Towards Design and Control of High Autonomy Manufacturing Systems

Jerzy W. Rozenblit

Dept. of Electrical and Computer Engr.
The University of Arizona
Tucson, AZ 85721, U.S.A

Witold Jacak

Institute of Systems Science
Johannes Kepler University
A-4040 Linz, Austria

Abstract

In this paper, requirements for design of high autonomy manufacturing systems are stipulated. Efforts towards amalgamating the autonomous architecture and its real world component (i.e., a flexible manufacturing system) are presented. Planning and control principles derived from discrete event modeling techniques are summarized.

1 Introduction

Autonomy as a design goal can be defined as the ability of a system to function independently, subject to its own laws and control principles [1, 7, 9, 24, 28]. A salient requirement in high autonomy systems design is the integration of *planning, scheduling, diagnosis, and control* functions. These functions, when integrated, can support the operation of a complete system. Architectures that foster this approach have been proposed in the literature by a number of authors [1, 22, 32, 38]. Whereas initial work in high autonomy systems stemmed from automatic intelligent control, new approaches emerge which take a distinct, simulation modeling approach [22, 36, 37, 38]. The generic autonomous architecture (described in detail in [7, 24, 36]) provides a framework in which the following design requirements can be met:

- the system must plan and re-plan to realize its goals
- the system must be able to execute its plans
- it must monitor its environment
- it must have cognitive and diagnostic capabilities

The architecture has three major layers: the *system interface, the planning, scheduling, and reasoning layer*, and *the control and sensing layer.* We refer the reader to [7, 24, 36] for more details.

Antsaklis et. al. [1] define functional modules of an autonomous control architecture as *management and organization level* that determines the overall system's goals and supports interaction with the system's external environment through the interface unit, *coordination level* that supports decision making, planning, and scheduling, and *execution level* that carries out control actions determined at higher levels through automatic controllers and actuators. Zeigler [36] encapsulates knowledge in the form of models that can be employed at different levels of control abstraction in an autonomous system to support its objectives. The resulting structure is termed *model-based architecture.*

In the ensuing sections we examine how the characteristics of such architectures are embedded in design, planning, and control of advanced manufacturing systems.

2 Automated Manufacturing Systems

In recent years the use of programmable, flexible automated machines and robots has enabled partial or complete automation of machining and assembly of products. The economic importance of manufacturing has led to extensive efforts to improve the efficiency and cost effectiveness of automated production system. More systematic approaches to design and planning of manufacturing systems and processes are needed to further enhance performance and enable their cost-effective, real world implementations.

A flexible manufacturing system (FMS) is a set of machines connected by a flexible material handling facility (such as a robot, a crane, or an automated

0-8186-2675-5/92 $3.00 © 1992 IEEE

guided vehicle) and controlled by a computer [19]. Flexible manufacturing systems (FMS) systems possess a number of unique features and characteristics which require that their design and operation strategies be substantially different from those used in conventional job shop and transfer line facilities [20]. One of the distinguishing features of a FMS is its degree of automation of machines and material handling systems.

FMSs are typically constructed as hierarchical structures [18, 23] whose functionality closely matches that of the autonomous architecture. Jones and McLean [18] proposed a generic manufacturing system called Automated Manufacturing Research Facility (AMRF). The underlying design philosophy in building AMRF was that the system had to exhibit high degree of control hierarchy, modularity, and high flexibility. The real-time production control system was designed along the following principles: a) the system was partitioned into a structure in which the control processes were functionally decomposed, b) it was designed to respond in real-time to performance data derived from sensors attached to machines, c) it was implemented in a distributed computer environment. We discuss an architecture derived from AMRF in detail in Section 3.

Maimon [23] defines a hierarchical system for controlling short-term production activities in an FMS. The system consists of the user interface, databases, the control and the communication layers. The user communicates with the system using a manufacturing language. The configuration, world status, process plans, and short-term requirements databases are used in the resource planning and spatial process optimization. The control layer employs a dynamic scheduler, the process sequencer, and the dynamic resource allocator. The scheduler determines which part to produce and a preferable route for it. The sequencer generates detailed internal movements of parts. The resource allocator ensures that parts are assigned resources in a conflict free manner. The communication layer is responsible for monitoring the production process, sensing, collecting, and transmitting data to higher levels. This system has been successfully tested on a number of case studies. Although one must specify a new control system for each implementation, the architecture facilitates efficient real-time problem solving at each level of the hierarchy.

In more recent work, Duan et. al. [6] specify EMM-networking (Extended Moore Machine) model for modeling hierarchical flexible manufacturing systems. They show how the model can be implemented in an object-oriented software environment for simulation, software development, and real-time control.

Combacau and Courvoisier [5] combine the rule-based problem solving paradigm and Petri net modeling formalism for control and monitoring of hierarchical FMSs. They successfully integrate diagnosis capabilities and real-time control.

Our current efforts focus on the development of a comprehensive framework for design, planning and control of automated (e.g., manufacturing, diagnostic, testing, etc.) systems [14, 15, 30]. The framework termed FMS CAD is to integrate several layers of support methods and tools such as automatic generation of different plans of sequencing operations, selection of devices to carry out the operations and their physical layout, synthesis and interpretation of robots' motion programs, simulation testing and verification of design variants, and simulation modeling of the overall manufacturing system architecture. In Section 4, we describe our progress in some of those areas. Prior to that, we explain the hierarchical FMS architecture in more detail.

3 Hierarchical FMS Architecture

Following the definition of AMRF, we are developing methods for workcell task planning and control [15, 16] in a hierarchical FMS architecture depicted in Figure 1.

The control hierarchy consists of the following levels: *Facility, Cell,* and *Workstation and Material Handling Equipment (e.g. Robot).* The levels in the control architecture have the following functions [18]:

- Facility: implements the manufacturing engineering, resource, information and task management functions. Manufacturing engineering functions are to a large extent carried out by human personnel. CAD tools are used to develop part specifications for the process planning system.
- The control functions at the cell level are: job sequencing, scheduling, material handling, supervision and coordination of the physical activities of workstations and robots.
- At the workstation level, machining, assembly, and material operations are performed.

Control mechanisms are established is such a way that an upper level component issues commands to its lower level descendants. It receives feedback upon

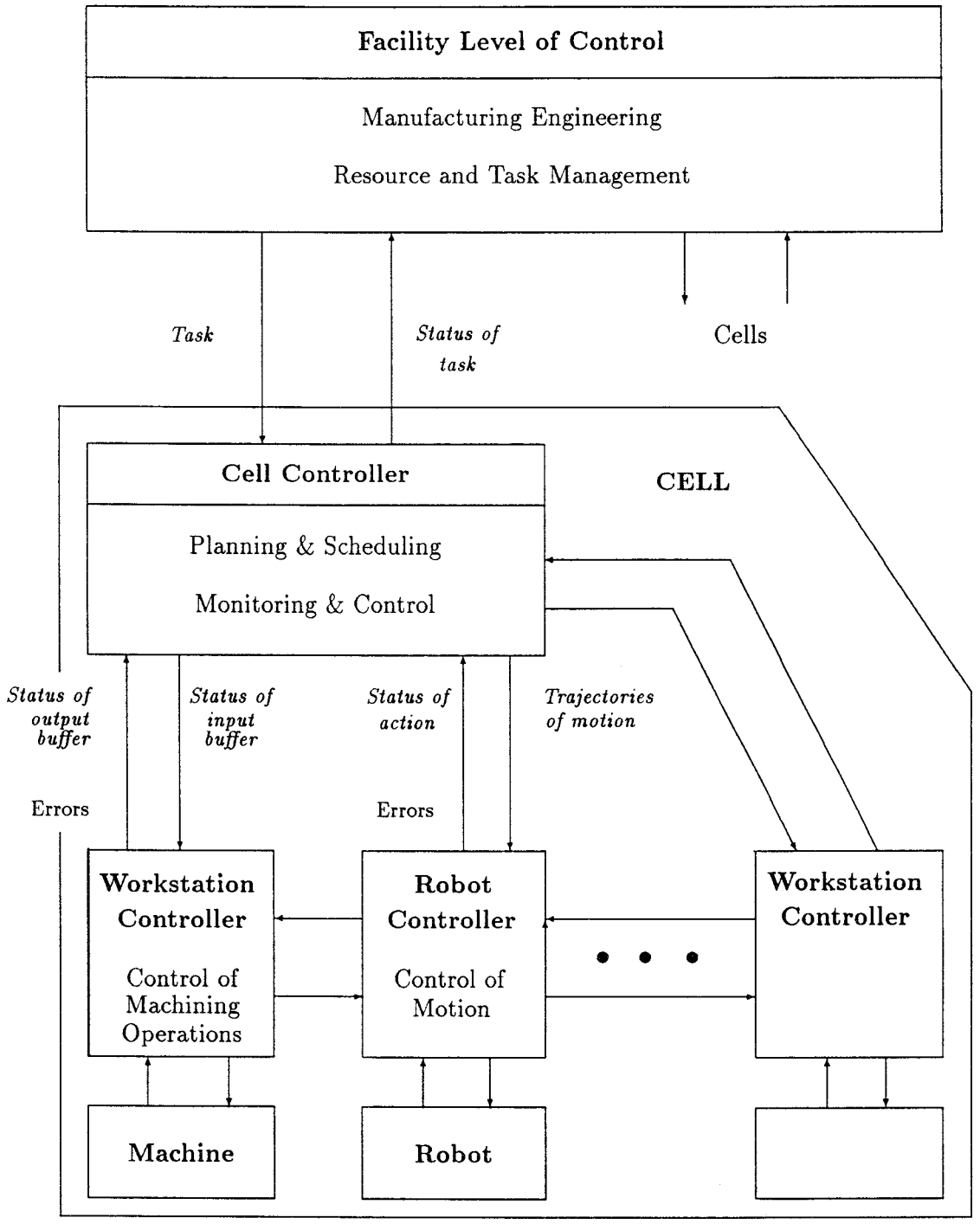

Figure 1: Hierarchical FMS Architecture

the completion of command execution by the descendents. The physical components at each level are computer systems and control devices, connected by a communication network such as a local area network (LAN) with a manufacturing automation protocol (MAP) [19].

To achieve high autonomy in the above architecture the extent of a system's interaction with its operator(s) should be minimized. The lesser the interaction (intervention of the human operator in the system's operation), the higher the system's autonomy [7]. Another autonomy criterion can be defined as the level of detail and abstraction that the human operator has to employ when assigning tasks and how long the workstations and robots can function on their own without any intervention from their operator [24].

Within the FMS CAD framework, we have been developing methods for simulation based workcell control. We now proceed to summarize our results.

4 Planning and Control of FMS Workcell

We have defined an intelligent control system of a manufacturing cell, which can plan tasks and motions of robots that service the cell. The system consists of two basic layers: the *Task Planning* and the *Task-Level Programming* layer. Task planning is based on the description of technological operations and their precedence relation. The resulting fundamental plan describes the decomposition of a manufacturing task into an ordered sequence of robot actions. The implementation of the plan is carried out using a *task-level programming* approach [14].

4.1 Task Level Planning

Task-Level Planning is carried out based on a description of the technological operations, a description of the workcell and its resources such as machines, robots, fixtures, or sensors, and a description of the precedence relation over the set of operations. The resulting fundamental cell action plan describes a decomposition of the task into an ordered sequence of technological operations called *Process*. Then, the *Process* sequence is translated into an ordered sequence of robot and workstation actions which are used to realize the task. The solution is generated as follows: Two subproblems are defined. The first subproblem consists in finding an ordered, feasible sequence of technological operations which can be transformed directly into the sequence of robot and workstation actions. In the second subproblem we define the set of preconditions for each action, which guarantee that a deadlock condition does not occur among the actions being executed. The fundamental cell-action plan for a machining task determines the robot's program of manipulations required to carry out this task. Such a program is a sequence of motion, grasp, and sensors instructions expressed in the *Task-Oriented Robot Programming Language* (TORPL) [8, 35].

Let us denote components of a FMS as a set of technological devices, D, called *workstations* and production *stores*, M, connected by a flexible material handling facility, R, (such as robots or automated guided vehicles).

Task planning for FMSs is critically dependent on the task representation. Several task representation schemes have been proposed in the literature [10]. They include: lists of operations, triangle tables, and AND/OR graphs. We represent a task as the following triple:

$$Task = (O\ ,\ \prec\ ,\ \alpha)$$

where: $O = \{o_1, .., o_L\}$ is a finite set of technological operations required to process the parts, $\prec$ is a partial order (precedence relation) on O and $\alpha \subset O \times D$ is a relation of device (machine) assignment.

The partial order represents an operational precedence, i.e., $q \prec o$ means that the operation q is to be completed before the operation o can begin. $(o, d) \in \alpha$ means that the operation o can be performed on the workstation d. The parts are transferred between machines by the robots which service the manufacturing cell. A robot ($r \in R$) can service only the machines which lie in its service space.

A process can be realized by different sequences of technological devices (called *resources*) required by successive operations from the list *Process* at the time of their execution. This set of new sequences, denoted $\mathbf{P}$, is called *production routes*. A production route $\mathbf{p} \in \mathbf{P}$ is an ordered list of resources which has $2L + 2$ stages, where L denotes the length of the list *Process*. The route is created on-line during the execution of technological operations. The production route is denoted by:

$$\mathbf{p} = (m_0, res(o_1), res(o_2), ..., res(o_L), m_F)$$

where $res(o_i) = \alpha(o_i) = d_i$ if there exists a robot r which can transfer parts directly from d_i to d_{i+1} and $res(o_i) = (d_i, m_i)$ if the robot r can transfer parts only from the production store m_i to d_{i+1}. We assume that there always exists a robot transferring parts from d_i

to m_i. By m_0 and m_F we denote the feeder and the output conveyor, respectively.

The production rate is directly related to the order of operations in *Process*. Moreover, a *circular wait* deadlock may occur between pipeline processes [19]. Techniques used to avoid deadlocks result in the increased job waiting time, and consequently, decrease the production rate. Thus, the problem of finding the most efficient sequence of machines (route) is very important in planning the operation of a flexible manufacturing system. We have defined a route planning algorithm which takes into account the conditions for deadlock avoidance [16]. To avoid blocking, the production route **p** is partitioned into a unique set of Z sublists called *zones* which reflect shared and unshared resources. To avoid deadlocks, we use the *restricted allocation policy* proposed in [2, 4]. The complete formal explanation of the restriction allocation policy is presented in [16]. We have shown that the circular wait deadlock can never occur under the restricted allocation policy.

The route planning algorithm generates a list of *Processes* which contains only feasible, ordered sequences of technological operations realizing the *Task*. We have shown that if there exists a production route for a given *Task*, then the procedure finds the optimal ordered sequence of operations.

Each *Process* from the set *Processes* determines a different fundamental plan of robot and machine actions and a different law of workcell control. To minimize the flow time of jobs the variants of fundamental plan should be tested.

4.2 Task Level Programming

The implementation and interpretation of the fundamental plan is carried out using the *task-level programming* approach in which detailed paths and trajectories, gross and fine motion, grasping and sensing instructions are specified. Variant interpretations of the plan's TORPL-instructions result in different realizations of the robot actions. To create and verify all valid interpretations of the motion program, we again use a two-level system whose structure is shown in Figure 2.

The first level is a *Discrete Event Simulator* of the manufacturing process. The simulator uses the Discrete Event System Specification (DEVS) [39] formalism to model actions and technological devices. The second level is the *Motion Planner* [13, 21] employed to design each individual robot action. The planner creates variants of collision-free time-trajectories of the manipulator which executes each action. It uses robot-dependent planning techniques and the discrete dynamical system formalism [29, 12, 13].

The variants of *Process* obtained from the process planer can be tested by a simulator. To simulate, the system must have the knowledge of how individual robot actions are carried out in the process. This knowledge must be available in order to verify the possible sequence of technological operations from the set *Processes*. The most important parameters are the time it takes to complete an operation o, τ_d^o, and the time the robot requires to service a workstation. The time τ_d^o depends on the type of machine on which the o operation is being processed. It is fixed but can be changed by replacing the machine. Similarly, the times of PICK UP and PLACE operations are determined by the type of part and machine on which the part is processed.

The times of the robot's inter-operational moves (transfers), τ_r^{motion}, depend on the geometry of the work-scene and the cost function of the robot's motion. This cost function determines the dynamics of motion along the geometric tracks and the duration of the moves. These data must be accessible in order to simulate the entire production system.

4.3 Motion Planner

To create all valid interpretations of robot commands the motion planner for each individual robot action is used. It creates variants of collision-free time-trajectories of a manipulator that are used to perform the individual robot action. Such a planner uses robot-dependent planning techniques . The variants of motion interpretation obtained from the motion planner are tested by the simulator.

The motion planner interprets and simulates the commands based on a geometric model of the robot and its workscene. The planning component of this system automatically formulates the robot's motion trajectory. It also provides time parameters for the robot's actions. To generate the trajectories, we must have available the geometrical models of all workstations and stores of a cell as well as the models of the robot's kinematics and dynamics. The robot's motion trajectory planning process is decomposed into two subproblems:

- planning of collision-free geometric track of motion and
- planning of the motion dynamics along the computed track.

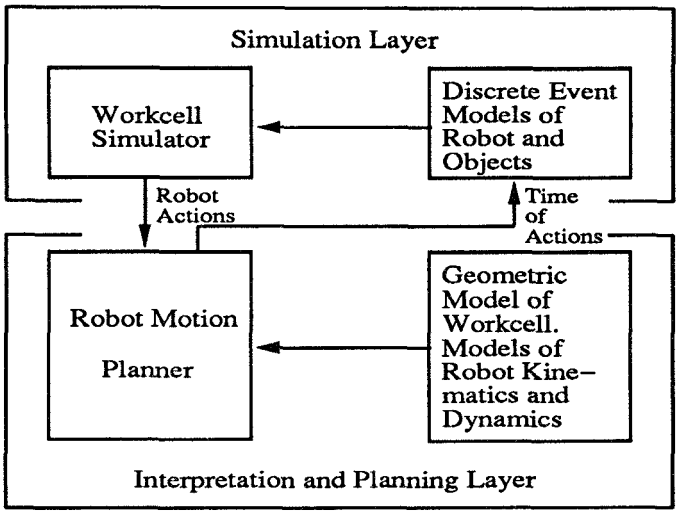

Figure 2: Structure of Robot Task Simulation System

The planner determines the collision-free track of the robot motion from the initial to the final effector location based on the geometric and kinematic description of the robot and its environment. This problem has been addressed in various ways and is widely reported in literature [12, 21, 25]. The methods which solve the problem in question depend on the assumed mathematical model of the robot's kinematics. Now, the optimal speed and acceleration of movements along computed track should be computed. This task is solved by the trajectory planner. The trajectory planner receives the geometrical tracks as inputs and determines the time history of position, velocity, acceleration and input torques which are then input to the trajectory tracker. On this level the robot is represented by the manipulator dynamics model [17, 33, 34].

Hence we obtain an optimal trajectory and the time of manipulator transfer movements. The time trajectories of motions allow us to establish times for each elementary action. For each motion's command we can change the geometry of movement or change the motion dynamics by selecting criteria of optimal trajectory planning . The variant interpretation obtained from the motion planning allows us to test and select the control law *Process* which minimizes the makespan.

Work is underway on the other phases of the FMS CAD framework. To select and configure devices, we apply the Knowledge Based Simulation Design Methodology [27, 29]. Recently, methods which combine qualitative and quantitative techniques have been developed for FMS spatial layout optimization [3].

5 Conclusions

The hierarchical FMS architecture closely reflects that of Antsaklis et. al. [1]. Indeed, autonomy subsumes the notion of automation. To achieve high autonomy in such an architecture, we believe that beyond the functional characteristics defined in Section 3, the following requirements must be addressed at each level.

The facility level should support dynamic task generation, specification of dynamic priorities and task migration, and hierarchical structuring of distributed decision making processes.

The cell layer must ensure that all the quality requirements are met in a generated process plan. It should ensure low production cost, maximize machine utilization, in other words, it should bring about high production efficiency. A replanning ability is required under failure.

The workcell level design must adhere to conventional automatic control principles, i.e., it has to ensure control quality and accuracy, it should have adaptive capabilities, and failure detection and identification algorithms.

References

[1] P.J. Antsaklis, K.M. Casino and S.J. Wang. Towards Intelligent Autonomous Control Systems: Architecture and Fundamental Issues. *Journal of Intelligent and Robotic Systems*, vol. 1, no. 4, 315-342, 1989.

[2] Z.Banaszak, E.Roszkowska. Deadlock Avoidance in Concurrent Processes. *Foundations of Control*, **18**(1-2), 3-17, 1988.

[3] M. Chierotti. Factory Design Layout: A Combine AI and Simulation Approach. Master's Thesis, ECE Department. University of Arizona, Tucson, Arizona, December 1991.

[4] E.G. Coffman, M. Elphick, A. Shoshani. System Deadlock.*Computing Surveys*, **3**(2), 67-78, 1971.

[5] M. Combacau and M. Courvoisier. Process Failure Diagnosis in FMS Real-Time Control: An Approach Combining Rule-Based Systems and Petri-Nets. *Proc. of the 2nd Conference on AI, Simulation, and Planning in High Autonomy Systems*, 174-180, IEEE Press, April 1991.

[6] N. Duan, S Kumara and D. Medeiros. EMM-Networking Model for FMS Modeling, Simulation, and Control. *Proc. of the 2nd Conference on AI, Simulation, and Planning in High Autonomy Systems*, 253-262, IEEE Press, April 1991.

[7] W. K. Erickson and P.C. Cheeseman. Issues in Design of an Executive Controller Shell for Space Station Automation. *Optical Engineering*, **25**(11), 1194-1199, 1986.

[8] B. Faverjon. Object Level Programming of Industrial Robots. *IEEE Int. Conf. on Robotics and Automation*, **2**, 1406-1411. 1986.

[9] P.A. Fishwick, J.W. Rozenblit and B.P. Zeigler (Editors). *Proc. of the 2nd Conference on AI, Simulation, and Planning in High Autonomy Systems*, IEEE Press, April 1991.

[10] L.S. Homem De Mello and A. C. Sanderson. AND/OR Graph Representation of Assembly Plans. *IEEE Trans. on Robotics and Automation*, vol. 6, no 2, 188-199, 1990.

[11] W. Jacak. Strategies for Searching Collision-Free Manipulator Motions: Automata Theory Approach. *Robotica*, **7**, 129-138, 1989.

[12] W. Jacak. A Discrete Kinematic Model of Robot in the Cartesian Space. *IEEE Trans. on Robotics and Automation*,**5**(4), 435-446, 1989.

[13] W. Jacak. Discrete Kinematic Modeling Techniques in Cartesian Space for Robotic System. in: *Advances in Control and Dynamics Systems*, ed. C.T. Leondes, Academic Press, 1991.

[14] W. Jacak and J.W. Rozenblit. Automatic Simulation of a Robot Program for a Sequential Manufacturing Process, *Robotica*, **10**, 45-56, 1992.

[15] W. Jacak and J.W. Rozenblit. Workcell Task Planning and Control, *Proc. of the 11th European Meeting on Cybernetics and Systems Research*, Vienna, April 1992.

[16] W. Jacak and J.W. Rozenblit. Planning of Machining Processes in an N-batch Manufacturing System, *European AI Conference*, July 1992, (submitted)

[17] W.Jacak, I.Dulęba, P.Rogaliński. A Graph-Searching Approach to Trajectory Planning of Robot's Manipulator, *Robotica*, (in print), 1992.

[18] A.T. Jones, C.R McLean. A Proposed Hierarchical Control Model for Automated Manufacturing Systems, *Journ. of Manufacturing Systems*, **5**(1), 15-25, 1986.

[19] A. Kusiak *Intelligent Manufacturing Systems.* Prentice Hall. 1990.

[20] J.E. Lenz. *Flexible Manufacturing.* Marcel Dekker, Inc., 1989.

[21] T. Lozano-Perez. Task-Level Planning of Pick-and-Place Robot Motions. *IEEE Trans. on Computer* **38**(3), 21-29, 1989.

[22] C.J. Luh and B.P. Zeigler. Abstraction Morphisms for Task Planning and Execution. *Proc. of the 2nd Conf. on AI, Simulation, and Planning in High Autonomy Systems*, IEEE Computer Society Press, 50-59, 1991.

[23] O. Maimon. Real-time Operational Control of Flexible Manufacturing System, *Journ. of Manufacturing Systems*, **6**(2), 125-136, 1987.

[24] NASA, *The Space Station Program.* NASA Publication, 1985.

[25] P.G.Ranky, C.Y.Ho *Robot Modeling. Control and Applications with Software*, Springer Verlag. 1985.

[26] J.W. Rozenbit. Design for High Autonomy: An Overview, *Applied Artificial Intelligence*, **6**(1), 1-19, 1992.

[27] J.W. Rozenblit and B.P. Zeigler. Design and Modeling Concepts. *International Encyclopedia of Robotics, Applications and Automation*, (ed.

Dorf, R.) John Wiley and Sons, New York, 308-322, 1988.

[28] J.W. Rozenblit and B.P. Zeigler (Editors). *Proc. of the 1st Conference on AI, Simulation, and Planning in High Autonomy Systems*, IEEE Press, April 1991.

[29] J. W. Rozenblit and B.P. Zeigler, Knowledge-Based Simulation Design Methodology: A Flexible Test Architecture Application, *Transactions of the Society for Computer Simulation*, **7**(3), 195-228, 1990.

[30] J. W. Rozenblit and W. Jacak. Simulation Based Planning of Robot Tasks in Flexible Manufacturing, *Proceedings of the 2nd AI, Simulation and Planning in High Autonomy Systems Conference*, 166-173, IEEE Computer Society Press, April 1991.

[31] G.N. Sardis. Intelligent Robotic Controls, *IEEE Trans. on Automatic Control*, AC-28, no. 5, 1983.

[32] K.Shin, N.McKay, A Dynamic Programming Approach to Trajectory Planning of Robotic Manipulators. *IEEE Trans. on Automatic Control*, **31**(6), 491-500, 1986.

[33] K.Shin, N.McKay, Minimum Time Control of Robotic Manipulator with Geometric Path Constrains. *IEEE Trans. on Automatic Control*, **30**(6), 531-541, 1985.

[34] R. Speed. Off-line Programming of Industrial Robots. *Proc. of ISIR 87*, 2110-2123, 1987.

[35] B.P. Zeigler. Endomorphic Modeling Concepts for High Autonomy Architectures. *Applied Artificial Intelligence*, **6**(1), 19-33, 1992.

[36] B.P. Zeigler, B.P., *Object-Oriented Simulation with Hierarchical Modular Models: Intelligent Agents and Endomorphic Systems*, Academic Press, 1990.

[37] B.P. Zeigler and S.D. Chi. Model-Based Architectures for Autonomous Systems. *Proc. of the 1990 IEEE Symp. on Intelligent Control.* Philadelphia, PA, 27-32, 1990.

[38] B.P. Zeigler. *Multifacetted Modelling and Discrete Event Simulation*, Academic Press, 1984.