

# Design and Implementation of Distributed Scheduling Systems for Complex Manufacturing Processes: a Software Agent-based Perspective

L. Mönch

Chair of Enterprise-wide Software Systems  
Department of Mathematics and Computer Science  
FernUniversität in Hagen  
Hagen, Germany

**Abstract** - *Complex manufacturing systems are characterized by an over time changing product mix, prescribed due dates, a huge number of machines, a large number of routes for a diverse product mix, and internal and external disturbances. We have to take into account these properties in course of designing and implementing scheduling systems, i.e., these systems have to be adapted without large efforts to changing requirements of the manufacturing process. In this paper, we discuss the question how we have to design and construct scheduling systems for the complex manufacturing systems domain. We show that information systems based on software agents are an appropriate way to develop distributed scheduling systems that can adapt to various circumstances of the manufacturing process. Furthermore, we show that agent-based frameworks offer a way to implement the scheduling systems in an efficient manner.*

**Keywords:** Distributed Scheduling, Requirement Analysis, Agent-based Systems, Packaged Enterprise-wide Software Systems, Kernel-Shell-Model

## 1 Introduction

Over the last years, markets have been changed from stock driven to customer demand driven markets. Shorter innovation cycles, more customized product specifications, and shorter delivery times are the result. This situation requires new enterprise-wide scheduling systems [13], [9], [10]. Taking this point of view into account there are three sources for the research described in this paper.

The first source is given by concrete needs from a practitioner's point of view. The manufacturing of modern products of the high-technology is performed in a highly automated way. The existing manufacturing processes are complex and take place in a globalized world. Therefore, a production control that is mainly performed manually by people from the production control department and from foreman on the shop floor is out of date.

The second source is given by technology pressure from an information technology point of view. This pressure is

provided by increased capabilities of computers, novel hardware-software combination like the live cache technology, and innovative software technologies like, for example, middleware. Furthermore, new approaches from Operations Research (OR) and Artificial Intelligence (AI) have to be taken into account. The renaissance of these methods is influenced to a certain degree by the further development of the information technology. Enterprise-wide scheduling systems are not a vision far away in our days.

The third source is determined by the software crisis, i.e., existing software for scheduling of manufacturing processes does not fulfill the requirements of the users from a functional and also a software quality point of view. The study [9] reports that more than a third of the interviewed companies use extended and modified versions of existing packaged software for scheduling purposes for the semiconductor manufacturing domain. Leading companies in US, Asia, and Europe participate in this study. Friedrich presented a similar pessimistic view for Advanced Planning and Scheduling (APS) Systems used in the process industry in [9]. The time needed for development and go live of these systems is long. The developed software systems are not flexible enough. It is hard to extend and maintain them. Furthermore, we have to face with interoperability problems with other information systems.

Based on these sources it follows that the construction, implementation, and the maintenance of software systems for scheduling is a non-trivial research object of the academic information systems community.

The paper is organized as follows. In Section 2, we present requirements for scheduling software from architecture and also from a more algorithmic point of view. Then we describe several approaches for reusing software artifacts in scheduling systems in Section 3. We show that frameworks offer a lot of advantage from a reuse point of view. We present some details of a distributed

scheduling system for production control in the high-tech industry that is implemented based on software agents in Section 4.

## 2 Problem setting and requirements for production control systems

In this paper, we consider complex discrete manufacturing systems of flexible job shop type. Therefore, the base system consists of groups of parallel machines. For the base process, dynamic lot arrivals, sequence-dependent set-up times, and batch processes are typical. Batching means that several lots have to be processed at the same time on the same machine. Complex manufacturing systems can be decomposed into work areas. Each work area consists of several groups of parallel machines. The production control for lots of one work area is performed usually in an autonomous manner, i.e. with own rights and own goals. Information systems are the carrier of the algorithms. Therefore, it makes sense to differentiate between requirements from an information system and also from an algorithm point of view. Because of the changing base systems and base processes, the ability to extend the software from a functional point of view is essential. Modern software systems for production control have to interact with many other information systems on the shop floor. Therefore, they should be able to communicate with other software systems. Because of the physical decomposition of the base system, production control systems have to be a carrier for distributed scheduling approaches. Scheduling approaches can be customized by an appropriate parameter setting. In some cases, this parameterization is not able to adapt the algorithms to new circumstances. It is therefore necessary to change the algorithms. This situation requires a strict separation of the structure of the production control system from the algorithms. Usually the performance of new algorithms is unknown for a given base system and base process. Therefore, it is necessary to use discrete event simulation to assess the performance of new production scheduling approaches in an ex-ante manner.

Production control systems have to be a carrier of highly efficient production control approaches. The used production control schemes are usually heuristics because most of the scheduling problems to be solved are NP hard [17]. The production control problems to be addressed have to be related to the entire manufacturing system because we have to achieve global goals like minimizing tardiness or maximizing throughput. Distributed hierarchical approaches seem to be appropriate to deal with the complexity [18]. Therefore, the representation of hierarchies has to be supported.

Usually, there is only a small period of time for making decisions. Therefore, the used scheduling approaches have to fulfill the any-time property, i.e., a feasible solution of the scheduling problem has to exist for each point of time  $t > 0$ . Usually, iterative procedures are necessary. The algorithms have to be encapsulated. Because of changing base system and base process conditions they have to be replaced under certain circumstances.

In this paper, we are interested in describing different ways to construct software for production scheduling. It will be turned out that agent-based frameworks are appropriate to build software systems for production control that fulfill the discussed requirements.

## 3 Reuse of Software Artifacts for Production Control

Packaged software for production control is widely used. Packaged software is software that may be applied in many enterprises. Packaged software systems have to own an appropriate data model. Hierarchies of production control objectives have to be modeled [17]. The use of packaged software requires the quasi standardization of business processes [4]. However, very often production planning and control is one of the core business areas of an enterprise. Usually, a successful company is not willing to open its production control strategy for other companies. In summary, it is difficult to use packaged software for production control tasks directly.

Specialized modeling languages contain language constructs for modeling production scheduling problems on a very high abstraction level. Examples for this approach are given by the products of the software vendor ILOG [11] for the solution of mixed-integer programs and constraint satisfaction problems. However, focusing only on the combinatorial complexity problem is not enough to construct a scheduling system. Other questions like integration of the algorithms into information systems and data providing for running the algorithms are also important.

Source code generators provide a third type of reuse approaches. Here, a meta-language is used for input specification purposes. This approach focuses usually on a specific domain. It is used, for example, for the construction of dispatch-rule-based production control software in the semiconductor manufacturing domain [1]. Beside integration and data providing problems it is usually difficult to construct an appropriate meta-language that is generic and specific at the same time. Furthermore, the maintenance of large-scale software systems that are based on generated source code is challenging.



Agent behaviors, behavior states, transition of behavior states, and the action item list allow that an agent may provide services for other agents

### 4.3 Infrastructure Issues

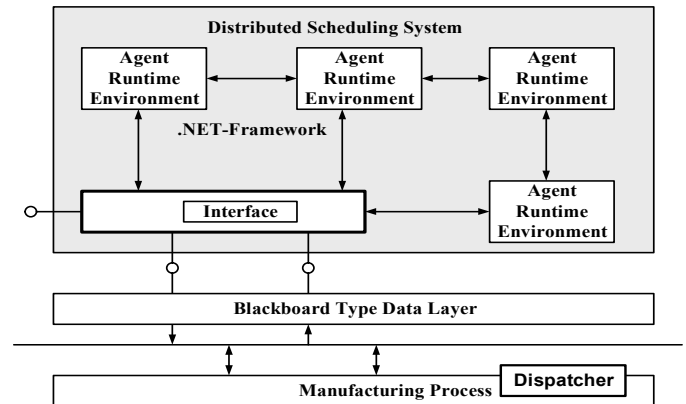
An Agent Runtime Environment had been developed that extends the FIPA Abstract Architecture [7] to ensure compatibility with existing standards. This runtime environment provides the infrastructure for agent execution and allows the agents a concurrent execution on the same host. The FIPA Abstract Architecture specifies an Agent Directory, a Message Transport System, an Agent Communication Language (ACL), and a Service Directory as mandatory parts.

In the framework, the FIPA Agent Directory and Service Directory are encapsulated by the Directory Service Agent. Furthermore, the Message Transport and ACL processing is encapsulated by the Agent Communicator. The two additional parts implemented in the Runtime Environment are the Agent Container that holds all local agents hosted within the environment and the Agent Management System (AMS) responsible for creating, destroying, or migrating the agents hosted on that platform.

Communication among autonomous agents is a key feature of multi-agent-systems to enable cooperation and coordination processes. The communicational part of the framework is organized in a way that makes it possible to compose and restrict these features depending on the chosen organizational structure of the multi-agent-system. We have to introduce communicational rights to the multi-agent-system framework because we implement the basic system using some kind of peer-to-peer service announcement system instead of the FIPA suggested global directory facilitator as a single point of failure. We use the .NET remoting framework for implementing the communication abilities. The Directory Service Agent is important for the environment. It manages the hierarchical as well as the local organization of the agents within a multi-agent-system. It is also responsible for establishing connections between different remote runtimes and the organizational communication with other known runtime environments. To optimize the overall system performance, several runtime environments can run on local and remote computers to distribute a multi-agent-system. When different runtimes are known to each other, the agents on these runtimes can communicate and cooperate with respect to their communicational rights. These rights are also used for publishing the agent's services that are provided within the multi-agent-system. A peer-to-peer service communication is utilized to distribute the different agent services among the known runtime environments. To store the remote service information, a local service-blackboard

at each runtime is updated. Each agent can explore this blackboard in order to find an agent offering the required service.

A distributed scheduling system contains usually several runtimes. Furthermore, we use a blackboard-type data layer that connects the scheduling system with the manufacturing process. The blackboard contains objects like lots, routes, and machines. These objects are updated in an event-driven manner from the manufacturing process. The blackboard acts like a mirror. Figure 2 shows the interaction between distributed scheduling system, blackboard, and manufacturing process.



**Figure 2:** Basic Architecture of a Distributed Scheduling System

In order to build a distributed agent hierarchy across different platforms, a unique identifier, called Hierarchy Identifier (HI), is introduced. This identifier is the organizational knowledge base for each agent. It provides the agents with a system wide unique hierarchy name that does not depend on the runtime where the agent is hosted. Furthermore, the HI stores the potential parent agent of a hierarchically organized agent and keeps track of its subsidiary child agents. The hierarchical organization of the agents is performed automatically as soon as an agent gets alive. The hierarchical organization system owned by each single runtime helps the agent to find its place within the hierarchy by looking for the appropriate parent and child agents.

### 4.4 Decoupling of Structure and Algorithms

The required separation of production control system from production control algorithms can be achieved by differentiating between decision-making agents and staff agents according to the PROSA reference architecture [19]. Staff agents support the decision-making agents in course of their decision-making process. Staff agents usually encapsulate scheduling and monitoring functionality. They

offer a broader view on the production control problem under consideration. Staff agents serve as a container for implementing different OR and AI algorithms. Therefore, they are essential in order to ensure the required exchangeability of the algorithms. In principle, we have to instantiate simply a new staff agent in order to use a new production control algorithm.

Decision-making agents are responsible for choosing appropriate parameters and feeding them to the staff agents. The different interactions between decision-making agents and staff agents can be described in a rather generic way. The differentiation of decision-making and staff agents offers the additional advantage that the any-time property of scheduling algorithms from Section 2 can be ensured more easily. We summarize the fulfillment of the requirements of Section 2 by the agent-based framework in Table 1. We used the outlined framework to implement the FABMAS prototype [16]. FABMAS is a hierarchically organized multi-agent-system for production control of semiconductor manufacturing processes. Semiconductor manufacturing processes are examples for complex manufacturing processes.

A three-layer hierarchical production control approach is suggested for the FABMAS system due to the physical decomposition of the shop floor of a wafer fab into work areas and on the next level into machine groups that contain parallel machines. Hierarchical communicational rights are applied due to the structure of the feedback enabled scheduling process. Production control guidelines and schedules are communicated top-down while process information is provided bottom-up.

#### 4.5 Interpretation within the Kernel-Shell-Model

The outlined agent-based framework and distributed production control schemes based on this framework may

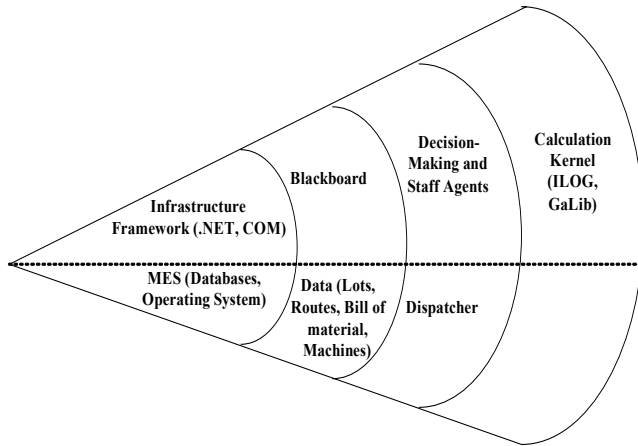
be interpreted within the kernel-shell-model [12], [14]. The kernel-shell-model differentiates between general and enterprise specific IT functionality clusters. We use Figure 3 to show how we can use this model in the situation described in this paper.

Furthermore, we want to demonstrate by Figure 3 that the suggested agent-based framework can be used to extend the scheduling functionality of manufacturing execution systems (MES). We use the dotted line in Figure 3 to divide the kernel-shell-model into two parts. The lower part models the MES, whereas the upper part is used to describe the agent-based framework.

The inner ring forms the technical kernel. The MES provides access to relational data bases and to the operating system. Infrastructure frameworks like the .NET remoting framework or COM-technology are used to implement communication among different agents. The inner shell is used for data storage. We use appropriate data structures contained in a process data model to model lots, machines, and routes together with their states. On the other hand, the agent-based framework has a blackboard-type data layer that contains similar objects as the process data model. Note that this shell is still rather generic. The next shell is given by dispatching functionality provided by the MES. Scheduling functionality is provided by the decision-making and staff agents of the agent-based framework. This shell is highly customized to a specific enterprise. Scheduling algorithms that are encapsulated in staff agents provided the outer shell. These algorithms are typically given by calculation kernels like ILOG or GaLib. We can see from Figure 3 that the suggested agent-based framework is based on infrastructure frameworks (mainly for communication purposes) and calculation kernels.

**Table 1:** Fulfillment of the Requirements

<b>Requirement</b>	<b>Feature of the Agent-based Framework</b>
Extendibility of the software at runtime	New agents can be released at runtime.
Scheduling system is carrier of distributed scheduling approaches.	Staff agents can implement scheduling functionality in a distributed manner.
Separation of system structure and algorithms	Decision-making agents are supported by staff agents.
Ex-ante performance assessment	Blackboard-type data layer between distributed scheduling system and simulated base system and base process
Representation of distributed hierarchies	Hierarchy identifier
Scheduling systems should be carrier of efficient scheduling approaches.	Heuristics based on local search, constraint satisfaction techniques
Any-time property of the approaches	Heuristics that work iteratively
Situation-dependent parameterization of scheduling approaches	Algorithms of machine learning may be encapsulated within staff agents.
Ability to exchange scheduling approaches easily	Staff agents can be removed and released at run time.



**Figure 3:** Interpretation of the Agent-based Framework within the Kernel-Shell-Model

## 5 Conclusions

In this paper, we present ideas how to construct distributed scheduling systems. We show that software agents are appropriate for building distributed scheduling systems. Agent-based frameworks offer a compromise between the rigid packaged software solutions and the highly flexible but too costly individual software solutions. We show that agent-based frameworks have to build on top of infrastructure frameworks and frameworks with highly specialized computing capabilities. We show that the kernel-shell-model can be used for constructing distributed scheduling systems. In future research, we have to address the problem of the development of appropriate process data models. These models could be based on characteristics [Di05] in order to reduce the amount of data for representing products, routes, and bills of material. However, the algorithms also have to be changed in this case.

Furthermore, the problem of designing adaptive staff agents has to be considered in more detail. First experiences with machine learning techniques are quite promising. However, the generation of training data based on simulation experiments is very time-consuming. It is not clear how a learning of the agents has to be organized in a dynamic manner.

## 6 References

- [1] Brooks Real-Time Dispatcher, [www.brooks.com/-documents.cfm?documentID=3455](http://www.brooks.com/-documents.cfm?documentID=3455), 2006.
- [2] M. Caridi and S. Cavalieri, "Multi-Agent Systems in Production Planning and Control: An Overview," *Production Planning & Control*, Vol. 15, No. 2, pp. 106-118, 2004.
- [3] J.Th. Dickersbach, *Characteristic Based Planning with mySAP SCM: Scenarios, Processes, and Functions*, Springer, Heidelberg, 2005.
- [4] J. Dorn, J., "Planung von betrieblichen Abläufen durch Standardsoftware – ein Widerspruch?", *WIRTSCHAFTSINFORMATIK*, Vol. 42, pp. 201-209.
- [5] M.E. Fayad, D.C. Schmidt, and R.E. Johnson, *Building Application Frameworks: Object-Oriented Foundation of Framework Design*, Wiley, New York, 1999.
- [6] A. Fink and S. Voß, "HotFrame: A Heuristic Optimization Framework," In S. Voß, D.L. Woodruff (Eds.), *Optimization Software Class Libraries*, Kluwer, Boston, pp. 81-154, 2002.
- [7] Foundation for Intelligent Physical Agents (FIPA), <http://www.fipa.org>, 2006.
- [8] J.W. Fowler and M. Pfund, *State of the Art Scheduling Survey Results*, Arizona State University, Modeling and Analysis of Semiconductor Manufacturing Laboratory, 2002.
- [9] S. Friedrich, "Stand der IT-Unterstützung für das Supply Chain Management in der Prozessindustrie," In T. Sprengler, T., S. Voss, H. Kopfer (Eds.), *Logistik Management: Prozesse, Systeme, Ausbildung*, Physica, Heidelberg, pp. 145-159, 2004.
- [10] G. Knolmayer, P. Mertens, and A. Zeier, *Supply Chain Management Based on SAP Systems: Order Management in Manufacturing Companies*, Springer, Berlin, 2002.
- [11] C. Le Pape, "Implementation of Resource Constraints in ILOG SCHEDULE: A Library for the Development of Constraint-Based Scheduling Systems," *Intelligent Systems Engineering*, Vol. 3, No. 2, pp. 55-66, 1994.
- [12] M. Lohmann, B. Schmitzer, and P. Mertens, "Kern-Schalen-Modell mit Fokus auf E-Commerce," *Proc. 3. Workshop "Komponentenorientierte betriebliche Anwendungssysteme"*, Frankfurt am Main, pp. 23-38, 2001.
- [13] T. McLaren, M.M. Head, and Y. Yuan, "Supply Chain Management Information Systems Capabilities. An Exploratory Study of Electronics Manufacturers," *Information Systems and e-Business Management*, Vol. 2, pp. 207-222, 2004.
- [14] P. Mertens and M. Lohmann, "Branche oder Betriebstyp als Klassifikationskriterien für die Standardsoftware der Zukunft? Erste Überlegungen, wie künftig betriebswirtschaftliche Standardsoftware entstehen

könnte, " *Proc. Verbundtagung Wirtschaftsinformatik*, Aachen, pp. 110-135, 2000.

[15] L. Mönch and M. Stehli, "ManufAg: a Multi-Agent-System Framework for Production Control of Complex Manufacturing Systems," *Journal of Information Systems and e-Business Management*, Vol. 4, No. 2, pp. 159-185, 2006.

[16] L. Mönch and M. Stehli, J. Zimmermann, and I. Habenicht, "The FABMAS Multi-Agent-System Prototype for Production Control of Waferfabs: Design, Implementation, and Performance Assessment," To appear in *International Journal of Production Planning and Control*.

[17] M. Pinedo and B.P.-C. Yen, "On the Design and Development of Object-Oriented Scheduling Systems," *Annals of Operations Research*, Vol. 70, pp. 359-378, 1997.

[18] C. Schneeweiss, *Distributed Decision Making*, Springer, New York, Heidelberg, Berlin, 2003.

[19] H. Van Brussel, J. Wyns, P. Valckenaers, L. Bongaerts, and P. Peeters, "Reference Architecture for Holonic Manufacturing Systems: PROSA," *Computers in Industry*, Special Issue on Intelligent Manufacturing Systems, Vol. 37, No. 3, pp. 225-276, 1998.

[20] M. Wall, "Galib: A C++ Library of Genetic Algorithms Components, " <http://lancet.mit.edu/ga/>, 2006.

[21] M. Wooldridge, *An Introduction to Multiagent Systems*, Wiley, Chichester, 2002.