AN AGENT-BASED ARCHITECTURE FOR SOLVING DYNAMIC RESOURCE ALLOCATION PROBLEMS IN MANUFACTURING

Lars Mönch, Marcel Stehli
Technical University of Ilmenau
Institute of Information Systems, Helmholtzplatz 3
D-98684 Ilmenau, Germany
E-mail:{Lars.Moench|Marcel.Stehli}@tu-ilmenau.de

Roland Schulz
TewiSoft GmbH
Ehrenbergstrasse 11
D-98684 Ilmenau, Germany
E-mail: Roland.Schulz@tewisoft.de

KEYWORDS
Manufacturing, Shop-Floor-Control, Agent-Based Modeling, Object-Oriented Frameworks, System Architecture

ABSTRACT

In this paper, we present results of agent-based modeling for the solution of dynamic resource allocation problems in manufacturing. After a concise problem description, we start by identifying the agents necessary for the required shop floor control functionality. Then we describe the architecture and implementation of a multi-agent-system prototype based on the object-oriented framework JAFMAS, a blackboard-type data layer and the discrete event simulator ParSimONY. We present results of computational experiments with our prototype multi-agent-system.

INTRODUCTION

Recently, distributed approaches to shop floor control have attracted a greater interest in the community (Roy and Anciaux 2001, Tranvouez et. al 2001). The two main reasons are the following: Firstly, important steps have been made towards a theoretical foundation and experimental evaluation of multi-agent-systems over the last five years. Secondly, the development of modern middleware and the further dissemination of the Java programming language have greatly reduced the implementation effort for this kind of system.
In this paper, we study a resource allocation scenario that stems from flexible manufacturing. Problems of this type have been discussed in the context of multi-agent-systems, for example in (Ottaway and Burns 2000) and in (Dewan and Joshi 2001).
Traditionally, in the context of shop-floor control, one uses a contract-net based solution for resource allocation problems. The existence of managers (clients) and contractors (servers) is essential for this approach. The basic idea comprises manager agents looking for task-solving entities and contractor agents offering task-solving abilities. The manager agents announce tasks, receive and evaluate bids from potential contractors. The contractor agents receive task announcements of the manager agents, evaluate his capability to respond and, if possible, respond with a bid and finally perform the task if his bid is accepted. The basic manager-contractor approach has the drawback that the acceptance competence is exclusively with the manager agent. An approach to solve this problem is the coordinator architecture introduced, for example, by (Zelewski 1997). Here, in addition to the manager and contractor role one integrates the coordinator (mediator) role into the multi-agent-system. Because the coordinator has the character of a centralized entity, the coordinator will be a bottleneck in large systems. Problems arise also in the case of a failure of the coordinator. However, for relatively small problems the coordinator architecture shows some advantage over the pure manager-contractor architecture.
Contract-net based resource allocation approaches can be used at the lowest level of distributed, hierarchical shop-floor control systems. Bidding procedures must be used in situations, where no valid schedule is available. The analysis and design results for such a system in the semiconductor manufacturing domain are described in (Mönch 2001).
The aim of our research was twofold. Firstly, we are interested in developing an architecture that allows for integrating a multi-agent-system and a discrete event simulator. For that purpose we extend the generic architecture suggested by Mönch et al. 2002 to the present case. Secondly, in order to carry out a validation of our research, we are interested in extending the results of a bidding procedure suggested by Mönch and Stehli 2002 from the static to the dynamic case. In this paper, we report about these two aspects of our research.

PROBLEM DESCRIPTION

We are interested in solving the following problem from manufacturing: Given n jobs, which have to process on m different machines, we are interested in minimizing the value of the performance measure Average Weighted Tardiness (AWT) of the n jobs:

$$T := \frac{1}{n}\sum_{i=1}^{n} w_i \max(0, c_i - d_i), \qquad (1)$$

where we denote by

w_i :	weight of job i,
c_i :	completion date of job i,
d_i :	due-date of job i.

Note that this performance measure is attractive in due-date oriented manufacturing environments (for example in semiconductor manufacturing).

We assume that each single machine is able to perform processing steps of more than one product. Furthermore, in order to investigate a more realistic scenario we consider the following process restrictions:

- We suppose the case of dynamic job arrival, i.e., each job i has an additionally attribute, the ready time r_i.
- There exists sequence-dependent setup times, which usually are a multiple of the pure processing time.
- In the manufacturing environment under investigation, we find charge production. Here, we define a charge as a temporary collection of different jobs with the aim to process these jobs at the same time on the same machine. The basic decision is whether to wait for jobs in order to obtain a full charge or to process the (non-full) charge immediately.
- Machine failures occur in a stochastic manner.
- The material handling system is assumed always available.
- Transportation times are assumed included into the processing times of the single jobs.

For problems of this type, there is a vast literature. However, we found only a small number of papers dealing with questions like how to choose costs and how to implement distributed prototype multi-agents-systems for such situations (cf. Brennan and William 2000 for the implementation of such a scenario using DCOM technology).

AGENTIFICATION OF THE PROBLEM

Multi-Agent-Systems offer a way to obtain the desired compromise between centralized control and fully decentralized control (cf. Baker 1998 and Van Brussel et al. 1998). Starting with the PROSA architecture (Van Brussel et al. 1998), we distinguish between decision-making agents and staff agents. Decision-making agents solve decision problems while the staff agents try to support them in the course of the decision-making process. In the PROSA architecture, we find order, product and resource agents as abstract classes.

Starting from the results related to the PROSA architecture, we identified three basic agent types in our application scenario:

- Each job agent represents a single job. We use job agents to monitor the job's production progress.
- Machine agents represent a single machine on the shop floor. The machine agents keep track of the production progress of the machine and its response to the requests of the mediator agent.
- Mediator agents are required to realize a matching between the job agents and the machine agents. A mediator agent is a kind of a staff agent as suggested in the PROSA architecture. It supports the decision-making entities job agent and machine agent during the fulfillment of their own goals.

Note, that we do not describe any further staff agents besides the mediator agent, because we are more interested in implementation issues and, in this paper, try to keep the scenario as simple as possible. However, as explained in (Mönch 2001) it is extremely important to identify the proper hierarchies and staff agents in order to develop a multi-agent-system for a special application domain. Our research does not model product agents explicitly because we consider product information in our study as static information. Note, that this is usually not true in more realistic application scenarios. We show the used agents in Figure 1.

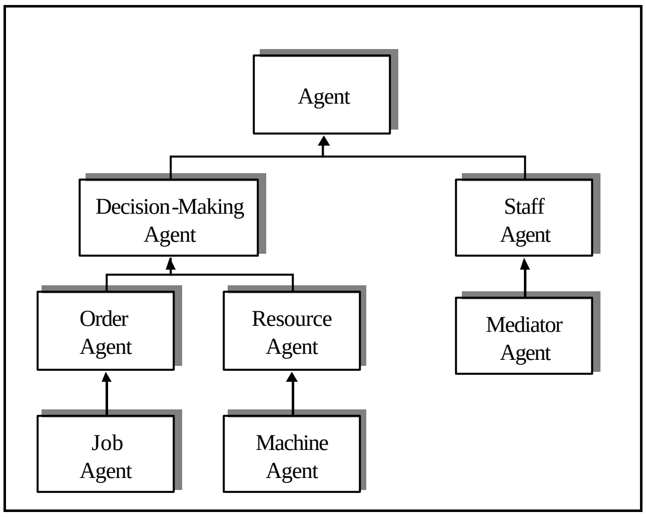

Figure 1: Agentification of the Manufacturing Problem

BIDDING SCHEME

We are interested in a robust bidding scheme that mimics the situation in a real shop floor. The basic idea that was suggested by two of the present authors (Mönch and Stehli 2002) is the introduction of properly chosen machine costs. We use these machine costs to give each job agent a certain amount of currency. For the sake of completeness we repeat the main ingredients of our bidding scheme.

Introduction of Machine Hour Rates

Machine hour rates are calculated as follows (cf. Warnecke et al. 1996). In a first step, we calculate the following two quantities for each machine of the shop floor (usually we choose one year as a fixed period T):

a) $t_{fail,i}$ is an estimate of the time (per period T) that the i-th machine is unavailable due to technical reasons. The time for repair, preventative maintenance, breaks and holidays is included into the quantity $t_{fail,i}$.

b) $t_{tech,i}$ is an estimate of the time that the i-th machine is unavailable due to technological reasons. The quantity $t_{tech,i}$ is determined by tests (especially in semiconductor manufacturing) in order to choose appropriate parameters of the machine.

We obtain the amount of time of machine i, designated to processing jobs (per period T) as follows:

$$t_{t,i}(T) := T - (t_{fail,i} + t_{tech,i}) . \qquad (2)$$

In a second step, it is necessary to calculate the costs that accumulate at a single machine. We have to distinguish between variable and fixed costs. Variable costs depend on the production consumption of the machine. Examples for variable costs C_{var} (per period T) are:

- costs for special materials for processing of process steps on the machine (for example, special gases in semiconductor manufacturing)
- costs for using the machine during the production process (for example for electrical energy)
- costs for maintenance, which depend on the number of completed process steps.

Fixed costs C_{fix} are necessary in order to model expenses that are independent of a machine's usage (for example, rental costs for the shop floor, wear and tear of the machine).

We determine the machine hour rate mh_i for machine i by using the following formula:

$$mh_i := \frac{C_{var,i} + C_{fix,i}}{t_{t,i}(T)}, \qquad (4)$$

i.e., the machine hour rate is given by the cost of the machine related to one single time unit of usage.

Machine hour rates are implemented in many enterprise resource-planning systems

Costs for Producing Jobs of a Single Product

We denote the set of products by $P := \{p_1, ..., p_m\}$. Each product p_i has a process flow $S := (s_{1i}, s_{2i}, ..., s_{ni})$. Here, we denote by s_{ki} the k-th process step of p_i. The time t_{ki} is required for processing s_{ki} on machine ki. The costs for the production of a job of product p_i are calculated as follows:

$$C(p_i) := \sum_{k=1}^{n} aver(mh_{ki}) aver(t_{ki}) . \qquad (5)$$

We use the notation $aver()$ in order to indicate, that we use average values for machine hour rates and for processing times in the case of parallel machines.

Costs for Job Agents

The goal of the job agents is to optimize the purchase of scarce machine capacity provided by the machine agents. The achievement of this goal is controlled by the objective function

$$g_J := \max(c_J - d_J, 0), \qquad (6)$$

where we denote by

c_J :	completion time of job J,
d_J :	due-date of job J.

Note, that the objective function of a single agent (6) is a part of the objective function of the whole production system (1).

Suppose, that agent A_J is designated to produce a job of product p_i. The budget

$$B(A_J) := (1 - w_t - w_m) \frac{C(p_i)}{\sum_{k \in P} \lambda_k C(p_k)} + w_t T_J + w_m M_J, \quad (7)$$

where

T_J :	measure of the due-date priority of job J,
M_J :	measure of the importance of job J from management point of view,
w_t :	weight for due-date priority,
w_m :	weight for management importance,
λ_k :	frequency of jobs of product p_k,

is given to the agent after release into the shop floor. Here, $w_t + w_m \le 1$, $w_t, w_m \ge 0$ and $\sum_{k \in P} \lambda_k = 1$ and $\lambda_k \ge 0$ are valid.

The due-date priority of a job J is given by the ratio:

$$T_J := \frac{\text{remaining processing time}}{\text{due-date-actual time}} . \qquad (8)$$

We calculate the management importance of job J as follows:

$$M_J := \begin{cases} \frac{1}{4}, & \text{if job J is not important} \\ \frac{1}{2}, & \text{if job J is important} \\ \frac{3}{4}, & \text{if job J is very important} \end{cases} . \quad (9)$$

The job agent pays a price P to the corresponding machine agent for carrying out a process step of the job on the machine represented by the machine agent. This price reduces the budget $B_{old}(A_J)$ of the job agent by P. The new budget is denoted by $B_{new}(A_J)$. In order to avoid a hold of jobs, that have a too small budget, a refresh of the budgets is carried out after a fixed amount of time. We consider only these jobs for budget update that required no machine capacity for a time $t_{threshold}$. We determine the new budget $B_{new}(A_J)$ as follows:

$$B_{new}(A_J) := \max\left(\frac{\sum_{k=l+1}^{n} aver(mh_{ki}) aver(t_{ki})}{\sum_{k \in P} \lambda_k C(p_k)}, B_{new}(A_J) \right), (10)$$

where we suppose, that the job has finished the process step s_l and the next process step is s_{l+1}. The update scheme (10) ensures, that each job agent has at least that budget required to pay the machine costs based on (5) for its remaining process steps.

Costs for Machine Agents

The goal of the machine agents is full utilization of the machines. This goal includes especially the minimization of setup-times and a maximization of the load, because this yields an increase of the dynamic capacity of the shop floor. Therefore, we consider the following combined objective function

$$G(M_i) := \min_{k \in I} \left(\alpha_1 SC_k + \alpha_2 (1 - UM_k) \right), \qquad (11)$$

where $\alpha_1 + \alpha_2 = 1$ and $\alpha_i \geq 0$ are valid. Here, we use the following notation:

I:	Set of indices of the jobs (charges) that are waiting in front of the machine,
SC_k :	normalized setup costs,
UM_k :	normalized fullness of charge k ,
α_i :	weight of the i-th objective.

A machine agent offers the job agent (product p_i , actual process step s_{ki}), processing capabilities at the price:

$$P(M_i) := (1 - w_s - w_{ch}) \frac{mh_i\, t_{ki}}{\sum_{k \in P} \lambda_k C(p_k)} + w_s SC_k + w_{ch}(1 - UM_k) , \qquad (12)$$

where $w_s + w_{ch} \leq 1, w_s, w_{ch} \geq 0$ hold. We denote by w_s the weight of the setup costs and by w_{ch} the weight of the fullness of the charge.

Note, that in case of a full load the necessary price to be paid is lower than in case of a non-full charge. Shorter processing times lead also to lower prices under the assumption of identical machine hour rates.

OVERALL ARCHITECTURE OF THE SYSTEM

Multi-Agent-System Prototype

An agent is a software entity that allows for autonomous decisions. By using communication capabilities, an agent demonstrates also cooperative behavior (Nwana et al. 1998).

We used the object-oriented framework JAFMAS (Chauhan 1997) as a starting point for a prototype of a multi-agent-system. The framework facilitates the development of multi-agent-systems by hiding most of the implementation details of communication capabilities. JAFMAS is a white-box framework (cf. (Pree 1996)) based on Java software technology. An abstract basic class agent is used in JAFMAS. Each agent contains references to objects of different classes that support commu nication. The MulticastCom object provides functionality to implement a multicast communication in the multi-agent-system. Concrete classes must be derived from the abstract classes DirectedComImpl and DirectedComIn in order to use direct communication abilities of the agents.

For communication purposes, each agent contains references of different conversation objects. A certain conversation consists of a set of conversation rules that specify the behavior of the agent in a situation dependent manner. We refer the reader to the work (Stehli 2002) and the paper (Mönch and Stehli 2002) for the details of the implementation of the multi-agent-system prototype.

Discrete Event Simulator for Shop-Floor Emulation

We use a simulation environment in order to emulate the behavior of a shop-floor (cf. Mönch et al. 2002 for a generic framework handling this situation). A real shop-floor communicates with the shop-floor control and/or scheduling system over a message bus. Additionally, an event-based online communication between the simulation and the shop-floor control software has to be established in order to realize the control of the material flow in the simulator based on decisions of the shop-floor control software.

We decided to use the simulation package ParSimONY (Preiss and Wan 1999). ParSimONY is a discrete event simulator written in the Java programming language that was especially developed for distributed simulation. Classes that allow for the simulation of manufacturing systems were added by Schulz (cf. Schulz 2001). A subscription/notification mechanism was added in order to send information from the simulation model to a data layer. For that purpose special events were defined. These events allow for a monitoring of machine and lot states. The event handling mechanism was implemented in the component PSY_Event (see Figure 3). The use of a discrete event simulator for performance evaluation of multi-agent-systems was described by Henoch and Ulrich 2000 and by Brennan and William 2000.

Blackboard-Type Data Layer

We adapt the data layer described by Mönch et al. 2002 to the present situation. The scheduling system described there has to be replaced by the multi-agent-system. Instead of using the discrete event simulator AutoSched AP we will use the simulator ParSimONY. We can distinguish between two types of data in the blackboard. We collect static data like

- information about process flows
- setup information
- existing machines
- processing times

and dynamic data like

- lot release information

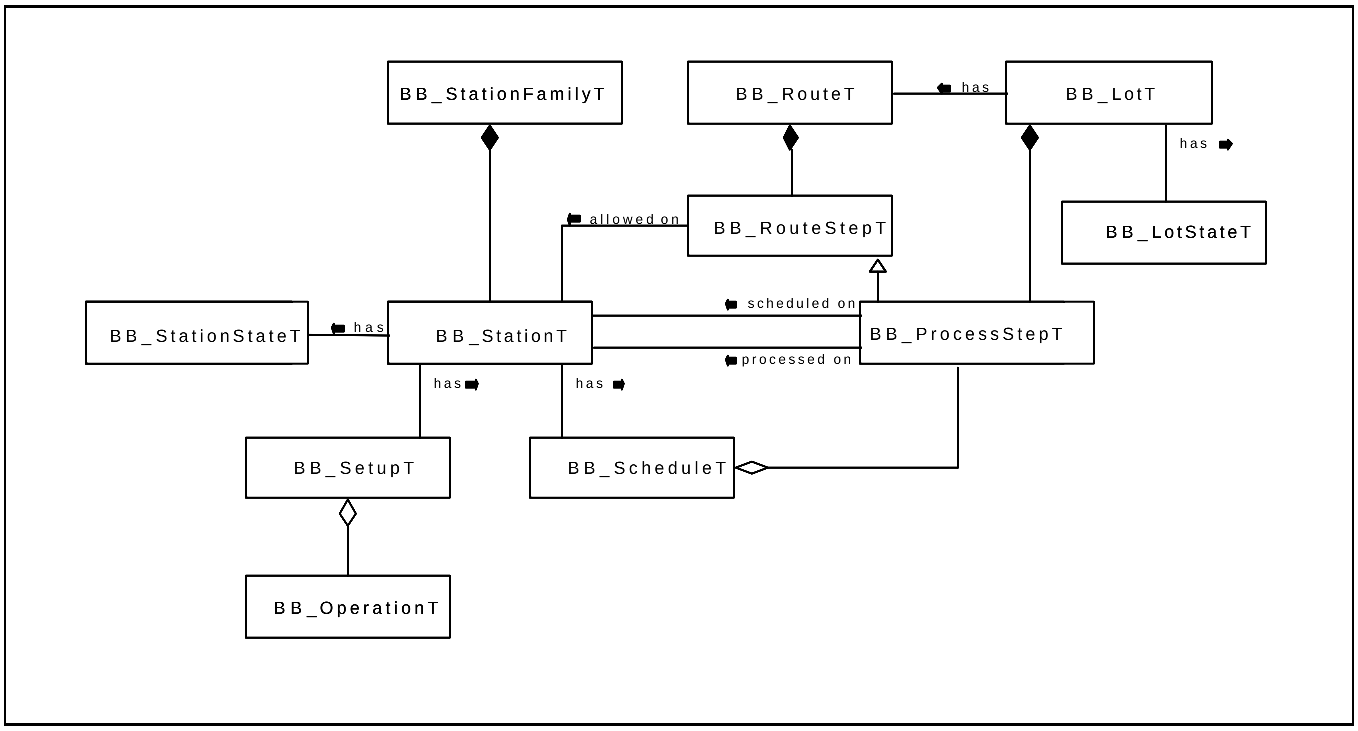

Figure 2 UML Class Diagram of the Black-Board

- lot states
- machine states
- setup states of a certain machine

in the blackboard. The blackboard is used in different situations.

1. The multi-agent-system reads information from the blackboard.
2. We initialize the blackboard at the beginning of the simulation run by reading all required information from the corresponding simulation objects.
3. We update the objects of the blackboard during the simulation run in an event-driven manner.

The blackboard was developed in the Java programming language. The UML class diagram for the blackboard is presented in Figure 2.

A timer starts the multi-agent-system in the case of time-driven triggering. This approach allows for the application of rolling horizon approaches. On the other hand, events of the shop-floor, i.e., simulation events, trigger the start of the multi-agent-system. The multi-agent-system uses only date stored in the blackboard in order to calculate schedules. No internal information from the simulation model is going to be used. By using this architecture, in principle, the simulator could be replaced by an arbitrary manufacturing execution system of a real factory. We can see the described architecture in Figure 3.

RESULTS OF COMPUTATIONAL EXPERIMENTS

Experimental Design

We assume a manufacturing system that consists of four different machines. We see the corresponding machine data of the scenario in Table 1. We consider the case of three different products. The corresponding product data can be found in Table 2.

Performance Evaluation

We use the performance measure AWT given by (1) in our experiments. The product mix uses the same number of jobs of each product. We compare our bidding scheme based on machine hour rates (MHR) with the First-In-First-Out (FIFO) dispatching rule. The percentage of important jobs was 80%.

Results of simulation runs can be seen in Figures 4. MHR outperforms the simple heuristic that is widely used in shop-floor control. MHR is better suited to the task of distributing the jobs over machines with different characteristics.

In order to measure the performance of our approach more sophisticated dispatching rules (for example the apparent tardiness cost rule Pinedo 1995) and other contract net approaches should be included into the experiments.

Table 1 Machine Data

Machine	Process Step	Duration (Time Units)	Machine Hour Rates
1	A	4	7
	B	6	7
2	A	6	5
	B	8	5
	C	3	5
3	B	6	7
	C	5	7
	D	6	7
4	D	8	5

Table 2 Product Definitions

Product	Process Flow
1	A-B-C
2	B-D-C
3	A-D-B-C

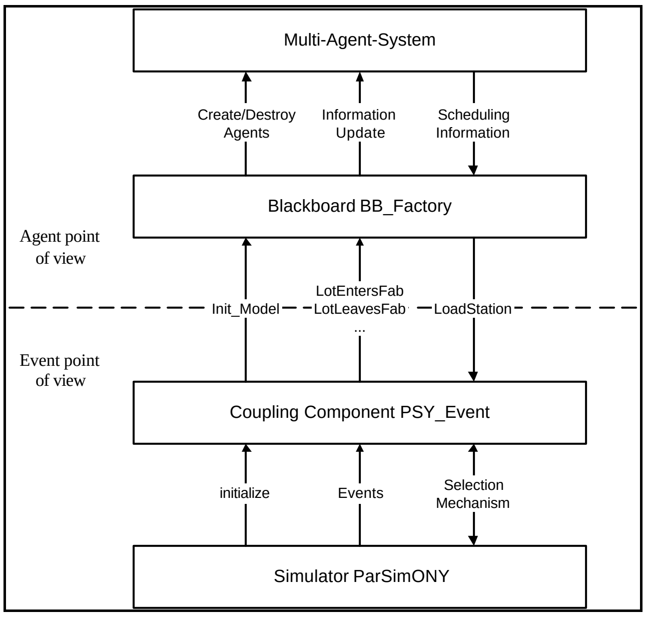

Figure 3 System Architecture

SUMMARY

In this paper, we present the results of agent-based modeling for dynamic resource allocation problems in manufacturing. We described an architecture including the multi-agent-system, a blackboard and a discrete event simulator. We report on the prototypical implementation. We present the results of some computational experiments in the last part of the paper.

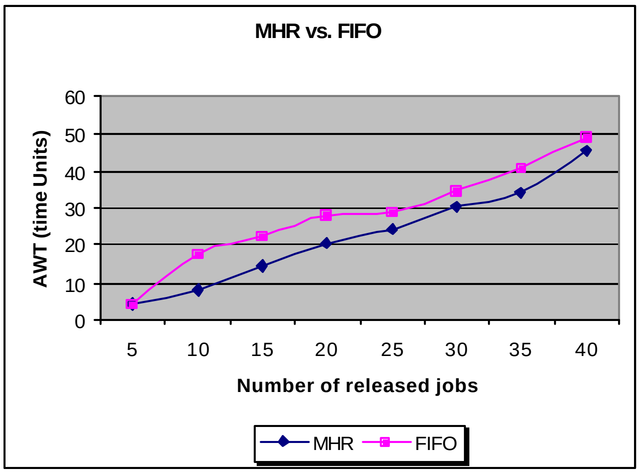

Figure 4: Performance Comparison based on AWT

ACKNOWLEDGEMENT: The first and the second author gratefully acknowledge the support of the Deutsche Forschungsgemeinschaft (DFG) - Priority Research Program 1083 “Intelligent Agents and Realistic Commercial Application Scenarios” for this work. The authors would like to thank Frank Mock for his valuable programming efforts.

REFERENCES

Baker, D. A. 1998. "A Survey of Factory Control Algorithms which Can be Implemented in a Multi-Agent Heterarchy: Dispatching, Scheduling, and Pull". *Journal of Manufacturing Systems*, Vol. 17/No. 4, 297-320.

Brennan, R. W. and O. William. 2000. "A Simulation Test-Bed to Evaluate Multi-Agent Control of Manufacturing Systems". In *Proceedings of the 2000 Winter Simulation Conference*, J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds., 1747-1756.

Chauhan, C. 1997. JAFMAS: "A Java-based Agent Framework for Multiagent Systems – Development and Implementation". *PhD* Dissertation, University of Cincinnati.

Dewan, P. and S. Joshi. 2001. "Implementation of an Auction-Based Distributed Scheduling Model for a Dynamic Job Shop Environment". *International Journal of Computer Integrated Manufacturing*, Vol. 14, No. 5, 446-456.

Henoch, J. and H. Ulrich. 2000. "HIDES: Towards an Agent-Based Simulator". In *Proceedings of the Workshop 2000 “Agent Based Simulation”*, eds.: C. Urban, 259-263.

Krothapalli, N. K. C. and A. V. Deshmukh. 1999. "Design of Negotiation Protocols for Multi-Agent Manufacturing Systems". *International Journal of Production Research*, Vol. 37, No. 7, 1601-1624.

Mönch, L. 2001. "Towards an Agent-Based Production Control in the Semiconductor Industry". In *Proceedings 13th European Simulation Symposium (ESS 2001)*, Marseille, 941-945.

Mönch, L., O. Rose and R. Sturm. 2002. "Framework for the Performance Assessment of Shop-Floor Control Systems". In *Proceedings of the International Conference on Modeling and Analysis of Semiconductor Manufacturing (MASM 2002)*. Tempe, 95-100.

Mönch, L. and M. Stehli. 2002. "Agent-Based Modeling and Implementation of Resource Allocation Scenarios in Manufacturing". In *Proceedings 3rd International Workshop on Agent-Based Simulation*, Passau, 149-154.

Nwana, H. S., L. Lee, and N. R. Jennings. 1998. "Coordination in Multi-Agent Systems". In *"Software Agents and Soft Computing – Towards Enhancing Machine Intelligence"*, Nwana, H.S., Azarmi, N. (Eds), Lecture Notes in Artificial Intelligence, Springer, 42-58.

Ottaway, T. A. and J. R. Burns. 2000. "An Adaptive Production Control System Using Agent Technology". *International Journal of Production Research*, Vol. 38, No. 4, 721-737.

Parunak, H. V. Dyke, A. D. Baker, and S. J. Clark. 1998. "The AARIA Agent Architecture: From Manufacturing Requirements to Agent-Based System Design". In *Proceedings ICAA’98, Workshop on Agent-Based Manufacturing.*

Pinedo, M. 1995. *Scheduling: Theory, Algorithms and Systems*. Prentice Hall, N.J.

Pree, W. 1996. *Framework Pattern.* SIGS Books, New York City.

Preiss, B. R. and K. W. C. Wan. 1999. "The Parsimony Project: A Distributed Simulation Testbed in Java". In *Proceedings of the 1999 International Conference On Web-Based Modelling & Simulation.* 89-94.

Roy, D. and D. Anciaux. 2001. "Shop-Floor Control: a Multi-Agent Approach". *International Journal of Computer Integrated Manufacturing,* Vol. 14, No. 6, 535-544.

Stehli, M. 2002. "Konzeption und Implementierung eines kontraktnetzartigen Allokationsmechanismus unter Benutzung eines objektorientierten Frameworks". Diplomarbeit, Technische Universität Ilmenau, Institut für Wirtschaftsinformatik.

Schulz, R. 2001. "Einsatz der Parallelen und Verteilten Simulation zur Simulation von Produktionssystemen". In *Proceedings "Simulation und Visualisierung 2001".* 67-78.

Tranvouez, E., A. Ferrarini, and B. Espinasse. 2001. "A Multi-Agent Modelling and Simulation of Workshop Disruptions Management by Cooperative Rescheduling Strategies". In *Proceedings 13th European Simulation Symposium (ESS 2001),Marseille,* 917-924.

Van Brussel, H., J. Wyns, P. Valckenaers, L. Bongaerts, and P. Peeters. 1998. "Reference Architecture for Holonic Manufacturing Systems: PROSA". *Computers in Industry, Special Issue on Intelligent Manufacturing Systems,* 37(3), 225-276.

Warnecke, H. J., H.-J. Bullinger, R. Hichert, and A. Voegele. 1996. *Kostenrechnung für Ingenieure.* Carl Hanser Verlag, München, Wien.

Zelewski, S. 1997. "Elektronische Märkte zur Prozeßkoordination in Produktionsnetzwerken". *Wirtschaftsinformatik* 39, (1997) 3, 231-243.

AUTHOR BIOGRAPHIES

LARS MÖNCH is an Assistant Professor in the Department of Information Systems at the Technical University of Ilmenau, Germany. He received a master’s degree in applied mathematics in 1994 and a Ph.D. in the same subject from the University of Göttingen. He worked at Softlab GmbH, Munich, in the area of object-oriented software development. His research interests are in simulation-based production control of semiconductor wafer fabs, applied optimization and artificial intelligence applications in manufacturing. He is a member of GI (German Chapter of the ACM), GOR (German Operations Research Society) and SCS. His email address is `<Lars.Moench@tu-ilmenau.de>`.

MARCEL STEHLI is a Ph.D. student in the Department of Information Systems at the Technical University of Ilmenau, Germany. He received a master’s degree in Information Systems in 2002. His research interests are in agent-based modeling applied to manufacturing problems and in object-oriented software development. His email address is `<Marcel.Stehli@tu-ilmenau.de>`.

ROLAND SCHULZ is a software development engineer at TewiSoft GmbH, Ilmenau, Germany. Prior to his current position he was a Ph.D. student in the Department of Information Systems at the Technical University of Ilmenau, Germany. He will finish his Ph.D. dissertation in autumn 2002. His research interests are in distributed and parallel simulation and in simulation applications in manufacturing. His email address is `<Roland.-Schulz@tewisoft.de>`.