The International Journal of Flexible Manufacturing Systems, 5 (1993): 53–88
© 1993 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Developing Manufacturing Control Software: A Survey and Critique

JARIR K. CHAAR
IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598

DANIEL TEICHROEW
IOE Department, The University of Michigan, Ann Arbor, MI 48109-2117

RICHARD A. VOLZ
CS Department, Texas A&M University, College Station, TX 77843-3112

Abstract. The complexity and diversity of manufacturing software and the need to adapt this software to the frequent changes in the production requirements necessitate the use of a systematic approach to developing this software. The software life-cycle model (Royce, 1970) that consists of specifying the requirements of a software system, designing, implementing, testing, and evolving this software can be followed when developing large portions of manufacturing software. However, the presence of hardware devices in these systems and the high costs of acquiring and operating hardware devices further complicate the manufacturing software development process and require that the functionality of this software be extended to incorporate simulation and prototyping.

This paper reviews recent methods for planning, scheduling, simulating, and monitoring the operation of manufacturing systems. A synopsis of the approaches to designing and implementing the real-time control software of these systems is presented. It is concluded that current methodologies support, in a very restricted sense, these planning, scheduling, and monitoring activities, and that enhanced performance can be achieved via an integrated approach.

Key Words: manufacturing control software, software methodologies, planning, scheduling, monitoring, simulation

1. Introduction

The evolution of manufacturing software has always been closely linked to the technological advances in manufacturing and computer hardware. Consequently, the integration of manufacturing devices has been preceded by the automation of these devices: supplying them with numerical control units and interfacing them to appropriate software libraries and databases.

Traditionally, the sequence of operations executed by the control software of manufacturing components has been captured by ladder logic diagrams (see, for example, Bollinger and Duffie 1988; Elsenbrown 1988; Gayman 1988). These diagrams specify the input and output procedures of the Programmable Logic Controller (PLC) that drives and cycles other operations of a manufacturing device. All possible combinations of PLC inputs must be captured by the ladder diagram. Consequently, these diagrams grow so complex that locating the cause when a problem is detected becomes extremely difficult.

Stecke (1985) outlines the various design, planning, scheduling, and control problems that need to be addressed at the different stages of the life cycle of a flexible manufacturing system. Design problems include determining the type and number of machine tools, the capacity of the material handling system, and the size of the buffers of a system. Planning problems include deriving the process plans of manufacturing jobs, allocating pallets and fixtures, and assigning the appropriate sets of machine tools to these jobs. Scheduling problems include determining the optimal input sequence of jobs and machine tool usages for a given job mix. Control problems are concerned with monitoring the system to ensure that production requirements and due dates are met, and that unreliability problems are handled.

To aid in these processes, software packages for handling the various aspects of modern manufacturing such as Computer-Aided Engineering (CAE), Computer-Aided Design (CAD), Computer-Aided Manufacturing (CAM), Computer-Aided Process Planning (CAPP), Material Requirement Planning (MRP), and Manufacturing Resource Planning (MRP II) packages have been created.

Recent activities in the manufacturing systems area attempt to integrate such packages into flexible and modular *Computer-Integrated Manufacturing* (CIM) systems (Savolainen, 1988). The task of integrating the various devices of a manufacturing system requires adequate hardware support (e.g., computer networks) and adequate communication software support (e.g., communication protocols). Furthermore, additional software is required for coordinating the operations of these various devices and for implementing the control strategies of their integrated systems.

Considerable effort is also being spent in studying manufacturing as an application domain and in developing architectures, frameworks, and reference models for manufacturing systems (Iscoe et al. 1991). This work is being carried out in particular under the auspices of ISO and CAMI. An in-depth analysis of the potential of this effort is outside the scope of this paper. Other standardization efforts include network protocols such as MAP and TCP/IP and programming languages and life-cycle methodologies such as IDEF. All of these activities are outside the scope of the paper.

The increasing use of industrial general-purpose computers in controlling modern manufacturing systems has also lead to a search for alternatives to ladder logic diagrams. The adoption of the Petri net-based function charts as a standard for describing the control logic of manufacturing devices is considered a major step in this direction (International Electrotechnical Commission, 1984).

In the research domain, manufacturing control software has been modeled as a variant of Petri nets and no distinction between planning, scheduling, and monitoring is made. A restricted version of planning is performed by the software engineer when designing the Petri net sequence of operations for processing a job in a manufacturing system. The set of execution rules of Petri nets provide a scheduling strategy for processing a manufacturing job. Monitoring is implicitly performed by checking for a specified set of conditions prior to the execution of the operations of the net. However, capturing all the possible conditions of a manufacturing system in a Petri net can result in very complicated diagrams of the control software of this system.

This paper reviews some of the recent software development activities in computer-integrated manufacturing. In particular, recent attempts to design and implement software algorithms for process planning, cyclic scheduling, simulating, and monitoring the operations of manufacturing systems are discussed. The design and implementation of the control software of these systems has proven to be very complex and costly (Ayres 1989). Therefore, a set of approaches for developing this real-time control software have been researched and implemented. This paper presents a synopsis of these approaches, and concludes that current methodologies support planning, scheduling, and monitoring the operations of manufacturing systems only in a very restricted sense.

The approaches have been grouped by the functional areas of process planning, scheduling, simulation, and monitoring. However, in practice, many of these approaches overlap and may cover more than one area. The survey covers only a selected set of generic approaches to developing manufacturing control software. Hence, the functional details of many commercial tools for developing manufacturing control software, such as CIMPLICITY by GEF, Plantworks by IBM, and NAS by DEC, are not included.

Section 2 reviews some approaches to the process planning problem of manufacturing software. Section 3 presents attempts to solve the cyclic job-shop scheduling problem. Section 4 discusses the vital role of simulation in modern manufacturing, and section 5 stresses the importance of monitoring the operations of such manufacturing systems. Attempts to integrate the process planning and scheduling activities of manufacturing software are outlined in section 6. This is followed, in section 7, by a detailed discussion of a selected set of approaches to developing manufacturing control software; these approaches integrate, in a strict sense, the functions of planning, scheduling, and monitoring of this software. Section 8 concludes with a summary and critique of these approaches and introduces an approach to developing the control software of efficient and dependable manufacturing systems.

When designing efficient and dependable manufacturing systems, an integrated approach to the planning, scheduling, and monitoring activities of these systems is required. Efficiency is the acheivement of higher throughput in a system while meeting all deadlines. Dependability is the ability to detect and correct faults, such as hardware failures and their effects, such as the divergence of actual operations from planned operations. A methodology for developing the control software of such systems has been developed by Chaar et al. (1990, 1991). This methodology supports the planning, scheduling, monitoring, and failure recovery activities for manufacturing systems, in addition to the traditional activities of the software life-cycle. The simulation and prototyping phases are also supported by the methodology. The planning, scheduling, and monitoring aspects of the methodology are assisted by a set of software tools. These tools are integrated in a Computer-Aided Software Engineering (CASE) environment for manufacturing software.

2. Process Planning

In a discrete parts manufacturing system, process planning generates a sequence of operations for processing a job by the system (Allen 1987). This job may produce a single part, a pallet of many parts, or a mechanical assembly of parts, and a complex process in a process manufacturing system. Process planning is a special case of planning in Artificial Intelligence. It results in a sequence of operations that is labeled a *process plan* and that can be automated.

The major issues involved in the general planning problem are summarized by Georgeff (1987), and the use of logic and deduction in solving this problem is thoroughly discussed by McDermott (1987). This section reports attempts to automate process planning in manufacturing systems. The major features of STRIPS, a planner that forms the basis of many recent planning approaches are also presented.

Preiss and Shai (1989) use the logic programming language Prolog in deriving the process plans of jobs in a manufacturing system. The structure of the system is described in terms of a set of Prolog facts, and the behavior of this system is described in terms of Prolog rules. The goal state of the system is specified for the Prolog interpreter, which attempts to prove this goal state. If a series of facts and rules that lead from the initial state of the system to its goal state can be found, this series forms a process plan of the system. A set of heuristics for speeding-up the proof process can be implemented as Prolog procedures.

Fikes and Nilsson (1971) proposed a planning system that attempts to find a sequence of operators that transform a system from an initial state to a final (goal) state. This planning system is known as the Stanford Research Institute Problem Solver (STRIPS) system, and represents explicitly the preconditions of operators and their effects on the state of the system. The state of a system is represented by a set of first-order predicates. The preconditions of an operator form a first-order predicate calculus formula, and the effect of this operator on the state of the system is captured by an add list and a delete list. When the operator is invoked, the predicates of the add list will be satisfied by the state of the system while the predicates of the delete list will not.

An operator of a plan generated by STRIPS reduces the gap between the current state of a system and its goal state. STRIPS has a linear planning system because of the linear ordering imposed on the operators of the generated plan. A mechanism for reusing portions of the plan sequences of STRIPS has been implemented by Fikes et al. (1971, 1972). Moreover, Sacerdoti (1974) extended the STRIPS system to handle planning in a hierarchy of abstract problem spaces; the extended system has been called ABSTRIPS. At the top of the hierarchy, ABSTRIPS generates a plan using more abstract (simplified) operators. The abstract plan is refined through successively less abstract problem spaces until a plan is formulated in the original problem space. Operators are simplified by omitting preconditions. The operators at the highest (more abstract) level have the fewest preconditions. The plan is refined by restoring preconditions to the problem spaces and solving the subproblems corresponding to the added preconditions.

Descotte and Latombe (1985) designed and implemented GARI, a system that generates the process plans used in manufacturing parts from the drawings of these parts. GARI consists of an expert knowledge base and a planner. Knowledge is coded as a set of production rules called *manufacturing rules*. The syntax of such rules is identical to that of STRIPS operators. The left-hand side of a rule consists of conditions about the part to be manufactured, the available machines, and/or the machining plan. The right-hand side contains *pieces of advice* representing technological and economical preferences. Each piece of advice is weighted according to its importance in the derived process plan of a part.

The GARI planner operates by iteratively constraining a loosely constrained initial plan built from the model of the part to be machined. If an inconsistent (i.e., overconstrained) plan is produced, then the planner makes use of a selective backtracking procedure, which reasons about the weights attached to the applied constraints, for deciding which compromise is best. The above strategy is adopted because process planning, in general, requires taking into account a great variety of pieces of knowledge. These pieces may not be pure constraints but rather preferences among which compromise is necessary.

A classification of plan sequences and an approach to the generation of linear and monotone plans for mechanical assemblies has been proposed by Wolter (1989). These plans are linear because their actions allow for moving only one part of the assembly at a time, and are monotone because parts are always moved to their final positions. The approach starts by proposing a set of insertion trajectories for each part in the assembly. Next, for each trajectory, the set of parts that would block the trajectory if they were already in their final positions is determined. A valid assembly plan is derived by selecting one trajectory for each part, and ordering these trajectories such that none of the previous constraints is violated.

A hierarchical process planning system that decides the sequence of machining operations that must be performed when manufacturing metal parts has been designed by Nau (1987). In analogy with ABSTRIPS, Semi-Intelligent Process Selector (SIPS) presents its knowledge in the form of a hierarchy of processes. Each process in the hierarchy is represented by a frame and may consist of one or more machining operations. A frame specifies the relevance and cost of a process, any restrictions on the application of this process, and any operations that must be performed prior to executing the process. Backward search is used to generate the sequence of machining processes and their operations from the ultimate goal to be achieved by the system. This sequence of operations represents a path of the search tree that has both minimal processing cost and does not violate any constraints on the processing sequence of the operations of this system.

Agnetis et al. (1990) and Arbib et al. (1991) determine the routes to be followed by parts in a manufacturing system from their process plans. The approach assumes that a process plan is used to capture the operations of a job and their precedence in a tree whose leaves represent the components loading and whose root represents the final product unloading. The routing problem is then treated as a partitioning problem on the nodes of the process plan tree with the objective of balancing machine workloads and minimizing part transfers in the system.

A CAD system, a process planner, and a CNC program generator have been integrated by Hancock (1988), and used in a sheet metal processing application. Moreover, a knowledge-based approach to the automation of process planning has been researched by Wang and Wysk (1988). This approach captures the decisions of the process planner in a set of rules, and accesses a CAD system when executing these rules.

Jones et al. (1987) group the standards, codes, and existing procedures used in metal welding into a set of databases that are accessed by a set of expert systems when planning a sequence of metal welding steps. Consequently, a combination of multiple disjoint but mutually dependent expert systems cooperate to solve a larger problem. Each expert system knows how to perform its task, and a top-level module must coordinate the problem-solving activity by calling on the individual expert system modules when their capabilities are needed.

3. Cyclic Job Shop Scheduling

The efficient use of machines in a manufacturing system hinges upon the existence of a timely source of inputs to the system and a scheduling strategy for effectively sequencing the flow of these inputs through the system. The process of developing scheduling strategies that avoid internal conflicts and accommodate high input rates is referred to as the *shop scheduling problem*. Three classes of shop scheduling problems have been identified in the literature (Garey and Johnson 1979); they are the open shop scheduling problem (see, for example, Gonzalez and Sahni 1976), the flow shop scheduling problem (see, for example, Garey et al. 1976; Lenstra et al. 1977; Sahni and Cho 1977; Papadimitriou and Kanellakis 1980; McCormick et al. 1986) and the job shop scheduling problem (see, for example, Garey et al. 1976; Graves et al. 1983).

In open shops, each job has to be independently processed once by each machine with no constraints on the order in which operations are performed on a particular job. Therefore, an open shop schedule must determine not only the time at which each job enters the system, but also the order in which each individual job is routed among the machines. For flow shops, however, all jobs follow the same routing and each job is processed exactly once by each machine. Therefore, the flow shop schedule needs to determine only the time at which each job enters the system.

The classical job shop scheduling problem differs from both open shop and flow shop problems in one important respect: the flow of work is not unidirectional (i.e., an individual job may revisit some machines several times and may not visit others at all). *Cyclic job shop scheduling* is a special case of the general job shop scheduling problem in which the time between successive job initiations forms a periodic sequence. This section presents some of the most recent attempts to solve the cyclic job shop scheduling problem when a set of identical jobs or a mix of jobs of several kinds is processed by a manufacturing system. Optimum cyclic schedules are easier to find and control than optimum general schedules. Their optimality is asymptotically achieved as the number of their jobs grows.

Subramanyam and Askin (1986) propose an expert systems approach to scheduling in flexible manufacturing systems. A set of rules is proposed for initiating the processing of jobs in these systems based on machine utilization, number of jobs in the queue of a machine, and the due-dates of these jobs. The use of expert systems in scheduling flexible assembly cells has also been studied by Van Brussel et al. (1990).

A heuristic method of initiating the processing of jobs in a flexible manufacturing system with the aim of minimizing the processing time of these jobs is proposed by Liu and Labetoulle (1988). The time interval that each job requires for its execution is calculated. The processing of the job with the shortest time interval is then initiated on the machine with the lowest utilization that can start this processing (i.e., shortest job first scheduling). Operation precedence and machine utilization are used next in completing the processing of initiated jobs.

Stecke and Solberg (1981) present a set of heuristics for loading (allocating operations and tooling to machines) and scheduling the processing of a job mix in a flexible manufacturing system with the aim of improving the system's throughput. Better throughput is achieved by minimizing the number of transfers in the system rather than attempting to balance the workload of its machines. Further improvement is accomplished by choosing a loading strategy that involves some degree of pooling or duplicate operation assignments.

Agnetis et al. (1990) present a set of rules that derive an optimal schedule for processing a job mix on a two-machine flow shop. The primary objective of this approach is to maximize machine utilizations (minimize their idle time) and its secondary objective is to minimize the size of the buffer required to hold the work-in-process of the flow shop. The job classes of the mix are first partitioned into overlapping batches. In particular, two job classes are selected at a time. The jobs of these two classes are scheduled in ratios that minimize both machines idle time and buffer requirements.

Stecke and Kim (1991) present a flexible approach to scheduling job mixes in flow shops. An integer programming technique, used to balance machine workloads in flexible manufacturing systems, is used to dynamically generate a schedule for the job classes of the mix with the aim of maximizing the utilization and makespan of the system. The approach minimizes tool changeover time and the number of fixtures in the flow shop. Furthermore, breakdowns are handled by solving the integer program formulation of the problem subject to a new set of constraints. The size of the integer programs is dictated by the number of machines in the shop. Hence, this size, in general, will not be too large for real systems.

When processing a batch of identical jobs, Roundy (1988) defines a cyclic schedule as a schedule that completes one job every τ units of time and repeats itself every τ units of time, for some $\tau > 0$. Two measures of quality for these cyclic schedules are proposed; the cycle time τ (a measure of the throughput) and the flow time Φ (the time to completely process one job). Thus, Φ/τ is a measure of the work-in-process inventory. Moreover, a branch-and-bound algorithm for creating a cyclic schedule that attempts to minimize cycle time τ is presented.

Graves et al. (1983) propose an algorithm for scheduling batches of identical jobs in re-entrant flow shops. Their re-entrant flow shops are equivalent to job shops, and their proposed algorithm performs cyclic job shop scheduling. Given the flow time of a job, the algorithm initiates the processing of this job as soon as this processing does not conflict with that of current jobs. However, this cyclic job shop scheduling strategy generally does not result in an optimal usage of the machines in the shop.

Karmakar and Schrage (1985) model the cyclic scheduling of a job mix on a single machine as an integer programming problem, and propose a branch-and-bound algorithm for finding an optimal solution to this problem. The problem formulation attempts to minimize the total cost of job processing, work-in-process inventory, and machine setups.

Maxwell and Singh (1986) address the problem of cyclically scheduling the processing of a job mix on a set of identical machines, given the flow time of each of the job types in the mix. They show that when the flow times of all job types in the mix are similar or small compared to the minimum cycle time of a cyclic job shop schedule, the search for a feasible schedule can be restricted to those schedules in which the same job type is always assigned to the same machine, thereby saving setup times.

Bispo and Sentieiro (1988) proposed an algorithm for the cyclic job shop scheduling of job mixes where the percentage of each job type in the mix is known a-priori. They first partition a job mix into a number of identical batches. The percentage of each job type in a batch is set to equal that of the job mix. The A* search algorithm is then used to generate a schedule that minimizes the processing time of a sample batch. This schedule is used to process the jobs of a batch after the processing of this batch has been initiated. The processing of a batch is initiated as soon as its schedule does not conflict with the schedules of other batches in process.

Shin and Zheng (1990) model an automated assembly line as a flow shop in which machines have no buffers, the constraints created by the presence of a material transport system are captured, and each batch of production is represented as a job mix. A job whose flow through the machines has n feedback loops is modeled as a job mix of n jobs of the flow shop and these jobs are individually scheduled. For an assembly line with two machines, the problem is formulated as an integer programming problem, and a solution that minimizes the cycle time of the schedule is derived. Heuristic rules are also provided for deriving suboptimal solutions to the problem of scheduling an assembly line with $m > 2$ machines, with the object of miminizing the cycle time of the schedule.

Timed marked graphs are used by Dubois and Stecke (1990a, 1990b) in evaluating the performance of manufacturing systems; these systems are viewed as deterministic, discrete-event, and dynamic systems that repetitively perform a set of activities with fixed durations. Timed marked graphs form a special type of Petri nets where each place has exactly one input transition and one output transition. Places in such Petri nets correspond to arcs of the marked graphs, while transitions correspond to vertices. The time delay of a place of the net is assigned to its corresponding arc in the graph.

The firing of a vertex of the marked graph is *enabled* as soon as each input arc (place) of this vertex contains at least one token. Vertices without an input arc are enabled unconditionally. The *firing* consists of removing one token from each input arc and adding one token to each output arc of the vertex.

When modeling the dynamic behavior of manufacturing systems, the nodes of the timed marked graph capture the activities (manufacturing operations) of these systems and their arcs express precedence constraints due to either processing sequences or the ordering of parts on machines, and their tokens represent the machines and the fixtures (resources) of these systems. This representation assumes that the activities of each resource and the resources of each activity are linearly ordered and have fixed durations.

To find a cyclic schedule that maximizes the utilization of the resources of a manufacturing system, the shortest path in the timed marked graph model of this system is computed. The worst case complexity of the algorithm used to compute this path is $O(n^3)$, where n is the number of resources (machines and fixtures) of this system.

Pinto et al. (1983) design a branch-and-bound algorithm for minimizing the capital investment and labor costs of an assembly line given the cycle time of the task to be performed by this assembly line. The problem is formulated as an integer programming problem that attempts to minimize the number of workstations of the assembly line in order to maximize the utilization of these workstations, and minimize the cost of running them (labor cost) and the initial capital investment (workstation cost).

4. Simulation

Simulation is an analysis tool that can be applied effectively to a variety of shop floor design and real-time shop floor control problems. Simulation can support longer-term system design evaluations of resource requirements, equipment needs, inventory buffer sizes and availabilities, and sensitivity analysis of a variety of product demand and equipment processing time probabilities. Simulation can also support shorter-term decisions involving equipment scheduling, shop order release, and work order scheduling. This section presents a synopsis of current attempts to use simulation in scheduling and performance evaluation of manufacturing systems.

Two approaches to designing manufacturing systems simulators have been implemented: the *language preprocessing approach* and the *data-driven approach*. Language preprocessing systems, such as SIMAN IV (Sturrock and Pegden 1989), WITNESS (Gilman and Billingham 1989), AUTOMOD (Farnsworth et al. 1986), and SLAM II (O'Reilly and Nordlund 1989) are essentially very-high-level programming languages. The user can develop a model of a factory system by manipulating some very-high-level structures of these languages, usually via a graphical interface. Such systems provide a good deal of flexibility but have several drawbacks. First, because they are programming languages, models must still be compiled, linked, and debugged after the preprocessing step. Second, they often require reprogramming. Third, their structures are adequate for modeling simple systems. Consequently, the user may have to code some special routines in the host language of the preprocessor when modeling complex ones.

The data-driven simulation approach provides both flexibility and ease of use. However, new capabilities cannot be generated by the users. This approach is implemented in SIMFACTORY (Klein 1986, 1988), which provides primitives for modeling the factory layout, the process plans, the work cells, queues, conveyors, vehicles, transporters, human resources, parts, production schedules, and maintenance actions of a manufacturing system. Furthermore, process plans are assigned their own reject probabilities; their rejects may be ignored, scrapped, or reworked.

One of simulation's shortcomings has been that outputs from a simulation model typically take the form of summary statistics or simple graphs. Although these outputs are necessary to measure and draw conclusions about the performance of a manufactuirng system, they provide little insight into the dynamic interactions between the components of the system. Hence, animation can be used in conjunction with other analysis aids to gain a detailed understanding of the operation of a model; subtle errors that might not be apparent from standard simulation output become obvious when the operation of a system is displayed graphically. These considerations motivated the development of many new simulation/animation systems as well as animation interfaces to existing simulation languages. GAME, a graphical editor and animator dedicated to discrete flow manufacturing systems, has been implemented in *C*++ by Breugnot et al. (1991). The CINEMA animation system (Healy 1986; Poorte and Davis 1989) has been designed to work with SIMAN IV and SIMFACTORY has been coupled with an animated display.

Jain and Foley (1986) propose a set of guidelines for designing a generic simulator that can vastly reduce the time for developing simulation models of manufacturing systems. The simulation models are implemented as components that are expressed in a simulation language for manufacturing. Furthermore, their generic simulator can modify the schedule of a manufacturing system in response to a drop in the system performance below acceptable limits.

The concepts of object-oriented programming, logic programming, and discrete-event simulation are used by Ruiz-Mier and Talavage (1987) in modeling the significant aspects of complex systems such as manufacturing systems. The system is decomposed into a set of objects. The operations of each object are then captured by a set of logic-based rules. Objects communicate with each other by exchanging messages whenever events occur in the system. These messages are well-formed formulas of predicate logic that affect the state of the object that receives them. The use of logic programming allows the simulator to infer the set of operations that must be performed by the devices of a manufacturing system while implementing a designated task.

Object-oriented programming is also used by Hollinger and Bell in specifying manufacturing systems and simulating their behavior (1987). The system is partitioned into a hierarchy of objects, and timed Petri nets are used to model the behavior of this system. The sequence of operations that must be performed by the objects of the system is captured in a timed Petri net of the system. A place of this net captures the conditions that precede the execution of an operation of the system. A probability distribution is used to model the duration of this operation; an operation is represented as a transition of the net. The performance of the system is then simulated by executing the operations of this timed Petri net.

Levas and Jayraman (1987) have implemented an object-oriented environment, WADE, for modeling and simulating work cell applications. WADE does not assist the design process by providing explicit information to action planning, selection, design, layout, interconnection, and programming of equipment. Instead, the designer somehow synthesizes a design of his application and uses WADE to model his conception and evaluate its performance.

A Fortran-based simulator for experimenting with different control strategies of flexible manufacturing systems has been designed by Manalis et al. (1987). The control strategies are coded as a set of rules for processing orders, and for managing the queues and the workload of the devices of a system. The simulator is of the fixed-interval increment type; each time the clock is advanced, the system checks for the completion or the start of any scheduled operations.

ESPNET is an expert system-based Petri net simulator (Duggan and Browne 1988; Wadhwa and Browne 1989), that can be used in prototyping manufacturing control software. ESPNET takes a timed Petri net for its input and generates an OPS5 simulator as output. The OPS5 code is then used to estimate the performance of a manufacturing system, and the utilization of its resources.

Schriber and Stecke (1987, 1988) use mathematical programming to schedule a job mix with the aim of maximizing machine utilizations in a flexible manufacturing system. Simulation is then used to estimate the degradation of these utilizations due to their dependency on factors ignored in the mathematical programming solution. These factors include the fixtures, load/unload stations, buffers, transport vehicles, machine breakdowns and scheduled machine maintenance, due dates, and lateness penalties.

5. Monitoring

In a manufacturing system, the monitoring process includes collecting the information necessary for determining the current state of the actual system, checking the feasibility of performing the current set of scheduled operations, and dispatching these operations. This process also supports detecting and correcting any faulty situation that might occur while performing these scheduled operations. This section discusses the different approaches used in monitoring manufacturing systems.

Weck (1983) suggests that the development of an effective monitoring system is key to achieving more effective machine operation and better quality products. Monitoring systems take charge of the diverse supervisory tasks performed by the operating personnel, and offer functions to locate the type and reasons for failure. An ESPRIT project in Europe (see Meyer 1986; Huber and Buenz 1988) is studying the role of expert systems in planning the quality assurance and the preventive maintenance of manufacturing systems.

A PLC-based production monitoring system has been designed and implemented by Russell (1990). The Programmable Logic Controller (PLC) is used to collect data on the status of its controlled machine. This data is stored in an appropriate set of tables of the PLC buffer. These tables are transferred, at fixed time intervals to the database of the host computer, and can be interpreted by the operator to detect any problems in the machine that is controlled by this PLC.

Moore et al. (1984) develop an expert system that performs forward and backward chaining inference in a real-time environment with dynamic data obtained from measurements taken from a distributed process control system. The measurements are provided by a set of procedures that are coded in C and interfaced to the inference engine of the expert system. Forward chaining is used to decide the next set of actions to be executed by the process control system during the normal course of operation of this system. A fault of the system is diagnosed by focusing on its causes and consulting the knowledge base of the system in order to generate the appropriate sequence of routines for recovering from this fault. This knowledge base is interactively built by the user of the expert system.

Artificial Intelligence techniques are also used by Bu-Hulaiga and Chakravarty (1988) in monitoring flexible manufacturing systems. A frame-based representation of the devices of the system is derived first. The frame of a device is composed of a number of slots that enclose the components of this device. A rule-based control algorithm manages the frame-based representation of the system while tracking the operations performed by the devices of this system. Corrective measures are derived by the control algorithm whenever the actual level of production of the system falls behind the level specified in the production plan of this system. The same frame-based representation is also used by Weber and Moodie (1988) to represent the control component and the information management component of manufacturing systems.

An expert system for monitoring the operation of flexible manufacturing cells that produce a wide variety of parts is designed by Teng and Black (1989). The normal course of operation of these cells is modeled as a Petri net. The Petri net is translated into a set of conditions-actions rules that are repeatedly executed by checking their conditions and performing their corresponding actions. The detection of any disturbance during the execution of these rules is followed by a search for the causes of this disturbance. A knowledge-base is then used to determine the set of rules that can help recover from this disturbance, and resume the normal course of operation of the cell.

Larner (1990) proposes a framework in which Artificial Intelligence and non-Artificial Intelligence techniques can be applied in implementing advanced control applications of manufacturing systems. The framework, called *Distributed, Operating-System based, Blackboard Architecture for Real-Time control* (DOSBART) broadens some features of blackboard architecture to provide a tool for building monitoring software for manufacturing systems. Blackboards are shared data structures that are accessed and updated by the various processes of a knowledge-based system. In DOSBART, blackboard architectures are extended to handle events, task control, deadlines, concurrency, and resource allocation. DOSBART has provided intelligent monitoring of a production line and a paper pulp process by comparing the observed data to that produced by a queuing-theory model.

An Operations Research-oriented approach to replanning the sequence of operations of a manufacturing system whenever a fault is detected in this system is proposed by Bean et al. (1991). Their approach can be classified as a semi-dynamic approach to monitoring manufacturing systems because a preplanned set of operations is executed by the system until replanning this sequence is dictated by the detection of a faulty condition in this system. In contrast, fully automated monitoring is advocated by previous Artificial Intelligence-based approaches.

The strategy of Bean and Birge attempts to reschedule the operations of a preplanned schedule when disruptions prevent the use of the preplanned schedule. Rescheduling has the goal of matching up with the preplanned schedule at some time in the future, i.e., reschedule so that the completed jobs and inventory positions are identical to what would have occurred in the preplanned schedule.

The strategy deals with disruptions such as machine breakdown, tool unavailability, change in release dates, or unexpected new jobs that can be introduced to a manufacturing system, and the match-up approach seeks to compensate for the disruption to minimize total tardiness. The time required to match up with the preplanned schedule depends on the slacks available in the preplanned schedule itself. Moreover, it is assumed that the disruptions are spaced far enough apart to allow match-up between disruptions.

Chintamaneni et al. (1988) propose a hierarchical model for supporting fault recovery in manufacturing systems. In this model, each level in the hierarchy handles all the failures that occur at this level plus the failures signaled by the lower levels of the hierarchy. If recovery cannot be completely provided at this level, the corresponding failure is signaled to the next upper level of control of the hierarchy. Besides the failures explicitly signaled, the upper levels should be aware of other fault tolerance activities that are being performed at this level.

Adlemo et al. (1990) propose three different methods for achieving fault tolerance in automated manufacturing systems. Hardware reconfiguration can be performed by replacing the faulty units in a system. Plan reconfiguration can be performed by creating an alternative sequence of operations for a job. Schedule reconfiguration can be performed by assigning the operations of the current plan to other alternate or duplicate units of the system.

A systematic method for monitoring manufacturing devices is proposed by Takata and Sata (1986). This method adopts the model of a device as a basis for controlling the device and for handling any failures. In particular, the state diagram is proposed as a model for sequential devices, and the Boolean equations that express the output variables of a system in terms of the input variables of this system are proposed as models for combinational devices. These models are used to locate all the possible causes of a failure, and to implement an appropriate routine for recovering from this failure.

An approach to automatic error detection and diagnosis in manufacturing systems is proposed by Chang et al. (1989, 1990). This approach captures the set of possible faults that can occur during the execution of an action and the known and probable effects of these faults as the nodes of a tree. In this tree, the nodes representing the possible faults of an action are connected to this action, and the nodes representing the effects of a fault are connected to the node representing this fault. An algorithm for identifying the occurrence of one or more faults by checking their known and probable effects is also presented.

Hasegawa et al. (1990) propose the use of a layered Petri net to describe both the normal operation and the exception handling of flexible manufacturing systems. Their Petri net structure consists of a supervisory layer and an operational layer. Both layers are represented by safe Petri nets. A *mode token* flows on the supervisory layer, and a *process token* flows on the operation layer. The Petri nets of the operation layer are enclosed in the places of the supervisory layer. The presence of a mode token in a place of the supervisory layer enables the execution of the operations that are specified in the operational layer Petri net of this place. The occurrence of an exception in the system moves this mode token to the place of the supervisory layer where this exception can be handled by executing the operations of the operational layer Petri net associated with this new mode of operation of the system.

Viswanadham and Johnson (1988) develop a method for fault detection in automated manufacturing systems that uses Petri nets and AND-OR fault trees. The normal course of operation of a system is modeled as a timed version of predicate-transition Petri nets. This net has two types of transitions; transitions that execute instantaneously, and transitions that execute for a nonzero, but finite duration of time. A mean value μ and a standard deviation σ are coupled with the noninstantaneous transitions of the net. Whenever the event of a transition does not occur during the time interval $[\mu - 2\sigma, \mu + 2\sigma]$, a fault is detected in the device where the event must occur. The detection of a fault is followed by consulting the AND-OR fault tree of this transition. This tree encloses at its highest level the faulty event, connects through a set of AND gates and OR gates the various events of the system that can lead to this faulty event, and captures the primal causes of these events at its leaves.

The primal causes of a faulty event are found by identifying the gate that feeds into this faulty event and by searching all the events feeding into the current gate of the tree. If the current gate is an AND gate, then all feeds are causative and true. If further resolution is required (i.e., if a feed does not lead directly to a leaf), call each such feed a top event and repeat the search.

If the current gate is an OR gate, then any of the feeds can be causative. Select the one with the greatest probability of occurrences and determine if it is true. If the feed is true and primal, the fault is found and the search is terminated. If the feed is true and more resolution is required, call this feed a top event and repeat the search. If the feed is false, select the next probable feed and repeat the algorithm until all feeds of the current OR gate have been checked.

6. Integrating Process Planning and Scheduling

This section presents some of the attempts at merging the planning and scheduling activities of real-time manufacturing control software. The development of Material Requirement Planning (MRP), Manufacturing Resource Planning (MRP II), and shop floor control systems presents a major attempt in this direction. MRP monitors material requirements on a weekly, sometimes daily, basis. MRP II monitors material, labor, and machine capacity requirements in the same time periods. Shop floor control, however, monitors labor, material, and machine capacity requirements as well as the actual environment—what machines and labor are actually present and available for work—on a shift, if not hourly, basis. MRP,

MRP II, and shop floor control have been integrated together as part of the rule-based software environments marketed by companies such as the PROMIS Systems Corporation (Debolt 1988) and CONSILIUM.

Harhalakis et al. (1988) model the integration of Computer-Aided Design (CAD), Computer-Aided Process Planning (CAPP), and Manufacturing Resource Planning (MRP II) as a place-transition Petri net. This integration is crucial to achieving better coordination and higher productivity in manufacturing. In particular, the use of Petri nets allows any concurrency conflicts or deadlocks in performing the activities of CAD, CAPP, and MRP II to be detected.

The Manufacturing Resource Planning (MRP II) system is a set of modules that establish a Master Production Schedule (MPS) for a system based on customer orders, sales, and forecasts. MPS is then translated into a Material Requirements Plan by checking the inventory of the system. The Master Production Schedule and the Material Requirements Plan are used by the Shop Floor Control (SFC) module in generating a detailed production schedule for the system.

A Computer-Aided Design (CAD) system consists of a set of modules that can assist a designer in drafting, analyzing, and optimizing a part design; it produces a part design specification. On the other hand, a Computer-Aided Process Planning (CAPP) system is a set of modules that transform part design specifications into the manufacturing instructions required to convert the raw material into a finished part. CAPP selects tools and machines, determines operation sequences, calculates operation and setup times, and identifies any jigs and fixtures needed for fabrication or assembly work. This information is then used by the SFC module of MRP II in automatically generating the production schedule of the system.

A common database is used in integrating the above MRP II, CAD, and CAPP systems. This technique of using a common data repository for capturing the static and dynamic requirements of manufacturing systems is also advocated by Ketcham et al. (1988) and by Moyne et al. (1989, 1991).

Jablonski et al. (1990) propose a flexible and reactive scheme for integrating CAD and CAPP systems. The scheme generates, in the form of parametric NC program templates, the sequence of processing steps of a part from a feature description of this part. A Manufacturing Resource Scheduling System (MRSS) assigns manufacturing resources to each processing step thus converting an abstract processing step into an executable manufacturing operation. During the production of the part, the rescheduling of manufacturing resources in reaction to failures can be performed because the NC program templates refer to abstract manufacturing resources instead of real manufacturing equipment; thus, these resources can simply be mapped to different equipment to achieve the rescheduling.

7. Developing Shop Floor Control Software

This section presents some of the approaches to developing shop floor control software that attempt to integrate the functions of planning, scheduling, and monitoring of this software. Finite-state machines, Petri net variants, and rule-based systems are used by these approaches to model the control software. The applicability of some of these approaches is enhanced by the presence of a software environment that can be used to enforce their underlying concepts.

Ben-Arieh et al. (1988) attribute the hierarchical nature of the real-time control problem in flexible manufacturing systems to the complexity of the task of controlling these systems. The two lowest levels of this control hierarchy are identified as the cell level and the workstation level, respectively. Part routing is tackled at the cell level, while equipment control is dealt with at the workstation level.

Knowledge-based techniques are presented for generating a real-time control strategy for each of these two levels. The workstation control method uses a decision table to map the current state of a workstation into values for the decision variables of this workstation. The cell control method consults a knowledge-based routing system in order to introduce or route parts dynamically in a mixed machining and assembly facility. Knowledge-based techniques are also used by Devedžić (1990) in controlling the robots of flexible manufacturing cells.

Shires and Bastos (1988) implement a decision support environment for performing real-time operational control in manufacturing systems. The sequence in which manufacturing orders and operations are carried out by a system is determined by the dispatching and the sequencing functions of this system. Dispatching determines the sequence in which orders are introduced into the system. Sequencing determines the sequence in which operations of different orders are executed by this system. Operational control, also referred to as factory control, implements the planned orders and operations by scheduling the execution of a particular operation on a particular facility.

Factory control is achieved by applying a set of rules to dynamically schedule based on the current state of the system, the execution of the orders of the dispatching function and the operations of the sequencing function. The list of scheduling rules is then presented to the system operator who decides on the set of operations that must be executed next.

7.1. System 90

System 90 is a software environment for developing the real-time control software of transfer lines that is based on the concept of Zone logic (Fisher 1989). Zone logic is a new technique for specifying the control software of transfer lines that cures many of the deficiencies of Ladder logic (Kompass 1987; Vasilash 1987). This techique has been implemented by AEG and Septor, and is expected to increase to 90 percent the uptime of the machines of transfer lines.

Zone logic requires a programmer to declare all the valid physical states (zones) of a machine. The rules of zone logic subsequently interpret these valid zones, and provide machine control, error detection, guided restart sequences, and a journal of all machine activities.

Zone logic adopts an object-oriented approach to modeling the machines of a transfer line. Each machine is implemented as a set of objects called *mechanisms*. A mechanism is associated with a mechanical motion unit of the machine, and its state is captured by a zone table. This zone table specifies the I/O values of each valid zone of the mechanism, a time limit on the validity of a zone, and the set of possible zones that the mechanism can assume once this time limit expires. A library of zone tables can eventually be built and used in specifying other machines of the transfer line.

The occurrence of a fault in the system results in an invalid zone of the mechanism being assumed by the system. By detecting the faulty I/O condition(s) of the current valid zone, the transfer line is stopped, and a message can be generated by the system to help the operator diagnose the fault. The sequence of operations required in restarting the machine can also be stored in a database and used to guide the operator through the diagnosis process.

Constraints on the motion of a mechanism and its interference with other mechanisms of the machine are captured in a mechanism interference table and a set of zone interface tables. The mechanism interference table simply specifies the mechanisms that interfere with each other. For each such pair of mechanisms, a zone interference table specifies the zones where these mechanism interferences occur.

A process table is used to define the sequence of operations that must be executed to process a part type by the transfer line. This table specifies the machine and mechanism that carry out each operation of the sequence. A process table may impose more constraints on the mechanisms of a machine by linking the use of these mechanisms to the sequence of operations for processing a part type. Furthermore, the zone of the mechanism when performing this operation is determined. A job produces a single part of the specified type. Furthermore, the operations that have already been completed when processing this job are tracked throughout the transfer line.

The operations that need to be performed again on a faulty part can be run through the machine without reprogramming. Moreover, process tables allow mixed part type manufacturing and can be used in conjunction with the mechanism zone tables of a machine to restart this machine automatically.

7.2. ISIS and OPIS

The Intelligent Scheduling and Information System (ISIS) is a constraint-directed job shop scheduling system that has been developed by Bourne and Fox (1984). In ISIS, the functions performed by the various devices of a manufacturing system are defined as operators. The planning system, working with the goal of transforming a raw product description into a final part shape, determines various sequences of these operators. The generation of these sequences is guided by the presence of a set of *constraints* that encode the knowledge of the production engineers who determine the order in which the devices of a manufacturing system must be employed to achieve effective tolerances and surface finishes of the product. Constraints such as due date, cost, quality, accuracy, and size can be expressed in ISIS. Due to the conflicting nature of certain constraints (e.g., cost versus quality), there may not be a schedule that satisfies all of them. Hence, ISIS considers relaxations of such constraints when generating and selecting schedules.

ISIS carries out a hierarchical, constraint-directed search through the space of possible schedules. The search moves through more and more detailed levels of analysis, with specific types of constraints coming into play at each level. ISIS is capable of incrementally scheduling orders as they are received by the plant as well as reactively scheduling orders that are affected by machine breakdowns and other dynamic changes in the shop.

Recently, the ideas of ISIS influenced the development of an Opportunistic Intelligent Scheduler System (OPIS) (Le Pape and Smith 1987; Smith 1988). Based on the source of the fault in a system, the task of resolving this fault is assigned to one of the many knowledge sources of OPIS. The implications of this fault resolution on the sequence of operations of the original schedule, the duration of this schedule, and the state of the system are considered when the operations of this resolution are merged with the original schedule. OPIS implements a top-down hierarchical approach to schedule generation and a bottom-up approach to rescheduling in the presence of faults (Ow et al. 1988).

Both ISIS and OPIS can be classified as rule-based software environments for developing manufacturing control software. This hierarchical view of the control structure of a manufacturing system reduces the search spaces of these planners. However, their real-time performance may suffer whenever they are used to control the operation of a large factory; the large number of operators that are required to capture the functions performed by this factory might radically increase the size of the search space of both ISIS and OPIS, and consequently, slow down their job shop scheduling activity.

7.3. Knowledge-Based Scheduling

Shaw and Whinston (1985a, 1985b, 1986, 1987, 1988) develop a knowledge-based scheduler for determining the machine assignments and manufacturing paths of the jobs dispatched into a manufacturing cell once the product mix of these jobs and their job entry sequence have been determined. Their approach models the state of the cell as a set of first-order predicates, and the operations that affect this state as a set of rules similar to the STRIPS operators; each operation is specified by an add list of state predicates, a delete list of state predicates, precondition predicates, the resource used by the operation, and the duration of this operation. The plan of each job in the mix is created using the A* heuristic search procedure.

The above plans compete for the resources (the machines) of the cell. Consequently, precedence constraints between conflicting operations of different jobs must be established so that the activities of these operations are well coordinated. Because of these added precedence constraints, the resultant schedule is a partially ordered network of operators; accordingly, this process is referred to as *nonlinear planning* (as opposed to linear planning where the output is a set of linearly sequenced operators). The combined schedule is next revised to maximize the parallelism while avoiding harmful interactions among the plans of the jobs of this schedule. The same nonlinear planning strategy can also be used to modify the previous schedule because of unexpected breakdowns in the system.

This knowledge-based scheduler is used at the cell level in a large manufacturing system while a bidding algorithm is used to distribute the processing of the jobs in the product mix to the various cells of this system (see Shaw and Whinston 1985b, 1988). Jobs are assigned to a given cell whenever this cell bids for them and has a reasonable workload. Cell bidding is based on a priority list of the jobs that can be processed by each cell of the system. The same bidding algorithm is used by Baker (1988) to integrate MRP II and the job shop control of manufacturing systems, by Duffie and Piper (1986) to implement a nonhierarchical control scheme for manufacturing systems, and by Ting (1990) in his proposed cooperative shop floor control model.

7.4. The AISPE Environment

The Advanced Industrial Software Production Environment (AISPE) consists of a set of software tools for supporting the specification, rapid prototyping, and simulation of process control systems such as manufacturing systems and robotized assembly cells (see Bruno and Marchetto 1985, 1986; Bruno and Balsamo 1986; Bruno and Morisio 1987b). AISPE can be used in performing the steps of a methodology based on *PROcess Translatable* (PROT) nets, a modified verison of predicate-transition Petri nets.

The PROT net model is oriented towards the specification of concurrent processes and their synchronizations. Their associated semantics allow the implementation of these processes and their synchronizations to be generated from the net automatically. Each process describes an autonomous activity to be performed cyclically: it consists of a sequence of states and state transitions. The transition from one state to another can be subjected to the influence of the other processes in the system. The features of a PROT net are

- *Types*: A PROT net models the interactions among the instances of various process types. Such instances synchronize and exchange information at transitions. In order to distinguish instances of the same process type and to support data exchange among the instances of different process types, data structures are associated with process types through type declarations.
- *Places*: Places are assigned types because they represent the states of the corresponding process types. Each process type consists of an ordered sequence of all the places of that type. Such a sequence is, in general, a tree called the *process tree*; however, it is a simple sequence if there are no fork places. The root of this tree corresponds to the initial state of the process, and a path from the root to a leaf represents the execution of a single cycle of the process. A cycle consists of the ordered sequence of the successive states assumed by an executing process, where the initial and final states are identical.
- *Tokens*: Tokens represent process instances. The presence of a token in a place indicates that the corresponding process is in the state identified by this place. Tokens carry pieces of information—called attributes—whose structure is specified in the type declaration. Tokens can be given priorities according to their attributes.
- *Transitions*: Transitions perform synchronizations among process instances at their incoming arcs; data exchange can also be specified by means of actions inscribed on outgoing arcs. A transition is called simple if it is not decomposable into a PROT subnet. PROT subnets enable top-down hierarchical structuring and model lower-level processes of the hierarchical structure in detail. Predicates can be inscribed on transitions: they are Boolean expressions to be evaluated on the attributes of the tokens present in the input places of the transitions. In case of ambiguity, a particular attribute is fully qualified by prefixing the name of this attribute with the name of a particular place of the PROT net.

  Transitions having some input places in common are said to be mutually exclusive because the firing of one of them disables the others. Additional pieces of information, such as timing requirements and the firing priority in the case of mutually exclusive transitions, can be associated with transitions. Timed transitions have a delay associated with them; nontimed transitions are called *immediate transitions*.

- Arcs: Arcs connect transitions to places and places to transitions. Actions can be inscribed on outgoing arcs of transitions. Basically, actions are assignments that can change the values of the attributes carried by the tokens that caused the transition to fire.
- *Initial Marking*: This is the initial distribution of tokens into places. All tokens of the same type must be put into a single place. The initial marking specifies the initial values of the attributes of each token as well.
- *Transition Firing*: A transition firing takes place when there is at least one token in each of its input places and the predicate inscribed on it is satisfied. The firing of an immediate transition consists of removing the tokens satisfying the predicate from the input places and adding tokens to the output places after performing the actions inscribed on the outgoing arcs.

  The firing of a timed transition consists of removing the tokens satisfying the predicate from the input places and, after waiting the associated time delay and performing the actions inscribed on the outgoing arcs, adding tokens to the output places.

The methodology, based on PROT nets and supported by AISPE, includes the following activities:

1. *Specification*: The PROT net gives an operational specification of a system based on the concept of *process* and *synchronization*. The system is partitioned into a set of components, in accordance with the object-oriented software design paradigm (see Booch 1986). Each component is modeled as a PROT net. These PROT nets are then connected together by sharing places of the same type, called *communication places*.
2. *Validation*: PROT nets can be analyzed in order to determine properties such as absence of deadlocks, place boundedness, and so on.
3. *Evaluation*: Timing can be associated with each transition of a PROT net, thus allowing the performance evaluation of the model to be carried out in order to discover critical aspects. Both analytical and simulation tools can be used to investigate a net augmented with timing requirements (see Bruno and Biglia 1985; Bruno and Morisio 1987a).
4. *Prototyping*: A prototype of the system can be generated automatically in a high-level general-purpose language, a rule-based language, or a simulation language. An Ada program skeleton can be derived automatically from the PROT net. Programmers are thereby relieved of the error prone coding of synchronizations among concurrent processes (see Bruno and Balsamo 1986; Bruno and Marchetto 1986). A simulation prototype can be also derived from the net if the target language is a simulation language supporting the concurrent processes approach to simulation (see Bruno 1984). PROT nets can also be translated into an OPS5 rule-based language prototype (see Bruno and Elia 1986).
5. *Implementation*: The prototype obtained from the net can be used as a framework for the final implementation.

Although AISPE models a manufacturing system as a set of components, the dynamic behavior of these components is rigidly specified by their PROT nets; planning a new sequence of operations for a manufacturing system may require performing substantial modifications on the PROT net models of this system. Consequently, the PROT net approach improves the practice of developing manufacturing control software by adopting object-oriented design techniques; however, this approach does not generate flexible control software for these systems.

AISPE does not provide adequate support for the cyclic scheduling of the resources of a system. The cyclic scheduling strategy associated with PROT nets is based on firing the transitions of these nets and does not guarantee the optimality of the cyclic schedules. Furthermore, capturing in graphical form all the possible states that may be assumed by a manufacturing system while performing a set of operations increases the complexity of the PROT net model of this system.

7.5. The FLEXIS DESIGNER Environment

The FLEXIS DESIGNER is a GRAFCET-based interactive environment for developing manufacturing control software (see Dove 1988; Wilczynski 1988). GRAFCET is the equivalent of function charts, a standard for describing the control logic of manufacturing devices (International Electrotechnical Commission 1984). In FLEXIS, a system is decomposed into actors. An actor is a self-contained, abstract, object-oriented program that communicates with other actors by sending and receiving messages, carries out tasks one at a time, schedules its own tasks, and drives the devices of the system.

Actors in the FLEXIS environment are implemented in a GRAFCET diagram (see Group de Travail Systèmes Logiques de l'AFCET 1977). GRAFCET is a functional requirements specification model derived from Petri nets. GRAFCET is suitable for modeling real-time systems because it overcomes two drawbacks inherent in Petri nets: nondeterministic evolution and infinite creation of tokens. A GRAFCET diagram (figure 1) is composed of steps and transition connected via directed arcs. Steps in a GRAFCET diagram can only be active or inactive (there are never several tokens in one step). Moreover, transitions in a GRAFCET diagram are always deterministic.

A step specifies a set of actions, and is active when executing these actions. Otherwise, this step is inactive. Level actions are executed for the period of time during which their step is activated. When their step is activated, pulse actions are executed for the fixed duration of time associated with them. The execution of an action might be conditional on input variables and/or the states of some specified steps. Tokens are used to show the active steps being carried out at a given time.

A transition has a Boolean condition associated with it called its receptivity. The receptivity of a transition can be a function of input variables, the states of some specified steps, the state of level variables, and/or the state transition of edge-triggered variables. Level variables are denoted by Y or $\bar{Y}$, positive-edge-triggered variables are denoted $Y\uparrow$ and negative-edge-triggered variables are denoted by $Y\downarrow$. Time is included in a receptivity condition by specifying a time origin and duration. The origin is the time of the last activation of a preceding step. A receptivity condition can be used to assign priorities implicitly to step activations. When the receptivity of a transition is true, it can fire by deactivating its input steps and activating its output steps.

To model complex actors, macro steps are used to enclose other GRAFCET diagrams. Macro steps allow a hierarchical presentation of these complex actors because they can

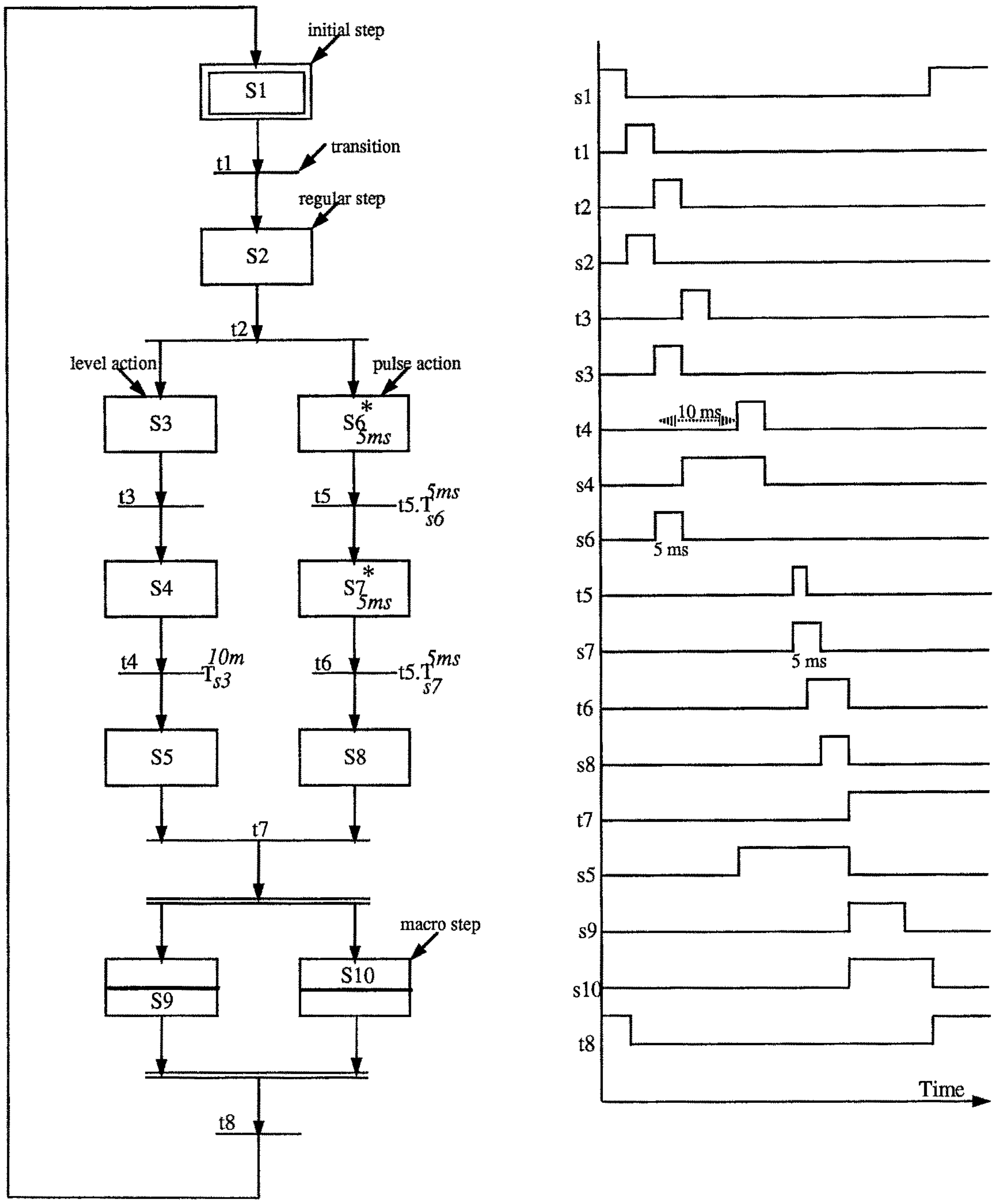

Figure 1. A GRAFCET (standard function chart) diagram.

enclose other macro steps of the hierarchy. In the FLEXIS environment, regular steps represent arbitrary control actions of an actor in the form of embedded C source code. Macro steps represent embedded GRAFCET macro programs as control actions. Transitions act as gates on the flow of control through a GRAFCET program. Associated with each transition is a C expression, which determines whether control can pass through the transition. Steps and transitions can be linked to form a control flow path. The timing capabilities of GRAFCET diagrams are not implemented in the FLEXIS DESIGNER environment. Consequently, it is not possible to evaluate the performance of an actor, nor to verify its timing properties.

In GRAFCET diagrams, a transition followed by two or more steps denotes a synchronous branch; when the transition is true, all of the steps are initiated simultaneously. A transition preceded by two or more steps is a synchronous join. Control cannot proceed after a synchronous join until all the preceding steps have completed and the single transition becomes true. Synchronous branch and join are denoted by double horizontal lines.

A single step followed by two or more transitions denotes an asynchronous branch. A single step preceded by two or more transitions is an asynchronous join. Control can flow through any of the transitions whose conditions are true. Because of the nature of the asynchronous branch, multiple tokens can be produced from an asynchronous join. In GRAFCET, asynchronous branch and join are denoted by single horizontal lines. In an asynchronous branch, the special transition condition of *otherwise* must always be included. This transition condition becomes true if the conditions of all other transitions in a set of asynchronous branches are false.

The methodology supported by the FLEXIS DESIGNER environment partitions a manufacturing system into a set of objects. Each object is implemented as a set of actors that can collaborate to execute the actions of this object, and to link the object with other objects of the system. An actor is modeled with a GRAFCET diagram and the regular steps of this GRAFCET diagram are implemented in C. This actor is incrementally compiled and tested by displaying, in the form of tokens, the flow of control in its GRAFCET diagram. This interactive approach helps in dynamically detecting deadlock and race condition problems. A prototype of the actors of the system can be designed and tested before a complete implementation is carried out. It is recommended to fully test the implementation of an actor that encloses a hardware device of the system prior to connecting this actor to its device and operating this device.

The FLEXIS DESIGNER enviornment has been used in designing the control software of many operational systems (see, for example, Thomas and McLean 1988; Judd et al. 1990). In particular, the use of FLEXIS in designing the controllers of the Automated Manufacturing Research Facility (AMRF) of the National Bureau of Standards (see Jun and McLean 1988; Lee and Yang 1988) revealed some of the limitations of this environment (see Thomas and McLean 1988). A system can be elegantly designed in GRAFCET, but can be very difficult to implement in a procedural language such as C because the latter does not offer concurrency constructs at the language level; the use of Ada may cure this problem. Furthermore, in contrast with PROT nets, the GRAFCET model does not support any data representation. To overcome this limitation, the FLEXIS DESIGNER environment captures the data flow in a GRAFCET design in C source code. However, FLEXIS does not support parameter passing; all program variables are global variables. Hence, macro steps are not reusable; they can only be used once after they have been defined.

GRAFCET-based environments suffer from the same limitations of PROT net-based environments. However, the FLEXIS DESIGNER handles fault recovery at the actor level only. The conditions of a fault are captured with an appropriate transition, and the code that must be executed to recover from this fault is enclosed in the step of the GRAFCET diagram of the actor that will be executed when the conditions of this transition become true.

7.6. Operation-Resource Petri Nets

A methodology for specifying the control logic of the discrete controllers used in manufacturing systems is presented in Willson and Krogh (1990). These controllers range from simple Programmable Logic Controllers (PLCs) to general-purpose computers. Their specifications are expressed in a rule-based formalism and are automatically translated into Petri net models. The Petri net models are analyzed, in turn, to verify the properties of these specifications and then are translated into executable control programs.

The rule-based formalism used to specify the control logic of discrete manufacturing controllers targets the control functions performed at the machine level and the cell level by Programmable Logic Controllers (PLCs). The basic elements of this rule-based formalism are *state variables* and *transition rules*. For each state variable, a set of discrete values or a range of values is defined, and a set of transition rules that describe the conditions under which a state variable changes value is specified. These transition rules are composed of a set of *state preconditions* and a set of *state postconditions*; when the preconditions of a rule are satisfied, the postconditions of this rule are made true.

A Petri net model is automatically generated from the above rules and is subsequently analyzed to detect any inconsistencies and incompleteness in the rule-based specification of the system. This Petri net model captures both the discrete event evolution of manufacturing systems and their underlying continuous processes (see Beck 1985; Beck and Krogh 1986). On the cell level, a manufacturing system is viewed as the interaction of discrete, asynchronous, concurrent events that must be coordinated to enforce precedence relations and allocate resources. On the machine level, the actual operations performed by the system are viewed as continuous-parameter processes that continuously evolve with time. Hence, the resources of the system are assigned discrete states and continuous-parameter or qualitative attributes. The values of these attributes can be modified whenever a continuous process that uses their resources is executed.

To capture both aspects, a class of place-transition Petri nets is defined with two types: *operation places* (o-places) and *resource state places* (rs-places). The o-places correspond to the operations of the modeled system, and the rs-places correspond to the discrete states of the resources of this system. A token present in an o-place indicates that the operation is in progress. A token in an rs-place indicates that the resource is in the discrete state associated with this place. Graphically, o-places are represented as boxes and rs-places are represented as circles. Along any path in a valid net, the o-places and rs-places alternate. Hence, the *transitions* of the net can be decomposed into two classes: *input transitions*, which immediately precede o-places and have a single outgoing arc to an o-place, and *output transitions*, which immediately follow from o-places and have a single incoming arc from an o-place. All arcs from rs-places enter input transitions, and all arcs from output transitions enter rs-places. Because of this restriction, the boxes for o-places can be eliminated for simplicity from the graphical representation and replaced by associating with the corresponding transition a time duration equal to the execution time of this operation.

The analysis of the net model of the discrete controller specification is followed by translating this specification into executable control programs. This translation can generate Instruction List (IL) programs that are executed sequentially and repeatedly by Programmable Logic Controllers (PLCs) (see Ekberg and Krogh 1987), or C programs that can be executed on a general-purpose computer to emulate the operation of a discrete controller (see Krogh et al. 1988; Pathak 1988).

7.7. Colored Petri Nets

Colored Petri nets are generalization of place-transition nets in which a color is associated with each token to indicate its identity. Also, a set of colors is associated with each place and each transition. The set of colors associated with a place indicates the colors of tokens that can be present at the place. A transition can fire with respect to each of its colors. The input and output arcs of a transition are labeled with functions that determine the colors of the tokens removed from its input places and the colors of the tokens deposited in its output places when the transition fires with respect to a particular color.

Graphically, places are represented by ellipses and transitions by rectangular boxes. The color set associated with a place or a transition is indicated on one of its sides. PROT nets can be classified as colored Petri nets if the colors associated with the tokens of colored Petri nets are interpreted as the types of the tokens of PROT nets. Hence, colored Petri nets share the same advantages and disadvantages with PROT nets.

Colored Petri nets have been used by Kamath and Viswanadham (1986) and by Viswanadham and Narahari (1987) in the modeling, analysis and synthesis of flexible manufacturing systems. The flow of control of each function performed by a system is modeled with a colored Petri net. The nets of the functions performed by the system are analyzed on an individual basis for boundedness, conservativeness, and liveness. These nets are then grouped into a complete colored Petri net of the system.

Martínez et al. (1987) model the software that coordinates the functions performed by the devices of a manufacturing system in terms of colored Petri nets. This software is integrated with the software implementing the local control of these devices and the software implementing the real-time scheduling algorithms of the system. The firing of a transition of the net with respect to a single color or a set of colors is translated into orders to the local controllers of the devices of the system. The message sent to these local controllers is coded by the transition and color that define each firing. Orders are subsequently executed as tasks by the devices of the system and result in freezing the colored token in the net until the local controllers report the conclusion of all tasks.

The control software consults with the real-time scheduler of the system in solving conflicts in transition firings and to report detected breakdowns and alarms. In communicating a conflict, information on the transitions and colors that produced the conflict are included. The real-time scheduler responds with the transition and the place chosen for correcting the conflict. The real-time scheduler is designed as an expert system composed of a set of facts and rules that are formulated in terms of the markings of the colored Petri net and the colors of the tokens of this net.

Structured timed colored Petri nets are used by Camurri and Franchi (1990) in modeling the control software of flexible manufacturing systems. These nets are realized by incorporating timed transitions into the above colored Petri nets.

7.8. Structured Adaptive Colored Petri Nets

Corbeel et al. (1985), and Gentina and Corbeel (1987) model the control software of flexible manufacturing systems as structured adaptive colored Petri nets, and link this model to a rule-based control strategy of the system. Their approach starts by specifying the functional requirements of the system in terms of a set of rules of first-order predicate logic. The initial state of the system and the production goals of this system are predicates of first-order logic. Deduction is used to select the set of control rules that achieve the production goals of the system. These rules specify the functions that must be performed by the devices of this system and ignore the transport actions that precede the execution of these functions. A structured adaptive colored Petri net is generated from the previous set of derived rules, and Petri net theory is used to check their properties. The structured adaptive colored Petri net is augmented next with the actions of a transport strategy that is designed to increase the level of concurrency in the system. The actions of the augmented net are implemented in a high-level procedural language.

Structured adaptive colored Petri nets extend the capabilities of colored Petri nets by being able to modify their own firing conditions. In these nets, the name of a place can be associated with the firing conditions of a transition. If these firing conditions do not contain the name of any place of the net, the firing rules are identical to those of place-transition nets. However, when the name of a place is included in the firing conditions of a transition of the net, the number of tokens to be moved from or to the place connected to this transition equals the actual number of tokens in the place included in the firing conditions if this place is not empty. Hence, structured adaptive colored Petri nets can capture, in a compact form, the interactions of the components of a manufacturing system.

Jarari (1990) extends structured adaptive colored Petri nets to model the error recovery of shop floor controllers. A timing constraint is attached to each transition of the net. Violating the timing constraint of a transition signals the occurrence of a fault during the execution of the operation associated with this transition.

7.9. Extended Petri Nets

Crockett et al. (1987) and Zhou et al. (1989, 1990) model the control sequencing information of industrial controllers as extended Petri nets. In their hierarchical model of manufacturing systems, these controllers are able to process commands sent from the higher levels of the hierarchy, synchronize the activities of the devices of their lower levels, and report back the status of these devices.

Extended Petri nets are coupled with features that make them more suitable for modeling manufacturing control software. In particular, types are associated with the places and the tokens of these nets. In an extended Petri net, a *status place* indicates the status of a resource or a part in a manufacturing system. Typical uses of such a place are to indicate whether a resource is available or whether a process has been completed; it is a nontimed place. An *action place* is a timed place that encloses an implementation of a certain procedure being performed by the system. A *switch place* is a nonprimitive place that checks certain conditions or evaluates a set of expressions in order to resolve a conflict in the net; a conflict occurs when a token can take one of several arcs leaving a place. A *source place* denotes the initialization of a token, and introduces a new resource, information, or part into the system. In contrast, a *sink place* indicates the termination of a token, and signals that a part, resource, or piece of information has left the system. For every source in the net, there is a corresponding sink.

Subnets are also introduced as a convenience for modeling hierarchies in manufacturing systems. Further extensions to extended Petri nets are proposed by Ahuja and Valavanis (1988). These extensions help distinguish between the flow of control, information, resources, and parts in the net. A different class of tokens is defined and initialized to indicate the presence of each class of parts, resources, or information. A generic class of tokens indicates the status of the net at different stages of its execution, and represents the flow of control through this net. A set of arcs is associated with each class of tokens, and multiple classes of tokens may reside in any place. An alternative to the token classes of extended Petri nets has been proposed by Kasturia et al. (1988). This alternative uses colored tokens to distinguish between the flow of control, information, resources, and parts in an extended Petri net.

The execution of the control software of industrial controllers has two interacting parts: the execution of the extended Petri net model (firing enabled transitions, marking and unmarking places) and the execution of procedures that affect the physical system being controlled. The execution of the model determines when the procedures are executed, and the results of the procedure influence the execution of the model.

7.10. The SECOIA Environment

An approach for monitoring flexible manufacturing systems has been designed and implemented by Atabakhche et al. (1986) and Sahraoui et al. (1986, 1987) as a component of the Specification and Emulation for COmputer Integrated Automation (SECOIA) project. This approach defines error detection, diagnosis, and handling as the basic functions of a monitoring system, and implements an approach to monitoring manufacturing systems that integrates Petri nets and rule-based systems.

High level Petri nets specify the normal course of operation of a manufacturing system, and permit the detection of faults during the operation of this system. The heuristic knowledge required for faults diagnosis and handling is stored in a knowledge base. This knowledge is accessed when the state reported by the system sensors is different from the state portrayed by the Petri net. The knowledge-based system is coupled with a backward chaining inference engine that attempts to diagnose and handle the fault by restoring the system state to that of the Petri net. The Petri net interpreter is implemented as a forward chaining inference engine. This engine checks for the conditions associated with the transitions of the net and executes the actions specified in the places of this net.

8. Conclusion

This paper presented a synopsis of ongoing research in the process planning, scheduling, and monitoring disciplines of manufacturing. Planning the sequence of operations of a job in a manufacturing system is viewed as a special case of the general planning problem of Artificial Intelligence where capturing the physical and logical constraints of the system and the duration of the operations performed by the devices of this system is vital to the generation of feasible and/or optimal plans for this system.

Two fundamentally different approaches to scheduling the processing of a set of jobs in a manufacturing system have been reviewed. The first appraoch assumes the presence of a process plan for each job to be scheduled and generates, a-priori, a complete schedule for processing these jobs by using some Operations Research techniques. Integer programming is extensively used by this approach. However, an integer program is approximated by a linear program whenever the solution time of the integer program is very large. The second approach extends the planning approach to cover the operations of all jobs in a set of jobs. Advocates of this approach try to perform real-time control of the factory floor by basing their scheduling decisions on the current state of the factory. Each one of the two approaches has its merits and its shortcomings. The first approach advocates an off-line scheduling strategy and is more likely to generate optimal schedules. However, the execution of this preplanned schedule may be affected adversely by the occurrence of a fault in the system. The on-line scheduling strategy of the second approach is more adaptive and solves the problem of invalidating the preplanned sequence of operations of a schedule due to the occurrence of a fault. However, achieving optimality when applying this second approach is unlikely, given the complexity of the planning problem and the short interval of time for determining the set of operations that can be executed next by the manufacturing system.

The use of expert systems in performing the monitoring activity of real-time manufacturing control software is a common strategy of all current approaches to monitoring. The normal course of operation of a manufacturing system is captured as a variant of Petri nets. The control sequence of this net is then followed until a fault is detected in the system. The detection of this fault triggers a search of the knowledge base of an expert system for the appropriate set of corrective measures for recovering from this fault. The efficiency of this search can be increased by classifying the set of probable faults that may occur in the manufacturing system.

A number of methodologies for developing shop floor control software have been discussed in this paper. The steps of these methodologies cover the phases of the software life-cycle and can be implemented with the help of a set of software tools that form the software environments of these methodologies. Tools for verifying the functional and timing requirements of manufacturing control software, prototyping, automatically generating subsets of the implementation code of this software, and simulating the performance of manufacturing systems have been integrated in these software environments. Table 1 is a summary of the major methodologies.

The choice of Petri nets for modeling the structural and behavioral properties of manufacturing control software permits the use of the theoretical properties of these nets in software verification. This choice, however, offers a limited support for planning, scheduling, and monitoring the operations of efficient and dependable manufacturing systems. The creation of a Petri net model for a manufacturing system is equivalent to planning the sequence of operations of a job. Firing the transitions of this Petri net is equivalent to scheduling the execution of the operations of this job whenever performing these operations becomes

Table 1. Approaches to developing manufacturing control software.

Acronym	Specification Language	Environment/ Tools	Developer/ Marketer	Status
System 90	Zone logic	Industrial computers	Septor	Commercial product
ISIS/OPIS	Rule-based	Unix	Fox & Smith	Academic research
KBSc	STRIPS	Unix	Shaw & Whinston	Academic research
AISPE	PROT nets	Unix	Bruno et al.	Academic research
FLEXIS DESIGNER	GRAFCETs	Unix	Savoir	Commercial product
o-r Petri nets	o-r Petri nets	Unix	Krogh et al.	Academic research
Colored Petri nets	Colored Petri nets	Unix	Many	Academic research
SA Colored Petri nets	SA Colored Petri nets	Unix	Many	Academic research
Extended Petri nets	Extended Petri nets	Unix	Many	Academic research
SECOIA	High-level Petri nets/rules	Unix	Courvoisier et al.	Academic research
Integrated approach to efficient and dependable manufacturing systems	Component-oriented rule-based language	Unix	Chaar, Volz & Davidson	Academic research

feasible. Monitoring is performed by repeatedly checking the set of conditions that precede the execution of every operation of the control sequence of the net. A fault is detected whenever the actual conditions and the net conditions are not identical, and this fault is handled by executing the appropriate set of operations, as specified by this net.

It is clear from the synopsis of the selected approaches mentioned in the paper that there has been a great deal of effort devoted to methods, algorithms, and software packages for manufacturing, which is a very diverse area. Hence, designing efficient and dependable manufacturing systems has always been a major goal of modern computer-integrated manufacturing. However, current efforts have not resulted yet in a consistent and integrated approach to developing manufacturing control software. In particular, these methods have not yet been widely used and where applied little feedback and published data are available. Also, the impact of these methods has been masked by political, economic, labor, and organizational considerations. However, this does not mean that these methods will not have a positive impact on future manufacturing when properly implemented.

An integrated approach to planning, scheduling, and monitoring the operations of efficient and dependable manufacturing systems is essential to the development of the shop floor control software of these systems. In these systems, efficiency is achieved by increasing the level of concurrency of the operations of a process plan, and by scheduling the execution of these operations with the intent of maximizing the utilization of the resources of their systems. On the other hand, dependability requires monitoring the operations of these systems. This monitoring activity facilitates the detection of any faults that may occur when executing the scheduled operations of a plan, recovering from these faults, and, whenever feasible, resuming the original schedule of the system.

Consequently, the life-cycle model of control software must include all of these activities, and a methodology for developing this software must provide a coherent approach to performing these activities. Such methodology together with a set of software tools that enhance its applicability have been proposed by Chaar et al. (1990, 1991). The tools aid in planning the set of operations of a manufacturing job, generating a cyclic schedule for processing a batch of jobs, and monitoring the operations of the system while this batch is being processed.

The syntax and semantics of a component-oriented rule-based language for specifying the formal models of manufacturing systems has been proposed (see Chaar and Volz 1989). A model captures the state of a component of the system in a set of first-order logic predicates, and it captures the semantics of the operations performed by this component in a set of rules that determine the preconditions and postconditions of an operation (see Naylor and Maletz 1986; Naylor and Volz 1987; Ben Hadj-Alouane et al. 1990). These models are used in planning the sequence of operations of each class of jobs to be manufactured by these systems.

To achieve efficiency, the reservation table technique is used to create optimum cyclic job-shop schedules for processing a batch of identical jobs or a mix of jobs from several classes on these systems (see Chaar and Davidson 1990). A reservation table is derived from the plan of a job. This table is then used to determine the theoretical maximum job initiation rate and the set of all possible initiation strategies for the batch. In some cases, this theoretical maximum rate is achieved by increasing the flow time of the job. The above technique inherently allows multiple devices to be reserved concurrently, it can deal with transport time explicitly, and it achieves higher initiation rates by including cycles that involve multiple job initiations.

To achieve dependability, a plan-oriented fault detection and correction strategy is proposed. This strategy can automatically handle any combination of faults that may occur when monitoring the operations of manufacturing systems. A fault-tree is consulted prior to executing the scheduled operations of a plan, and the faults that affect the execution of these operations are handled subsequently. Resuming the original cyclic schedule is attempted, whenever feasible.

References

Adlemo, A., Andréasson, S.-A, Andréasson, T. and Carlsson, C., "Achieving Fault Tolerance in Factory Automation Systems by Dynamic Configuration," *The Proceedings of The First International Conference on Systems Integration*, Morristown, New Jersey, pp. 396-402 (April 23-26, 1990).

Agnetis, A., Arbib, C., Lucertini, M. and Nicoló, F., "Part Routing in Flexible Assembly Systems," *IEEE Transactions on Robotics and Automation*, Vol. 6, No. 6, pp. 697-705 (December 1990).

Agnetis, A., Arbib, C. and Stecke, K.E., "Optimal Two-Machine Scheduling in a Flexible Flow System," *The Proceedings of the 1990 International Conference on Computer Integrated Manufacturing*, Troy, New York (May 21-23, 1990).

Ahuja, J.S. and K.P., Valavanis, "A Hierarchical Modeling Methodology for Flexible Manufacturing Systems Using Extended Petri Nets," *The Proceedings of the 1988 International Conference on Computer Integrated Manufacturing*, Troy, New York, pp. 350-356 (May 23-25, 1988).

Allen, D.K., "An Introduction to Computer-Aided Process Planning," *CIM Review*, pp. 7-23 (Fall 1987).

Arbib, C., Lucertini, M. and Nicoló., F., "Work-Load Balance and Part-Transfer Minimization in Flexible Manufacturing Systems," *International Journal of Flexible Manufacturing Systems*, Vol. 3, No. 1, pp. 5-25 (February 1991).

Atabakhche, H., Simonetti Barbalho, D., Valette, R. and Courvoisier, M., "From Petri Net Based PLCs to Knowledge Based Control," *Proceedings of the IECON'86*, pp. 817-822 (1986).

Ayres, R.U., "Technology Forecast for CIM," *Manufacturing Review*, Vol. 2, No. 1, pp. 43-52 (March 1989).

Baker, A.D., "Complete Manufacturing Control Using a Contract Net: A Simulation Study," *The Proceedings of the 1988 International Conference on Computer Integrated Manufacturing*, Troy, New York, pp. 100-109 (May 23-25, 1988).

Bean, J.C., Birge, J.R., Mittenthal, J. and Noon, C.E., "Matchup Scheduling with Multiple Resources, Release Dates and Disruptions," *Operations Research*, Vol. 39, No. 3 (May-June 1991).

Beck, C.L., "Modeling and Simulation of Flexible Control Structures for Automated Manufacturing Systems," Master's thesis, The Robotics Institute, Carnegie-Mellon University (December 1985).

Beck, C.L. and Krogh, B.H., "Models for Simulation and Discrete Control of Manufacturing Systems," *The Proceedings of the 1986 IEEE International Conference on Robotics & Automation*, San Francisco, California, pp. 305-310 (March 1986).

Ben-Arieh, D.H., Moodie, C.L. and Chu, C.-C., "Control Methodology for FMS," *IEEE Journal of Robotics and Automation*, Vol. 4, No. 1, pp. 53-59 (February 1988).

Ben-Hadj-Alouane, N., Chaar, J.K. and Naylor, A.W., "The Design and Implementation of the Control and Integration Software of a Flexible Manufacturing System," *The Proceedings of The First International Conference on Systems Integration*, Morristown, New Jersey, pp. 494-502 (April 23-26, 1990).

Bispo, C.F.G. and Sentieiro, J.J., "An Extended Horizon Scheduling Algorithm for the Job-Shop Problem," *The Proceedings of the 1988 International Conference on Computer Integrated Manufacturing*, Troy, New York, pp. 249-252 (1988).

Bollinger, J.G. and Duffie, N.A., *Computer Control of Machines and Processes*, Addison-Wesley Publishing Company, pp. 369-420 (1988).

Booch, G., "Object-Oriented Development," *IEEE Transactions on Software Engineering*, Vol. SE-12, No. 2, pp. 211-221 (February 1986).

Bourne, D.A. and Fox, M.S., "Autonomous Manufacturing: Automating the Job-Shop," *Computer*, Vol. 17, No. 9, pp. 76-86 (September 1984).

Breugnot, D., Gourgand, M., Hill, D. and Kellert, P., "GAME: An Object-Oriented Approach to Computer Animation in Flexible Manufacturing System Modelling," Rutan, A.H., (ed.), *The Proceedings of the 24th Annual Simulation Symposium*, New Orleans, Louisiana, pp. 217-227 (April 1991).

Bruno, G., "Using Ada for Discrete Event Simulation," *Software Practice and Experience*, Vol. 14, No. 7, pp. 685-695 (July 1984).

Bruno, G. and Biglia, P., "Performance Evaluation and Validation of Tool Handling in Flexible Manufacturing Systems Using Petri Nets," *1985 International Workshop on Timed Petri Nets*, pp. 64-71 (1985).

Bruno, G. and Marchetto, G., "Rapid Prototyping of Control Systems Using High Level Petri Nets," *The Proceedings of the 8th International Conference on Software Engineering*, pp. 230-235 (August 1985).

Bruno, G. and Balsamo, A., "Petri Net-Based Object-Oriented Modeling of Distributed Systems," *OOPSLA '86: Object-Oriented Programming Systems, Languages and Applications Conference Proceedings*, Portland, Oregon, pp. 284-293 (1986).

Bruno, G. and Elia, A., "Operational Specification of Process Control Systems: Execution of PROT Nets Using OPS5," Kugler, H.-J., (ed.), *Information Processing (IFIP) 86*, pp. 35-40 (1986).

Bruno, G. and Marchetto, G., "Process-Translatable Petri Nets for the Rapid Prototyping of Process Control Systems," *IEEE Transactions on Software Engineering*, Vol. SE-12, No. 2, pp. 346-357 (February 1986).

Bruno, G. and Morisio, M., "The Role of Rule Based Programming for Production Scheduling," *1987 IEEE International Conference on Robotics and Automation*, Raleigh, North Carolina, pp. 545-550 (March 1987a).

Bruno, G. and Morisio, M., "Petri-Net Based Simulation of Manufacturing Cells," *1987 IEEE International Conference on Robotics and Automation*, Raleigh, North Carolina, pp. 1174-1179 (March 1987b).

Bu Hulaiga, M.I. and Chakravarty, A.K., "An Object-Oriented Knowledge Representation for Hierarchical Real-Time Control of Flexible Manufacturing," *International Journal of Production Research*, Vol. 26, No. 5, pp. 777-793 (1988).

Camurri, A. and Franchi, P., "An Approach to the Design of the Hierarchical Control System of FMS, Combining Structured Knowledge Representation Formalisms and High-Level Petri Nets," *The Proceedings of the 1990 IEEE International Conference on Robotics & Automation*, Cincinnati, Ohio, pp. 520-525 (May 1990).

Chaar, J.K. and Volz, R.A., "On the Ada Implementation of a Component-Oriented Rule-Based Specification Language for Manufacturing Systems Control Software," *The Proceedings of the Fifth Annual Conference on Artificial Intelligence and Ada (AIDA-89)*, Washington, D.C., pp. 39-50 (November 1989).

Chaar, J.K. and Davidson, E.S., "Cyclic Job Shop Scheduling Using Reservation Tables," *The Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, Cincinnati, Ohio, pp. 2128-2135 (May 1990).

Chaar, J.K., "*A Methodology for Developing Real-Time Control Software for Efficient and Dependable Manufacturing Systems,*" PhD thesis, The University of Michigan, Ann Arbor, Michigan, Technical Report RSD-TR-20-90 (1990).

Chaar, J.K., Volz, R.A. and Davidson, E.S., "An Integrtaed Approach to Developing Manufacturing Control Software," *The Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, California, pp. 1979-1984 (April 1991).

Chang, S.J., DiCesare, F. and Goldbogen, G., "The Generation of Diagnostic Heuristics for Automated Error Recovery in Manufacturing Workstations," *The Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, Scottsdale, Arizona, pp. 522-527 (May 1989).

Chang, S.J., DiCesare, F. and Goldbogen, G., "Analysis of Diagnosability of Failure Knowledge in Manufacturing Systems," *The Proceedings of the 1990 IEEE International Conference on Robotics & Automation*, Cincinnati, Ohio, pp, 696-701 (May 1990).

Chintamaneni, P.R., Jalote, P., Shieh, Y.-B and Tripathi, S.K., "On Fault Tolerance in Manufacturing Systems," *IEEE Network*, Vol. 2, No. 3, pp. 32-39 (May 1988).

Corbeel, D., Gentina, J.C. and Vercauter, C., "Application of An Extension of Petri Nets to Modelization of Control and Production Processes," *Advances in Petri Nets*, pp. 162-180 (1985).

Crockett, D., Derochers, A., DiCesare, F. and Ward, T., "Implementation of a Petri Net Controller for a Machining Workstation," *The Proceedings of the 1987 IEEE International Conference on Robotics & Automation*, Raleigh, North Carolina, pp. 1861-1867 (March 1987).

Debolt, J.R., "Paperless Production via CIM Software," *The Proceedings of the Automated Clean Room Processes*, pp. 241_1-241_8 (March 1988).

Descotte, Y. and Latombe, J.-C., "Making Compromises among Antagonist Constraints in a Planner," *Artificial Intelligence*, Vol. 27, pp. 183-217 (1985).

Devedžić, V., "A Knowledge-Based System for the Strategic Control Level of Robots in Flexible Manufacturing Cells," *International Journal of Flexible Manufacturing Systems*, Vol. 2, No. 4, pp. 263-287 (July 1990).

Dove, R.K., "Process Design Automation: The Key to the 90's," *The Proceedings of AUTOFACT'88 Conference*, Chicago, Illinois, pp. 10-13-10-30 (October 30-November 2, 1988).

Dubois, D. and Stecke, K.E., "Dynamic Analysis of Repetitive Decision-Free Discrete-Event Processes: The Algebra of Timed Marked Graphs and Algorithmic Issues," *Annals of Operations Research*, Vol. 26, pp. 151-193 (1990a).

Dubois, D. and Stecke, K.E., "Dynamic Analysis of Repetitive Decision-Free Discrete-Event Processes: Applications to Production Systems," *Annals of Operations Research*, Vol. 26, pp. 323-347 (1990b).

Duffie, N.A. and Piper, R.S., "Nonhierarchical Control of Manufacturing Systems," *Journal of Manufacturing Systems*, Vol. 5, No. 2, pp. 137-139 (1986).

Duggan, J. and Browne, J., "ESPNET: Expert-System-Based Simulator of Petri Nets," *IEE Proceedings D*, Vol. 135, No. 4, pp. 239-247 (July 1988).

Ekberg, G. and Krogh, B.H., "Prototype Software for Automatic Generation of On-Line Control Programs for Discrete Manufacturing Processes," Technical Report CMU-RI-TR-87-3, The Robotics Institute, Carnegie-Mellon University (1987).

Elsenbrown, B., "Programmable Controllers Move to Systems Solutions," *Manufacturing Engineering*, pp. 59-61 (January 1988).

Farnsworth, K.D., Norman, V.B. and Norman, T.A., "Integrated Software for Manufacturing Simulation," Wilson, J., Henriksen, J. and Roberts, S., (eds.), *The Proceedings of the 1986 Winter Simulation Conference*, Washington, D.C., pp. 184-191 (1986).

Fikes, R.E., "Monitored Execution of Robot Plans Produced by STRIPS," *The Proceedings of the IFIP Congress 71*, Ljubljana, Yugoslavia, pp. 189-194 (August 23-28, 1971).

Fikes, R.E. and Nilsson, N.J., "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," *Artificial Intelligence*, Vol. 2, pp. 189-208 (1971).

Fikes, R.E., Hart, P.E. and Nilsson, N.J., "Learning and Executing Generalized Robot Plans," *Artificial Intelligence*, Vol. 3, pp. 251-288 (1972).

Fisher, J.P., "Zone Logic—Increased Machine Productivity through Artificial Intelligence," *The Proceedings of the 18th Annual International Programmable Controllers (IPC) Conference* (April 1989).

Garey, M.R., Johnson, D.S. and Sethi, R., "The Complexity of Flowshop and Jobshop Scheduling," *Mathematics of Operations Research*, Vol. 1, pp. 117-129 (1976).

Garey, M.R. and Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, New York (1979).

Gayman, D.J., "An Old Favorite Gets New Standards," *Manufacturing Engineering*, pp. 55-58 (January 1988).

Gentina, J.C. and Corbeel, D., "Coloured Adaptive Structured Petri Net: A Tool for the Automatic Synthesis of Hiearchical Control of Flexible Manufacturing Systems," *The Proceedings of the 1987 IEEE International Conference on Robotics & Automation*, Raleigh, North Carolina, pp. 1166-1173 (March 1987).

Georgeff, M.P., "Planning," *Annual Review of Computer Sciences*, Vol. 2, pp. 359-400 (1987).

Gilman, A.R. and Billingham, C., "A Tutorial on SEE WHY and WHITNESS," MacNair, E.A., Musselman, K.J. and Heidelberger, P. (eds.), *The Proceedings of the 1989 Winter Simulation Conference*, Scottsdale, Arizona, pp. 192-200 (1989).

Gonzalez, T. and Sahni, S., "Open Shop Scheduling to Minimize Finish Time," *Journal of the Association of Computing Machinery*, Vol. 23, pp. 665-679 (1976).

Graves, S.C., Meal, H.C., Stefek, D. and Zeghmi, A.H., "Scheduling of Re-Entrant Flow Shops," *Journal of Operations Management*, Vol. 3, No. 4, pp. 197-207 (August 1983).

Group de Travail Systèmes Logiques de l'AFCET, "Pour une Représentation Normalisée du Cahier des Charges d'un Automatisme Logique," *Automatique et Informatique Industrielles*, 6(61/62) (November/December 1977).

Hancock, T.M., "Integration of Design, Planning, and Manufacturing Subsystems in Sheet Processing," *The Proceedings of the 1988 International Conference on Computer Integrated Manufacturing*, Troy, New York, pp. 138-143 (May 23-25, 1988).

Harhalakis, G., Lin, C.P., Moy, K.Y. and Hillion, H., "A Facility-Level CIM System," *The Proceedings of the 1988 International Conference on Computer Integrated Manufacturing*, Troy, New York, pp. 253-263 (May 23-25, 1988).

Hasegawa, M., Takata, M., Temmyo, T. and Matsuka, H., "Modeling of Exception Handling in Manufacturing Cell Control and its Application to PLC Programming," *The Proceedings of the 1990 IEEE International Conference on Robotics & Automation*, Cincinnati, Ohio, pp. 514-519 (May 1990).

Healy, K.J., "CINEMA Tutorial," Wilson, J., Henriksen, J. and Roberts, S., (eds.), *The Proceedings of the 1986 Winter Simulation Conference*, Washington, D.C., pp. 207-211 (1986).

Hollinger, D. and Bel, G., "An Object-Oriented Approach for CIM Systems Specification and Simulation," bϕ, K. Estensen, I., Falster, P. and Warman, E.A., (eds.), *Computer Applications in Production and Engineering CAPE'86*, IFIP, pp. 445-460 (1987).

Huber, A. and Buenz, D., "Using GRAI to Specify Expert Systems for the Control and Supervision of Flexible Flow Lines," Browne, J., (ed.), *Knowledge Based Production Management Systems (IFIP) 86*, pp. 295-308 (1988).

International Electrotechnical Commission, "Preparation of Function Charts for Control Systems," (November 1984).

Iscoe, N., Williams, G.B. and Arango, G., "Domain Modeling for Software Engineering," *Proceedings of the 13th International Conference on Software Engineering*, Austin, Texas, pp. 340-343 (May 1991).

Jablonski, S., Reinwald, B., Ruf, T. and Wedekind, H., "Flexible and Reactive Integration of Manufacturing Systems through Dynamic Resource Allocation," *The Proceedings of The First International Conference on Systems Integration*, Morristown, New Jersey, pp. 503-509 (April 1990).

Jafari, M.A., "Petri Net Based Shop Floor Controller and Recovery Analysis," *The Proceedings of the 1990 IEEE International Conference on Robotics & Automation*, Cincinnati, Ohio, pp. 532-537 (May 1990).

Jain, S. and Foley, W.J., "Basis for Development of A Generic FMS Simulator," Stecke, K.E. and Suri, R., (eds.), *Proceedings of the Second ORSA/TIMS Conference on Flexible Manufacturing Systems: Operations Research Models and Applications*, The University of Michigan, Ann Arbor, pp. 393-403 (August 12-15, 1986).

Jones, J.E., White, D.R., Xiaoshu, X., Oberly, P.A. and Funk, T.L., "Development of an Off-Line Weld Planning System," *Texas Instruments Technical Journal*, pp. 47-53 (Winter 1987).

Judd, R.P., Vanderbok, R.S., Brown, M.E. and Suter, J.A., "Manufacturing System Design Methodology: Execute the Specification," *Proceedings of CIMCON'90-NIST Special Publication 785*, pp. 135-152 (May 1990).

Jun, J.S. and McLean, R., "Control of an Automated Machining Workstaton," *IEEE control Systems Magazine*, pp. 26-30 (February 1988).

Kamath, M. and Viswanadham, N., "Applications of Petri Net Based Models in the Modelling and Analysis of Flexible Manufacturing Systems," *The Proceedings of the 1986 IEEE International Conference on Robotics & Automation*, San Francisco, California, pp. 312-317 (March 1986).

Karmarkar, U.S. and Schrage, L., "The Deterministic Dynamic Product Cycling Problem," *Operations Research*, Vol. 33, No. 2, pp. 326-345 (March-April, 1985).

Kasturia, E., DiCesare, F. and Desrochers, A., "Real Time Control of Multilevel Manufacturing Systems Using Colored Petri Nets," *The Proceedings of the 1988 IEEE International Conference on Robotics & Automation*, Philadelphia, Pennsylvania, pp. 1114-1119 (April 1988).

Ketcham, M.G., Smith, J.M. and Nnaji, B.O., "An Integrated Data Model for CIM Planning and Control," *The Proceedings of the 1988 International Conference on Computer Integrated Manufacturing*, Troy, New York, pp. 338-342 (1988).

Klein, B., "A SIMFACTORY Tutorial," Wilson, J., Henriksen, J. and Roberts, S., (eds.), *The Proceedings of the 1986 Winter Simulation Conference*, Washington, D.C., pp. 193-196 (1986).

Klein, B., "Factory Planning and Production Analysis using SIMFACTORY," Abrams, M., Haigh, P. and Comfort, J., (eds.), *The Proceedings of the 1988 Winter Simulation Conference*, pp. 112-114 (1988).

Kompass, E.J., "Distributed Machine Control Uses Zoned Logic, Isolated Controllers, Fiber Optics," *Control Engineering* (August 1987).

Krogh, B.H., Willson, R. and Pathak, D., "Automated Generation and Evaluation of Control Programs for Discrete Manufacturing Processes," *The Proceedings of the 1988 International Conference on Computer Integrated Manufacturing*, Troy, New York, pp. 92-99 (May 23-25, 1988).

Larner, D.L., "Factories, Objects & Blackboards," *AI Expert*, pp. 38-45 (April, 1990).

Le Pape, C. and Smith, S.F., "Management of Temporal Constraints for Factory Scheduling," Technical Report CMU-RI-TR-87-13, The Robotics Institute, Carnegie-Mellon University (June 1987).

Lee, K. and Yang, C., "Mare Island Flexible Manufacturing Workstation," *The Proceedings of the 1988 International Conference on Computer Integrated Manufacturing*, Troy, New York, pp. 9-18 (1988).

Lenstra, J.K., Rinnooy Kan, A.H.G. and Brucker, P., "Complexity of Machine Scheduling Problems," *Annals of Discrete Mathematics*, Vol. 1, pp. 343-362 (1977).

Levas, A. and Jayaraman, R., "WADE: An Object-Oriented Environment for Modeling and Simulation of Workcell Applications," Technical report, Manufacturing Research Department, IBM Research Division, T.J. Watson Research Center (November 1987).

Liu, Z. and Labetoulle, J., "A Heuristic Method for Loading and Scheduling Flexible Manufacturing Systems," *The Proceedings of the International Conference on Control '88*, Austin, Texas, pp. 195-200 (April 1988).

Mamalis, A.G., Bilalis, N.G. and Konstantinidis, M.J., "On Simulation Modeling of FMS," *Simulation*, Vol. 48, No. 1, pp. 19-23 (January 1987).

Martínez, J., Muro, P. and Silva, M., "Modeling, Validation and Software Implementation of Production Systems Using High Level Petri Nets," *The Proceedings of the 1987 IEEE International Conference on Robotics & Automation*, Raleigh, North Carolina, pp. 1180-1185 (March 1987).

Maxwell, W.L. and Singh, H., "Scheduling Cyclic Production on Several Identical Machines," *Operations Research*, Vol. 34, No. 3, pp. 460-463 (May-June 1986).

McCormick, S.T., Pinedo, M.L., Shenker, S. and Wolf, B., "Sequencing in an Assembly Line with Blocking to Minimize Cycle Time," Technical report, School of Operations Research and Industrial Engineering, Columbia University (1986).

McDermott, D.V., "Logic, Problem Solving, and Deduction," *Annual Review of Computer Sciences*, Vol. 2, pp. 187–229 (1987).

Meyer, W., "Knowledge-Based Realtime Supervision in CIM—The Workcell Controller," *ESPRIT'86: Results and Achievements*, pp. 33–52 (1986).

Moore, R.L., Hawkinson, L.B., Knickerbocker, C.G. and Churchman, L.M., "A Real-Time Expert System for Process Control," *Proceedings of the First Conference on Artificial Intelligence Applications*, pp. 569–574 (1984).

Moyne, J.R., McAfee, Jr., L.C. and Teorey, T.J., "An Application of Entity-Relationship Modeling Techniques to the Automated Manufacturing Process," *Proceedings of the Second International Conference on Data and Knowledge Systems for Manufacturing and Engineering*, Gaithersburg, Maryland, pp. 206–215 (October 1989).

Moyne, J.R., Teorey, T.J. and McAfee, Jr., L.C., "Time Sequence Ordering Extensions to the Entity Relationship Model and their Application to the Automated Factory Process," *Data and Knowledge Engineering*, Vol. 6, No. 5, pp. 421–433 (September 1991).

Nau, D.S., "Automated Process Planning Using Hierarchical Abstraction," *Texas Instruments Technical Journal*, pp. 39–46 (Winter 1987).

Naylor, A.W. and Maletz, M.C., "The Manufacturing Game: A Formal Approach to Manufacturing Software," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-16, No. 3, pp. 321–334 (May/June 1986).

Naylor, A.W. and Volz, R.A., "Design of Integrated Manufacturing System Control Software," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-17, No. 6, pp. 881–897 (November/December 1987).

O'Reilly, J.J. and Nordlund, K.C., "Introduction to SLAM II and SLAMSYSTEM," MacNair, E.A., Musselman, K.J. and Heidelberger, P. (eds.), *The Proceedings of the 1989 Winter Simulation Conference*, Washington, D.C., pp. 178–183 (1989).

Ow, P.S., Smith, S.F. and Thiriez, A., "Reactive Plan Revision," *Proceedings AAAI88 Seventh National Conference on Artificial Intelligence*, St. Paul, Minnesota, pp. 77–82 (August 21–26, 1988).

Papadimitriou, C.H. and Kanellakis, P., "Flowshop Scheduling with Limited Temporary Storage," *Journal of the Association of Computing Machinery*, Vol. 27, pp. 533–549 (1980).

Pathak, D.K., "Automated Development of Control Software for Discrete Manufacturing Systems," Master's thesis, The Robotics Institute, Carneigie-Mellon University (May 1988).

Pinto, P.A., Dannenbring, D.G. and Khumawala, B.M., "Assembly Line Balancing with Processing Alternatives: An Application," *Management Science*, Vol. 29, No. 7, pp. 817–830 (July 1983).

Poorte, J.P. and Davis, D.A., "Computer Animation with CINEMA," MacNair, E.A., Musselman, K.J. and Heidelberger, P. (eds.), *The Proceedings of the 1989 Winter Simulation Conference*, Washington, D.C., pp. 147–154 (1989).

Preiss, K. and Shai, O., "Process Planning by Logic Programming," *Robotics & Computer-Integrated Manufacturing*, Vol. 5, No. 1, pp. 1–10 (1989).

Roundy, R., "Cyclic Schedules for Job Shops with Identical Jobs," Technical Report 766, School of Operations Research and Industrial Engineering, Cornell University (July 1988).

Royce, W.W., "Managing the Development of Large Software Systems: Concepts and Techniques," *Proceedings of the 9th International Conference on Software Engineering (a reprint of 1970 article)*, Monterey, California, pp. 328–338 (March 1987).

Ruiz-Mier, S. and Talavage, J., "A Hybrid Paradigm for Modeling of Complex Systems," *Simulation*, Vol. 48, No. 4, pp. 135–141 (April 1987).

Russell, D., "Integration of PLCs and Databases for Factory Information Systems," *The Proceedings of The First International Conference on Systems Integration*, Morristown, New Jersey, pp. 730–737 (April 23–26, 1990).

Sacerdoti, E.D., "Planning in a Hierarchy of Abstraction Spaces," *Artificial Intelligence*, Vol. 5, pp. 115–135 (1974).

Sahni, S. and Cho, Y., "Complexity of Scheduling Shops with No Wait in Process," Technical Report 77–20, Computer Science Department, The University of Minnesota (1977).

Sahraoui, A., Courvoisier, M. and Valette, R., "Some Considerations on Monitoring in Distributed Real-Time Control of Flexible Manufacturing Systems," *Proceedings of the IECON'86*, pp. 805–810 (1986).

Sahraoui, A., Atabakhche, H., Courvoisier, M. and Valette, R., "Joining Petri Nets and Knowledge Based Systems for Monitoring Purposes," *The Proceedings of the 1987 IEEE International Conference on Robotics & Automation*, Raleigh, North Carolina, pp. 1160–1165 (March 1987).

Savolainen, T., "Software Technologies in Computer Integrated Manufacturing," *Computers in Industry*, Vol. 11, No. 1, pp. 3–22 (November 1988).

Schriber, T.J. and Stecke, K.E., "Using Mathematical Programming and Simulation to Study FMS Machine Utilizations," Thesen, A., Grant, H. and Kelton, W.D., (eds.), *The Proceedings of the 1987 Winter Simulation Conference*, Atlanta, Georgia, pp. 725-730 (1987).

Schriber, T.J. and Stecke, K.E., "Machine Utilizations Achieved Using Balanced FMS Production Ratios in a Simulated Setting," *Annals of Operations Research*, Vol. 15, pp. 229-267 (1988).

Shaw, M.J. and Whinston, A.B., "Automated Planning and Flexible Scheduling: A Knowledge-Based Approach," *The Proceedings of the 1985 International Conference on Automation & Robotics*, St. Louis, Missouri, pp. 890-894 (1985a).

Shaw, M.J. and Whinston, A.B., "Task Bidding and Distributed Planning in Flexible Manufacturing Systems," *Proceedings of the IEEE Second Conference on Artificial Intelligence Applications*, Miami, Florida, pp. 184-189 (1985b).

Shaw, M.J. and Whinston, A.B., "Application of Artificial Intelligence to Planning and Scheduling in Flexible Manufacturing," Kusiak, A., (ed.), *Flexible Manufacturing Systems: Methods and Studies*, pp. 223-242 (1986).

Shaw, M.J., "Knowledge-Based Scheduling in Flexible Manufacturing Systems," *Texas Instruments Technical Journal*, pp. 54-61 (Winter 1987).

Shaw, M.J. and Whinston, A.B., "A Distributed Knowledge-Based Approach to Flexible Automation: The Contract Net Framework," *International Journal of Flexible Manufacturing Systems*, Vol. 1, No. 1, pp. 85-104 (September 1988).

Shin, K.G. and Zheng, Q., "Scheduling Job Operations in an Automatic Assembly Line," *The Proceedings of the 1990 IEEE International Conference on Robotics & Automation*, Cincinnati, Ohio, pp. 176-181 (May 1990).

Shires, N. and Bastos, J.-M., "Factory Control Integrated with Operational Planning," *The Proceedings of the International Conference on Control '88*, Austin, Texas, p. 201-206 (April 1988).

Smith, S.F., "A Constraint-Based Framework for Reactive Management of Factory Schedules," Oliff, M.D., (ed.), *Proceedings of the First International Conference on Expert Systems and the Leading Edge in Production Planning and Control*, pp. 113-130 (1988).

Stecke, K.E. and Solberg, J.J., "Loading and Control Policies for a Flexible Manufacturing System," *International Journal of Production Research*, Vol. 19, No. 5, pp. 481-490 (1981).

Stecke, K.E., "Design, Planning, Scheduling, and Control Problems of Flexible Manufacturing Systems," *Annals of Operations Research*, Vol. 3, pp. 3-12 (1985).

Stecke, K.E. and Kim, I., "A Flexible Approach to Part Selection in Flexible Flow Systems Using Part Mix Ratios," *International Journal of Production Research*, Vol. 29, No. 1, pp. 53-75 (1991).

Sturrock, D.T. and Pegden, C.D., "Introduction to SIMAN," MacNair, E.A., Musselman, K.J. and Heidelberger, P. (eds.), *The Proceedings of the 1989 Winter Simulation Conference*, Washington, D.C., pp. 129-139 (1989).

Subramanyam, S. and Askin, R.G., "An Expert Systems Approach to Scheduling in Flexible Manufacturing Systems," Kusiak, A., (ed.), *Flexible Manufacturing Systems: Methods and Studies*, pp. 243-256 (1986).

Takata, S. and Sata, T., "Model Referenced Monitoring and Diagnosis—Application to the Manufacturing System," *Computers in Industry*, Vol. 7, pp. 31-43 (1986).

Teng, S.-H. and Black, J.T., "An Expert System for Manufacturing Cell Control," *Computers & Industrial Engineering*, Vol. 17, No. 1-4, pp. 18-23 (1989).

Thomas, B.H. and McLean, C., "Using GRAFCET to Design Generic Controllers," *The Proceedings of the 1988 International Conference on Computer Integrated Manufacturing*, Troy, New York, pp. 110-119 (May 23-25, 1988).

Ting, J.J., "A Cooperative Shop-Floor Control Model for Computer-Integrated Manufacturing," *Proceedings of CIMCON'90—NIST Special Publication 785*, pp. 446-465 (May 1990).

Van Brussel, H., Cottrez, F. and Valckenaers, P., "SESFAC: A Scheduling Expert System for Flexible Assembly Cells," *The Proceedings of the 1990 IEEE International Conference on Robotics & Automation*, Cincinnati, Ohio, pp. 1950-1955 (May 1990).

Vasilash, G.S., "Rule-Based Breakthrough for Transfer Line Control," *Production*, pp. 34-37 (May 1987).

Viswanadham, N. and Narahari, Y., "Coloured Petri Net Models for Automated Manufacturing Systems," *The Proceedings of the 1987 IEEE International Conference on Robotics & Automation*, Raleigh, North Carolina, pp. 1985-1990 (March 1987).

Viswanadham, N. and Johnson, T.J., "Fault Detection and Diagnosis of Automated Manufacturing Systems," *The Proceedings of the 27th Conference on Decision and Control*, Austin, Texas, pp. 2301-2306 (December 1988).

Wadhwa, S. and Browne, J., "Modeling FMS with Decision Petri Nets," *International Journal of Flexible Manufacturing Systems*, Vol. 1, No. 3, pp. 255–280 (July 1989).

Wang, H.-P. and Wysk, R.A., "A Knowledge-Based Approach for Automated Process Planning," *International Journal of Production Research*, Vol. 26, No. 6, pp. 999–1014 (1988).

Weber, D.M. and Moodie, C.L., "A Knowledge-Based System for Information Management in an Automated and Integrated Manufacturing System,"' *Robotics & Computer-Integrated Manufacturing*, Vol. 4, No. 3/4, pp. 601–617 (1988).

Weck, M., "Machine Diagnostics in Automated Production," *Journal of Manufacturing Systems*, Vol. 2, No. 2, pp. 101–106 (1983).

Wilczynski, D., "A Common Device Control Architecture—The Savoir Actor," *The Proceedings of the AUTOFACT'88 Conference*, Chicago, Illinois, pp. 10-3–10-40 (October 30–November 2, 1988).

Willson, R.G. and Krogh, B.H., "Petri Net Tools for the Specification and Analysis of Discrete Controllers," *IEEE Transactions on Software Engineering*, Vol. 16, No. 1, pp. 39–50 (January 1990).

Wolter, J.D., "On The Automatic Generation of Assembly Plans," *The Proceedings of the 1989 IEEE International Conference on Robotics & Automation*, Scottsdale, Arizona, pp. 62–68 (May 1989).

Zhou, M., DiCesare, F. and Desrochers, A.A., "A Top-Down Approach to Systematic Synthesis of Petri Net Models for Manufacturing Systems," *The Proceedings of the 1989 IEEE International Conference on Robotics & Automation*, Scottsdale, Arizona, pp. 534–539 (May 1989).

Zhou, M.C. and DiCesare, F., "A Petri Net Design Method for Automated Manufacturing Systems with Shared Resources," *The Proceedings of the 1990 IEEE International Conference on Robotics & Automation*, Cincinnati, Ohio, pp. 526–531 (May 1990).