Virtual Process Design Techniques for Intelligent Manufacturing

Witold Jacak

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

Jerzy Rozenblit

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

Abstract

The design techniques discussed here are concerned with computer integrated, cellular manufacturing systems. Such systems consist of automated robotic cells. Planning and control within a cell is carried out in off- and on-line modes by a hierarchical controller. A workstation and its control modules are called Computer Assisted Workcell (CAW). An intelligent CAW should determine control laws so that manufacturing tasks can be carried out optimally. Techniques are presented to synthesize high autonomous computer assisted workcells. At the core of the CAW's architecture are three layers: a) execution, b) coordination, and c) organization. The architecture and its layers are discussed in detail.

1 Introduction

The basic building block of a flexible manufacturing system (FMS) is a robotic manufacturing cell. Cellular manufacturing systems consist of three main components: *a production system, a material handling system,* and *a hierarchical, computer assisted control system.*

We term a cellular manufacturing system *intelligent* if it can make decisions based upon the simulation of a needed solution in a virtual world, or upon the experience gained in the past from both failures and successful solutions. In this paper, a *computer integrated cellular system* consisting of automated robotic cells is discussed. Planning and control within a cell is carried out off- and on-line by a hierarchical controller. The cell and its controller constitute a *Computer Assisted Workcell* (CAW).

Given a technological task, an intelligent Computer Assisted Workcell should be able to determine control algorithms and laws so that: a) a task is realized, b) deadlocks are avoided, c) the flow time is minimal, d) the work-in-process is minimal, e) geometric constraints are satisfied, and f) collisions are avoided.

The control laws which govern the operation of a CAW are structured hierarchically. We distinguish three basic levels of control: a) *the workstation (execution)* level, b) the *cell (coordination)* level, and c) *the organization level.* This follows the classification of intelligent control systems often cited in the literature [5], [6], [7], [9].

The organizer accepts and interprets related feedback from the lower levels, defines the strategy of task sequencing to be executed in real-time, and processes large amounts of knowledge with little or no precision. Its functions are: reasoning, decision making, learning, feedback, and long term memory exchange.

The coordination level defines part routing in the logical and geometric aspects and coordinates the activities of workstations and robots. The robots coordinate the activities of the equipment in the workstation. This level is concerned with the formulation of the actual control task to be executed by the lowest level. The execution level consists of device controllers. It executes the programs generated by the coordinator. The hierarchical structure of a computer assisted cell is shown in Figure 1.

There are two major problems in the synthesis of such complex control systems: one is the *coordination and integration* of all levels of the system. The other is that of *automatic programming* of the elements of the system. Thus, we distinguish two major aspects of the control problem: the *logical* and the *operational control.*

The intelligent control of CAW is synthesized and executed in two phases, namely, *planning and off-line simulation phase* and *on-line simulation-based monitoring and control phase.*

In the first phase, a hierarchical simulation model of a robotic workcell, called a *virtual cell,* is created. Workcell components such as NC-machine tools, robots conveyers, etc., are modelled as elementary

0-8186-4020-0/93 $03.00

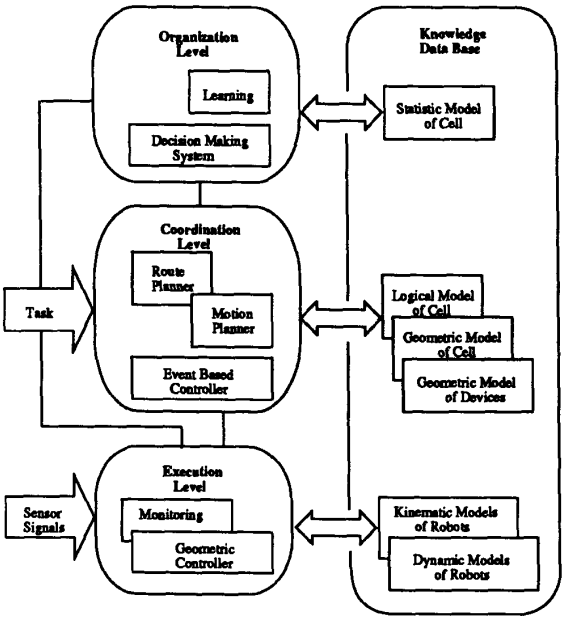

Figure 1: Hierarchy of Control in Computer Assisted Cell

DEVS systems [10]. This type of simulation is used for verification and testing of different variants of task realizations (processes) obtained from the route planner. To simulate variants of a process, the system must have the knowledge of how individual robot actions are carried out. For detailed modeling of cell components, we employ continuous simulation and motion planning methods [1], [2]. The geometrical interpretations of cell-actions are obtained from the motion planner and are tested in a geometric cell-simulator. This allows us to select the optimal task realizations which establish the logical control of the system.

In the second phase, a real-time, discrete event simulator of a CAW is used to generate a sequence of future events of the virtual cell in a given time-window. These events are compared with the current states of the real cell and are used to predict robot motion commands and to monitor the process flow. In this paper, our emphasis is on the design of a discrete event CAW controller.

2 Virtual Workcell

A real cell is a fixed, physical group of machines, D, stores, M, and robots, R. A *virtual cell* is its formal representation (computer model). To synthesize CAW's control, we first specify technological tasks realized in the cell. A *technological task* realized by a robotic cell is represented by a triple:

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

where: O is a finite set of *technological operations* required to process the parts, $\prec \subset O \times O$ is a partial order (*precedence relation*) on the set O, and $\alpha \subset O \times (D \cup M)$ is a relation of device or store assignment.

The partial order represents an operational precedence. $(o, d) \in \alpha$ means that the operation o can be performed on the workstation d. If $(o, m) \in \alpha$, then m is the production store from the set M where the parts can be stored after the operation o has been completed.

The virtual cell has a hierarchical structure which is comprised of models. The models represent knowledge about a real production environment.

Geometric Model of Workcell: The lower level of representation describes the geometry of a virtual cell. A geometric model has two components: (1) the workcell objects models and (2) the workcell layout model.

The geometry model of each object is created by using solid modeling [8]. Solid modeling incorporates the design and analysis of virtual objects created from primitives of solids stored in a geometric database. The complex virtual objects of a workcell (e.g., technological devices, robots, auxiliary devices or static obstacles) are composed of solid primitives such as cuboids, pyramids, regular polyhedrons, prisms, and cylinders.

The workcell's objects can be placed in a robot's workscene at any position and in any orientation. The virtual objects (devices, stores) are loaded from a library (model base) into the Cartesian base frame. They can be located anywhere in the cell using translation and rotation operations in the base coordinate frame [1], [2].

Logical Model: Based on the geometric model of a workcell, a logical structure of the cell must be created. The group of machines is divided into subgroups, called *machining centers* serviced by separate robots. Parts are transferred between machines by the robots from the set R, which service the cell. A robot $r \in R$ can service only those machines that are within its service space $Serv\_Sp(r) \subset E_0$ (E_0 - Cartesian base

frame). The set of devices which lie in the r-th robot service space is denoted by $Group(r) \subset D \cup M$. More specifically, a device belongs to the group serviced by a robot r (i.e., $d \in Group(r)$) if all positions of its buffer lie in the service space of the robot r. Consequently, the logical model of a workcell is represented by: $Cell_{Logic} = (D \cup M,\ R,\ \{Group(r) \mid r \in R\})$

The virtual cell concept is the basis for designing the CAW planners. We employ two types of planning algorithms: a) technological route planners and b) motion planners. For details, the reader is referred to [3].

3 Discrete Event-Based, Real-Time Cell Controller

Process planning is based on the description of a task, its operations and their precedence relations. As a result, the fundamental plan of cell-actions, i.e., an ordered sequence of technological operations, is created [3]. Then, a real-time, event-based simulator of a CAW is synthesized. The simulator generates a sequence of future events of the virtual cell in a given time-window. These events are compared with the current states of the real cell. They are used to predict motion commands of robots and to monitor the process flow. The simulation is event-driven and is based on the DEVS system concept introduced by Zeigler [10]. The structure of a CAW is shown in Figure 2. We now proceed to describe its underlying formal concepts.

Each workstation $d \in D$ is modelled by two coupled, atomic discrete event systems (DEVS) called active and passive models of a device, i.e., $DEVS_d = (Dev_d^A, Dev_d^P)$, where Dev_d^A and Dev_d^P are active and passive models, respectively.

An active atomic DEVS is the basis for simulation. The passive atomic DEVS is a register of the real states of the workstation. It acts as a synchronizer between the real and the simulated processes.

3.1 Active Model

An active model is the following structure:

$$Dev_d^A = (X_V(d), S_V(d), \delta_{int}^d, \delta_{ext}^d, \mathrm{ta}^d)$$

$X_V(d)$ is a set, the external input virtual event types,
$S_V(d)$ is a set, the sequential virtual states,
δ_{int}^d is a set of functions, the internal transition specifications,
δ_{ext}^d is a function, the external transition specification,

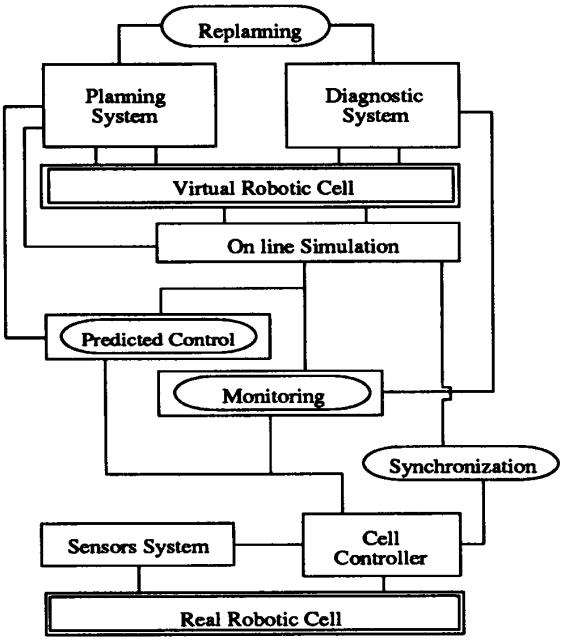

Figure 2: Structure of Computer Assisted Cell

ta^d is a function, the time advance function,
with the following constraints:
$E_V(d) = \{(s,t) | s \in S_V(d), 0 \leq t \leq \mathrm{ta}^d(s)\}$ is the total virtual, event-set of the system specified by Dev_d^A;
δ_{int} is a set of parameterized internal state-transition functions: $\delta_{int} = \{\delta_{int}^u | u \in U\}$ and $\delta_{int}^u : S_V \rightarrow S_V$;
δ_{ext} is an external state-transition function $\delta_{ext} : E_V \times X_V \rightarrow S_V$;
ta^d is a mapping from S_V to the non-negative reals with infinity: $\mathrm{ta} : S_V \rightarrow R$

Each workstation $d \in D$ can have a buffer. The capacity of this buffer is $C(d)$. If a workstation has no buffer, then $C(d) = 1$. Let NC(d) (NC program register) denote the set of operations performed on the workstation d, i.e., $\mathrm{NC}(d) = \{o \in O | (o,d) \in \alpha\}$.

The virtual state set of a workstation d is defined by $S_V(d) = SD_d \times SB_d$, where SD_d denotes the state set of the machine and SB_d denotes the state set of its buffer. The state set SD_d is defined as $SD_d = S_{ready} \cup S_{busy} \cup S_{done}$ where:

- $S_{ready} = \{ready\}$ signifies that *machine is free*
- $S_{busy} = \{(busy, q) | q \in \mathrm{NC}(d)\}$ and $(busy, q)$ sig-

nifies that *machine is busy processing the operation q*

- $S_{done} = \{(done, q) | q \in \mathrm{NC}(d)\}$ and $(done, q)$ signifies that *machine has completed the operation q and is not free*

The state set of a buffer SB_d is specified as

$$SB_d = (O \times \{0, 1, \#\})^{C(d)}.$$

Let $C(d) = K$ and $b_d = (b_i | i = 1, ..., K) \in SB_d$, then

$$b_i = \begin{cases} (0,0) & \Leftrightarrow \text{i} - \text{th position of the buffer is free} \\ (0,\#) & \Leftrightarrow \text{i} - \text{th position of the buffer is reserved for a part in process} \\ (q,0) & \Leftrightarrow \text{i} - \text{th position of the buffer is occupied by a part before q} \\ (q,1) & \Leftrightarrow \text{i} - \text{th position of the buffer is occupied by a part after q} \end{cases}$$

We assume that the i-th position denotes the location at which a part is placed in the buffer.

Given the virtual state set, we define the internal state-transition functions δ_{int} to model the workstation. They are parameterized by an external parameter u which is loaded into the workstation from a higher level of control called the *workcell management system*. The parameter $u \in U$ is the operation's choice function. It represents the priority strategy of a workstation, i.e.,

$$u : 2^{\mathrm{NC}(d)} \rightarrow \{o\} \subset \mathrm{NC}(d)$$

and $u(\emptyset) = \emptyset$, $u(\{o\}) = \{o\}$. The choice function u defines a priority rule under which the operations are chosen from the device's buffer.

Let $s(d) = (s_d, (b_1, b_2, ..., b_K)) = (s_d, b) \in S_V(d)$ and $Wait(d) = \{q | ((\exists b_j)((o,0) = b_j\}$ be the set of operations waiting to be processed.

Then the internal transition function δ^u_{int} : $S_V(d) \rightarrow S_V(d)$ is specified as follows:

$$\delta^u_{int}(s(d)) = \begin{cases} ((done, q), (b_1, b_2, ..., b_K)) \\ \Leftrightarrow s_d = (busy, q) \\ ((busy, q), (b_1, ., b_{i-1}, (0, \#), b_{i+1}, ., b_K)) \\ \Leftrightarrow s_d = ready \wedge \\ u(Wait(d)) = \{q\} \wedge (q, 0) = b_i \\ (ready, (b_1, b_2, ..., b_K)) \\ \Leftrightarrow s_d = ready \\ \wedge\ Wait(d) = \emptyset \\ (ready, (b_1, ., b_{i-1}, (q, 1), b_{i+1}, ., b_K)) \\ \Leftrightarrow s_d = (done, q) \wedge b_i = (0, \#) \end{cases}$$

3.1.1 Modeling the Workstation/Robot Interaction

The interaction between the robots and workstation is modelled by the external state transition function. The set of external virtual events for the workstation's model Dev^A_d is defined as follows:

$$X(d) = \{e^1_d(q,i), e^2_d(q,i), e^0 \mid q \in \mathrm{NC}(d) \wedge i = 1, ..., K\}$$

where:

1. $e^1_d(q,i)$ = PLACE $(q,0)$ ON *i-th position* - signifies that a part before the operation q is placed on the i-th position of d-workstation's buffer,
2. $e^2_d(q,i)$ = PICKUP $(q,1)$ AT *i-th position* - signifies that a part after the operation q is removed from the i-th position of d-workstation's buffer,
3. e^0 = DO NOTHING (empty event)

The external transition function δ^d_{ext} for each workstation d is defined below.
Let $s(d) = (s_d, (b_i | i = 1, .., K)) \in S_V(d)$ then
$\delta_{ext}((s(d), t), e^0) = s(d)$
$\delta_{ext}((s(d), t), e^1_d(q,i)) =$
$(s_d, (b_1, ., b_{i-1}, (q,0), b_{i+1}, ., b_K)) \Leftrightarrow b_i = (0,0)$
$\delta_{ext}((s(d), t), e^2_d(q,i)) =$
$(s_d, (b_1, ., b_{i-1}, (0,0), b_{i+1}, ., b_K)) \Leftrightarrow b_i = (q,1)$
$\delta_{ext}((.,.),.) = failure$ for all other states

3.1.2 Time advance

The time advance function for Dev^A_d determines the time needed to process a part on the d-th workstation. It is defined as follows:
Let $s(d) = (s_d, (b_i | i = 1, .., K))$. Then

$$\mathrm{ta}^d(s(d)) = \begin{cases} \tau_{process}(q) & \Leftrightarrow s_d = (busy, q) \\ \tau_{load} + \tau_{setup}(q) & \Leftrightarrow s_d = ready \\ & \wedge\ u(Wait(d)) = \{q\} \\ \tau_{unload} & \Leftrightarrow s_d = (done, q) \\ \infty & \text{otherwise} \end{cases}$$

$\tau_{process}(q)$ denotes the tooling/assembly time of the operation q for the workstation d. $\tau_{load}, \tau_{setup}(q)$, τ_{unload} denote the loading, setup, and unloading times for d, respectively.

At time moment t, let the workstation be in the active state s which has begun at time moment t_o. After this time, the workstation transfers its state from

s into $\delta^u_{int}(s)$, which will be active in the next time interval $[t_o + \text{ta}(s), t_o + \text{ta}(s) + \text{ta}(\delta^u_{int}(s))]$.

It is easy to observe that the external events cannot change the advance time for an active state. Let $x = (e, t)$ be an external event different from e^0, which occurs at time moment $t \in [t_o, t_o + \text{ta}(s)]$. The workstation changes its state to state $s' = \delta_{ext}((s, t), e)$ but the advance time of the new state s' is equal to $\text{ta}(s') = t_o + \text{ta}(s) - t$.

3.2 Passive Model

A passive model is a finite state machine:

$$Dev^P_d = (X_R(d), Q_R(d), \varphi^d_{ext}, \lambda^d)$$

where:
$X_R(d)$ is a set, the external input real event types,
$Q_R(d)$ is a set, the sequential real states,
φ^d_{ext} is a function, the external real-state transition specification,
λ^d is a function, the updating function,
with the following constraints:
(a) $\varphi^d_{ext} : Q_R(d) \times X_R(d) \rightarrow Q_R(d)$;
(b) $\lambda : UP \times Q_R(d) \times S_V(d) \rightarrow S_V(d)$;

$UP = \{0, 1\}$ where 0 denotes that updating is not required and 1 denotes that updating should occur.

The interaction between external sensors and the workstation is modelled by the state transition function φ_{ext}. The external sensor system generates real events which can be used to synchronize the simulated technological process with the real process.

The set of real events for the workstation's model Dev^P_d can be defined as follows: $X_R(d) = \{re^1_d(q, i), re^2_d(q, i), re^3_d(q, i), re^4_d(q, i) | q \in \text{NC}(d) \wedge i = 1, ..., K\}$ where:

- $re^1_d(q, i)$ - signifies that a part before the operation q is placed on the i-th position of d-workstation's buffer
- $re^2_d(q, i)$ - signifies that a part after the operation q is removed from the i-th position of d-workstation's buffer,
- $re^3_d(q, i)$ - signifies that a part before q is loaded into the machine and processing is begun,
- $re^4_d(q, i)$ - signifies that the machine has completed q, the machine is unloaded and the part is placed on the i-th position of d's buffer.

We assume that the state set $Q_R(d)$ of the passive model is the same as that of the active one, i.e., $Q_R(d) = S_V(d)$.

Hence, the state transition function φ^d_{ext} for the workstation d can be defined as follows:
Let $q(d) = (s_d, (b_i | i = 1, .., K)) \in Q_R(d)$. Then
$\varphi_{ext}(q(d), re^1_d(q, i)) =$
$(s_d, (b_1, ., b_{i-1}, (q, 0), b_{i+1}, ., b_K))$
$\varphi_{ext}(q(d), re^2_d(q, i)) =$
$(s_d, (b_1, ., b_{i-1}, (0, 0), b_{i+1}, ., b_K))$
$\varphi_{ext}(q(d), re^3_d(q, i)) =$
$((busy, q), (b_1, ., b_{i-1}, (0, \#), b_{i+1}, ., b_K))$
$\varphi_{ext}(q(d), re^4_d(q, i)) =$
$(ready, (b_1, ., b_{i-1}, (q, 1), b_{i+1}, ., b_K))$

This transition function reflects real state changes of the workstation. The output function λ produces the real state in the active model of the workstation when updating is required, i.e.,

$$\lambda(q, s, 1) = q \in S_V(d) \;\; \& \;\; \lambda(q, s, 0) = s \in S_V(d)$$

A production store model can be defined in a similar manner.

Each robot $r \in R$ is modelled by two coupled systems, again called active and passive models.

$$REVS_r = (Rob^A_r, Reg^P_r)$$

where Rob^A_r is the active discrete event model and Reg^P_r is the passive model. The active model is used for simulation while the passive one represents the register of the robot's real states. It acts as a synchronizer between the real and simulated processes.

The virtual workstations and robots (DEVS models) together with the real devices and robots represent the execution level of CAW control system. The structure of a CAW's execution level is illustrated in Figure 3.

Each function δ_{int} is parameterized by an external parameter g which is loaded into the cell controller from the workcell's organizer. The parameter $g \in Strategies$ is the operation's choice function, and represents the priority strategy of a CAW, i.e.,

$$g : 2^{OP} \rightarrow \Re$$

The choice function g defines a type of priority rule under which the operations are be chosen for processing.

4 Real Time Monitoring

The external events generated sequentially by the cell controller and robots activate the workcell's devices and coordinate the transfer actions. The discrete

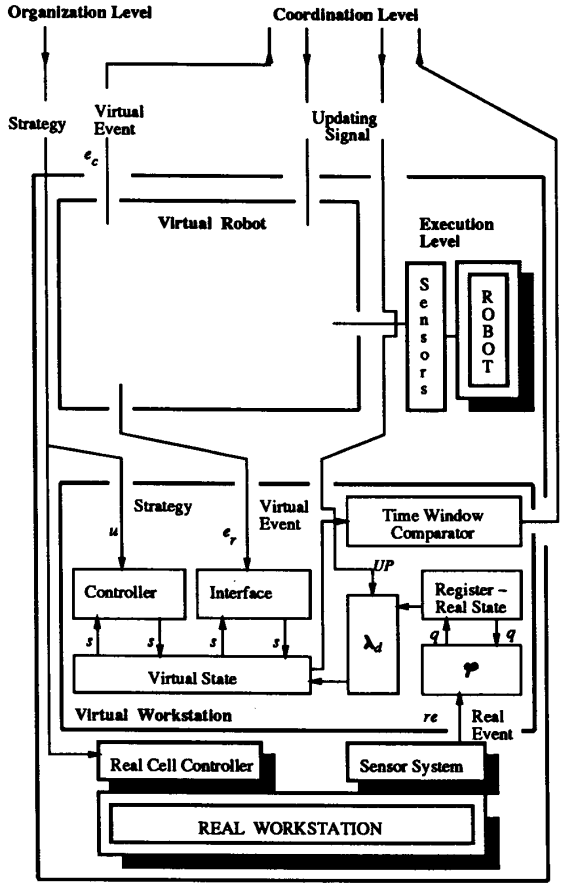

Figure 3: The Execution Layer of CAW

event model generates a sequence of future events of the virtual cell in a given time-window, which we call a *trace*. A trace in a time window is denoted as a sequence of pairs (state,time), i.e.,

$$tr_{[t_o,t_o+T]} =< (s_1,t_1),(s_2,t_2),(s_3,t_3),..(s_n,t_n) >$$

where $t_1 \geq t_0 \wedge t_n \leq t_0 + T$.

The events from a trace are compared with the current states of the real cell. They are used to predict the motion commands and to monitor the process flow.

Let $tr_T(d)$ be a trace of virtual events of device d in the time-interval $T = [t_o, t_o + T]$. t_o is the moment of the last updating:

$$tr_T(d) =< (s_o,t_o),(s_1,t_1)....,(s_k,t_k) >$$

where $t_k \leq t_o + T$ and s_i is the virtual state of a DEVS model of the device d. The monitoring process is carried out as follows: Let $q(t)$ be the current real state of d, modelled by a passive Dev_d^P automaton and $t \geq t_o$. If $q(t) \neq s_j$ for $t \in [t_j - \tau_0, t_j + \tau_0]$, then we invoke the *diagnosis* module. Otherwise, we call the *updating* procedures.

Let $q_d(t)$ be the current real state of d at time t, recorded by a passive Dev_d^P automaton and $s_d(t') \in S_V(d)$ be a virtual state of d's simulator at time t'. If $q(t) = s(t')$ and $t' \neq t$ but $t \in [t'-\tau_0, t'+\tau_0]$ where τ_0 is the tolerance time-window, then the synchronization between the real and virtual cell should be performed. The updating is then defined as a synchronization of every device and robot in the workcell. In [4], we further discuss various updating modes.

In addition to updating, we introduce the notion of diagnosis in a CAW. Let $q_d(t)$ be the current real state of d at time t, recorded by a passive Dev_d^P automaton and $s_d(t') \in S_V(d)$ be a virtual state of d-device simulator in time t'. Let $q(t) = s(t')$ and $t' \neq t$ and $t < t' - \tau_0$ or $t > t' + \tau_0$ where τ_0 is the tolerance time-window. In this case, the diagnosis of the real cell should be carried out.

5 Summary

We have introduced an architecture for design and control of intelligent, flexible manufacturing systems. Called a Computer Assisted Workcell (CAW), the architecture offers support methods and tools at the following layers of control: organization, coordination, and execution. We distinguish two major aspects of the control problem: the logical, and the operational control. To synthesize the control laws, we decompose the problem into two phases, namely, the planning and off-line simulation phase, and the on-line,

simulation-based monitoring and control phase. In the first phase, a hierarchical simulation model of robotic workcell, termed a virtual cell is created. This kind of simulation is intended for the verification and testing of different variants of task realizations.

In the second phase, a real-time, discrete event simulator of CAW is used to generate a sequence of future events of the virtual cell in a given time-window. These events are compared with the current states of the real cell and are used to predict motion commands of robots and to monitor the process flow.

References

[1] 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

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

[3] W. Jacak and J. Rozenblit. Model-Based Workcell Task Planning and Control. *IEEE Transactions on Robotics and Automation*, (in review), 1993

[4] W. Jacak and J. Rozenblit. CAST Tools for Intelligent Control in Manufacturing Automation. In *Lecture Notes on Computer Science*, (ed. F. Pichler), (in print), Springer-Verlag, 1993

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

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

[7] A. Meystel. Intelligent Control in Robotics. *J. Robotic Syst.*, 5(5), 1988

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

[9] G. N. Saridis. Intelligent Robotic Control. *IEEE Trans. Autom. Contr.*, **AC-28**(5), 1983

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