Simulation for Hierarchical Management and Control in Manufacturing Systems

B. Janusz, W. Reithofer, J. Raczkowsky
Institute for Real-Time Computer Systems and Robotics
Department of Computer Science, University of Karlsruhe
Kaiserstrasse 12, D-76128 Karlsruhe, Germany
bjanusz@ira.uka.de, rhofer@ira.uka.de

Abstract

Various methods for controlling manufacturing systems have been proposed, but until now the problem could not be considered as being solved. This may be due to the lack of a sound mathematical foundation. Therefore, an algebraic modelling method for describing manufacturing systems has been developed in the Esprit Basic Research Project HIMAC. The aim of this paper is to show clearly, how the developed methodology can be validated by the use of simulation. It is structured as follows: First, the need for the new modelling approach is justified. Next, the manufacturing algebra and factory dynamics are described as far as their understanding is needed in the following. The presentation of our simulation test bed is the core of the paper. Finally, an example illustrates all considerations in detail.

1 Introduction

Computer Integrated Manufacturing (CIM) requires models suitable for computers [Rembold 93]. The basic purpose of a computer in a CIM system is to support the human in designing, planning and controlling the manufacturing system as far as possible. This comprises a large variety of more specific subfunctions, and the very complex integration of subsystems. In order to perform this task the computer system uses various models of the real system.

In terms of managing a plant, the common goal is to perform the complete manufacturing process as optimally as possible. One step is to compute a production plan. This problem has been handled extensively in the past. One can assume that existing tools for production planning and control offer the opportunity to compute a nearly optimal production schedule off-line [Glaser 92]. The next step is to implement this production schedule, which is a very complex task. This may be related to the high complexity, the randomness and the uncertainty of the manufacturing environment, but is also due to the lack of a systematic approach based on a sound theoretical foundation. Of course we have mathematical descriptions of some aspects of the manufacturing process, eg., Markov chains or queueing systems [Kleinrock 75], [Law 91], [Bolch 89], and we can describe dependencies and the course of events of parallel processes with Petri nets [König 88]. However, a formal description of the manufacturing system state and a formal state equation which describes the evolution of the system state, and which is also manageable, is not known.

The goal of the ESPRIT Basic Research project HIMAC (Hierarchical Management and Control of Manufacturing Systems, [Canuto 93]) is the development of an integrated and coherent core of theories and related algorithms on hierarchical management and control. The expected results of the project can be subdivided into three main stages. Manufacturing algebra offers a tool for modelling the manufacturing objects and structure. Factory dynamics are expressed by state equations. A hierarchical theory of production management and control constitutes the basis for manufacturing control. A simulation test bed is used for testing in order to validate the theoretical results.

The manufacturing algebra is used as a mathematical tool for describing the manufacturing processes in a static way. The factory state equation, based on a factory state vector, describes the evolution of the manufacturing process.

This new modelling approach has been chosen with the intention of building up a solid mathematical basis which allows the design of closed-loop control systems. In order to validate the results and test different control policies, a test bed has been built up. This test bed uses a commercial simulation system which is integrated into a closed-loop control configuration. Several industrial enterprises will be selected according to batch size, product mix and technological sector and will serve as test cases [Reithofer 94].

2 Manufacturing Algebra and Factory Dynamics

Manufacturing algebra is intended to provide a mathematical tool for describing products and the relevant manufacturing processes in a factory independently of any technological sector. Only algebraic relations between variables will be defined. The resulting algebra should be used at the product design stage, where the main concern is *what* and *how* to produce.

The time evolution of production processes is described by the factory dynamics. Factory dynamics is a mathematical tool for desribing *where* and *when* the manufacturing operations defined by the algebra are performed in order to produce the planned products at the right time and in the right quantities.

In the following, only the fundamentals of the theory are presented. More details can be found in [Canuto 93] and [Canuto 95].

2.1 The Algebra

The manufacturing algebra is based on a pair of basic elements:

- a finite set of *(manufacturing) objects* and
- a definition of *manufacturing operation.*

Objects: The set O of objects consists of all movable objects in a factory, i.e. raw materials, semifinished and finished products, fixtures, equipment, tools etc. A basic assumption is that the set O has a finite cardinality n_0.

An n_0-dimensional integer vector q called the *quantity vector*, is defined over the set O. If the i-th component $q(i)$, denoting the quantity of the i-th object type, is positive, it indicates *availability* of the i-th object; a negative quantity indicates *need* of the object. The set Q of the quantity vectors, equipped with the usual vector addition and scalar multiplication, is the *quantity space.*

Manufacturing operations: Given a set of manufacturing objects O, a manufacturing operation a is defined by any ordered pair $a = (v, y)$, where $v \in Q$ describes the objects used, $y \in Q$ the objects produced by operation a.

Each operation has two associated values:

- the *working time* and
- the *manufacturing operation cost,*

so that constraints given by the manufacturing system can be described.

The *balance* of an operation $a = (v, y)$ is the difference between the output vector of operation a and the appropriate input vector:

$$b = y - v. \tag{1}$$

Quantities which appear with a negative sign in the balance vector indicate used objects, whereas quantities with a positive sign indicate the produced objects.

2.2 Factory Dynamics

In factory dynamics, the model is formulated axiomatically. Any abstract factory is assumed to be a network of three dynamic elements: *storing units*, *transport units* and *manufacturing units.*

Storing units: A storing unit is described as a vector of object quantities. These quantities represent the state of the storing unit. It is possible to apply storing and drawing operations to a storing unit. The resulting effects of the operations are delayed by a certain time. Therefore, the state at time t necessarily has to take into account all commands of drawing and storing operations, which at time t have already been given but were not yet carried out. By knowing the

time when these commands will be carried out, the evolution of the quantities in the stores can be foreseen. In this way, the state of the store is made up of the object quantities present at time t and the object quantities expected at the successive instants $t+1, t+2, \ldots$ up to $t+n_j$. The maximum of all possible time delays (*horizon*), which is assumed to exist, is denoted by n_j. The minimum time delay is 1.

Therefore, the state of the factory storing units at time t is denoted by three indices: $x_t(i,j,k)$, whereby

- i is the number of the store unit,
- j is the number of time steps after a command has been given and
- k is the object type.

Transport and manufacturing units: A *transport unit* connects two different stores and performs operations by drawing objects from the input store and storing them in the output store.

A *manufacturing unit* connects two (not necessarily different) stores and performs operations by drawing objects from the input store and storing, in general other, objects in the output store.

An *operation* can be identified by its number s. The units performing operations are identified by their numbers r in the list of the manufacturing and transport units of the factory. The commands, i.e the operations which should be actuated at time t are then described by the vector $u_t(r,s)$, whereby $u_t(r,s) = 1$ stands for executing operation s by unit r.

The operations are described in conformity with the manufacturing algebra. The quantity vectors expressed with the three-index notation are used as one-dimensional vectors.

The balance vector can also be used: $b(i,j,k;r,s)$ are integer numbers denoting the used or the produced quantities of objects, if operation s is performed on unit r.

The factory state equation: If the current state is x_t, and u_t contains all operations to be executed at time t, the subsequent state can be calculated by

$$x_{t+1} = Ax_t + Bu_t. \qquad (2)$$

The *balance matrix* B consists of the balance vectors of all manufacturing operations which can be performed in the factory. The *shift matrix* is denoted by A. The task of this matrix is to shift all objects which are expected to be produced at time $t+j$ to the places corresponding to the time $t+j-1$.

The state equation is the starting point for the dynamic analysis and the control design. Up to now, controllability properties, steady state solutions and a few feedback properties have been derived in [Canuto 94b].

3 The HIMAC Simulation Test Bed

In order to validate the theoretical results and to test different control policies, a simulation test bed has been built up. Simulation is used in order to avoid tests on a real manufacturing system. An appropriate "test bed" is depicted in Figure 1. Three interfaces are needed to implement the

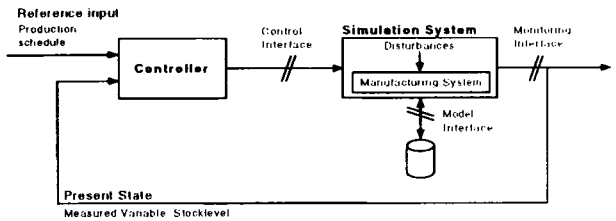

Figure 1: Closed-loop feedback control.

links between the simulator and its environment. The *static* aspect of the theory, i.e. the algebra, concerns the *model interface*. This interface facilitates the transformation of an algebraic description of a manufacturing system into its simulation model and vice versa. After the simulation model of the system has been constructed, the execution of a simulation experiment can be controlled via the *control* and the *monitor* interfaces. Therefore, it is clear that these interfaces deal with the *dynamic* behaviour of the system.

By carrying out an extensive market analysis, we found out, that SIMPLE++ [Simple 94] is

the most suitable simulation system for our purposes. It offers an integrated graphical user interface. "Integrated" means that modelling, simulation and animation are possible at any time, i.e. it is even possible to vary the model during the simulation run. The modelling is performed graphically interactively by use of building blocks. The simulation system offers *basic building blocks*, *application-specific building blocks* and *user-defined building blocks*. The latter can by designed by aggregating existing building blocks. This feature is used in our application to construct HIMAC-specific building blocks. HIMAC-specific building blocks incorporate our particular modelling paradigm and implement the simulator-side adaptions necessary for the model, the control and the monitoring interfaces. Methods can be programmed in the language "SimTalk". The simulation run is managed by an event-manager.

3.1 Statics: Model transformation

Two types of model transformation are possible:

- **Simulation model → Algebraic description**

  After establishing the simulation model graphically, the algebraic description may be needed for the controller design. This transformation is implemented within the HIMAC building blocks. After triggering a special method, a file containing the elements of the algebra with all corresponding information like manufacturing time and cost, capacity etc. is generated. This file can be directly translated into the vectors and matrices needed for the state equations.

- **Algebraic description → Simulation model**

  In the future it will become more and more necessary to adapt manufacturing systems to a complex and uncertain environment. This may result in the development of bottom-up design tools for manufacturing systems. As an output, these tools may generate a system description in an algebra-like language. To make this system description accessible to the efficiency analysis, it must be transformed into a simulation model. To solve this problem the algebraic representation is first compiled into an equivalent SimTalk file and then sent to the simulation system via a remote procedure call.

3.2 Dynamics: Controlling and Monitoring the Simulation Experiment

The *reference input* of the controller is the output of the production planning unit. This can be e.g. a production schedule represented by a Gantt-chart. The task of the closed-loop control is to control the implementation of the operations as close as possible to this schedule. Hence, the controller determines the next operation that has to be executed within the next scanning period and sends this commands it to the simulation system. The simulator replicates the industrial environment. Of paramount importance is its ability to generate disturbances. These disturbances are specified by statistical distributions. Therefore, the simulated evolution of the manufacturing system may differ from the expected behaviour according to the factory state equation. Hence, the controller needs some information about the real system evolution. To meet this need, the simulated state of the manufacturing system is recorded by a snapshot which captures the actual stocklevel of every storing unit.

4 Example

In order to illustrate the implementation and the use of the interfaces described above, a simple example is presented. The examined manufac-

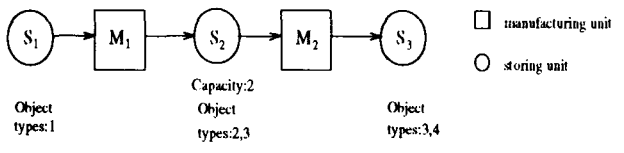

Figure 2: Manufacturing system.

turing system consists of three storing units and

two manufacturing units as depicted in Figure 2. Four manufacturing operations are feasible. Two different end products may be formed from one raw product: Object O_4 is produced by performing operation FOP_1 with unit M_1 and operation FOP_3 with unit M_2. Analogous to this, object O_5 is produced by executing operation FOP_2 with unit M_1 and operation FOP_4 with unit M_2. The semifinished objects are O_2 and O_3.

4.1 Generating the Algebraic Description

After the simulation model has been designed by using the HIMAC-specific building blocks, a file containing Table 1 can be generated. This table lists the algebra elements for this particular model. With this table it is possible to derive automatically the list of objects, the list of manufacturing units, the list of transport units and the list of manufacturing operations. Also, the layout of the factory is implicitly represented.

Index	Input	Output	Duration	Unit
FOP_1	O_1/S_1	O_2/S_2	1	M_1
FOP_2	O_1/S_1	O_3/S_2	2	M_1
FOP_3	O_2/S_2	O_4/S_3	2	M_2
FOP_4	O_3/S_2	O_5/S_3	1	M_2

Table 1: Algebraic-equivalent description of the manufacturing system.

4.2 Generating the Simulation Model

As mentioned in Section 3.1, it is also very important to generate an executable simulation model out of the algebraic description of the manufacturing system. Let us assume the description is given by a table like Table 1. The first step ist to compile this table into a representation which can be understood by the simulation system. In our case, this compilation results in a SimTalk program (see Figure 3). This program can be handed over from an external process to the simulation system via a remote procedure call. The instructions *OP.erzeugen*, *LA.erzeugen* and *MA.erzeugen* create the manufacturing operations, the storing and the manufacturing units

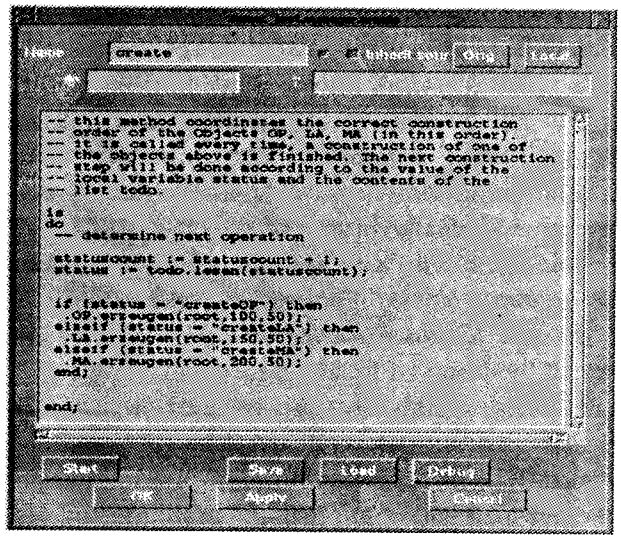

Figure 3: Method generating operations, storing and manufacturing units.

respectively according to the input list given by Table 1.

Now, the units must be connected in a way that corresponds to the layout of the factory. For this purpose, additional procedures are carried out. Figure 4 shows an example of a procedure called by *MA.erzeugen*. In the loop, all given manufacturing units are considered. For each unit, the predecessor unit (*actmachine.vorgaenger*), the successor unit (*actmachine.nachfolger*) and the operation to be performed (*actmachine.OPlocal*) are defined.

Figure 5 shows the generated manufacturing system. At the bottom, the storing and manufacturing units are shown. The upper row contains the event manager, which starts the simulation, the files with lists of operations, storing and manufacturing units, the protocol file, in which all events appearing during a simulation are taken down, and the manager file. The last file contains the procedures which were triggered by a remote procedure call to start the formation of the modelled manufacturing system.

4.3 Dynamics

The two previous sections concern the static aspects. The dynamic behaviour, i.e the control and the monitoring interfaces, are the contents

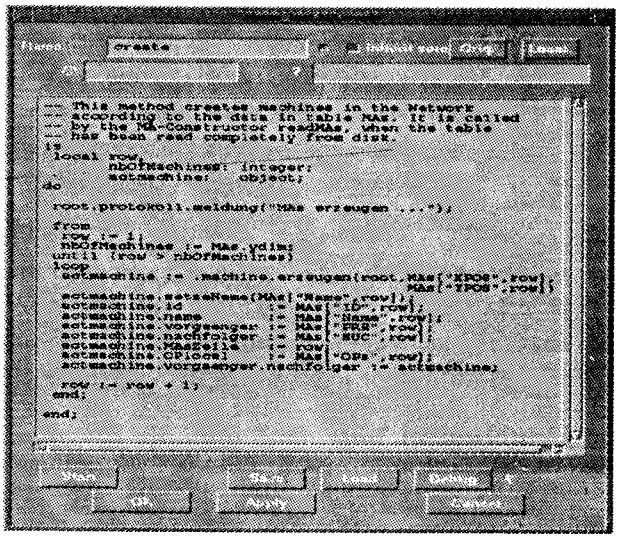

Figure 4: Creation of the connections between a manufacturing unit and its predecessor, its successor and the operation to be performed.

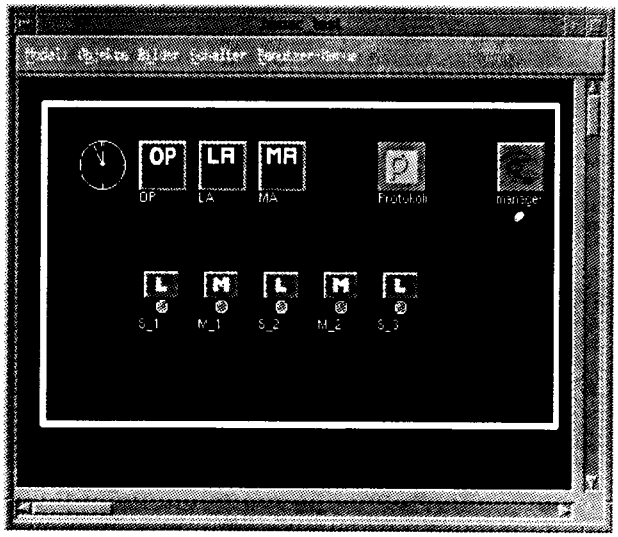

Figure 5: Simulation model of the example in Figure 2.

of this section. To describe the state equations, the space-time arrangement has to be taken into account: two objects of the same type are different, if they are not of the same place and/or the same time. Therefore, the vector describing the present state must be large enough to consider all possibilities: an object can be stored in each storing unit and at each time instant between 1 and the horizon n_j. The dimensionality of the vector is the result of the multiplication of the number of storing units, the horizon and the number of objects. In the example, the cardinality is $3 * 2 * 5 = 30$. Because of their size, the vectors and matrices are represented in their transposed form:

$$x_t^T = (\overbrace{\underbrace{x_{111}, x_{121}}_{O_1}, \underbrace{x_{112}, x_{122}}_{O_2}, x_{113}, \ldots, \underbrace{x_{115}, x_{125}}_{O_5}}^{S_1},$$

$$x_{211}, x_{221}, \quad \ldots, \quad \overbrace{x_{314}, \underbrace{x_{324}}_{*_2}}^{*_1}, x_{315}, x_{325})$$

$*_1$: objects available at time t

$*_2$: objects available at time $t+1$

The matrix B consists of all balance vectors:

$$B^T = \begin{pmatrix} \text{-1000000000} & \text{0010000000} & \text{0000000000} \\ \text{-1000000000} & \text{0000010000} & \text{0000000000} \\ \text{0000000000} & \text{0000000000} & \text{0000000000} \\ \text{0000000000} & \text{0000000000} & \text{0000000000} \\ \text{0000000000} & \text{0000000000} & \text{0000000000} \\ \text{0000000000} & \text{0000000000} & \text{0000000000} \\ \text{0000000000} & \text{00-10000000} & \text{0000000100} \\ \text{0000000000} & \text{0000-100000} & \text{0000000010} \end{pmatrix} \quad (3)$$

The balance vectors are deduced from Table 1. For this purpose, the input vector is substracted from the output vector, which also considers the working time. If an operation is not provided for a manufacturing unit, the balance vector is a zero vector.

Matrix A is always constructed in the same way (see 4), because it is only a "shift matrix". Only the diagonal parts are significant. The multiplication Ax results in a shift of the objects not yet available, but which are being produced, by one step in the state vector. The time until the availability of these objects becomes shorter. The objects in x_{i1k} and x_{i2k} are added together, because during the transition from state x_t to x_{t+1}, objects in x_{i2k} become available. Objects in x_{i1k} are already available at time t.

$$A = \begin{pmatrix} \begin{matrix} 1100000000 \\ 0000000000 \\ 0011000000 \\ 0000000000 \end{matrix} & & \\ \vdots & 0 & 0 \\ \begin{matrix} 0000000011 \\ 0000000000 \end{matrix} & & \\ & \begin{matrix} 1100000000 \\ 0000000000 \end{matrix} & \\ 0 & \vdots & 0 \\ & \begin{matrix} 0000000011 \\ 0000000000 \end{matrix} & \\ & & \begin{matrix} 1100000000 \\ 0000000000 \end{matrix} \\ 0 & 0 & \vdots \\ & & \begin{matrix} 0000000011 \\ 0000000000 \end{matrix} \end{pmatrix} \quad (4)$$

Given the matrices A and B and a controller, which delivers the commands u_t at time t, the future states can be computed.

In the example, the reference input of the controller is the production schedule represented by the Gantt-chart in Figure 6. The controller deter-

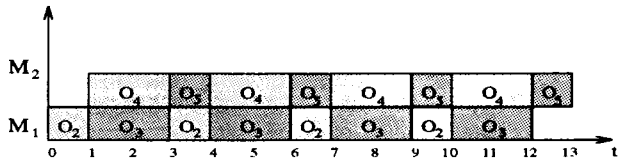

Figure 6: Gantt-chart for an optimal controller.

mines the next operation that has to be executed as close as possible to this schedule, i.e the commands are:

$$\begin{aligned} u_0^T &= (10000000), \quad u_1^T = (01000010) \\ u_2^T &= (00000000), \quad u_3^T = (10000001) \\ &\vdots \end{aligned} \quad (5)$$

It is assumed, that at time $t = 0$ nine objects O_1 are stored in storing unit S_1.

In (6), the results of the execution of the first eight commands are shown:

$$\begin{aligned} x_0 &= (9000000000 \quad 0000000000 \quad 0000000000) \\ x_1 &= (8000000000 \quad 0010000000 \quad 0000000000) \\ x_2 &= (7000000000 \quad 0000010000 \quad 0000000100) \\ x_3 &= (7000000000 \quad 0000100000 \quad 0000001000) \\ x_4 &= (6000000000 \quad 0010000000 \quad 0000001010) \\ x_5 &= (5000000000 \quad 0000010000 \quad 0000001110) \\ x_6 &= (5000000000 \quad 0000100000 \quad 0000002010) \\ x_7 &= (4000000000 \quad 0010000000 \quad 0000002020) \\ x_8 &= (3000000000 \quad 0000010000 \quad 0000002120) \end{aligned} \quad (6)$$

At time $t = 8$, six objects O_1 are already used up, two objects O_4 and two objects O_5 are produced, i.e. they are stored in storing unit S_3, and one object O_3 and one object O_4 are being produced.

5 Conclusion

The work presented in this paper is based on a manufacturing algebra which facilitates a static description of the manufacturing system by the definition of objects, operations and functions. Factory dynamics are expressed by a state equation. The equation is the core of this model.

Two different transformations between formal description and simulation model and the aim of these transformations were presented:

- From a simulation of a manufacturing system, an algebraic description is derived. This model description allows the design of closed-loop control systems.
- From a given algebraic description, a simulation model of the manufacturing system can be generated. For this purpose, the algebraic representation is translated into a SimTalk file, which is sent to the simulation system via a remote procedure call. This model transformation is needed if one wants to test and access a manufacturing system which was designed generically.

A simple example shows clearly, how these transformations were implemented.

Our present work concerns two examples of increasing complexity: The modelling and simulation of an assembly line and of a tool machine producing manufacturing system.

6 Acknowledgements

The research work was performed at the Institute for Real-Time Computer Systems and

Robotics (IPR), Prof. Dr.-Ing. U. Rembold and Prof. Dr.-Ing. R. Dillmann, Faculty for Computer Science, University of Karlsruhe. The research is supported by the Commission of the European Communities under ESPRIT-BR No. 8141 HIMAC.

References

[Bolch 89] G. Bolch: *Leistungsbewertung von Rechnersystemen mittels analytischer Warteschlangenmodelle.* B. G. Teubner, Stuttgart, 1989.

[Bernhart 92] W. Bernhart: *Montageplanung in CIM.* K-Feldmann, TÜV-Rheinland, 1992.

[Canuto 93] E. Canuto, F. Donati, M. Vallauri: *Factory Modelling and Production Control.* International Journal of Modelling and Simulation, 13 (4,1993), p.162 - 166.

[Canuto 94a] E. Canuto, F. Donati, M. Vallauri: *A New Approach to Modelling Manufacturing Systems.* Submitted to ISATA Int. dedicated Conf. on Lean Agile Manufacturing, Aachen (Germany), 31 Oct - 4 Nov 1994.

[Canuto 94b] E. Canuto, F. Donati, M. Vallauri: *Modelling Factory Dynamics for Production Control.* 2nd IEEE Mediterranean Symp. on New Directions in Control and Automation, Chania (Crete, Greece), June 1994.

[Canuto 95] E. Canuto, F. Donati, M. Vallauri: *Manufacturing Algebra and Factory Dynamics.* Internal Report #6, HIMAC–EP 8141, January 1995

[Glaser 92] H. Glaser, W. Geiger, V. Rohde: *PPS - Produktionsplanung und -steuerung; Grundlagen - Konzepte - Anwendungen.* Gabler, Wiesbaden (Germany), 1992.

[Kleinrock 75] L. Kleinrock: *Queueing Systems - Volume I: Theory.* John Wiley & Sons, 1975

[König 88] R. König, L. Quäck: *Petri-Netze in der Steuerungs- und Digitaltechnik.* Oldenburg Verlag, München Wien, 1988.

[Lavenberg 83] S. S. Lavenberg: *Computer Ierformance Modelling Handbook.* Notes in Computer Science and Applied Mathematics, Academic Press, 1983.

[Law 91] A. M. Law, W. D. Kelton: *Simulation Modeling and Analysis.* MacGraw-Hill Series in Industrial Engineering and Management Science, 1991.

[Reithofer 94] W. Reithofer, J. Raczkowsky, E. Canuto: *HIMAC - Hierarchical Management and Control in Manufacturing Systems.* CIMMOD/CIMDEV conference, Torino, September 1994.

[Rembold 93] U. Rembold, B. O. Nnaji, A. Storr: *Computer Integrated Manufacturing.* Addison-Wesley, 1993.

[Simple 94] SIMPLE++, Referenzhandbuch, AESOP Stuttgart, 1994.