

# **Knowledge-Based Design of Scheduling Systems**

Jürgen Sauer

Universität Oldenburg, FB Informatik

Escherweg 2

D-26121 Oldenburg, Germany

sauer@informatik.uni-oldenburg.de

## **Abstract**

A Scheduling system shall support the human scheduler in his daily work in creating and revising schedules of the production. Several criteria influence the requirements for such a system, e.g., the area of production and the preferences of the producing company as well as the need for presentation of specific information and sophisticated scheduling algorithms. Most of the existing scheduling systems have been built from scratch. To improve the building of scheduling systems an intelligent support system for the implementation of hybrid scheduling systems is presented which integrates different scheduling techniques, knowledge about their appropriateness for specific problem scenarios and knowledge about the design of scheduling systems. Using this knowledge it helps the user designing adequate scheduling systems for different application scenarios.

## **1. Introduction**

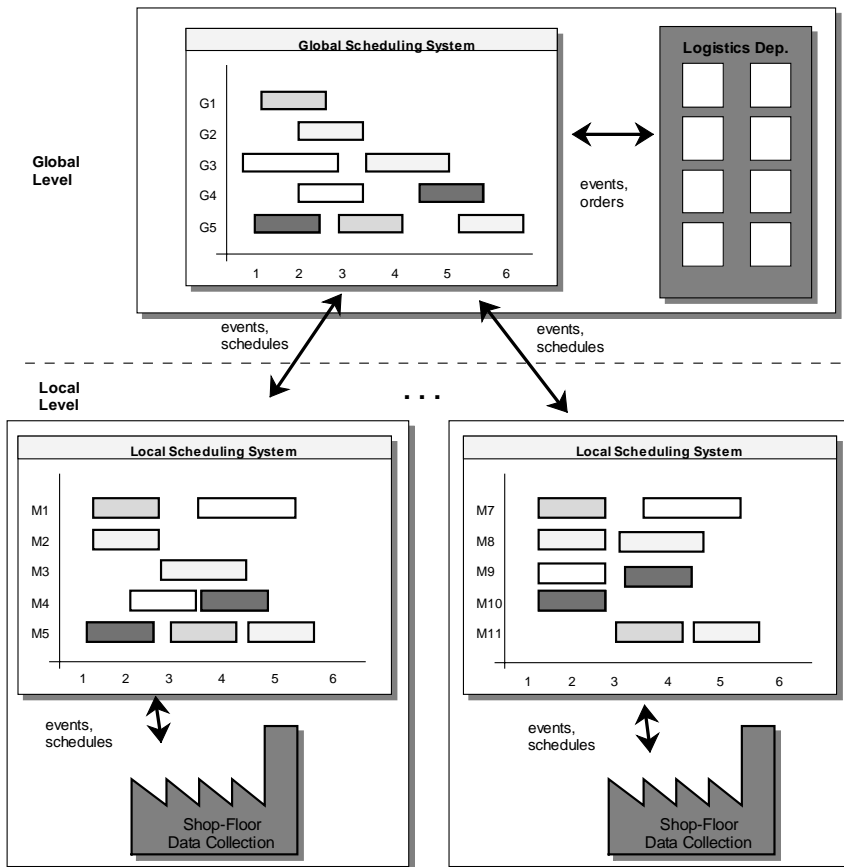
The scheduling of production processes of a manufacturing enterprise is one of the significant tasks to be performed to achieve competitive production, which means e.g., to deliver products in time or to use resources efficiently. In recent years, an increasing interest in the use of artificial intelligence technologies in the area of production scheduling could be observed [1]. The task of scheduling is the temporal assignment of orders (e.g., for manufacturing products) to resources (e.g., machines) where a number of goals and conditions have to be regarded. The goals are mostly economically oriented trying

to reduce costs or to increase productivity. The conditions can be separated in hard constraints which have to be fulfilled (e.g., using the correct routings and only the possible machines), and soft constraints which should be fulfilled (e.g., meeting the due dates or using preferred machines). If alternative routings and machines are available then the complexity of the problem increases.

Scheduling covers the creation of a schedule of production processes over a longer period (predictive scheduling) and the adaptation of an existing schedule due to actual events in the scheduling environment (reactive scheduling). Additionally, scheduling has an important interactive dimension as we will always find humans in the application areas who have to decide and control within the scheduling process.

In different application areas scheduling is performed on several hierarchical levels. These levels normally correspond to the organizational hierarchy within the enterprise. On the material requirements planning (MRP) level most of the input data for scheduling tasks is created (e.g., amounts of products, intermediates, and required dates of finishing or delivery).

Especially in a multi-site production environment (see figure 1) these input data are used on a global scheduling level where the products are distributed to different production sites according to global goals like meeting due dates, minimizing work in progress, or minimizing inventory [2]. Additionally, new orders and other events (e.g., new shifts) originated from the logistics department as well as the events from the local levels (e.g., due date violations) have to be regarded. The task of the global scheduling system is to provide the frame of the schedule for the local scheduling systems and to react to the events of the global scheduling environment. On the local level, the global schedule is used to create detailed schedules for the production of intermediate products. All the goals and constraints of the local area have to be regarded, e.g., meeting due dates, maximizing machine utilization or minimizing work in progress. Again, like on the global level, the local scheduling systems have to react to the events of the local scheduling environment (e.g., machine breakdowns).



**Figure 1: A multi-site scheduling scenario.**

The information exchange between the scheduling levels includes the global schedule as well as global events that have to be regarded on the local level and vice versa. On the local level an additional communication with the shop floor is necessary providing the data of the shop floor as input for reactive scheduling.

Due to the difficulty of the problem domain, the objective in a real-world scheduling environment is the determination of a "good" and feasible solution, not an optimal one. Very important for performing this task is the (heuristic) knowledge of the human domain experts who are able to solve distinct scheduling problems and to judge the feasibility of schedules by virtue of their gained experience.

## 2. Knowledge Based Scheduling Systems

Especially for the local scheduling level several knowledge-based systems have been presented, e.g., ISIS, OPIS, SONIA [1; 3-5], all using the experience of human experts and problem-specific knowledge of the application domain. Reactive scheduling is a newer area of interest [6], and scheduling on the global level (also called multi-site scheduling) [7] is a key research area today.

In spite of a large number of developed scheduling methods, only a few practical applications have entered into everyday use in industrial reality. Our experience obtained in projects addressing real-world scheduling problems showed that not only the applied scheduling methods but also several other features play a significant role for the acceptance of scheduling systems [4; 8]. Hence, a scheduling system should meet the following requirements:

- *information presentation*

The information necessary for the scheduling task has to be presented in an appropriate manner, showing specific information at a glance (e.g., capacities or alternative process plans). Moreover, it should be possible to monitor all scheduling actions in order to see the immediate consequences of specific decisions and to maintain consistency.

- *interaction*

Interaction shall allow for full manual control of the scheduling process. All decisions may be made by the user (e.g., selecting orders, operations, or machines), but a scheduling system should support the interactive as well as the automatic part of the problem solving activities (also called mixed-initiative scheduling [9]).

- *incorporation of scheduling expertise*

Main feature of a knowledge-based scheduling system is the identification and application of problem-specific knowledge for the solution of the addressed problem. Although most previous

knowledge-based approaches focused merely on predictive scheduling, reactive scheduling is the more significant problem in many applications and has to be supported algorithmically as well.

- *integration in the organizational environment*

Scheduling systems cannot be designed as stand-alone components since they have to communicate, interact, access the same data, and share information with their organizational environment. Therefore, knowledge-based systems have to be an integrated part of an existing information system, providing well defined interfaces to standard application systems such as database systems and computer networks.

- *participation of the user*

The incorporation of the user in all phases of the system design process is extremely important for the final acceptance of the scheduling system. All the other software engineering principles should be regarded as well.

- *communication between scheduling levels*

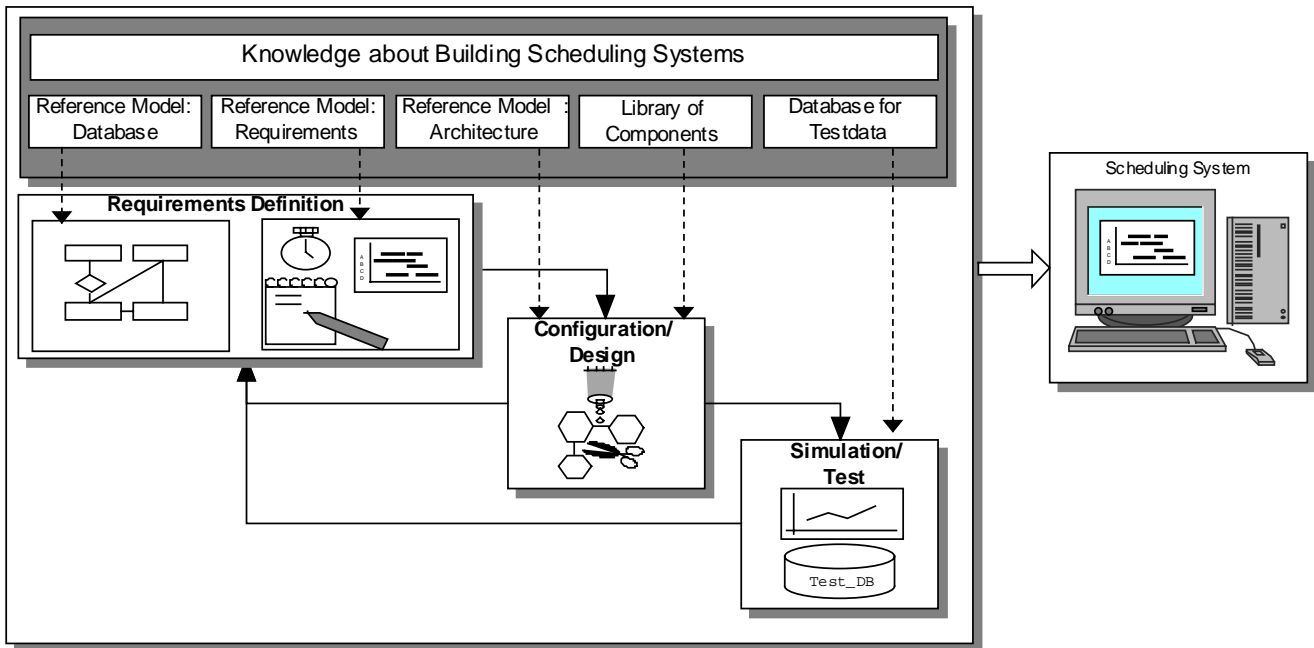
This is especially important in the multi-site scheduling area. Communication is the crucial task in order to maintain consistency, coordinate the activities between the different scheduling systems on the two scheduling levels, and to inform the participating systems of events which might affect them.

### **3. Building Knowledge-based Scheduling Systems**

Most of the existing knowledge-based scheduling systems have been built from scratch, often using the same or similar components within the different systems. All of them include sophisticated scheduling strategies, but the systems are not as modular and the components are not as reusable as they should be.

Here is the starting point of our project SSWB (scheduling systems workbench). The idea of SSWB is to give the designer of a new scheduling system as much advice and knowledge on this area as necessary to efficiently build hybrid knowledge-based scheduling systems. This includes knowledge

about the development process as well as different levels of scheduling knowledge. Figure 2 gives a sketch of the concept of SSWB.



**Figure 2: The concept of the scheduling systems workbench.**

The development process is quite similar to other software development projects. The process model is based on the iterative waterfall model and mainly consists of the phases requirements definition, design/ configuration and simulation/ test. It allows user participation in all of the phases of the development process.

Especially important within the process of building scheduling systems is knowledge about the design of user interfaces and knowledge about the main components and interfaces of scheduling systems.

Three levels of scheduling knowledge are involved in our approach. On the first level there is basic scheduling knowledge from Operations Research (OR) and Artificial Intelligence (AI) which is normally coded in scheduling algorithms. These algorithms reach from simple priority rule-based ones to sophisticated ones using problem solving techniques from OR and/ or AI. A wide variety of

scheduling strategies and algorithms has been developed and can be used [5; 10]. The scheduling knowledge is one vital part of the library of components used in the design system. On the second level there is knowledge about the appropriateness of scheduling algorithms based on specific problem situations, e.g., which algorithm is to be used if meeting due dates is the main goal of scheduling [11]. On the third level there is the general knowledge about building scheduling systems, e.g., which components are important. The knowledge on level two and three is called meta-knowledge and is integrated in the "knowledge about building scheduling systems" and the "reference model: xy" components. For the database, the requirements and the general architecture of a scheduling system so called reference models have been developed, from which the actual system description can be derived. The reference models have been designed using the experience from previous projects (e.g., [4]) and reports from literature (e.g., [12]).

Within the development process the workbench provides support for the modeling of the investigated application problem, the definition of the desired system requirements, the assembly of the scheduling system by selection of predefined components and algorithms, and the simulation of alternative algorithms or system configurations. Thus the three phases "requirements definition", "configuration/design", and "simulation/ test" are supported by SSWB.

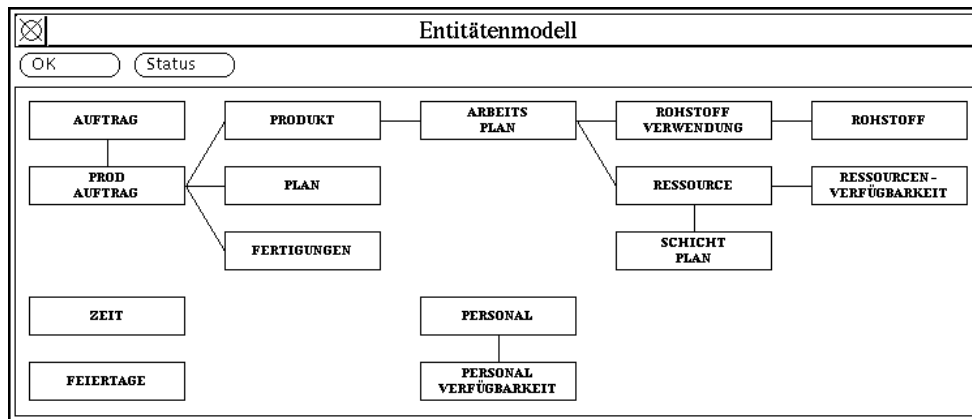
### *First Step: Requirements Definition*

The process starts with the requirements definition where several reference models are used to guide the modeling of the problem. The reference model for the database describes the objects of the scheduling area (e.g., products, orders, resources). Figure 3 shows the top level of the entity relationship scheme of the database. The reference model for system requirements is used to collect data on

- the different scheduling levels and goals/ evaluation criteria,
- the structure of the scheduling problem (e.g., discrete manufacturing, continuous processes),

- the time model (e.g., shifts, calendar),
- functional requirements (e.g., alternative machines, variants, priorities, transport),
- hard (e.g., technical restrictions) and soft constraints (e.g., preferable machines),
- user interface needs (e.g., windows for Gantt-chart, orders, routings),
- events for reactive scheduling and communication needs (global - local, local - shop floor).

Consistency checks are performed within and at the end of the phases of development.

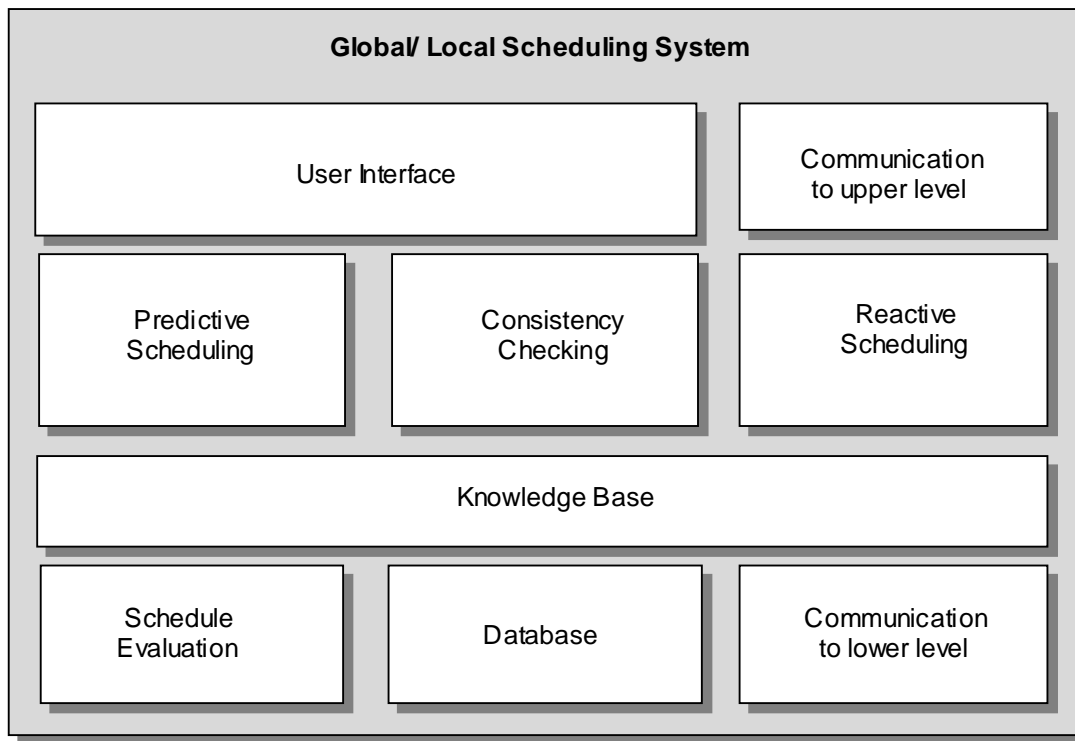


**Figure 3: The reference model of the database.**

### *Second Step: Configuration/ Design*

On the basis of the requirements defined a scheduling system is built in the configuration phase using components from a library of system components. A reference model for a scheduling system (see fig. 4) has been derived from several existing systems and consists of

- a user interface responsible for the tasks of information presentation and interactive scheduling,
- a scheduling component containing scheduling knowledge both for predictive and reactive scheduling tasks,
- a database component which implements an interface (e.g., to an existing database system), and
- interfaces to other scheduling levels responsible for communication with the systems on those levels.



**Figure 4: The reference model for a scheduling system.**

The component library contains predefined algorithmic solutions for the components of the scheduling system (e.g., windows for the user interface, several different scheduling algorithms for the different scheduling levels, an interface to the shop floor and to a database system as well as facilities for communication and coordination between the components and levels). The components are adopted from OR-/ AI-scheduling algorithms and from scheduling systems in practice. The afore mentioned meta-knowledge is used to select the right components of the scheduling system to be built. The collected information of the requirements definition is compared with meta-information about the appropriateness of specific components (e.g., scheduling algorithms for the specific application scenario). The best suited components are determined using several evaluation criteria and proposed for use in the system. The user can select the components from the proposal to build up a specific knowledge-based scheduling system.



interface to a database system (ProDBI to Oracle). The implemented system supports the three steps of development in the following way:

### *First Step: Requirements Definition*

Main task in this phase is the collection of data about the specific properties of the scheduling system and its environment. Several windows are used to collect data as mentioned in section 3. Figure 5 shows a part of the user interface used in the requirements definition phase, consisting of windows for adjusting the database reference model to the actual problem (Entitätenmodell, Status) and windows for collecting required data (e.g., goals) of the problem area (Bewertungskriterien, Zeitkriterien).

Table 1 gives an overview of some of the criteria and the possible values that are collected. The #-row shows how many values are allowed (1: exactly one, m: one or more), the bold written value is the default value. The criteria represent facts about the scheduling environment (criteria 1 to 14), the database (criteria 15 and 16) and the requirements of the algorithmic solution (criteria 17 to 23). Prolog facts are used to represent these facts.

### *Second Step: Configuration/ Design*

The design task is performed in two steps. First the architecture of the system is selected by defining the components it shall consist of. Default is the reference model of a scheduling system as presented in figure 4. In the prototype the design of systems is limited to local scheduling systems with simple user interfaces. In the second step the components of the system are specified. Up to now the user interface is quite fixed, but the other components are designed using the information collected previously. This is especially the case for the selection of the scheduling algorithms to be used. Therefore all scheduling algorithms of the component library are provided with their fulfillment of the criteria information. These are used to find the best suited algorithms for the scheduling system to be designed. Some of the

criteria are used as knock-out criteria (KO and KO IF in row Type of table 1) and others are used to calculate a fitness value of the remaining algorithms (functions Sel, Fun1, and Prio in Row Type of table 1). Result is a ranking of the algorithms which is presented in a window. Here the user can choose algorithms for the test phase.

No.	Criteria	Values	#	Type
1	Planungsart	<b>lokal</b> , reaktiv, global, verteilt	m	KO
2	Fertigungsstruktur	<b>Fließfertigung</b> , Werkstattfertigung, Gruppenfertigung	1	KO
3	Ereignisse	Neuer Auftrag, Auftrag löschen, Neue Startzeit/ Endzeit für Auftrag, Mengenänderung, Neue Priorität für Auftrag, Neues Produkt/ Variante in Auftrag, ...	m	Sel
4	Fertigungsart	Einmalfertigung, Einzelfertigung, Kleinserienfertigung, Serienfertigung, Großserienfertigung, <b>Massenfertigung</b>	1	Fun1
5	Produktionsart	Flow shop, Job shop, <b>Egal</b>	1	KO
6	Transport	Ja, <b>Nein</b>	1	KO IF
7	alternative Maschinen	<b>Ja</b> , Nein	1	KO IF
8	alternative Varianten	<b>Ja</b> , Nein	1	KO IF
9	Auftragsprioritäten	Ja, <b>Nein</b>	1	KO IF
10	Grobplan	Ja, <b>Nein</b>		
11	Laufzeit	<b>sehr niedrig</b> , niedrig, durchschnittlich, hoch, sehr hoch	1	Fun1
12	Produktionsvolumen	sehr niedrig, niedrig, durchschnittlich, <b>hoch</b> , sehr hoch	1	Fun1
13	Pufferzeiten	Arbeitsschritte, Produkt, Maschinen, <b>Keine</b>	m	KO
14	BDE	Ja, Nein	1	KO IF
15	Datenbank	Ja, Nein	1	KO
16	Entitäten	AUFTRAG, PRODAUFTRAG, PRODUKT, PLAN, FERTIGUNGEN, ARBEITSPLAN, ZEIT, FEIERTAGE, ROHSTOFFVERWENDUNG, RESSOURCE, ...	m	KO
17	Planungstyp	Anzahl der einzuplanenden Operationen, wertorientiert, rüstzeitorientiert, <b>Egal</b>	m	KO
18	Planungsstrategien	auftragsorientiert, ressourcenorientiert, operationenorientiert, <b>Egal</b>	m	KO
19	Einplanungsregeln	spt, lpt, edd, fifo, best fit, <b>Egal</b>	m	KO
20	Reihenfolge beim Einplanen	vorwärts, rückwärts, <b>Egal</b>	m	KO
21	optimale Lösung	Ja, <b>Nein</b>	1	KO IF
22	Kapazitätsanpassung	Überstunden, Zusatzschichten, Kurzarbeit, Personal einstellen, Auswärtsfertigung, überlappend fertigen, Rüstzeiten verringern, Übergangszeiten verringern, ...	m	
23	Bewertungsfunktion	Termineinhaltung, kurze Durchlaufzeiten, Rüstkostenminimierung, Reinigungszeitminimierung, Lagergrößentoptimierung, minimale Transportkosten, minimale Kapitalbildung, ...	m	Prio

**Table 1: Criteria of scheduling system and environment.**

### *Third Step: Simulation/ Test*

In this step a test database is created using the data collected previously. With the data of this database it is possible to check the algorithms selected regarding, e.g., runtime or results of the evaluation functions. Therefore the results of several of the evaluation functions are shown and a simulation of events occurring over time is possible. This finally leads to the selection of the "right" scheduling components for the proposed scheduling system.

## **5. Conclusion**

An approach and its implementation have been presented, that support the creation of knowledge-based scheduling systems. It combines knowledge about the design of scheduling systems with the necessary components to build a system that fits the users needs. The development process is adopted from software engineering and includes the modeling of the application area, the collection of the requirements as well as the preferences of the later users. Meta knowledge is incorporated to select the appropriate system architecture together with all the components within the scheduling system. The prototypical implementation showed the feasibility of the concept but is rather restricted. In an ongoing project the system is redesigned and extended using object-oriented technology.

Under investigation are the configuration of user interfaces, the integration of new and sophisticated scheduling approaches, using methods like fuzzy-logic, neural networks, genetic algorithms, multi-agents, the support for designing multi-site scheduling systems, and the communication facilities of the reference system.

## **References**

1. Smith, S. F. Knowledge-based production management: approaches, results and prospects. *Production Planning & Control*, 3(4). 1992.

2. Bruns, R., and Sauer, J. *Knowledge-Based Multi-Site Coordination and Scheduling*. In: R. D. Schraft, ed., *Flexible Automation and Intelligent Manufacturing 1995*. Begell House, Stuttgart. 1995, pp. 115-123.
3. Dorn, J., and Froeschl, K. A. *Scheduling of Production Processes*. Ellis Horwood. 1993.
4. Sauer, J., and Bruns, R. Knowledge-Based Scheduling Systems in Industry and Medicine. *IEEE-Expert*(February). 1997.
5. Zweben, M., and Fox, M. S. *Intelligent Scheduling*. Morgan Kaufman. 1994.
6. Kerr, R. M., and Szelke, E. *Artificial Intelligence in Reactive Scheduling*. Chapman & Hall. 1995.
7. Sauer, J. A Multi-Site Scheduling System. *AAAI's Special Interest Group in Manufacturing Workshop on Artificial Intelligence and Manufacturing: State of the Art and State of Practice*, Albuquerque. AAAI Press, Menlo Park. 1998.
8. Kempf, K. G., Le Pape, C., Smith, S. F., and Fox, B. R. Issues in the Design of AI-Based Schedulers: A Workshop Report. *AI Magazine, Special Issue*. 1991.
9. Hsu, W. L., Prietula, M., Thompson, G., and Ow, P. S. A mixed-initiative scheduling workbench: Integrating AI, OR, and HCI. *Journal of Decision Support Systems*, 9(3). 1993, pp. 245-247.
10. Sauer, J. Knowledge-Based Scheduling Techniques in Industry. In: L. C. Jain, R. P. Johnson, Y. Takefuji, and L. A. Zadeh, eds., *Knowledge-Based Intelligent Techniques in Industry* . CRC Press. 1999, pp. 53 - 84.
11. Sauer, J. Scheduling and Meta-Scheduling. In: C. Beierle and L. Plümer, eds., *Logic Programming: Formal Methods and Practical Applications* . Elsevier Science. 1995.
12. Kirn, S., and Schneider, J. STRICT: Selecting the Right Architecture. In: F. Belli and F. J. Radermacher, eds., *Industrial and Engineering Applications of AI and Expert Systems - Proc. of 5th IEA/AIE- 92* . Springer. 1992.