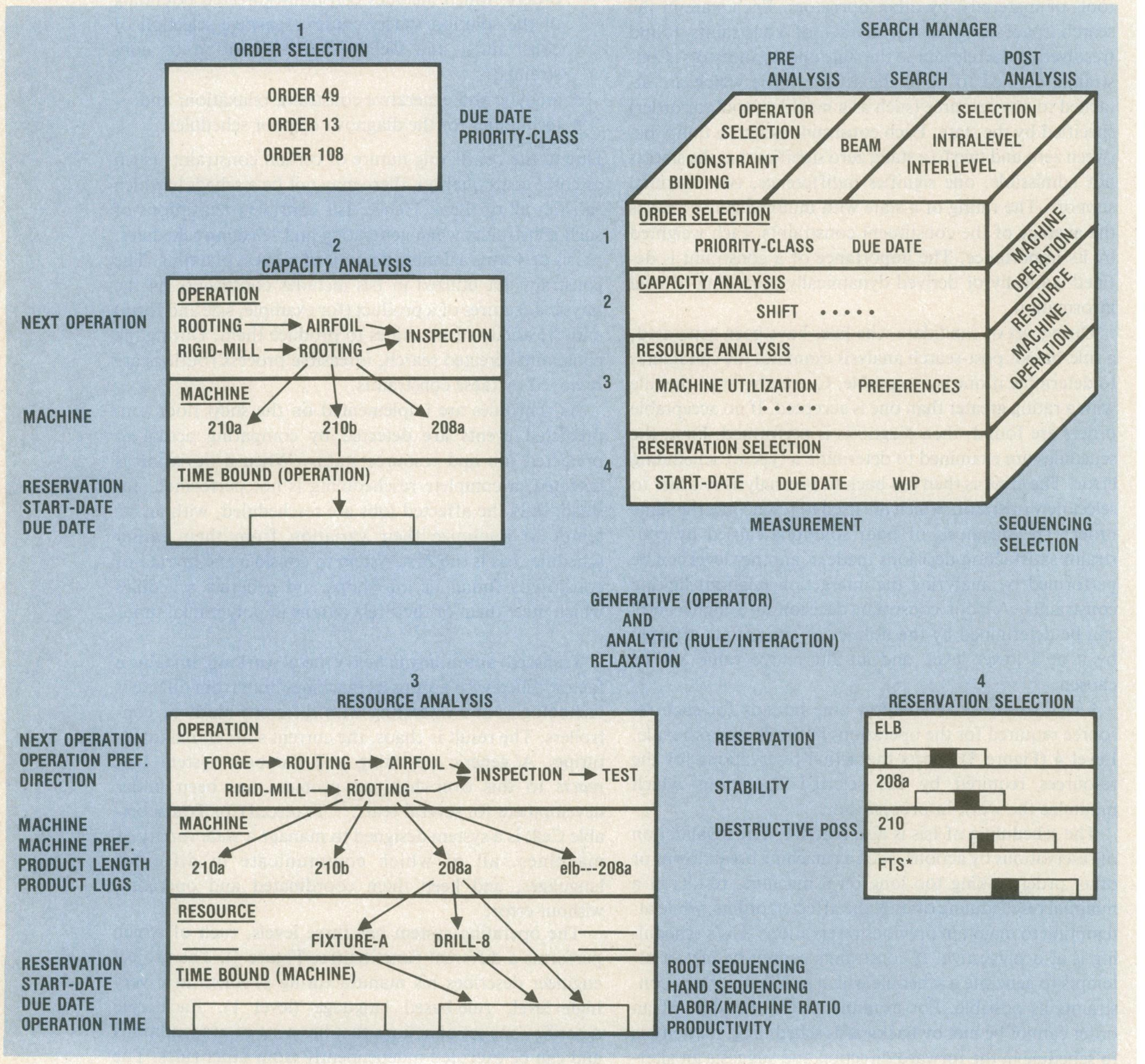


Figure 3. ISIS system architecture.

Transcell's first application is to control a manufacturing cell that manufactures preforms for turbine blades at a Westinghouse Electric Corporation turbine components plant (see Figure 5). The cell is the first major step in the manufacturing process and makes preforms by open die forging. Metaphorically, this cell is an automated blacksmith. It is composed of nine basic parts. There is a vision system (1) for locating billets, a robot (2) for transferring very hot parts and reaching into a rotary furnace (3) that operates around 2200°F. From the furnace, the robot (robot *A* in Figure 5) loads an open die forge (4) which literally beats a cylindrical shape into an elongated shape that approximates the final turbine blade geometry (this open die forge consists of hammers and two chucks, as shown in Figure 5). After the part formation, a second robot (5) moves the part to a gage station (6) that reconstructs a three-dimensional model of the part using a second vision system (7). The gaging results are used to make corrective actions in the part program for the open die forge and furnace. The second robot (robot *B* in Figure 5) moves the part to a cropper (8) that trims the excess material from the preform, and then to a stamper (9) that punches a batch code into the surface of the part, and finally to a rack, which is destined for the next cell. The cell is controlled by a star-shaped computer network with a Vax 11/750 at the center and four different kinds of machine controllers at the tips of the star, one per machine. The cell is monitored by 150 sensors, which are strategically located to detect error conditions in equipment before they cause real problems.

We have not been able to achieve complete autonomy in this cell because of the extreme dangers inherent in its failure. However, the system does effectively reduce setup times, increase machine utilization, and nominalize the

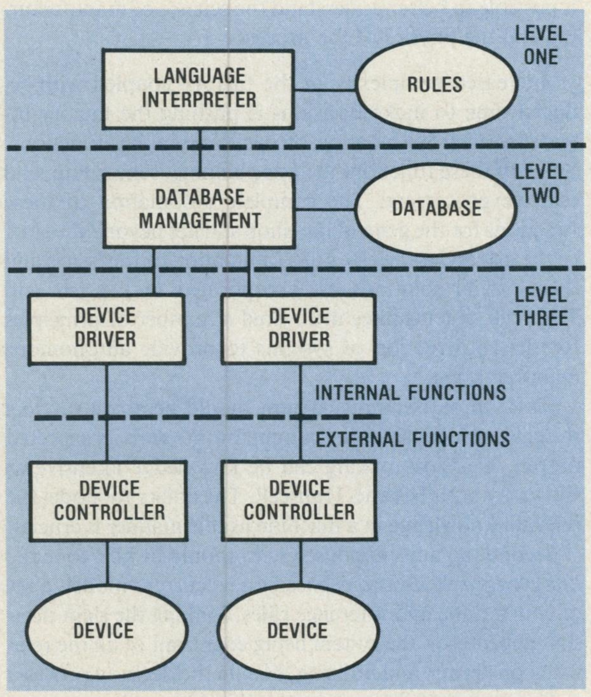


Figure 4. Three levels of control with the cell supervisor.

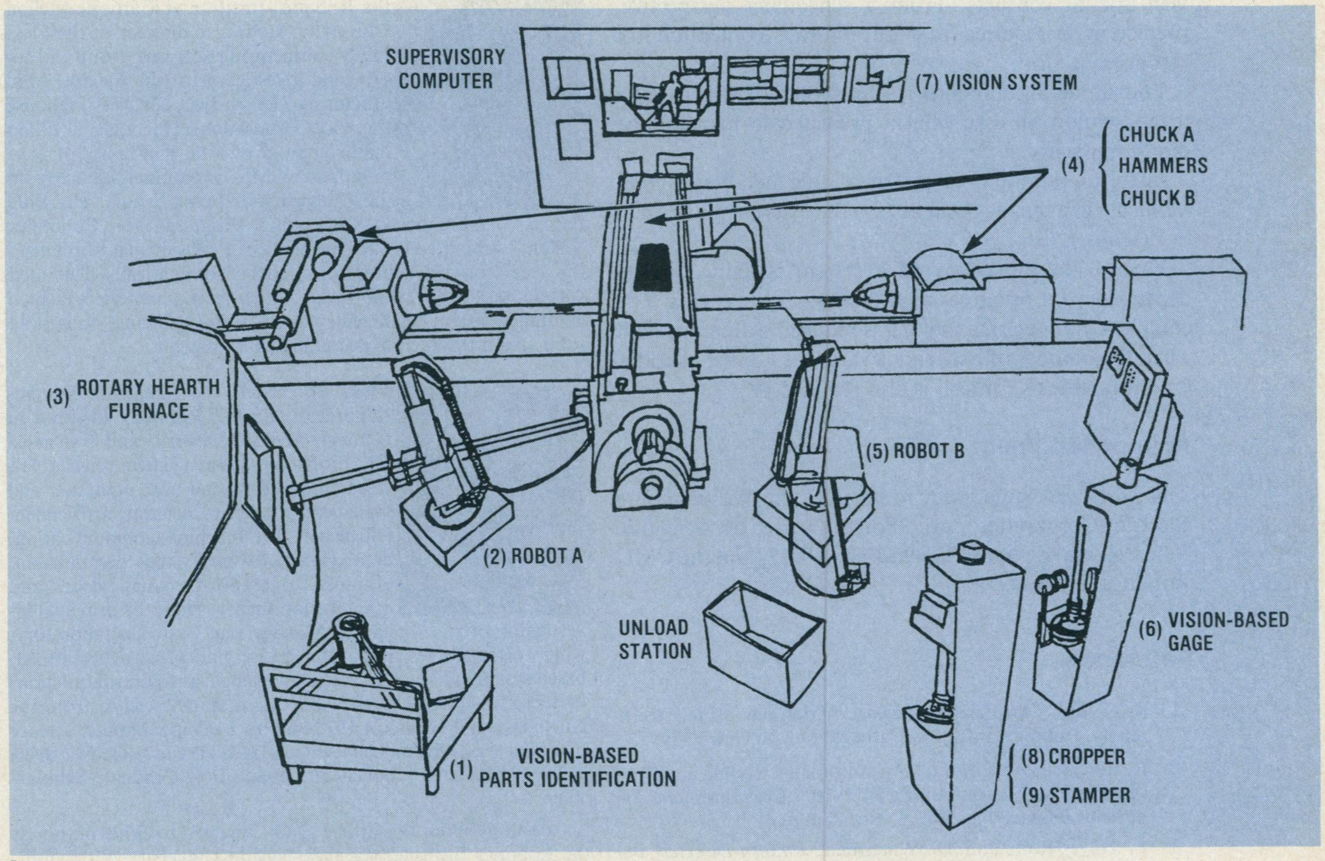


Figure 5. The initial test site for Transcell.

number of bad parts produced. However, by fall 1984 we are going to try our first experiments in "transporting" Transcell to a new control environment.

The biggest advantage in trying to build a completely autonomous system is the production of a tool that makes it possible to better understand the science of manufacturing and the physics of the process.

Increased complexity in the factory coupled with reduced time to make decisions is pushing the factory towards greater automation of decision-making functions. Some of these functions include planning, scheduling, and reactive processing. The complete automation of these functions for the general job-shop still lies beyond the state of the art. Nevertheless, our examination of the issues and their partial solutions as incorporated in the Isis and Transcell systems have uncovered a number of principles for the construction of systems to provide autonomous manufacturing.

First, an autonomous system should be able to *model* the environment at a level commensurate with its expected output. Decision making can be knowledge intensive as shown by both Isis and Transcell. The ability to model the relevant knowledge in a machine usable manner is crucial.

Second, an autonomous system should be able to make *knowledgeable decisions* based on its current model, a set of constraints, and inference rules. Making the right decision depends on the system being cognizant of all the relevant constraints and utilizing them in the decision process.

Third, an autonomous system should be able to *predict* what is going to happen next. The ability to predict events is very important in the manufacturing cell in order to prevent spatial conflicts. From a scheduling perspective, prediction is required to reduce future contention for resources.

Fourth, an autonomous system should be able to *react* to a situation, in case what it predicted to happen next didn't happen.

Fifth, an autonomous system should be able to *communicate* in a language suitable to the external party (man or machine).

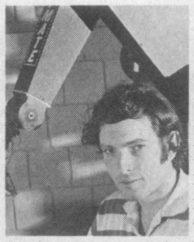
Finally, the autonomy of a system is limited by its dependency on resources provided by other systems. The complexity of decision making is reduced within a system when the number of exogenous variables it must consider (and is unable to control) is also reduced. ✻

Acknowledgment

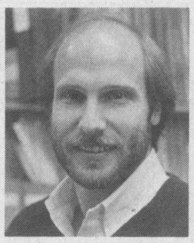
This work was supported, in part, by the Westinghouse Electric Corporation, Air Force Office of Scientific Research under contract F49620-82-K0017, and the CMU Robotics Institute.

References

1. M. S. Fox, "Artificial Intelligence in Manufacturing," tech. report, Robotics Institute, Carnegie-Mellon University.
2. C. Emerson and I. Ham, "An Automated Coding and Process Planning System Using a PDP-10." *Computers and Industrial Engineering*, Vol. 6, No. 2, 1982.
3. F. Hayes-Roth and D. Waterman, *Pattern-Directed Inference Systems*, Academic Press, New York, 1978.
4. M. S. Fox, "Constraint-Directed Search: A Case Study of Job-Shop Scheduling," PhD Thesis, tech. report, CMU-RI-TR-83-22, Robotics Institute, Carnegie-Mellon University, 1983.
5. A. Newell and H. A. Simon, "GPS: A Program that Simulates Human Thought," *Computers and Thought*, Eds. E. Feigenbaum and J. Feldman, McGraw-Hill, New York, 1963.
6. Y. Descotte and J. C. Latombe, "GARI: A Problem Solver That Plans How to Machine Mechanical Parts," *Proc. Seventh Int'l Joint Conf. Artificial Intelligence*, Vancouver B.C., 1981.
7. M. S. Fox, B. Allen, S. Smith, and G. Strohm, "ISIS: A Constraint-Directed Search Approach to Job-Shop Scheduling," *Proc. IEEE-CS. Conf. Trends and Applications*, National Bureau of Standards, Washington, D. C., tech. report, CMU-RI-TR-83-3, Robotics Institute, Carnegie-Mellon University, 1983.
8. R. W. Conway, "Priority Dispatching And Job Lateness in a Job Shop." *J. Industrial Engineering*, Vol. 16, 1965.
9. P. Haley and J. McDermott, "PTRANS: A Rule-Based Management Assistant." tech. report, Computer Science Dept., Carnegie-Mellon University, 1984.
10. D. W. Yen and P. K. Wright, "Adaptive Control in Machining – A New Approach Based on the Physical Constraints of Tool Wear Mechanisms," *J. Engineering for Industry*, ASME, Vol. 105, No. , 1983.
11. J. M. Wright, M. S. Fox, and D. Adam, "SRL-2 Users Manual", tech. report, report no. , Robotics Institute, Carnegie-Mellon University, 1984.
12. D. A. Bourne et al., "A Flexible Manufacturing Cell for Swaging." *Mechanical Engineering* - Comm. of ASME, Oct. 1982.



David A. Bourne is a research scientist in the Robotics Institute at Carnegie-Mellon University. He is also director of the Flexible Manufacturing Software group and co-principal investigator of the Flexible Manufacturing Cell project for Westinghouse Electric Corporation. He holds a BS in mathematics from the University of Vermont and an MS in computer science from the University of Pennsylvania. His PhD thesis, "Automatically Generating Programs for Computer Vision," submitted to the University of Pennsylvania, formalizes the basis for which programs can be automatically generated without task specific knowledge. Other research interests include computer vision, intelligent manufacturing systems, household robots, and flexible programming environments.



Mark Fox heads the Intelligent Systems Laboratory of the Robotics Institute at Carnegie-Mellon University and is an assistant professor of management science. His research interests span both computer and management science, including artificial intelligence, man-machine communication, databases, software system organization, graphics, operations management, and organization theory. He is the principal investigator of the Intelligent Management Systems Laboratory, which performs research in the design and construction of AI-based systems to automate the management and control functions of factories. Fox received a BSc in computer science from the University of Toronto in 1975 and his PhD in computer science from Carnegie-Mellon University in 1983. He was twice awarded a National Research Council of Canada Post-Graduate Scholarship.

Questions about this article can be directed to either author at the Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA 15213.