



Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Document
D-95-01

PuK - 95

Beiträge zum 9. Workshop "Planen und Konfigurieren"

**Susanne Biundo, Wolfgang Tank
(Hrsg.)**

Februar 1995

**Deutsches Forschungszentrum für Künstliche Intelligenz
GmbH**

Postfach 20 80
D-67608 Kaiserslautern, FRG
Tel.: (+49 631) 205-3211/13
Fax: (+49 631) 205-3210

Stuhlsatzenhausweg 3
D-66123 Saarbrücken, FRG
Tel.: (+49 681) 302-5252
Fax: (+49 681) 302-5341

Deutsches Forschungszentrum für Künstliche Intelligenz

The German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) with sites in Kaiserslautern and Saarbrücken is a non-profit organization which was founded in 1988. The shareholder companies are Atlas Elektronik, Daimler-Benz, Fraunhofer Gesellschaft, GMD, IBM, Insiders, Mannesmann-Kienzle, SEMA Group, and Siemens. Research projects conducted at the DFKI are funded by the German Ministry for Research and Technology, by the shareholder companies, or by other industrial contracts.

The DFKI conducts application-oriented basic research in the field of artificial intelligence and other related subfields of computer science. The overall goal is to construct *systems with technical knowledge and common sense* which - by using AI methods - implement a problem solution for a selected application area. Currently, there are the following research areas at the DFKI:

- ☐ Intelligent Engineering Systems
- ☐ Intelligent User Interfaces
- ☐ Computer Linguistics
- ☐ Programming Systems
- ☐ Deduction and Multiagent Systems
- ☐ Document Analysis and Office Automation.

The DFKI strives at making its research results available to the scientific community. There exist many contacts to domestic and foreign research institutions, both in academy and industry. The DFKI hosts technology transfer workshops for shareholders and other interested groups in order to inform about the current state of research.

From its beginning, the DFKI has provided an attractive working environment for AI researchers from Germany and from all over the world. The goal is to have a staff of about 100 researchers at the end of the building-up phase.

Dr. Dr. D. Ruland
Director

PuK - 95

Beiträge zum 9. Workshop "Planen und Konfigurieren"

Susanne Biundo, Wolfgang Tank (Hrsg.)

DFKI-D-95-01

© Deutsches Forschungszentrum für Künstliche Intelligenz 1994

This work may not be copied or reproduced in whole or in part for any commercial purpose. Permission to copy in whole or in part without payment of fee is granted for nonprofit educational and research purposes provided that all such whole or partial copies include the following: a notice that such copying is by permission of Deutsches Forschungszentrum für Künstliche Intelligenz, Kaiserslautern, Federal Republic of Germany; an acknowledgement of the authors and individual contributors to the work; all applicable portions of this copyright notice. Copying, reproducing, or republishing for any other purpose shall require a licence with payment of fee to Deutsches Forschungszentrum für Künstliche Intelligenz.

ISSN 0946-0098

PuK – 95

Beiträge zum 9. Workshop “Planen und Konfigurieren”

Susanne Biundo, Wolfgang Tank
(Hrsg.)

Vorwort

Der PuK, das Jahrestreffen der GI-Fachgruppe 1.5.3 *Planen und Konfigurieren*, ist seit nunmehr neun Jahren ein Forum, das Forscher und Anwender auf dem Gebiet des Planens und Konfigurierens zu Diskussion und Erfahrungsaustausch zusammenführt.

In diesem Jahr findet der PuK wieder als zweitägiger Workshop der *Deutschen Expertensystemtagung* (XPS) statt. Er hat zwei Schwerpunkte.

Zum erstenmal findet eine sogenannte *Standortpräsentation* statt. Hier stellen Arbeitsgruppen der *Universität Kaiserslautern* und Forschungsbereiche des *Deutschen Forschungszentrums für Künstliche Intelligenz* (DFKI GmbH) in ausgewählten Beiträgen ihre Arbeiten auf dem Gebiet des Planens vor.

Der zweite Schwerpunkt des Programms liegt im Bereich konkreter industrieller Anwendungen von Planungs- und Konfigurierungssystemen (*PuK in der Praxis*).

Der Hauptbeitrag zum Thema *Planen in der Praxis* von Jürgen Dorn und Reza Shams berichtet über langjährige Erfahrungen im Bereich der Feinplanung von Produktionsprozessen und stellt ein wissensbasiertes Planungssystem vor, das in der Stahlproduktion eingesetzt wird.

Im Bereich *Konfigurieren* nimmt hier das aktuelle Thema *Konfigurieren bei der Angebots- und Auftragsklärung* breiten Raum ein. Als eine erläuternde Einführung in diese Thematik haben wir den gleichnamigen Artikel von Wolfgang Tank dieser Sektion vorangestellt.

Susanne Biundo

Wolfgang Tank

Programmkomitee

Susanne Biundo, DFKI Saarbrücken

Axel Brinkop, FhG IITB Karlsruhe

Hartmut Dörner, Hanseatische AG Berlin

Andreas Günter, LKI Hamburg

Albert Haag, SAP Walldorf

Joachim Hertzberg, GMD St. Augustin

Josef Schneeberger, FORWISS Erlangen

Wolfgang Tank, MAZ Hamburg

Inhaltsverzeichnis

Standortpräsentation

Multiagent Planning in DAI Applications	3
<i>Klaus Fischer, Jörg P. Müller und Markus Pischel</i>	
Adaptives Planen in Intelligenten Assistenzsystemen	13
<i>Dietmar Dengler</i>	
CAPlan – Ein SNLP-basierter Planungsassistent	25
<i>Frank Weberskirch und Jürgen Paulokat</i>	
Controlling a Nonlinear Hierarchical Planner Using Case Replay	37
<i>Héctor Muñoz, Jürgen Paulokat und Stefan Wess</i>	

PuK in der Theorie

Akquisition von Kontrollwissen zur Zielauswahl für Causal Link Planer	49
<i>Reinhard Praeger, Vera Kettnaker und Frank Weberskirch</i>	
Innovation beim Konfigurieren	59
<i>Lothar Hotz und Thomas Vietze</i>	
Räumliche Beziehungen und Constraint-Systeme: Eine kurze Betrachtung über die Komplexität	69
<i>Manfred Kopisch</i>	

PuK in der Praxis

Scheduling High-Grade Steel Making	79
<i>Jürgen Dorn und Reza Shams</i>	
Erfahrungen mit einem CLP-basierten Ansatz zur Produktionsplanung	97
<i>Heinrich Petith und Astrid Schmücker-Schend</i>	
Stücklistenverifikation mit SPOCK	103
<i>Wolfram Gieseke, Peter Küsell und Norbert Lenschow</i>	
Integration von KONWERK in eine komplexe Anwendungsumgebung	113
<i>A. Angelus, Ch. Boley, A. Kathe und H. Reuter</i>	

Konfigurieren bei der Angebots- und Auftragsklärung

Konfigurieren bei der Angebots- und Auftragsklärung	125
<i>Wolfgang Tank</i>	
SINA – Ein praxisorientierter Ansatz zur angebotsgerechten Konfiguration von Investitionsgütern	133
<i>Andreas Kebbel und Mathias Krömker</i>	
Regelbasierte Angebotskalkulation auf Prozeßkostenbasis	143
<i>Joachim Fischer, Klaus Hausschulte, Karsten Jakuschona und Rainer Koch</i>	
Wissensbasierte Angebotsbearbeitung von Materialflußanlagen am Beispiel von Recyclinganlagen	155
<i>Klaus Richter und Olaf Wolter</i>	
Konfigurierung, Auswahl und Auslegung von statisch belasteten, hydrodynamisch geschmierten Gleitlagern in KONWERK	165
<i>Olaf Arnold</i>	

Standortpräsentation

Multiagent Planning in DAI Applications

Klaus Fischer, Jörg P. Müller, Markus Pischel*

DFKI, Stuhlsatzenhausweg 3, D-66123 Saarbrücken

Abstract

This paper describes aspects of multiagent planning in distributed AI (DAI) applications which have been developed in the AKA-MOD project at DFKI in Saarbrücken. Starting from a description of the underlying conceptual agent model INTERRAP two applications are presented: the MARS application models a group of cooperating transportation companies; the FORKS system, a robotics application, simulates an automated loading dock. The paper outlines how the planning problems which occur in these application domains have been solved.

1 INTRODUCTION

Distributed Artificial Intelligence (DAI) [1] explores either how a group of intelligent and autonomous computational systems (agents) coordinate their knowledge, goals, plans, and skills in order to achieve certain goals (multiagent systems (MAS), agent-centered approach), or how solving a certain problem can be distributed among a set of nodes (distributed problem solving, problem-centered approach). The DAI approach has turned out to be useful for dealing with applications for which a centralized design approach is inappropriate or even impossible, either due to an inherent distribution of knowledge and control, or due to the complexity of the system to be described.

In the AKA-MOD project the agent-centered approach was in the main focus. Agents are autonomous decision-makers with local and incomplete knowledge. They have to attain their own local goals, but interact incidentally or deliberately due to the environment they share. Thus, a multiagent system is described by two modeling layers, the local *agent layer* specifying the knowledge, the goals, plans, and actions of the agents on the one hand, and the global *system layer* specifying the behaviour of the system as a whole.

In the sequel we describe a testbed for the design of DAI applications which is one of the main results of the AKA-MOD project. Such a testbed has to provide a model defining the individual agent, its local problem-solving skills and its interaction capabilities (the agent layer), but also tools helping the system designer to keep a global view of the domain (the system layer). As regards the agent layer, we identified a couple of requirements to be imposed on the individual agent, which should be reflected in its underlying design architecture. Firstly, agents are to behave in a *situated* manner, i.e. they have to perceive unexpected events and to react appropriately to them (see e.g. [2]). Secondly, they are to act in a *goal-directed* manner in order to achieve their goals. In AI, this is normally achieved by devising plans for certain goals (see e.g. [10]). Thirdly, they are to solve their tasks *efficiently* and often have to satisfy real-time constraints. This requires access to a set of “hard-wired” procedures [7] with guaranteed execution properties. Fourthly, they are to cope with the presence of other agents. Whereas certain types of *interactions* can

*e-mail:{kuf,jpm,pischel}@dfki.uni-sb.de

often be performed by employing local mechanisms (e.g. obstacle or collision avoidance in a robot scenario, see [9, 11]), others (e.g. collaboration) require the adoption of joint goals, the generation and execution of joint plans, the exchange of relevant information (i.e. about goals and plans) (see e.g. [8]), and thus the explicit representation of models of other agents in terms of beliefs, goals, plans, and intentions [13]. Finally, they are to be *adaptive*, i.e. they must be able to learn in order to improve their performance and to survive even if the environment changes. These requirements have led to the development of the agent architecture INTERRAP, a layered architecture describing the individual agent.

The layer of system modeling is provided by the development environment MAGSY. MAGSY allows the definition of *basic agents* with specific functionalities, which can be used to construct more complex structures leading to a higher-level agent definition (in our case, this higher-level definition is INTERRAP). Moreover, it provides means of constructing, visualizing, evaluating, and debugging DAI scenarios, including the lower layers of communication, on top of which more complex protocols can be defined allowing interaction among heterogeneous agents.

Two applications have been implemented using the MAGSY-INTERRAP framework: the first system is FORKS, an interacting robot application. FORKS describes an automated loading dock, where forklift robots have to load and to unload trucks in a coordinated way, avoiding and resolving conflicts, and exploiting possibilities to collaborate. The second system is MARS, which models an agent society consisting of transportation companies and their trucks that have to carry out transportation orders.

2 THE INTERRAP AGENT ARCHITECTURE

The main idea of INTERRAP is to define an agent by a set of functional layers, linked by a communication-based control structure and a shared hierarchical knowledge base. The basic design elements of the agent are (1) its basic interface facilities, (2) patterns of behaviour, as well as (3) local plans and (4) joint, multi-agent plans.

Patterns of behaviour allow an agent to react flexibly to its environment, and to perform routine tasks without having an explicit symbolic representation of how the task is to be performed. Due to the latter feature, we can characterize patterns of behaviour as abstract actions having procedural character. Planning allows an agent to act in a goal-directed manner. Moreover, in a multi-agent context, planning is necessary to coordinate actions of agents. For instance, agents should be able to devise joint plans ([8, 12]) to cope with special situations, or they should at least be able to exchange partial global plans (see [3]).

Figure 1 shows the components of the INTERRAP agent model and their interplay. It consists of five basic parts: the world interface (WIF), the behaviour-based component (BBC), the plan-based component (PBC), the cooperation component (CC), and the agent knowledge-base. The *world interface* comprises the agent's facilities for perception, action, and communication. The *BBC* implements the basic behaviour of the agent and the execution and decision component. The BBC is closely linked to the WIF, and thus, to the actions and changes in the world. Patterns of behaviour can be activated both by external trigger conditions (bottom-up activation) and by the plan-based component (top-down activation). The *PBC* contains a planning mechanism which is able to devise local single-agent plans. The plans are hierarchical skeletal plans whose nodes may be either new subplans, or executable patterns of behaviour, or primitive actions. Thus, the PBC may activate patterns of behaviour in order to achieve certain goals. Finally, the *CC* contains a mechanism for devising joint plans. It has access to protocols, a joint

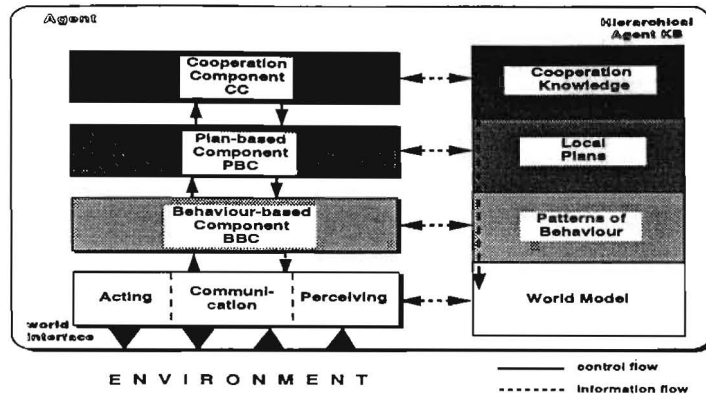


Figure 1: The INTERRAP Agent Model

plan library, and knowledge about communication strategies stored in the cooperation knowledge layer of the KB. CC, PBC, BBC, and the WIF establish the control of the agent. They interact and shift information and control among each other via communication.

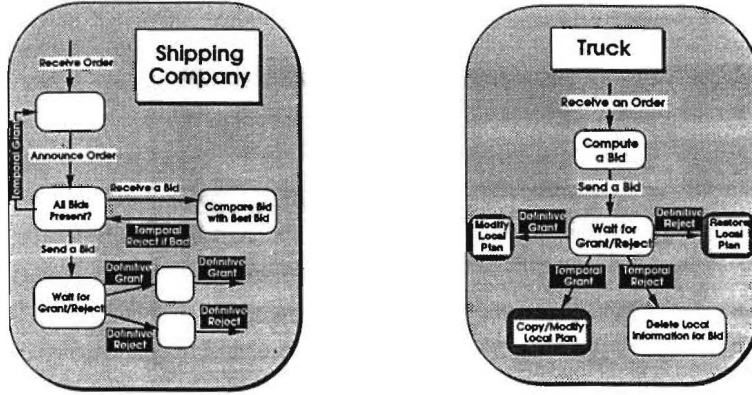
The *knowledge base* is structured hierarchically. It consists of four layers which basically correspond to the structure of the agent control. The lowest layer contains facts representing the world model of the agent. Layer two defines the primitive actions and the patterns of behaviour. Layer three contains local plans. Finally, layer four contains knowledge of and strategies for cooperation, e.g. beliefs about other agents' goals. The basic idea is that information is passed only from lower layers of the knowledge base to higher layers. For example, the plan-based component can access information about the world model, whereas the behaviour-based component does not have access to planning or cooperation information.

3 MARS: COOPERATIVE TRANSPORTATION SCHEDULING

The MARS scenario (Modeling a Multi-Agent Scenario for Shipping Companies) implements a group of shipping companies whose goal it is to deliver a set of dynamically given orders, satisfying a set of given time and/or cost constraints. The complexity of the orders may exceed the capacities of a single company. Therefore, cooperation between companies is required in order to achieve the goal in a satisfactory way.

The common use of shared resources, e.g. train or ship, requires coordination between the companies. Although each company has a local, primarily self-interested view, cooperation between the shipping companies is necessary in order to achieve reasonable global plans. Apart from *system agents*, which perform tasks such as the representation and visualization of the simulation world, the MARS agent society consists of two sorts of *domain agents*, which correspond to the logical entities in the domain: *shipping companies* and *trucks*.

In contrast to other approaches (cf. [4]) we decided to look upon trucks as agents. This view allows to delegate problem-solving skills to them (such as route-planning and local plan optimization). Furthermore, we obtain a logical distribution of the system even if we consider only a single company. Major patterns of cooperation which can be observed in real transportation business are (i) the exchange and retrieval of information among agent; (ii) task allocation between companies and trucks (contracting); (iii) balancing the load of the trucks of different companies by offering or requesting loading space; (iv) negotiating the conditions for orders between companies; (v) having organized the return



(a) (b)

Figure 2: Description of the ECNP from the point of view of a manager (a) and a bidder (b).

freight of a truck by a partner company in the destination region; (vi) establishing the connection between long-term and local traffic by cooperation.

3.1 Vertical Cooperation within a Shipping Company

The contract net protocol (CNP) [14] is a well known DAI problem solving technique which can be used to distribute tasks among a set of agent. However, the pure contract net protocol runs into problems if the tasks exceed the capacity of a single truck. In this case, the manager of the task, i.e. the shipping company agent, has to solve a knapsack problem, which for itself is in general NP hard. To overcome this problem, we decentralized task decomposition by developing and implementing an extension of the CNP, which is called the ECNP protocol. In ECNP, the two speech acts *grant* and *reject* are replaced by four new speech acts: *temporal grant*, *temporal reject*, *definitive grant*, and *definitive reject*. The ECNP is a natural, straightforward solution of the task decomposition problem.

We describe communication and cooperation between two agents by specifying patterns of interaction [11]. These patterns consist of a *protocol layer* and a *decision layer*. A flow chart representation is used to define the protocol layer of a pattern of interaction from the point of view of a single agent. Figure 2 shows the flow charts of the protocol layer of the ECNP protocol, (a) from the manager's point of view and (b) from the point of view of the bidders. The main difference to the CNP is that now the bidders, i.e. the trucks, are allowed to bid for only parts of an order.

Communication in the ECNP proceeds as follows. The manager (shipping company) announces an order to its trucks. It then receives bids for the order and selects the best one using an ordering function. The truck which gave the best bid is sent a temporal grant. All other trucks get temporal rejects. If the best bid does not cover the whole amount of an order, the shipping company reannounces the remaining part of the order. This procedure is repeated until the shipping company receives a bid that covers the whole amount of the order which was finally reannounced. At this stage, the shipping company has gathered a set of bids that covers the whole original order. This set is used to compute a bid for the whole task, that is returned to the broker agent. The latter grants the best of all bids received by the different companies. Based on this answer, the shipping company passes a definitive grant (or definitive reject, respectively) to all trucks which got temporal grants before. One can show that all but the last bid selected are

locally optimal choices for the shipping company [5]. This provides a valuable hint for further improvements of the ECNP.

On the other hand, the trucks must be able to cope with the temporal and definitive grant or reject messages. When a truck receives a temporal grant for the first time, it has to store a copy of its local situation, i.e. the currently valid plan, because it must be able to restore this situation once it gets a definitive reject. All subsequent temporal grants and temporal rejects are handled like the grants and rejects in the pure contract net protocol. If a truck is sent a definitive grant for an order, it removes the copy created above. If a truck gets a definitive reject, it has to remove all the local information gathered while receiving temporal grants and restore the situation before the first temporal grant.

Evaluating the problem solving performance of our system using benchmark test, we were able to prove the high quality of the solutions found by MARS using ECNP were [6]. However, we could even improve these already good results by adopting a bidding procedure called simulated trading, which perfectly completes our approach by providing the possibility to incrementally improve a valid solution, e.g. the one found using ECNP.

3.2 Horizontal Cooperation among Shipping Companies

To further optimise their local solutions shipping companies have to cooperate with each other. One possibility to cooperate can be based on the notion of *unbooked-leg cooperation*. We model this kind of cooperation as follows: trucks check their local schedule for unbooked legs (i.e. legs where they have a certain amount of free capacity left). They announce the free capacity to their company. The company offers this to a set of suitable other companies. Here, a partner model is used in order to achieve a *focus of address*. On the other hand, when a company receives a new order, it looks for unbooked-leg offers by other companies. If it finds a suitable offer, it applies for it. Since several companies may apply for one offer, the best application will win. However, this is more than a contract net, since companies are free to apply or not to apply, and since the frame conditions of the deal are negotiated (e.g. the price a company has to pay for taking the unbooked leg). Figure 3 shows a set of communication primitives for an unbooked-leg cooperation protocol.

Truck → Company:	
ANNOUNCE:	Announce an unbooked leg.
REVOKE:	Revoke a previously announced unbooked leg.
FINISH:	Indicates the end of the validity of an announcement, and triggers the confirmation of the leg to a suitable applicant.
Company → Truck:	
INSERT:	Tells the truck to update its plan when a suitable applicant has been found.
FREE:	Frees the truck from its obligation as regards a previously announced unbooked leg.
Company → Company:	
OFFER:	Offer free capacity to another company.
APPLY:	Apply for an offered unbooked leg.
GRANT:	Grant an application for an unbooked leg.
REVOKE:	Revoke an offer or revoke an application for an unbooked leg.

Figure 3: Protocol Primitives for Unbooked-Leg Cooperation

4 FORKS: INTERACTING ROBOTS IN THE LOADING DOCK

The FORKS application deals with the simulation of an automated loading dock. In the loading dock, there are shelves which may contain different types of goods. The agents in the scenario are automated forklifts which have to load and unload incoming trucks. We use a grid-based representation of the loading-dock. The primitive actions of the forklifts are moving from one place to another, turning around, getting and storing goods, and communicating with other agents. Each agent has a certain range of perception (for example 3×2 squares in front of it). This representation of the world simplifies many of the problems of mechanical control and geometrical trajectory planning (cf. [9]). On the other hand, if we use a grid size of about two meters, the representation is realistic enough to model robots for Flexible Transport Systems (FTS).

In the following, we will demonstrate how the autonomous forklifts in the FORKS application can be designed as INTERRAP agents.

4.1 The World Interface

The *world interface* comprises the agent's facilities for perception, action, and communication. The *actions* component controls the effectoric capabilities of the agent. Obviously, these capabilities are very domain-dependent. In the case of the forklift robots, these are *walk_ahead*, *turn_left*, *turn_right*, *put_box*, *grasp_box*. The *communication* unit contains the agent's communicative facilities. It controls the physical realization of sending and receiving messages. Outgoing messages have to be transformed from the agent's internal language into a knowledge interchange format (often called *interlingua*) which is understood by all agents. Corresponding to this, incoming messages must be transformed into the local agent language. This transformation is done by the *translator module*. The *perception* part of the WIF controls the vision and sensing facilities of the agent. Again the concrete implementation of this module depends strongly on the types of agents we want to model. In the case of a real robot environment, the perception part may include the transformation and processing of the data obtained by a camera. Sensor information can be received for example by infra-red, laser, or ultrasonic sensors. In our simulation system, perception is implemented as follows: each agent has a configurable range of perception ($n \times m$ squares). Since a change within this range of perception can be always associated with a certain square, each time something within the perceived fields of an agent changes, the simulation world sends the agent a symbolic description of what has changed within this square.

4.2 Integrating Reaction and Deliberation

Having defined the interface to the physical world, we now define how the information that is perceived by the agent is processed, how decisions are made based on this information, finally leading to the execution of actions. In the INTERRAP framework, the local behaviour of an agent is basically determined by two components and the way they interact. The behaviour-based and the plan-based component. In the following, we will describe both components by the example of the FORKS application.

4.2.1 The Behaviour-based Component

The *BBC* implements the basic reactive behaviour and the procedural knowledge of the agent. It has access to a set of executable patterns of behaviour (PoB). The BBC is closely linked to the WIF, and thus, to the actions and changes in the world. Patterns

of behaviour can be activated both by external trigger conditions and by the plan-based component. What is most important is that these patterns directly correspond to goals of the agent which are activated when a pattern is triggered. The BBC consists of a control component *bbc-control* and of a set of patterns of behaviour together with a priority mechanism which are represented and maintained in the *pattern maintenance unit*. The *bbc-control* serves as an interface to the neighbour modules and as an interpreter for the execution of the patterns of behaviour. It is responsible for processing messages from other modules. Moreover, in each loop of the BBC cycle, it has to monitor the execution of the currently active pattern of behaviour determined in the pattern maintenance unit. The pattern maintenance unit itself describes a mechanism for determining the situated priority of the patterns (see [11] for a detailed description of pattern priority). A pattern agenda is maintained and recomputed in each BBC cycle; the pattern with highest priority in this agenda is chosen for execution by the *bbc-control*. In the following, we will explain our concept of patterns of behaviour in more detail.

Patterns of behaviour (PoB) There are two basic classes of patterns of behaviour in our terminology. On the one hand, there is a class of patterns which are basically reactive (so-called *reactor patterns*). On the other hand, certain patterns of behaviour describe pieces of procedural knowledge, i.e. mechanisms which are not represented in a declarative manner, but which are basically procedures. These *procedure* patterns are abstract (pre-compiled) actions which can be activated by the planner. They are appropriate for representing *routine behaviours*, which do not require deep reflection or planning. Examples for such routine tasks are starting a car, or walking along a hallway, or moving straight ahead from one landmark to another.

In the loading-dock, each pattern of behaviour is implemented as a set of OPS-5 rules. Specific parts of this rule set represent the execution body, activation condition, termination, failure, and *holding* conditions of the PoB. Patterns for basic reactive behaviour in FORKS have been designed to recognize a new transportation order, recognize when a box has been found, detect a threatening collision, avoid a collision (*step aside* behaviour), detect a blocking conflict, or to resolve a blocking conflict (*random move* behaviour, *step aside* behaviour). Patterns representing abstract actions are implemented for moving to a landmark (variant of potential field using a weighted random function), grasping and storing a box, respectively, searching for a box in a certain area (around a truck, in a shelf), performing a *query* protocol with another agent e.g. in order to find out the goals of this agents or to get to know the location of a certain box, leaving a narrow shelf corridor and waiting outside, or for returning to the waiting zone.

4.2.2 The Plan-based Component

The PBC of the INTERRAP model contains the agent's local planning facilities. There are two "standard modes" of the PBC: the first is to devise a plan for a goal upon request from the BBC (message *do(Goal)*) and to control the correct execution of the plan. The second is to interpret the agent's part in a joint plan devised by the component (message *interpret(Plan)*). In the following, the different subcomponents of the PBC and their interplay are explained. The PBC consists of a planning control module *pbcontrol*, of a *plan generator* module, a *plan evaluator* module, and a *resource handler* module. The *pbcontrol* is the "head" of the PBC. Its main parts are the PBC interface, the plan interpreter, and a set of goal stacks. The interface receives messages from and sends messages to the BBC. The plan interpreter controls the processing and the decomposition of a plan. Furthermore, based on the information brought about by the plan evaluator, it

decides which goal to plan next. For this purpose, it maintains a set of goal stacks. This is necessary, because the planner may be called by several concurrent patterns of behaviour. Thus, for each goal, one goal stack is maintained. In each cycle, the interpreter chooses one of the goal stacks and processes the top goal of this stack. Processing a goal means either to expand the goal into subgoals, or to activate a PoB.

The task of the *plan generator* is to devise a (single-agent) plan for a given input goal of the agent. The plan generator is called by the *pb-control* by sending a command `make_plan(Goal, ?Planlist)`.

One of our major concerns when modeling the FORKS system was to keep the planning problem as simple as possible by incorporating domain knowledge. Therefore, the main task of the plan generator is to choose and instantiate a suitable plan from a plan library. The plan library consists of a set of entries

$$\text{plan-lib} ::= (\text{lpb-entry}_1, \dots, \text{lpb-entry}_k).$$

Each entry of the plan library is a tuple `lpb-entry(Goal, Type, Body)`. `Goal` is the reference name of the entry and specifies which goal (or rather: which plan step corresponding to a certain goal) is expanded by the specific entry. `Type` can be either *s* for *skeletal plan* or *b* for *executable pattern of behaviour*. For `Type = s`, the `Body` of the entry consists of a list of plan steps, which specify the expansion of the entry plan step. If `Type = b`, the `Body` denotes the name of an executable pattern of behaviour. In more recent work we investigate how the event calculus can be adopted to extend our model with a general purpose planner.

The *plan evaluator* is able to associate utilities with plans. If it receives a list of alternative plans, it returns a list of evaluated plans. The evaluation is used by the plan interpreter in order to decide which of the alternative plans to pursue.

4.3 Coordination and Cooperative Conflict Resolution

Some situations exceed the problem-solving capacities of the PBC. In these cases, control is shifted to the CC. The main functionality of the CC is that it has to devise a *joint plan* for a certain situation, given a description of the situational context (e.g. type of conflict, position of the conflicting agents) and of the mental context (current goals of the agent). The social context (goals of other agents involved) has to be provided and checked by the cooperation component by evaluating available information about other agents. Currently, goal information is acquired by starting a simple query protocol. This classification process results in a type of interaction, which is used as an index to an entry in a library of joint plans.

The basic parts of the CC are the *CC control*, the *joint plan generator*, the *joint plan evaluator*, the *joint plan translator*, and the *resource handler*.

The *CC control* serves as an interface between the CC and the PBC. It coordinates the work of the other submodules of the component. Furthermore, it contains a classifier which is able to map a goal description contained in a task `do(Goal)` received by the PBC to a type of interaction using information from the PBC and the partner model library.

The task of the *joint plan generator* is to return a set of joint plans for a given situation specification `make_plan(Type, OwnGoal, PartnerGoals)` so that each of the resulting plans satisfies `OwnGoal  $\cup$  PartnerGoals`. As mentioned above, our objective has been to keep planning tractable by utilizing domain-specific information, and to concentrate on the interplay among the modules. Therefore, the plan generator has to select a set of joint skeletal plans indexed by the situation specification from a joint plan library. This selection is done by pure matching.

Since joint plans are subject to negotiation, the agent must be able to evaluate a joint plan which has been proposed to it by another agent. On the other hand, in order to generate “reasonable” joint plans itself, the agent must have a measure for what is a reasonable plan. It is the task of the *joint plan evaluator* to determine whether a plan is reasonable by attributing a utility for the plan. The evaluator accepts as input a list $[P_1, \dots, P_k]$ of joint plans proposed for achieving a goal, and outputs a list of evaluated plans $[(P_1, e_1), \dots, (P_k, e_k)]$ where e_i is the utility ascribed to P_i . In the current FORKS implementation, the utility of a plan P is computed in a straightforward manner as $U(P) = -C(P)$, where $C(P) = \sum_{p \in P} c(p)$, where p are primitive actions in P , where we assume that a cost function c on primitive plan steps is given.

It is the task of the *joint plan translator* to transform a joint plan into a single-agent plan by projecting the agent’s part of the joint plan and by adding synchronization actions which guarantee that the constraints contained in the original joint plan are satisfied during plan execution. The organization of the translator depends on the way joint plans and single agent plans are represented. In section (4.4), we will show the functionality of the translator module by means of the loading dock example.

4.4 Control

As we have seen, the modules of the INTERRAP model exchange information and shift control by communication. In fact, this process can be described in two dimensions, a *bottom-up* dimension and a *top-down* dimension. As tasks get more and more complex, control for this task is shifted from lower layers to higher layers until a suitable one has been found (*bottom-up, competence-driven control*). At the same time, the lower layers remain active and thus can cope with new events. Then, for task execution, control flows back (*top-down, activation-driven*) from higher layers to lower layers, finally resulting in the execution of primitive actions in the WIF. We refer to [12] for a more detailed description of these matters.

5 CONCLUSION AND OUTLOOK

In this article, we described a general framework for the design of DAI applications and showed its usefulness by developing two systems for significantly differing domains: the transportation domain and interacting robots. An agent architecture for autonomous interacting systems and a DAI system development platform were presented and we described how planning problems are solved using this framework. In this respect, this article overviews some of the main results achieved during the past three years of DAI research in the AKA-MOD project at DFKI.

What became obvious during our work was that the clue to understanding (predicting, designing) the behaviour of a multi-agent system lies in the understanding of the decision processes of the individual agent; viewed in terms of the INTERRAP architecture, this means to formalize the decision processes in the behaviour-based layer (priority of PoB, suppression, goal activation), in the local planning layer (planning decisions, costs, and utilities), and in the cooperation layer (initiation of cooperation, decisions within negotiations, decisions wrt. the goals of other agents). Developing such a general model and developing tools that support the designer of a DAI application in modeling the agents’ decision layer are the primary tasks of our future research.

References

- [1] A. Bond and L. Gasser. *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann, Los Angeles, CA, 1988.
- [2] Rodney A. Brooks. A robust layered control system for a mobile robot. In *IEEE Journal of Robotics and Automation*, volume RA-2 (1), April 1986.
- [3] E. H. Durfee and V. R. Lesser. Negotiating task decomposition and allocation using partial global planning. In *Distributed Artificial Intelligence, Volume II*, pages 229–244, San Mateo, CA, 1989. Morgan Kaufmann Publishers, Inc.
- [4] J. Falk, S. Spieck, and P. Mertens. Unterstützung der Lager- und Transportlogistik durch Teilintelligente Agenten. *Information Management*, 2, 1993.
- [5] K. Fischer, N. Kuhn, and J. P. Müller. Distributed, knowledge-based, reactive scheduling in the transportation domain. In *Proc. of the Tenth IEEE Conference on Artificial Intelligence and Applications*, San Antonio, Texas, March 1994.
- [6] K. Fischer and H. J. Müller. Cooperative problem solving in the transportation domain. In Ulrich Derigs, editor, *Proc. of the Int. Conf. on Operations Research '94*. Springer-Verlag, 1995. To Appear.
- [7] M. P. Georgeff and A. L. Lansky. Procedural knowledge. In *Proc. of the IEEE Special Issue on Knowledge Representation*, volume 74, pages 1383–1398, 1986.
- [8] D. Kinny, M. Ljungberg, A. Rao, E. Sonenberg, G. Tidhar, and E. Werner. Planned team activity. In A. Cesta, R. Conte, and M. Miceli, editors, *Pre-Proceedings of MAAMAW'92*, July 1992.
- [9] J. P. Latombe. How to move (physically speaking) in a multi-agent world. In Y. Demazeau and E. Werner, editors, *Decentralized A.I. 3*. North-Holland, 1992.
- [10] D. McDermott. Robot planning. Technical Report 861, Yale University, Department of Computer Science, 1991.
- [11] J. P. Müller and M. Pischel. The Agent Architecture INTERRAP: Concept and Application. Technical Report RR-93-26, German Artificial Intelligence Research Center (DFKI), Saarbrücken, June 1993.
- [12] J. P. Müller and M. Pischel. Modeling interacting agents in dynamic environments. In *Proc. of the European Conference on Artificial Intelligence (ECAI94)*, pages 709–713. John Wiley and Sons, August 1994.
- [13] A. S. Rao and M. P. Georgeff. Modeling Agents Within a BDI-Architecture. In R. Fikes and E. Sandewall, editors, *Proc. of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, Cambridge, Mass., April 1991. Morgan Kaufmann.
- [14] R.G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. In *IEEE Transaction on Computers*, number 12 in C-29, pages 1104–1113, 1980.

Adaptives Planen in Intelligenten Assistenzsystemen

Dietmar Dengler,
DFKI GmbH,
Stuhlsatzenhausweg 3,
66123 Saarbrücken,
e-mail: dengler@dfki.uni-sb.de

Abstract

A generic planning system is introduced which allows for custom building of planners able to generate plans for different plan consumers in the context of intelligent assistance systems. All planners are adapted to the peculiarities of different plan consumers, to their domain knowledge, their typical behavior, their preferences, and their utilization of plans.

The necessary knowledge sources of the generic planner are fixed in order to enable it to produce plans of a certain specificity. Its control strategy is described in a formal specification language containing constructs which allow for the configuration of characteristic parts of the control strategy. The customized planners are defined by executable specifications.

An application of the approach to deductive planning based on a modal temporal logic is shown. It is shown in an example how needs of different plan consumers in an intelligent help system can be met by a deductive planner.

1 Introduction

As the complexity of interactive computer software increases in terms of large amounts of information to be conveyed and a wide range of task structures, intelligent user support systems become necessary to guide and assist users in accomplishing their tasks. The aim is to give the user support depending on the context and relative to his knowledge and experience. The tasks of a planner in such a context are multifaceted: it must be able to generate plans which serving as a basis for the monitoring of the user's performance and representing user-oriented optimal solutions for recognized suboptimal behavior. The plans are the input to automatic execution facilities or may serve as the basis for a tutoring or advice-giving component. Thereby, the results of the planner have to fulfil different quality requirements with respect to specificity and resource consumption and must be adapted to the current needs of the particular *plan consumers*, e.g., to their domain knowledge, their capabilities, preferences, and behavior.

To tackle the problem of generating plan variants and to serve all types of plan consumers effectively requires high flexibility in the planner. In our approach we take as a

basis a generic planning system from which a wide range of particular planners can be customized appropriate to generating plans of a certain quality. The control strategy of the planner is described in a formal specification language containing constructs which allow for the configuration of characteristic parts of the control strategy. We have implemented our approach by building a deductive planner based on a modal temporal logic. Since planning is done there by proving formal plan specifications, adaptation means to guide the proof process accordingly and to restrict the knowledge base of the logical calculus used.

Aspects of intelligent user support systems which justify the approach of an adaptive planning system are mentioned in section 2. Section 3 describes necessary knowledge sources for the planner according to the application domain considered. In section 4 the approach of how to realize the configuration of a planner is explained. Finally, an application of the approach to deductive planning is described in section 5.

2 Intelligent User Support

A scenario where the flexibility and adaptability of a planning system is actually needed occurs in the context of intelligent help systems for application software (cf. [20], [2]). Help systems are required both to instruct users in how to operate software and to promote some understanding of the workings of the application system. The users of interactive software systems differ in the fact that they pursue different tasks, and have different experiences, preferences, and levels of knowledge. Support provided by the system should therefore be adapted to the individual user.

A planner can be seen as a central component of the support system since all activities of this system which concern plans or procedural processes are connected with the planner. The output of the planner must be adapted to the requirements of certain plan consumers:

- The user of the application system; he asks for concrete or abstract plans, would like to have an optimized plan for a given suboptimal one, or wants the planner to verify the correctness of a plan he proposes.
- A reasoning system as a part of the support system, e.g. a plan recognizer needs plans as a basis to observe the user.
- An automatic action execution facility; it executes plans automatically and the plans don't need to contain any user-specific overhead, e.g. initiated by a request of the user, or by an offer for semantic plan completion.
- Either an advice-giving or tutor component; plans at various levels of detail and difficulty are needed to provide appropriate help to the user. The usage of user-unknown domain knowledge must also be under complete control.

Various possible requirements for plan quality are listed below (see also [9] for more details):

1. A plan should reach its goal with limited resources, e.g. with few basic actions, or there is a limit on resources which should not be exceeded.

2. There are restrictions and/or preferences with respect to the actions and control structures occurring in a plan.
3. A plan shows a certain degree of abstraction, or an increasing abstraction corresponding to the temporal course of the plan, e.g. a procedural advice can be better understood if it is detailed at the beginning and more abstract at the end; a plan recognizer prefers an abstract plan as a hypothesis for the user observation since it describes a large class of concrete plans in a compact way.
4. A plan contains supporting actions (e.g. information-providing actions) adapted to the knowledge of the consumer.
5. A plan has been generated using a certain problem solving strategy which is mirrored by structural properties of the plan, e.g. grouping of actions concerning the same objects.
6. A plan considers in its outlook the previous plan behavior of a user.
7. A plan is only a partial solution to a specified planning problem (i.e. the planner has to work incrementally).

Considering the plan features just mentioned it is obvious that a kind of user modelling is necessary in order to be able to produce plans which are adapted to the current user [16]. Pursuing the aim of providing intelligent support can also imply the necessity for interactive planning where a plan can be developed cooperatively with the user [9].

3 Knowledge Sources

In order to generate plans which are adapted to a particular plan consumer, structuring and access methods with respect to the domain knowledge must be available. The basic entities of plans are basic actions. The set of all available basic actions can be structured along different lines. Synonymous actions are collected to equivalence classes SC_i and preferences in such a class are expressed by a partial order $<_i$. The family $BS = (\langle SC_i, <_i \rangle | i \in I)$, I an index set, describes a basic structuring of the set of actions. Classes SC_i can be compounded to equivalence classes AC_{R_z} with respect to a relation R_z . Again, these classes are structured by a partial order $<_{R_z}$ on their elements. An instance of a relation R_z is e.g. "reaching the common goal z ". For I an index set, an indexed action set $A_{R,I}$ is defined as a family $A_{R,I} = (\langle AC_z, <_{R_z} \rangle | AC_z \text{ subclass of } AC_{R_z}, z \in I)$.

By a structural change of I (and of R respectively) a specialized indexing can be defined:

- The elements of I are atomic goals; one gets a map of atomic goals to realizing actions.
- The elements of I are sets of atomic goals; an indexing of actions with respect to commonly reachable goal sets is specified.
- The elements are pairs (G, C) , G, C sets of atomic goals; they represent conditioned indexes meaning that if goals G has to be reached by the simultaneous

presence of goals C then $R_{(G,C)}$ specifies an equivalence class on a part of **BS** with respect to the goals G . For example, a conditioned index can be used to specify the usage of positive effects of some actions.

Control structures are an instrument to assemble basic actions to plans. One can differentiate between three sets: *explicit* control structures CS_e appear directly in the application domain, *conceptual* control structures CS_c are only implicitly contained in the domain (e.g. various forms of iteration), and *abstract* control structures CS_a occurring in abstract plans are often needed to communicate with non-human plan consumers. Preferences in the different sets are again expressed by a partial ordering. *Typical plan behavior* can be considered as an extension of the action set structuring. Let P_G be an equivalence class of plans with respect to the relation “reaching the goal G ”, I an index set of pairs (G, C) with G a goal description and C a constraint, $<_{(G,C)}$ a partial order. The typical plan behavior is then described by $\mathbf{TP} = (\langle P_G, <_{(G,C)} \rangle | (G, C) \in I)$. The constraints C can be both domain dependent (e.g. conditions about the current system state) and domain independent conditions (e.g. user-specific conditions about the objects of the domain). Special indexes can be derived from $\mathbf{TP}$, e.g. which action to use in a specific situation, or how to order subgoals to simulate the user’s behavior. There is some knowledge about the human-computer interaction in the application domain considered. The assumption is that the user has that information about the system state which he can see on the screen. The planner has knowledge about how the user can access missing information and how to generate plans which pay attention to that.

Besides different kinds of indexes there is problem solving knowledge in the form of heuristics for subgoal ordering and there is a collection of resource-specific optimization strategies.

4 Configuration of a Planner

We will now describe our approach to the configuration of a planner. The idea is to have a two-level specification of its control strategy. A generic planner is defined by its specification in a specification language $\mathcal{L}_S$. Besides other constructs, $\mathcal{L}_S$ contains *choice points* cp_i as a language construct, whose possible instantiations are elements of $\mathcal{L}_S$. A choice point represents a place in a control strategy where planning-specific decisions are necessary. To abstractly describe the whole set of possible instantiations there is a language $\mathcal{L}_{CP_i}$ to specify a choice point. A term of this language is then part of a configuration description of the planner. A specific set of choice point descriptions is intended to describe the current requirements for plan quality which the planner has to reach. A concrete planner arises by replacing all choice points in the specification of the generic planner according to the configuration description. The planner then corresponds to a specification in a language $\mathcal{L}'_S$ obtained from $\mathcal{L}_S$ by omitting the choice point constructs. We have chosen $\mathcal{L}'_S$ in a way such that specifications can easily be executed, i.e. planning is done by executing the specification of the planner. The customized planner is able to generate plans of a specific quality.

The approach sketched above has the advantage that a customization of a planner can be automated and is even possible under certain circumstances during the execution of the planner, i.e. the planner can change its strategy during planning.

The specification language $\mathcal{L}_S$ is a tactic language in the tradition of metalogical frameworks (cf. [17]) and is formalized by an algebraic specification, i.e. $\mathcal{L}_S$ is given by a term algebra $T_\Sigma(X)$ with signature Σ and variable set X :

$$\begin{aligned} \Sigma = & \text{ sorts: } \text{spec}, \text{tactic}, \text{tactic_proc}, \text{string}, \dots \\ & \text{ opns: } \text{basic_step}_1, \dots, \text{basic_step}_n: \text{spec spec} \rightarrow \text{tactic} \\ & \quad \text{true}, \text{false}: \text{tactic} \\ & \quad \text{then: list(tactic) spec spec} \rightarrow \text{tactic} \\ & \quad \text{orelse: list(tactic) spec spec} \rightarrow \text{tactic} \\ & \quad \text{define_tac: string tactic spec} \rightarrow \text{tactic_proc} \\ & \quad \text{call_tac: string spec} \rightarrow \text{tactic} \\ & \quad \text{choice_tac: string} \rightarrow \text{tactic} \\ & \quad \dots \\ & \text{ eqns: } \forall s:\text{string } A,C:\text{tactic } B,D:\text{spec} \\ & \quad \text{define_tac}(s,A,B) \wedge \text{define_tac}(s,C,D) \rightarrow \\ & \quad A=C \wedge B=D \\ & \quad \dots \end{aligned}$$

The assumption is that a planning problem and its solution can be formalized in a certain logical language and they can be represented by a formal description of type *specification*. The basic entities of the tactic language are the constructs *basic_step*_{*i*} which represent elementary planning tasks. It is assumed that the input/output behavior of these tasks is given by known functions. There exist constructs of this kind of more than the type structures mentioned here, e.g. to describe access to different knowledge sources.

There are some control structures on tactics, like sequencing, choice making, iteration, and recursion. A special construct is *choice_tac* which identifies an adaptable choice point in the control strategy. A choice point *choice_tac*(*s*) is described by a language $\mathcal{L}_{CP}$, defined maximally by the following algebra:

$$\begin{aligned} \Sigma_s = & \text{ sorts: } \text{feature} \\ & \text{ opns: } f_1, \dots, f_n: \text{feature} \\ & \quad <: \text{feature feature} \rightarrow \text{feature} \\ & \quad ; : \text{feature feature} \rightarrow \text{feature} \\ & \quad \wedge: \text{feature feature} \rightarrow \text{feature} \\ & \quad \vee: \text{feature feature} \rightarrow \text{feature} \end{aligned}$$

A *feature* f_i abstractly denotes planning specific methods, strategies, or knowledge access. The various knowledge sources described in section 3 appear as features for choice points. For example, for a choice point “choose an action to reach an atomic goal” there can be features: *take_user_action_knowledge*, *take_system_action_knowledge*, *consider_user_preferences*, *take_action_avoiding_negative_interactions*, etc. The intended meaning of the operators of the choice point algebra is:

- $f_i < f_j$ means deterministic choice: try first feature f_i , if not successful try f_j
- $f_i ; f_j$ means sequencing: use first f_i , if successful use f_j

- $f_i \wedge f_j$ means intersection: describes that only solutions are admitted that arise according to feature f_i and feature f_j
- $f_i \vee f_j$ denotes a nondeterministic choice: decides nondeterministically to take one feature, if it cannot be successfully applied the other one is tried

There is a homomorphism $h_s : \mathcal{L}_{CP_s} \rightarrow T_\Sigma$ which translates abstract choice point descriptions into terms of the tactic language $\mathcal{L}'_S$. Let $T_{\Sigma'}(X)$ be the subalgebra of $T_\Sigma(X)$ where only the operation *choice_tac* is missing. Then, a *planning tactic* **PT** is described by a set of terms of $T_{\Sigma'}(X)$ where:

1. **PT** = $T \cup \{call_tac(planner, S)\}$,
where $T \subset \{\sigma(t_1, t_2, t_3) | \sigma = define_tac\}$
2. For all $pt \in \mathbf{PT}$ holds: there is subterm $pt' = call_tac(s, S)$ of pt iff there is a term $define_tac(s, U, S) \in \mathbf{PT}$

A current planning problem is solved by evaluating a term $call_tac(planner, plan_specification)$ into the standard boolean algebra. That may only result in *true* if the planner has worked successfully. The evaluation process is described by a homomorphism between $T_{\Sigma'}(X)$ and the boolean algebra and can be done by meta-interpretation.

A *configurable planning tactic* is described by a tuple

CT = $(\mathbf{PT}, \{(T_{\Sigma_{s_i}}, h_{s_i}) | s_i \text{ choice point in } \mathbf{PT}\})$, where **PT** a planning tactic with respect to $T_\Sigma(X)$. A *configuration description* is given by a set

CD = $\{t_i | t_i \in T_{\Sigma_{s_i}}, choice_tac(s_i) \text{ subterm of elements of } \mathbf{PT}\}$,
 $|\mathbf{CD}| \leq |\text{set of choice points in } \mathbf{CT}|$.

A configurable planning tactic originates from an appropriate planning strategy refinement process done by the planning system engineer. This process can be supported by abstract interpretation methods (cf. [7]) and the efficiency of the individual planners can be improved by partial evaluation and transformations (cf. [11]).

Configuration process

Given **CT** and **CD** a planning tactic **PT** has to be built. First, a complete and coherent configuration description is needed. There is a set of transformation rules which transform the set **CD**. Thereby macros are expanded, terms are rewritten, deleted, or added, e.g. since some simultaneously specified configuration parameters make no sense, or missing choice point specifications are supplemented using defaults [9]. A set of rewrite rules *RR* specifying how *choice_tac(s)*-terms have to be replaced by the corresponding tactics can be inferred according to the known homomorphisms h_s . Applying *RR* on the configurable tactic in **CT** results in a specific planning tactic **PT**.

5 A Deductive Planning System

We will now sketch how the approach of an adaptive planning system fits into the field of deductive planning. The system in which our approach is embedded is called PHI (cf. [5], [2]). It is a logic based tool for intelligent help systems and provides both a plan recognizer and a planning component which work in close cooperation. These two components considerably improve the performance of a help system. Regarding an active help mode—the help process is initiated by the system—the planning component

can generate plans which serve as hypotheses for the observation task of the plan recognizer. Plan recognition in this sense means that the user's actions observed have to be matched against a set of plans which represent possible user behavior with respect to the application system. The plan recognizer is among other things able to propose semantic plan completion if it is highly probably that the user pursues a particular plan [3]. Furthermore, the planning component can generate more optimal solutions for recognized plans known to be suboptimal. Regarding a passive help mode the planning component can directly generate optimized plans.

The logic LLP [5] is the basis for deductive planning in PHI. Plan specifications as well as plans are represented by certain classes of formulas in this logic. It is an interval-based modal temporal logic combining features of both traditional programming and temporal logics. It provides the modal operators $\bigcirc$ (next), $\Diamond$ (sometimes), $\Box$ (always), and the binary modal operator $;$ (chop) expressing the sequential composition of formulas. Apart from these operators, control structures are also available, as in programming logics.

Our deductive approach to plan generation can be summarized as follows: given a formal plan specification formula which describes the relation of preconditions and effects with respect to the plan which has to be found, the plan arises from a constructive proof of its specification.

Thereby, *plan specifications* are e.g. LLP formulas of the form

$$\forall x [\text{preconditions}(x) \wedge \text{Plan}_x] \rightarrow \text{goals}(x)$$

i.e., if *preconditions*(x) hold in a situation where we carry out Plan_x then we will reach *goals*(x), a formula containing temporal modalities. Plan_x is a metavariable for a plan formula. Its instantiation is obtained if the constructive proof of the specification formula ended successfully.

The basic actions are given by axiom schemata where instances are able to describe the primary effects of actions as well as their invariants. Control structures of plans are represented by logical and temporal control structures defined in LLP. The knowledge about these structures is represented by sets of deduction rules handling the introduction and elimination of these structures.

To perform the proofs a specific calculus for LLP, namely a sequent calculus, is used. The tactic language introduced in the last section can be used in a straightforward way to guide the necessary proofs. The most important basic entity of the language $\mathcal{L}_S$ becomes *apply_rule* performing the application of a sequent rule. Most of the operators of $\mathcal{L}_S$ are in the style of tactical theorem proving systems, e.g. cf. [17]. A current (deductive) planner is then given by the specification of a particular proof strategy for the underlying logic LLP. The different indexes on knowledge sources mentioned in section 3 provide restrictions, preferences, and constrained access to the nonlogical axioms and rules of the sequent calculus.

To demonstrate one aspect of consumer-specific planning we show in an example how supporting actions can be integrated in an abstract plan. The application domain where the example comes from is a subset of the UNIX operating system, namely its *electronic mail system*. The planner has to generate abstract plans which serve as hypotheses for the observation of the user by a plan recognizer. We regard the plan specification $\forall x \text{ pre}(x) \wedge P \rightarrow \Diamond \text{goal}(x)$. Abstract plans describe a class of concrete plans in a compact way. If the planner were to generate a concrete plan then two things must be proven, a modified plan specification, $\exists x \text{ pre}(x) \wedge P \rightarrow \Diamond \text{goal}(x)$, and whether the

plan is applicable in the current situation, $pre(t)$ for the substitution $\sigma = \{t/x\}$ from the proof of the specification.

The prover has access to a complete description of the current situation and can “compute” the bounded range of values for the variable x . But, the plan should simulate the behavior of the user, i.e. the user must also be able to follow the value restriction of x . In general, this is only possible by executing additional *informative* actions. The plan generated must correspond to this behavior. Using our proving approach it means that the plan specification must be automatically extended by goals corresponding to appropriate informative actions. The extension must guarantee that the original specification is still valid. It looks abstractly like

$$\forall x \ pre(x) \wedge P \rightarrow \Diamond(add_goal(x) \wedge \Diamond goal(x))$$

and causes P to become a sequence $P_{add_goal}; P_{goal}$ during planning. The point is, that $pre(x)$ can contain a subformula expressing a value restriction on x which is however not a necessary precondition for a plan which fulfils the original specification.

Let’s have a look at an example from the mail domain. The specification considered is:

$$\forall x \ sender(x) = s \wedge delete_flag(x) = 0 \wedge P \rightarrow \Diamond read_flag(x) = 1$$

It describes that a message x , which is not deleted and has sender s , should be read. A simple plan which fulfils that specification is the abstract single action plan $EX(type(x))$ which only presupposes that x is not deleted. But that plan doesn’t simulate the behavior of the user, who has to know which message fulfils all specified preconditions. The plan could be used as a basis of a concrete plan for the consumer *automatic action execution*. However, the user usually initiates some additional informative action. The plan specification can be extended according to knowledge about human-computer interaction in the domain. Using the meta-knowledge

$$\forall x, y \ member(x, screen) \wedge same_type(x, y) \rightarrow \\ instantiation_possible_by_user(y)$$

the specification becomes:

$$\forall x \ sender(x) = s \wedge delete_flag(x) = 0 \wedge P \rightarrow \\ \Diamond(member(x, screen) \wedge \Diamond read_flag(x) = 1)$$

A more reasonable instantiation for P is then the plan

$EX(from(s)); EX(type(x))$, where the *from* action just produces the missing information. The plan can still be more sophisticated if typical user behavior is also considered. For example, it can be available that after the user has read a message, typically he may save it if it is interesting. Assuming that the system does not exactly know what defines the concept “interesting message” the instantiation for P can then be:

$EX(from(s)); EX(type(x)); EX(save(x, +x)) \vee EX(empty)$.

The nondeterministic subplan appears since it is uncertain if the message is really interesting for the user.

Regarding an appropriate configuration of the planner in order to produce the different plans the choice point *work on subgoal* with respect to the last mentioned instantiation of P can be described as:

```
(informative_action_intro ; standard ;
(plan_behavior_certain < plan_behavior_uncertain ))
< standard.
```

“standard” designates a simple backward-chaining strategy to find a plan with respect to the current subgoal.

A central control component of the support system can direct the configuration of a current planner according to requirements of the plan consumers. But it is also possible that the consumers themselves initiate a configuration (cf. [9]).

6 Related Work

Improving the usability of computer systems is an important research goal of human-computer interaction (cf. [4]). Adaptive systems with their design variety help to approach this goal (cf. [12, 16]). The system should be able to adapt to the individual characteristics and needs of its users. Human-computer interaction systems of that kind where a planner is an integrated part can be members of the following classes: intelligent help system (cf. [20], [2]), intelligent assistant system ([6]), or intelligent tutoring systems (cf. [10]). In these systems the planner’s results have to be measured with respect to the user’s ongoing task, his knowledge about the domain, and his experience with the application system (cf. [22], [12]). A plan is then optimal if it is both well adapted to the user’s requirements and as short as possible according to the planner’s ability.

With the integration of general optimization techniques in a planning system the complexity problem of planning must be considered (cf. [1]). Often, only approximate solutions are realistic, cf. e.g. [13, 14, 23]). It is desirable to place at the planning system engineer’s disposal, a pool of optimization techniques from which the most promising candidate for the current application can then be taken. Adaptation of the planning system to the domain model can improve its efficiency (cf. [15]), e.g. by automatically building appropriate indexes on the domain knowledge (see section 3).

Generic planning systems have the advantage that their flexibility and expressive clarity allow for the easy development, debugging, and maintenance of particular planners.

There are some other approaches which partially consider aspects of a generic planning system. TEST (cf. [18]) is a specification tool for planners. Its framework can be used as a prototyping tool for new planners relying on an analysis of planning as theory extension. In the planning framework of Cranefield (cf. [8]) it is possible to define various different temporal operators and corresponding inference rules for combining plans to form larger compound plans. Furthermore, one has complete control over the planning process since the planning strategy is defined by tactics. The MRG system ([21]) provides a general framework to handle plans. It uses a tactic language to represent basic actions, events, goals as well as activities, like plan generation, execution and monitoring, and interaction with the real world. A user of the system is able to combine, and integrate various planning techniques. O-Plan2 ([19]) is an architecture providing different mechanisms to enable a planning and control system builder to select suitable implementation methods for describing and triggering choices, and posting constraints to restrict them during planning.

7 Conclusion

We introduce a generic planning system which allows one to customize special-purpose planners easily for application in the intelligent support system context. Since the planners must be able to generate results which fulfil specific quality criteria we took the approach that a control strategy for the particular planners can automatically be configured out of an individually specified criteria adjustment with respect to the generic planner. All planners are defined by formal executable specifications. In this way the refinement of the generic planner can easily be supported by abstract interpretation methods and on the other hand the efficiency of the individual planners can be improved by partial evaluation and transformations. The flexibility of the planning system makes it adaptable to the peculiarities of different plan consumers. This was sketched by the application of the approach to consumer-specific deductive planning in the context of intelligent help systems.

References

- [1] C. Bäckström and B. Nebel. Complexity results for SAS⁺ planning. In *Proc. of the 13th IJCAI*, Chambéry, 1993.
- [2] M. Bauer, S. Biundo, D. Dengler, J. Koehler, and G. Paul. PHI – a logic based tool for intelligent help systems. In *Proc. of the 13th IJCAI*, Chambéry, 1993.
- [3] M. Bauer and G. Paul. Logic-based plan recognition for intelligent help systems. In C. Bäckström and E. Sandewall, editors, *Proc. of the 2nd European Workshop on Planning*, 1993.
- [4] D. Benyon. Accommodating individual differences through an adaptive user interface. In M. Schneider-Hufschmidt, T. Küme, and U. Malinowski, editors, *Adaptive User Interfaces*, pages 149–165. North-Holland, 1993.
- [5] S. Biundo, D. Dengler, and J. Koehler. Deductive planning and plan reuse in a command language environment. In *Proc. of the 10th ECAI*, pages 628–632, Vienna, 1992.
- [6] G. A. Boy. *Intelligent Assistant Systems*. Academic Press, London, 1991.
- [7] P. Cousot and R. Cousot. Abstract interpretation frameworks. *Journal of Logic and Computations*, 2, 1992.
- [8] S. J. S. Cranefield. A logical framework for practical planning. In *Proc. of the 10th ECAI*, pages 633–637, Vienna, 1992.
- [9] D. Dengler. An adaptive deductive planning system. Technical Report RR-94-06, German Research Center for Artificial Intelligence (DFKI), 1994.
- [10] M. Elsom-Cook. Student modelling in intelligent tutoring systems. *Artificial Intelligence Review*, 7:227–240, 1993.

- [11] N.E. Fuchs. Specifications are (preferably) executable. *Software Engineering Journal*, 1992.
- [12] A. J. Kok. A review and synthesis of user modelling in intelligent systems. *The Knowledge Engineering Review*, 6, 1991.
- [13] A. L. Lansky. Localized representation and planning. In J. Allen, J. Hendler, and A. Tate, editors, *Readings in Planning*. Kaufmann, San Mateo, CA, 1990.
- [14] T. A. Linden. Transformational synthesis: An approach to large-scale planning applications. In *Proc. of the Workshop on Innovative Approaches to Planning, Scheduling and Control*, pages 126–136, San Diego, CA, 1990.
- [15] L. McCluskey and J. Porteous. Model adaption: an approach to tackling the efficiency problem in general planning systems. In *UK PSIG Proc.*, 1992.
- [16] M.F. McTear. User modelling for adaptive computer systems: a survey of recent developments. *Artificial Intelligence Review*, 7:157–184, 1993.
- [17] L. Paulson. The foundation of a generic theorem prover. *Automated Reasoning*, 5, 1989.
- [18] H. Reichgelt and N. Shadbolt. A specification tool for planning systems. In *Proc. of the 9th ECAI*, Stockholm, 1990.
- [19] A. Tate. The emergence of “standard” planning and scheduling system components - open planning and scheduling architectures. In C. Bäckström and E. Sandewall, editors, *Proc. of the 2nd European Workshop on Planning*, 1993.
- [20] C. Tattersall. A new architecture for intelligent help systems. In C. Frasson, G. Gauthier, and G. I. McCalla, editors, *Proc. of the 2nd Intelligent Tutoring Systems Conference*, pages 302–316. Springer, Berlin, Heidelberg, 1992.
- [21] P. Traverso, A. Cimatti, and L. Spalazzi. Beyond the single planning paradigm: Introspective planning. In *Proc. of the 10th ECAI*, pages 643–647, Vienna, 1992.
- [22] R. Winkels. User modelling in help systems. In D. H. Norrie and H.-W. Six, editors, *Computer Assisted Learning: 3rd International Conference, ICCAL '90, Hagen, FRG, June 1990*, pages 184–193. Springer, New York, 1990.
- [23] Q. Yang, D. Nau, and J. Hendler. Merging separately generated plans with restricted interactions. *Computational Intelligence*, 8:648–676, 1992.

CAPlan – ein SNLP-basierter Planungsassistent*

Frank Weberskirch, Jürgen Paulokat
Universität Kaiserslautern, FB Informatik
Postfach 3049, D-67653 Kaiserslautern
{weberski | paulokat}@informatik.uni-kl.de

Zusammenfassung

Im Rahmen dieser Arbeit beschreiben wir die wesentlichen Merkmale der CAPlan-Architektur, die die interaktive Bearbeitung von Planungsproblemen ermöglichen. Anhand des SNLP-Algorithmus, der der Architektur zugrunde liegt, werden die im Laufe eines Planungsprozesses auftretenden Entscheidungspunkte charakterisiert. Mit Hilfe von frei definierbaren Kontrollkomponenten kann das Verhalten an diesen Entscheidungspunkten festgelegt werden, wodurch eine flexible Steuerung des Planungsprozesses ermöglicht wird. Planungsziele und -entscheidungen werden in einem gerichteten azyklischen Graphen verwaltet, der ihre kausalen Abhängigkeiten widerspiegelt. Im Gegensatz zu einem Stack, der typischerweise zur Verwaltung von Entscheidungen eingesetzt wird, erlaubt die graphbasierte Repräsentation die flexible Rücknahme einer Entscheidung, ohne alle zeitlich danach getroffenen Entscheidungen ebenfalls zurücknehmen zu müssen.

1 Einleitung

Arbeiten aus dem Bereich der Planung liegt meistens die Annahme zugrunde, daß eine Planungsaufgabe von einem autonomen Planungssystem gelöst werden kann. Die in diesem Zusammenhang beschriebenen Planungsalgorithmen basieren auf einem allgemeinen Suchverfahren, mit dessen Hilfe der Raum möglicher Pläne schrittweise, ohne jegliche Interaktion mit der Außenwelt abgearbeitet wird. Ohne auf die Verwaltung von Planungsentscheidungen explizit einzugehen, legen die Darstellungen der Algorithmen einen Stack zu ihrer Verwaltung nahe, der auch in vielen Implementierungen von Planungsalgorithmen eingesetzt wird.

Bei der Anwendung eines Planungsverfahrens auf reale Problemstellungen führt das autonome Problemlösungsverhalten häufig zu Akzeptanzproblemen oder zu nicht adäquaten Lösungen. Komplexe Aufgabenstellungen können oftmals nur von interaktiven Planungssystemen sinnvoll gelöst werden, die einerseits in der Lage sind, Teilaufgaben automatisch zu bearbeiten und andererseits auf Benutzerinteraktionen opportunistisch zu reagieren, d.h. es wird kein autonomes Planungssystem benötigt, sondern ein Planungsassistent. Im einfachsten Fall einer Benutzerinteraktion handelt es sich um die Auswahl eines Ziels, mit dem der Planungsprozeß fortgesetzt werden soll oder um die Auswahl eines Schritts zur Erreichung eines Ziels. Beides kann mit Hilfe einer stackbasierten Implementierung eines Planungsverfahrens unterstützt und verwaltet werden. Soll darüberhinaus die interaktive Zurücknahme von Planungsentscheidungen möglich sein, zeigen stack-basierte Verfahren ein problematisches Verhalten. Aufgrund der vollständigen Anordnung der Entscheidungen müssen alle Entschei-

*Die vorgestellte Arbeit wurde teilweise durch die Deutsche Forschungsgemeinschaft, SFB 314: „Künstliche Intelligenz – Wissensbasierte Systeme“, Projekt X 9 gefördert.

dungen, die zeitlich nach der zurückgenommenen Entscheidung getroffen wurden, ebenfalls zurückgenommen werden, obwohl sie sich u.U. auf unabhängige Teilprobleme beziehen.

Im folgenden wird die Verwaltung von Planungsentscheidung in CAPlan beschrieben, die es ermöglicht, Entscheidungen zurückzunehmen, wobei die aktuelle Teillösung nur so weit wie notwendig modifiziert wird. Da sich die Implementierung von CAPlan weitgehend an SNLP orientiert, werden zunächst in Abschnitt 2 die wesentlichen Ideen von SNLP im Überblick dargestellt. Abschnitt 3 beschreibt die wichtigen Eigenschaften der CAPlan-Architektur und vergleicht das Verhalten von SNLP und CAPlan anhand eines Beispiels. Die Mechanismen und zentralen Datenstrukturen zur Verwaltung von Planungsaufgaben und Planungsentscheidungen sind Gegenstand von Abschnitt 4. Das Papier endet mit einer abschließenden Diskussion der dargestellten Mechanismen (Abschnitt 5).

2 SNLP

Zunächst sollen einige grundlegenden Begriffe und Notationen erläutert werden, die charakteristisch für die Planung mit SNLP-basierten Planern wie CAPlan sind (vgl. [Barrett & Weld, 1993] und [Knoblock & Yang, 1994]). Weiter wird auf die Behandlung von Interaktionen (Threats) eingegangen und der SNLP-Algorithmus vorgestellt.

2.1 Notationen

Planungsprobleme bestehen aus einer initialen Situation und einer Menge $\{g_1, \dots, g_n\}$ von Zielen. Ein *partiell geordneter Plan* ist ein Tupel (S, O, B) , wobei S eine Menge von Planschritten¹, O eine Menge von Ordnungs-Constraints auf den Planschritten aus S und B eine Menge von Bindungs-Constraints für Variablen ist. In S gibt es zwei ausgezeichnete Planschritte, s_0 und s_∞ ; sie repräsentieren die aktuelle Problemstellung dadurch, daß s_0 als Effekte die initiale Situation und s_∞ als Vorbedingungen die Menge der zu erfüllenden Ziele hat. Die aktuellen Planungsziele G (offene Ziele) lassen sich als eine Menge von Tupeln (p, s) mit der Eigenschaft definieren, daß p eine Vorbedingung von Schritt s ist, die noch nicht erfüllt ist.² Der *initialer Plan* zu einem Planungsproblem ist gegeben durch $(S_0, O_0, B_0) = (\{s_0, s_\infty\}, \{s_0 \prec s_\infty\}, \emptyset)$, die initiale Menge der Planungsziele durch $G = \{(g_i, s_\infty) | i = 1, \dots, n\}$.

Ein wesentlicher Punkt bei SNLP ist, daß die Gültigkeit jeder Vorbedingung eines Planschrittes durch die Erzeugung eines *Causal Links* garantiert wird. Wenn ein Schritt s_i einen Effekt hat, der verwendet wird, um die Vorbedingung p eines Schrittes s_j wahr zu machen, dann drückt der Causal Link $s_i \xrightarrow{p} s_j$ diesen Zusammenhang aus.

Ein Planschritt s_k interagiert mit einem Causal Link $s_i \xrightarrow{p} s_j$, wenn s_k einen Effekt q hat, der möglicherweise (d.h. eventuell erst mit zusätzlichen Bindungs-Constraints) identisch mit p oder der Negation von p ist. Dann heißt das Tupel $(s_k, s_i \xrightarrow{p} s_j)$ ein *Threat*. Man spricht von einem *positiven Threat*, wenn q möglicherweise identisch mit p ist; analog handelt es sich um einen *negativen Threat*, wenn q möglicherweise die Negation von p ist. Weiterhin soll zwischen *aktiven* und *potentiellen Threats* unterschieden werden. Ein Threat ist aktiv, wenn der Schritt s_k möglicherweise zwischen s_i und s_j liegt und die u.U. notwendigen Bindungs-Constraints konsistent mit dem aktuellen Plan sind. Ansonsten handelt es sich um einen potentiellen

¹Ein Planschritt ist ein partiell instantiiertes Operator-Schema der Domäne, welches Vorbedingungen, Effekte, neu eingeführte Bindungs-Constraints für Variablen und neu hergeleitete Teilziele bestimmt.

²Im Text wird p alleine offenes Ziel genannt, wenn das zugehörige s aus dem Kontext klar hervorgeht.

Threat, d.h. entweder gilt im aktuellen Plan $s_k \prec s_i$ oder $s_j \prec s_k$ oder die notwendigen Constraints führen zu einem inkonsistenten Plan. Potentielle Threats spielen erst beim Rückzug von Planungsentscheidungen eine Rolle (vgl. Abschnitt 4.3). SNLP unterscheidet sich z.B. von NONLIN [Tate, 1977] dadurch, daß aus Gründen der Systematik neben negativen auch positive Threats aufgelöst werden. SNLP betrachtet nur aktive Threats.

2.2 Auflösen von Threats

Tritt im Plan (S, O, B) ein aktiver Threat $t = (s_k, s_i \xrightarrow{p} s_j)$ auf, so kann dieser grundsätzlich aufgelöst werden, indem der interagierende Schritt s_k durch zusätzliche Ordnungen aus dem Bereich des Causal Link gebracht wird oder indem durch zusätzliche Variablenbindungs-Constraints die Unifikation verhindert wird. Aus dem aktiven wird so ein potentieller Threat.

Es ergeben sich konkret die folgenden Auflösungsmöglichkeiten eines Threats t , auf die im folgenden Algorithmus mit *Resolve-Threat*(t) zurückgegriffen wird: (P bezeichne die Menge von Variablen-Paaren (x, y) , die übereinstimmen müssen, damit der Threat entsteht)

- **Promotion:** $O' = O \cup \{s_j \prec s_k\}$, $B' = B$
- **Demotion:** $O' = O \cup \{s_k \prec s_i\}$, $B' = B$
- **Separation:** $O' = O \cup \{s_i \prec s_k, s_k \prec s_j\}$,
 $B' = B \cup \sigma$ mit $\sigma = \{\alpha \mid \alpha = \text{noncodesignate}(s) \cup \text{codesignate}(P - s), \emptyset \neq s \subseteq P\}$

Hier bezeichnen die Ausdrücke *codesignate*(R) bzw. *noncodesignate*(R) die Menge der Gleichheits- bzw. Ungleichheits-Constraints zwischen den Variablenpaaren $(x, y) \in R$. Charakteristisch für SNLP ist, daß aus Gründen der Systematik bei jeder Möglichkeit von Separation alle Variablenpaare aus P betrachtet werden müssen.

2.3 Algorithmus

SNLP sucht im Raum der partiell geordneten Pläne. Der folgende Algorithmus beschreibt in Anlehnung an die Algorithmen aus [McAllester & Rosenblitt, 1991], [Barrett & Weld, 1993] und [Knoblock & Yang, 1994] die Vorgehensweise von SNLP und bildet das Grundgerüst des Algorithmus von CAPlan.

Die Parameter des Algorithmus sind: ein partieller Plan (S, O, B) , eine Menge G von offenen Zielen, eine Menge L von Causal Links und eine Menge T von aktiven Threats. Der Algorithmus ist rekursiv und löst chronologisches Backtracking aus, wenn bei der Threat-Auflösung oder Operatorauswahl keine konsistente Möglichkeit (mehr) existiert. Ziel des Algorithmus ist es, für jede Vorbedingung eines Schrittes im Plan einen gegen alle Threats geschützten Causal Link zu erzeugen. Für ein gegebenes Planungsproblem wird der Algorithmus mit dem initialen Plan und der initialen Menge G von Planungszielen aufgerufen: $SNLP((S_0, O_0, B_0), G, \emptyset, \emptyset)$.

Algorithmus $SNLP((S, O, B), G, L, T)$

1. **Termination:** Falls $G = \emptyset$ und $T = \emptyset$, dann fertig (Erfolg).

2. **Threat-Behandlung:** (falls $T \neq \emptyset$)

(a) **Threat-Auswahl:** Sei $t \in T$ mit $t = (s_k, s_i \xrightarrow{p} s_j)$

(b) **Threat-Auflösung:** (Backtracking-Punkt)

Resolve-Threat(t): Dies löst t auf und verändert O zu O' und B zu B' .

- (c) **Threat-Aktualisierung:** $T' = T - \{t | t \in T \text{ kein aktiver Threat in } (S, O', B')\}$
- (d) **Rekursiver Aufruf:** $SNLP((S, O', B'), G, L, T')$

3. Zielarbeitung: (falls $G \neq \emptyset$)

- (a) **Zielauswahl:** Sei $(p, s_{need}) \in G$ das nächste offene Ziel.
- (b) **Operatorauswahl:** (Backtracking-Punkt)
Sei s_{add} ein Schritt (aus S oder neu) mit Effekt p , der nicht hinter s_{need} angeordnet ist. $S' = S \cup \{s_{add}\}$; $O' = O \cup \{s_{add} \prec s_{need}\}$; $B' = B \cup \{\text{Bindungs-Constraints, so daß } s_{add} \text{ Effekt } p \text{ hat}\}$; $G' = G - \{(p, s_{need})\} \cup \text{Preconditions}(s_{add})$, $L' = L \cup \{s_{add} \xrightarrow{p} s_{need}\}$.
- (c) **Threat-Erkennung:** $T' = \{t | t = (s_k, s_i \xrightarrow{p} s_j) \text{ aktiver Threat in } (S', O', B')\}$
- (d) **Rekursiver Aufruf:** $SNLP((S', O', B'), G', L', T')$

Bemerkung zum Algorithmus

Korrektheit, Vollständigkeit und Systematik³ sind nachgewiesene Eigenschaften von SNLP. Der obige Algorithmus ist jedoch (ebenso wie der Algorithmus POCL aus [Barrett & Weld, 1993]) eine Vereinfachung des ursprünglichen Find-Completion-Algorithmus aus [McAllester & Rosenblitt, 1991] und die Vollständigkeit kann hier nicht garantiert werden, da der Algorithmus u.U. nicht terminiert, weil es zu endlosem Herleiten neuer Teilziele kommen kann. Aus diesem Grund löste der ursprüngliche Algorithmus aus [McAllester & Rosenblitt, 1991] beim Überschreiten einer Kostenobergrenze chronologisches Backtracking aus und verhindert so endloses Suchen. Vollständigkeit wird dann durch *Iterative Deepening* [Korf, 1987] erreicht.

In [Weberskirch, 1994] wird der obige Algorithmus so erweitert, daß das Terminationsproblem gelöst ist, ohne daß eine solche Kostenobergrenze und ein Iterative-Deepening-Verfahren verwendet wird. Dazu werden zunächst die für eine Nichtterminierung verantwortlichen Teilzielrekursionen erkannt. Eine Teilzielrekursion tritt auf, wenn durch Anwendung eines Operators op auf ein Ziel g wieder ein Teilziel $g' = g$ hergeleitet wird (eventuell erst nach mehreren anderen Operatoranwendungen). In diesem Fall muß verhindert werden, daß g' wieder durch den Operator op bearbeitet wird, weil dies unweigerlich zur Rekursion führt. Außerdem wird das einfache chronologische Backtracking durch Backjumping [Ginsberg, 1993] ersetzt.

3 CAPlan

Aufbauend auf das Planungsverfahren aus Abschnitt 2.3, soll hier auf die Modifikationen und Erweiterungen eingegangen werden, die aus einem SNLP-Planer einen Planungsassistenten machen. Zunächst wird eine Übersicht darüber gegeben, welche grundsätzlichen Möglichkeiten es gibt, den SNLP-basierten Planer CAPlan zu steuern. Dann geht es um die Schwierigkeiten, die durch Benutzerinteraktionen (insbes. durch die Rücknahme beliebiger Planungsentscheidungen durch den Benutzer) bei SNLP entstehen, und um deren Lösung in CAPlan.

3.1 Steuerung von CAPlan durch Kontrollkomponenten

Betrachtet man den Algorithmus aus Abschnitt 2.3, so kann man folgende Punkte identifizieren, an denen während der Planung eine Auswahlentscheidung getroffen werden muß:

³Systematik meint, daß jeder Plan (S, O, B) nur maximal einmal erzeugt wird.

- **Threat-Auswahl:** Welcher Threat aus der Menge T soll bearbeitet werden?
- **Threat-Bearbeitung:** Wie soll ein Threat aufgelöst werden? (vgl. Abschnitt 2.2)
- **Zielauswahl:** Welches offene Ziel aus der Menge G soll bearbeitet werden?
- **Operatorauswahl:** Mit welchem Operator soll ein Ziel bearbeitet werden?

Charakteristisch für SNLP und damit für CAPlan ist, daß die Threat- und die Zielauswahl keine Backtracking-Punkte im Algorithmus sind. Prinzipiell spielt es also keine Rolle, welcher Threat oder welches offene Ziel zuerst bearbeitet wird. Ein geschickte Threat- bzw. Zielauswahl kann jedoch den Suchraum drastisch reduzieren.

Ein anderer Aspekt wird im SNLP-Algorithmus zu starr betrachtet: Threats werden immer sofort aufgelöst, bevor die Erfüllung von Teilzielen fortgesetzt wird. In [Peot & Smith, 1993] und [Smith & Peot, 1993] wird gezeigt, daß das nicht zwingend ist und es nachweisbar bessere Strategien gibt, die die Systematik von SNLP nicht beeinflussen [Harvey *et al.*, 1993].

In CAPlan wird dieser Steuerungsproblematik dadurch Rechnung getragen, daß es im Planungssystem eine frei definierbare Menge von *Kontrollkomponenten* gibt, von denen eine an allen oben genannten Stellen entscheidet, welche Alternative gewählt werden soll bzw. ob Threats oder Ziele bearbeitet werden sollen (vgl. [Weberskirch, 1994]). Dazu ist es in CAPlan notwendig, den Algorithmus aus Abschnitt 2.3 dahingehend zu modifizieren, daß die Schritte 2 und 3 nicht mehr in einer festen Reihenfolge durchlaufen werden, sondern die Reihenfolge von der Entscheidung der momentan aktiven Kontrollkomponente abhängig zu machen ebenso wie alle anderen Auswahlentscheidungen. Diese momentan aktive Kontrollkomponente kann auch während der Bearbeitung eines Planungsproblems beliebig gewechselt werden. Damit wird eine flexible Architektur geschaffen, da verschiedene Kontrollstrategien leicht in Form verschiedener Kontrollkomponenten definierbar und kombinierbar sind.

Momentan gibt es für CAPlan folgende Kontrollkomponenten:

- **SNLP:** Hier wird die Vorgehensweise des SNLP-Algorithmus aus Abschnitt 2.3 als Default-Kontrollkomponente zur Verfügung gestellt.
- **UC (User Control):** Mit dieser Kontrollkomponente arbeitet CAPlan als Planungsassistent. Anfallende Entscheidungen werden vom Benutzer getroffen. Der Grad an Interaktion mit dem Benutzer ist variabel; er kann sich z.B. auf den Aspekt der Ziel- oder Operatorauswahl beschränken oder sich auf alle o.g. Entscheidungspunkte beziehen.
- **CbC (Case-based Control):** Mit CAPlan/CbC [Muñoz *et al.*, 1994] steht eine Kontrollkomponente zur Verfügung, die episodisches Wissen in Form von Fallbeispielen zur Steuerung der Planung verwendet (fallbasierte Planung).

3.2 Beispiel für die Planung mit CAPlan

Nachdem der Algorithmus und der Steuerungsmechanismus von CAPlan vorgestellt wurde, soll nun an einem einfachen Beispiel demonstriert werden, wie CAPlan bei der Planung vorgeht und welche Defizite sich aus dem SNLP-Ansatz ergeben. Man gehe von einer Blocks-world-Domäne mit den Operatoren `PickUpBlock` und `PutDownBlock` aus (vgl. Abb. 1).⁴

⁴Zur Schreibweise folgende Bemerkung: `+Clear(y)` bzw. `-On(x,y)` als Effekt eines Operators heißt, daß der Operator `Clear(y)` bzw. `On(x,y)` gültig bzw. ungültig macht.

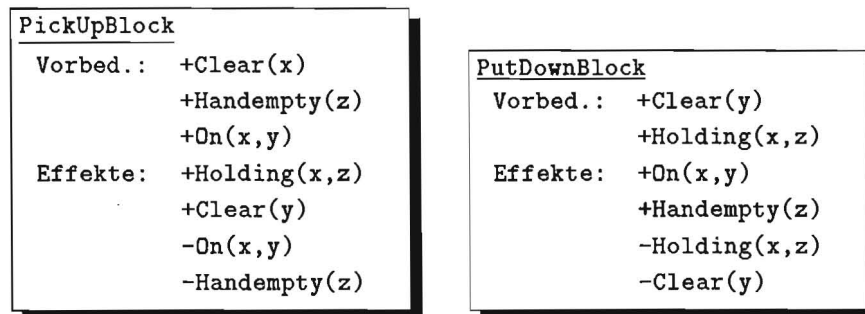


Abbildung 1: Operatoren PickUpBlock und PutDownBlock

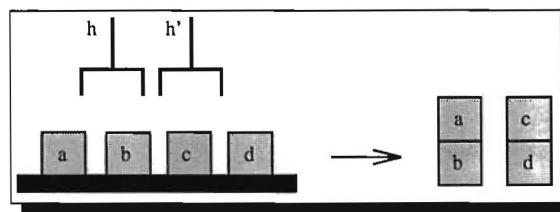


Abbildung 2: Beispiel für eine Problemstellung

Die Problemstellung sei wie in Abbildung 2 dargestellt: die Ziele $+On(a,b)$ und $+On(c,d)$ sind zu erreichen. Abbildung 3 zeigt einige Planzustände, die CAPlan durchläuft:⁵

Abbildung 3.a: Zunächst wird ein initialer Plan (Schritte START und FINISH) erzeugt und die Menge offener Ziele mit $G = \{(+On(a,b), FINISH), (+On(c,d), FINISH)\}$ initialisiert.

Abbildung 3.b: Hier wurde das Ziel $+On(c,d)$ ausgewählt und mit dem Operator PutDown-Block (im Plan STEP-1) bearbeitet. Man sieht den Causal Link für die Vorbedingung $+On(c,d)$ von FINISH. Außerdem hat der neue Schritt STEP-1 zwei Teilziele hergeleitet, die die Abbildung nicht zeigt.

Abbildung 3.c: Es wurde das offene Ziel $+Holding(c,x_0)$ ausgewählt und wieder unter Erzeugung eines neuen Schrittes (STEP-2) bearbeitet.

Abbildung 3.d: Hier wurden die Vorbedingungen von STEP-2 und die noch offene Vorbedingung von STEP-1 erfüllt, ohne daß ein neuer Schritt erzeugt werden mußte, da START bereits die geforderten Effekte hat. Die Planung für das erste Teilziel der Problembeschreibung ist damit fertig.

Abbildung 3.e: Schließlich sieht man den fertigen Plan. Er besteht aus zwei parallelen Teilplänen (Schritte STEP-2/STEP-1 bzw. STEP-4/STEP-3). Im Laufe der Planung sind Threats aufgetreten, deren Auflösung (durch Separation) nur indirekt dadurch zum Ausdruck kommt, daß bestimmte Variablenbindungen erfolgt sind.

Die Frage, die sich nun im Hinblick auf Benutzerinteraktionen stellt, ist die folgende: Was passiert, wenn der Benutzer beim fertigen Plan aus Abbildung 3.e entscheidet, z.B. die zweiten Planungsentscheidung, die STEP-2 (PickUpBlock) einführt, zurückzunehmen und stattdessen eine andere Alternative (PickUpBlock') zu wählen? Wir gehen hier davon aus, daß die Domänenmodellierung einen dritten Operator PickUpBlock' enthält. Um diese Frage

⁵In der Abbildung stehen Knoten mit abgerundeten Ecken für die Planschritte (START, FINISH, STEP-i), zwischen denen im Laufe der Planung Causal Links aufgebaut werden.

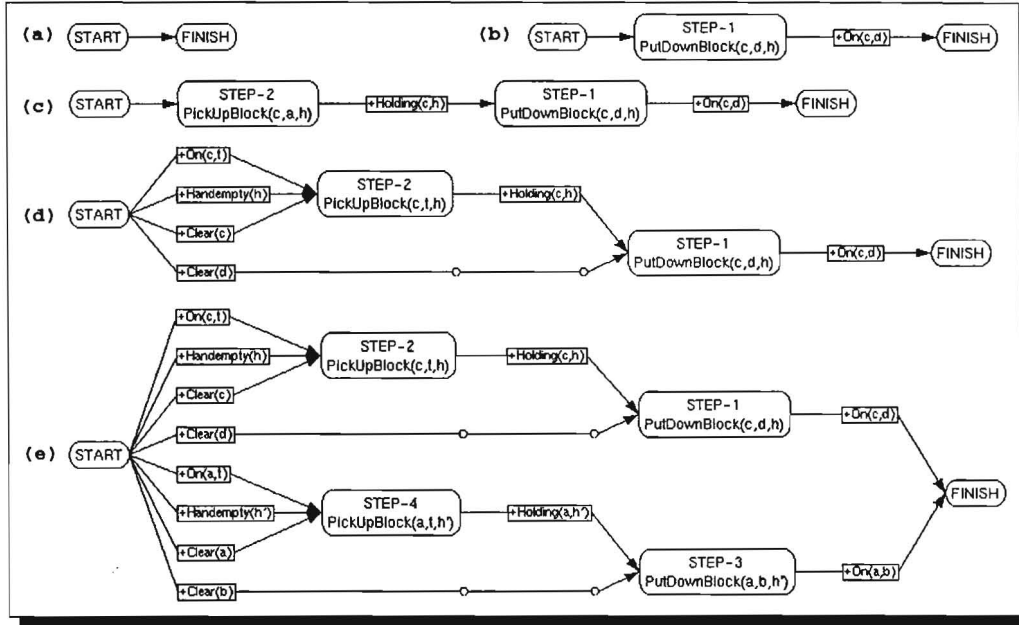


Abbildung 3: Einige erzeugte partielle Pläne zur Problemstellung aus Abbildung 2

zu beantworten, muß man den SNLP-Algorithmus unter dem Aspekt, wie er Backtracking durchführt, etwas genauer betrachten.

4 Repräsentation und Verwaltung von Entscheidungen

4.1 Stack von Causal Links und Threat-Auflösungen bei SNLP

Wie schon in Abschnitt 2.3 erwähnt, führt SNLP chronologisches Backtracking durch, um Inkonsistenzen im Plan aufzulösen. Die hierzu notwendige zeitliche Reihenfolge auf der Menge der Entscheidungen wird mit Hilfe eines Stacks verwaltet.

Bezogen auf das Beispiel aus Abbildung 2 sieht dies wie folgt aus: Nach dem letzten Planungsschritt aus Abbildung 3.e enthält der von SNLP benötigte Stack Einträge wie sie in Abbildung 4 aufgelistet sind. Ganz unten im Stack liegt der für das zuerst bearbeitete Ziel ($+On(c,d)$, $FINISH$) erzeugte Causal Link. Darüber liegen alle anderen erzeugten Causal Links und Threat-Auflösungen.

Aus der Verwendung eines Stack ergibt sich, daß SNLP nur in der Lage ist, zur jeweils letzten Entscheidung zurückzuspringen; daher kann auf die Anforderung, eine beliebige, vorher getroffene Entscheidung d zurückzuziehen nur so reagiert werden, daß alle Entscheidungen d_i , die nach d getroffen wurden, ebenfalls zurückgezogen werden.

Ist im Beispiel etwa die Entscheidung d_2 (vgl. Abb. 4) zurückzuziehen, so werden von SNLP auch die Entscheidungen $d_3, \dots, d_{15}$ zurückgezogen; es entsteht wieder der Plan aus Abbildung 3.b. Leider wird alles revidiert, was zur Erfüllung des Zieles ($+On(a,b)$, $FINISH$) bereits getan wurde ($d_7, d_9, d_{12}, \dots, d_{15}$), obwohl der durch diese Entscheidungen erzeugte Teilplan nach wie vor verwendet werden könnte, da er nicht von d_2 abhängt. Beim Rückzug einer Entscheidung wird der Plan also mehr als notwendig modifiziert, da auch unabhängige Entscheidungen zurückgenommen.

:	
d_{15}	$START \xrightarrow{+Clear(b)} STEP-3$
d_{14}	$START \xrightarrow{+Clear[a]} STEP-4$
d_{13}	$START \xrightarrow{+Handempty[h']} STEP-4$
d_{12}	$START \xrightarrow{+On[a,t]} STEP-4$
d_{11}	$x18 \neq d$
d_{10}	$x18 \neq c$
d_9	$STEP-4 \xrightarrow{+Holding[a,h']} STEP-3$
d_8	$x0 \neq x16$
d_7	$STEP-3 \xrightarrow{+On[a,b]} FINISH$
d_6	$START \xrightarrow{+Clear[d]} STEP-1$
d_5	$START \xrightarrow{+Clear[c]} STEP-2$
d_4	$START \xrightarrow{+Handempty[h]} STEP-2$
d_3	$START \xrightarrow{+On[c,t]} STEP-2$
d_2	$STEP-2 \xrightarrow{+Holding[c,h]} STEP-1$
d_1	$STEP-1 \xrightarrow{+On[c,d]} FINISH$

Abbildung 4: Stack von SNLP zu Abbildung 3.e

Daraus ergibt sich für CAPlan die Forderung, die einfache Verwaltung der abgearbeiteten Ziele in Form eines Stacks durch einen leistungsfähigeren Mechanismus zu ersetzen.

4.2 Graphförmige Repräsentation von Entscheidungen

Es soll nun zunächst davon ausgegangen werden, daß während des Planungsprozesses keine Threats auftreten. Um eine minimale Modifikation des bestehenden Planes zu ermöglichen, werden in CAPlan die getroffenen Entscheidungen in Form eines Graphen verwaltet.

Knoten stehen entweder für noch offene Ziele oder für getroffene Entscheidungen, d.h. für erzeugte Causal Links und (falls notwendig) für neu eingeführte Schritte oder für Auflösungen von Threats. *Kanten* beschreiben Abhängigkeiten zwischen Entscheidungen bzw. offenen Zielen, derart daß eine Entscheidung d_j von d_i abhängig ist, wenn d_j Nachfolger von d_i ist.

Nachfolger eines Entscheidungsknotens entstehen, wenn eine Entscheidung neben einem Causal Link $s_i \xrightarrow{p} s_j$ auch den benötigten Schritt s_i erzeugt. Für die Vorbedingungen dieses Schrittes s_i , die zunächst offene Planungsziele sind, werden Nachfolgerknoten erzeugt. Diese Knoten für offene Ziele werden durch einen Entscheidungsknoten ersetzt, sobald eine Entscheidung getroffen wird, wie das offene Ziel erfüllt wird. Wurde bei der Erzeugung eines Causal Links $s_i \xrightarrow{p} s_j$ der Schritt s_i nicht neu erzeugt, so wird eine Abhängigkeitskante von dem Entscheidungsknoten d_k , der s_i erzeugt hat, zu dem Knoten d_l , der den Causal Link erzeugt, eingefügt, falls d_k nicht bereits Vorgänger von d_l ist. Hierdurch entsteht ein gerichteter azyklischer Graph, der die Abhängigkeiten der Entscheidungen widerspiegelt.

Betrachtet man zunächst etwa den Planzustand aus Abbildung 3.d. wo der Teilplan für das erste Teilziel $(+On(c,d), FINISH)$ fertig ist, so zeigt Abbildung 5 den von CAPlan bis dahin aufgebauten Graph (hier nur ein Baum) von Entscheidungen. Es kommen folgende Aspekte

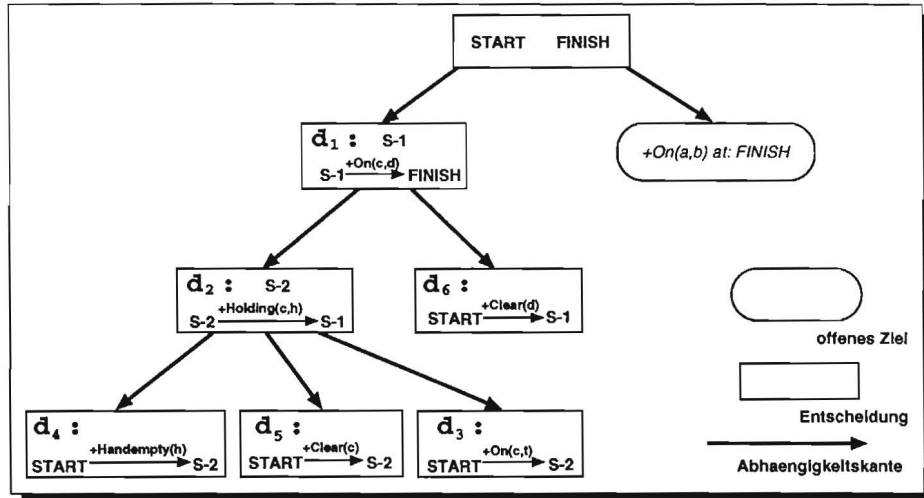


Abbildung 5: Graph von Entscheidungen zum Planzustand aus Abbildung 3.d. Die Namen der Schritte wurde hier abgekürzt, z.B. zu S-1 statt STEP-1.

zum Ausdruck: Die Wurzel steht für die Initialisierung des Systems mit dem Planungsproblem, wodurch die Schritte START und FINISH erzeugt werden. Es gibt zwei initiale Teilziele (Vorbedingungen von FINISH), die zu Nachfolgerknoten der Wurzel werden. Für das noch unbearbeitete Ziel $(+On(a,b), FINISH)$ existiert noch keine Entscheidung, was hier durch einen Knoten mit abgerundeten Ecken angedeutet ist. Bei der Bearbeitung des anderen Teilziels wurde ein neuer Schritt STEP-1 und der Causal Link $STEP-1 \xrightarrow{+On(c,d)} FINISH$ erzeugt, was als Entscheidung d_1 repräsentiert ist. STEP-1 hat außerdem zwei neue Teilziele erzeugt: $(+Holding(c,x0), STEP-1)$, $(+Clear(d), STEP-1)$. Bei den Entscheidungen im Baum, die Blattknoten sind, wurde ein bereits existierender Schritt im Plan (hier START) verwendet. Es brauchte keine zusätzliche Abhängigkeitskante eingefügt werden, da der Wurzelknoten, der START enthält, bereits Vorgänger aller Blattknoten ist.

Wird im Planzustand aus Abbildung 3.d, zu dem Abbildung 5 gehört, wieder die Entscheidung d_2 zurückgenommen wird (es gibt bis dahin die Entscheidungen $d_1, \dots, d_6$), so würde SNLP $d_2, \dots, d_6$ zurücknehmen. CAPlan verwendet den Baum aus Abbildung 5 und revidiert d_2 und alle Nachfolger (d_3, d_4, d_5). CAPlan erkennt, daß d_6 von der Rücknahme von d_2 unabhängig ist und nicht zurückgenommen werden muß.

4.3 Berücksichtigung von Threats

Die Bearbeitung von Threats ist im Algorithmus aus Abschnitt 2.3 ein Backtracking-Punkt und kommt daher im Stack aus Abbildung 4 vor. Genauso muß die Threat-Bearbeitung auch im Graph von Entscheidungen berücksichtigt werden, da bei der Threat-Auflösung Entscheidungen getroffen werden, bestimmte Ordnungen oder Bindungs-Constraints zu erzeugen.

Wird ein Threat erkannt, so wird im Abhängigkeitsgraph ähnlich wie für offene Vorbedingungen von Schritten ein neuer Knoten erzeugt, ein sog. *ProtectionGoal*, welches zunächst nur ausdrückt, daß der ihm zugeordnete Threat möglicherweise aktiv ist und aufgelöst werden muß. Aus der Definition eines Threats ergibt sich, daß ProtectionGoals immer von zwei anderen Entscheidungen abhängen müssen: einerseits von der Entscheidung, die den bedrohten Causal Link erzeugt hat, andererseits von der Entscheidung, die den Schritt erzeugt, der diesen Causal Link bedroht.

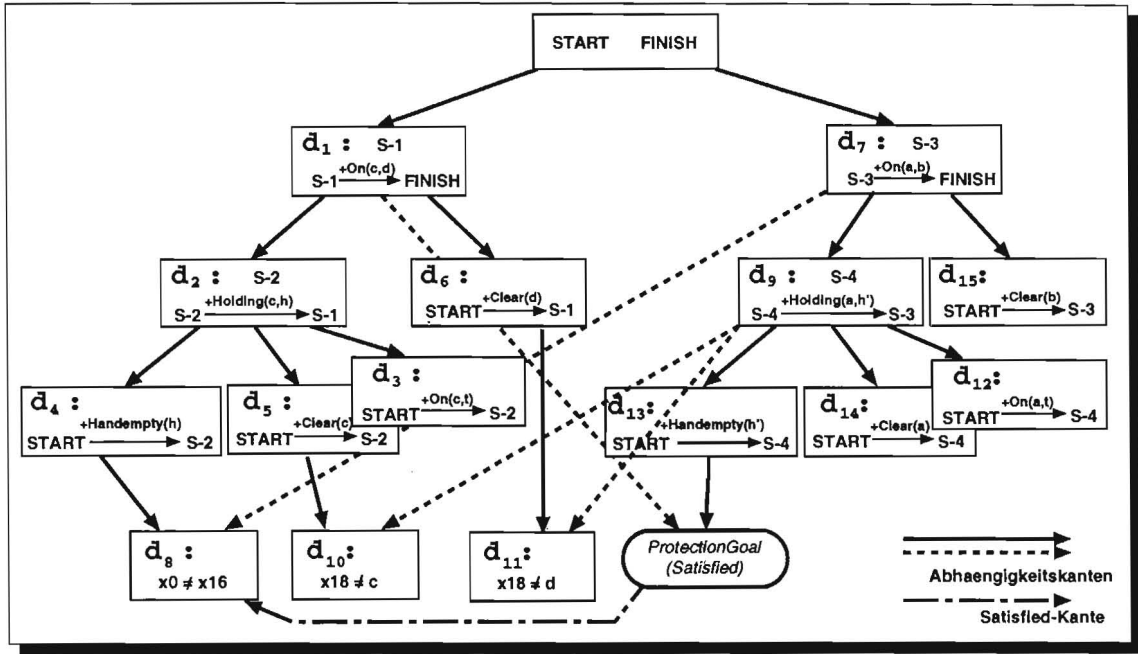


Abbildung 6: Abhängigkeitsgraph der Entscheidungen zum Planzustand aus Abbildung 3.e

Abbildung 6 zeigt diesen Sachverhalt am Beispiel aus Abschnitt 3.2. Der hier gezeigte Graph repräsentiert die Abhängigkeiten aller getroffenen Entscheidungen, wenn das Planungsproblem gelöst ist (vgl. Abb. 3.e). Im Laufe der Planung sind drei Threats aufgetaucht, die bearbeitet wurden. Abbildung 6 zeigt für jeden der Threats, welche Entscheidung getroffen wurde, um ihn aufzulösen (d_8, d_{10}, d_{11}). Betrachtet man z.B. den durch d_8 (Separation $x0 \neq x16$) aufgelösten Threat, so ist dieser Entscheidungsknoten abhängig von d_4 und d_7 , weil durch diese Entscheidungen der bedrohte Causal Link bzw. der bedrohende Schritt (gestrichelte Linie) erzeugt wurde. Die so repräsentierte Abhängigkeit hat zur Folge, daß d_8 automatisch zurückgenommen wird, sobald mindestens eine der Entscheidungen d_4 oder d_7 zurückgenommen wird. Die Rücknahme des Knotens hat zur Folge, daß alle Constraints, die mit der Entscheidung in Plan aufgenommen wurden, ebenfalls zurückgenommen werden. Hierdurch wird sichergestellt, daß der Plan nach einer Benutzerinteraktion nur die minimal notwendige Menge von Constraints enthält (Vollständigkeit der Planung).

Am gleichen Beispiel betrachtet, ergibt sich: wenn die Entscheidung d_2 zurückgenommen wird, so nimmt CAPlan automatisch die abhängigen Entscheidungen d_3, d_4, d_5, d_8 und d_{10} zurück. Im Gegensatz zu SNLP bleiben $d_6, d_7, d_9, d_{11}, \dots, d_{15}$ gültig, d.h. der Plan wird nur soweit modifiziert wie notwendig.

Ein weiterer Aspekt muß im Hinblick auf die Erzeugung von ProtectionGoals bedacht werden. Im Laufe des Planungsprozesses wird immer wieder nach Threats für Causal Links gesucht (vgl. Abschnitt 2.3) und dabei werden neben aktiven auch eine Reihe *potentieller Threats* entdeckt, potentiell deshalb, weil der aktuelle Plan (S, O, B) bereits Ordnungen oder Bindungs-Constraints enthält, die den Threat auflösen. Will man dem Benutzer gestatten, beliebige Entscheidungen wieder zurückzunehmen, so müssen auch diese potentiellen Threats repräsentiert werden, da sie infolge der Rücknahme von anderen Entscheidungen u.U. zu aktiven Threats werden können.

Potentielle Threats werden zunächst genauso wie normale Threats im Abhängigkeitsgraphen durch ein ProtectionGoal eingetragen, jedoch wird zusätzlich vermerkt, aufgrund welcher an-

deren Entscheidungen sie nur potentielle Threats sind. Dies geschieht mittels eines speziellen Typs von Kanten (Satisfied-Kante) zwischen einer Entscheidung und einem ProtectionGoal. Ein ProtectionGoal muß bearbeitet werden, wenn es keine Satisfied-Kante mehr besitzt, die auf eine gültige Entscheidung zeigt, d.h. der zugehörige Threat ist dann aktiv.

Sieht man sich das wieder am Beispiel an (vgl. Abb. 6), so wird die Auswirkung klar. Im zu diesem Graph gehörigen Plan (vgl. Abb. 3.e) gibt es eine ganze Reihe von solchen potentielle Threats, von denen aus Gründen der Übersicht in Abbildung 6 nur einer dargestellt ist. Dieses ProtectionGoal ist als bereits erfüllt (*satisfied*) markiert (Satisfied-Kante zu d_8). Nimmt man aber wie in allen Beispielen zuvor auch hier die Entscheidung d_2 zurück, so wird dadurch zunächst u.a. auch d_8 zurückgenommen (ungültig). Das hat zur Folge, daß dieses ProtectionGoal nun bearbeitet werden muß. Die im Abhängigkeitsgraph gespeicherte Information über potentielle Threats verhindert also, daß nach dem Zurücknehmen einer Entscheidung alle Causal Links nochmals auf noch zu bearbeitende Threats hin überprüft werden müssen, was einen nicht unerheblichen Aufwand bedeuten würde.

Verwaltung des Entscheidungsgraphen

Die Verwaltung des Abhängigkeitsgraphen in CAPlan geschieht aufbauend auf ein TMS-basiertes System namens REDUX [Petrie, 1991], welches einige für die Bereiche Planung und Konfiguration grundlegenden Mechanismen zur Verfügung stellt. Insbesondere ist es in der Lage, Entscheidungen, Abhängigkeiten zwischen Entscheidungen, die durch Operatoranwendung entstehen, und Rückzugsbegründungen von Entscheidungen zu repräsentieren und unterstützt dependency-directed Backtracking.

Für CAPlan mußte REDUX jedoch noch um einige Aspekte erweitert werden. Threats werden durch Ziele repräsentiert und die von REDUX aufgebaute, baumförmige Abhängigkeitsstruktur muß zu einem gerichteten azyklischen Graphen erweitert werden.

5 Diskussion

Das zentrale Ziel, das wir mit der CAPlan-Architektur verfolgen, ist die Unterstützung der interaktiven Lösung eines Planungsproblems. Hierbei können einerseits einfache Teilprobleme automatisch durch das Planungssystem und andererseits komplexe Aufgabenstellungen interaktiv mit dem Benutzer bearbeitet werden. Die Architektur gibt den Benutzerentscheidungen die höchste Priorität, so daß Sytementscheidungen, die der Benutzer als falsch oder suboptimal einschätzt, jederzeit von ihm modifiziert werden können. Durch dieses opportunistische Sytemverhalten können Aufgaben interaktiv gelöst werden, obwohl das zugrundeliegende Domänenmodel unvollständig oder partiell falsch ist. Da Wissen über die Präferenz von Alternativen zu jedem Zeitpunkt interaktiv in den Lösungsprozeß eingebracht werden kann, berücksichtigt diese Architektur die Erfahrung, daß die Expertise der meisten Anwendungsdomänen mit den heutigen Techniken nicht vollständig oder zumindest nicht in einem Schritt modelliert werden kann.

Wir haben gezeigt, daß vor allem die Rücknahme von Entscheidungen bei stackbasierten Planungssystemen mit unnötigen Folgeänderungen verbunden sein kann. Zu ihrer Vermeidung werden Entscheidungen in CAPlan mit Hilfe gerichteter azyklischer Graphen verwaltet, die eine Erweiterung der Teilzielbäume oder der UND/ODER-Bäumen darstellen. Sie unterscheiden sich jedoch wesentlich von UND/ODER-Graphen, da in diesen Graphen eine Teilaufgabe durch unterschiedliche Problemlösungsschritte erreicht werden kann. Teilzielbäume zur Verwaltung von Entscheidungen werden auch in anderen Systemen verwendet, z.B. REDUX

[Petrie, 1991] oder in einem System zur Integration von Aktionsplanung und Konfigurierung [Köhne, 1993]. Im Gegensatz zu CAPlan repräsentieren diese Systeme nur die eigentlichen Planungsziele, nicht jedoch die Ziele, eine Interaktion zwischen Planungsentscheidungen zu beheben. Dem REDUX-System liegt ein anderes Planungsmodell zugrunde, das nur negative Interaktionen berücksichtigt. Sie stellen Inkonsistenzen dar, die durch Backtracking behoben werden, so daß aus diesem Grund eine baumförmige Repräsentation ausreicht.

Einen interessanten Aspekt der REDUX-Architektur stellt die Verwaltung des Zielbaums mit Hilfe eines JTMS dar, wodurch die Aktualisierung der Entscheidungsstruktur auf die Label-Propagierung in einem TMS-Netzwerk zurückgeführt wird. Diese integrierte Abhängigkeitsverwaltung war der ausschlaggebende Grund, daß CAPlan aufbauend auf der modifizierten REDUX-Architektur realisiert wurde.

Literaturverzeichnis

- BARRETT, A. UND WELD, D.S. 1993. *Partial-Order Planning: Evaluating Possible Efficiency Gains*. Technical Report 92-05-01 Expanded Version. Dep. of Computer Science and Engineering, University of Washington, Seattle, WA 98195.
- GINSBERG, M.L. 1993. Dynamic Backtracking. *Journal of Artificial Intelligence Research*, **1**, 25–46.
- HARVEY, W.D., GINSBERG, M.L. UND SMITH, D.E. 1993. Deferring Conflict Resolution Retains Systematicity. *Aus: Proceedings of AAAI-93*.
- KNOBLOCK, C.A UND YANG, Q. 1994. Evaluating the Tradeoffs in Partial-Order Planning Algorithms. *Aus: Proceedings of the Canadian AI Conference*.
- KÖHNE, A. 1993. *Integration von Aktionsplanung und Konfiguration*. Dissertationen zur Künstlichen Intelligenz (DISKI), Universität Karlsruhe.
- KORF, R.E. 1987. Planning as Search: A Quantitative Approach. *Artificial Intelligence*, **33**, 65–88.
- MCALLESTER, D. UND ROSENBLITT, D. 1991. Systematic Nonlinear Planning. *Seiten 634–639 aus: Proceedings of AAAI-91*.
- MUÑOZ, H., PAULOKAT, J. UND WESS, S. 1994. Controlling non-linear hierarchical planning by case replay. *Aus: KEANE, M., HATON, J. UND MANAGO, M. (Hrsg.), Proceedings of the 2nd EWCBR*.
- PEOT, M UND SMITH, D. 1993. Threat-removal strategies for partial-order planning. *Seiten 492–499 aus: Proceeding of AAAI-93*.
- PETRIE, CH. 1991. *Planning and Replanning with Reason Maintenance*. Dissertation, University of Texas/Austin.
- SMITH, D UND PEOT, M. 1993. Postponing threats in partial-order planning. *Seiten 500–506 aus: Proceedings of AAAI-93*.
- TATE, A. 1977. Generating Project Networks. *Seiten 888–893 aus: Proceedings of IJCAI-77*.
- WEBERSKIRCH, F. 1994. *Realisierung eines nichtlinearen Planungssystem zur Unterstützung der Arbeitsplanerstellung bei der computerintegrierten Fertigung (CIM)*. Diplomarbeit, Universität Kaiserslautern.

Controlling a Nonlinear Hierarchical Planner using Case Replay*

Héctor Muñoz, Jürgen Paulokat and Stefan Wess
University of Kaiserslautern, Dept. of Computer Science
P.O. Box 3049, D-67653 Kaiserslautern, Germany
E-mail: {munioz|paulokat|wess}@informatik.uni-kl.de

Abstract

We describe a hybrid case-based reasoning system supporting process planning for machining workpieces. It integrates specialized domain dependent reasoners, a feature-based CAD system and domain independent planning. The overall architecture is build on top of CAPLAN, a partial-order nonlinear planner. To use episodic problem solving knowledge for both optimizing plan execution costs and minimizing search the case-based control component CAPLAN/CBC has been implemented that allows incremental acquisition and reuse of strategical problem solving experience by storing solved problems as cases and reusing them in similar situations. For effective retrieval of cases CAPLAN/CBC combines domain-independent and domain-specific retrieval mechanisms that are based on the hierarchical domain model and problem representation.

1 Introduction

Planning for machining workpieces is a crucial step in the process chain of product development in mechanical engineering because it strongly influences the overall product's costs. However, the domain's complexity currently doesn't allow any complete analytical model so that planning must be done based on experience. The task of process planning consists of selecting and ordering machining operations such that all features of a given workpiece can be manufactured by minimal or, at least, by low costs (Cheung & Dowd, 1988). Mostly, the features of a workpiece cannot be treated independently, because steps of a manufacturing plan possibly interact. Positive interactions can be utilized to decrease the cost of manufacturing a feature, e.g. sharing preparatory steps for several processing steps, or compounding manufacturing of several features in one processing step. While positive interactions should be utilized to minimize overall production costs, negative interactions must be resolved because they lead to inconsistencies. For example a processing step for a feature must not be used if it destroys or makes impossible to manufacture another one.

Theoretically, partial-order nonlinear planning is well suited to support the generation of process plans. Domain-independent conflict resolution techniques that are widely studied can be used for detecting negative interactions between steps. (Conflicts, threats (McAllester & Rosenblitt, 1991; Barrett & Weld, 1993) or clobbering a goal (Chapman, 1987) are used synonymously.) The truth criterion (Chapman, 1987) is an effective way of finding positive interactions. However, in most situations there are many possible treatments of interactions which one has to be chosen from. Choosing, however, is a critical step because it influences future planning and finally the execution costs of the overall plan. Past research in planning offers little support on this problem because it was mainly concerned with search for consistent solutions (e.g. (Chapman, 1987; McAllester & Rosenblitt, 1991; Barrett & Weld, 1993)) and the efficiency of the planning process itself (e.g. (Minton, 1988)). Until now some initial work is done on acquiring and representing expertise and controlling a planner to support optimization of plan execution costs (e.g. (Pérez & Carbonell, 1993; Borrajo & Veloso, 1994)).

A main characteristic of human planning for machining is the use of examples that have found to be a successful in similar situations (Humm, Schulz, Radtke, & Warnecke, 1991). In mechanical engineering,

*THIS PAPER WAS PUBLISHED IN THE PROCEEDINGS OF THE EWCBR-94. This research was partially sponsored by the Deutsche Forschungsgemeinschaft (DFG), Sonderforschungsbereich (SFB) 314: "Künstliche Intelligenz - Wissensbasierte Systeme", Project X9 (1991 - 1995).

there are numerous attempts to build up index structures to support the classification of workpieces and the retrieval of associated manufacturing plans, e.g. (Optiz, 1970). However, these index structures are intended for manual use and only utilize information about the geometry, material of workpieces and the applied technology, e.g. (Zhang, Wright, & Davies, 1988). They completely lack attempts to extract structural information from plans as, e.g., in (VeloSo & Carbonell, 1991) to make retrieval more informed.

In this paper, we describe CAPLAN/CBC the case-based control component of the first-principle planner CAPLAN and its application in the process planning domain. Similar to (Kambhampati, Cutkosky, Tenenbaum, & Lee, 1991), CAPLAN combines specialized reasoners with a general purpose planning approach. In CAPLAN/CBC this hybrid approach is applied to case-based planning by integrating domain-independent and domain-specific methods to organize the case base and to control case retrieval. It extends the ideas of (Paulokat, Praeger, & Wess, 1992), where only domain specific methods are used.

As in PRODIGY/ANALOGY (VeloSo, 1992) and contrary to other well-known case-based planning systems like CHEF (Hammond, 1986) case-based planning in CAPLAN/CBC means controlling the planning process of the first-principle planner by reusing control decisions of a case for solving the current problem.

The paper is organized as follows. In the next section, we describe the characteristics of the process planning domain. In section 3 we summarize the concepts of nonlinear, partial-order planning that influenced the architecture of CAPLAN/CBC. Further, we give a survey of the domain model. Section 4 describes our approach to case-based planning and in the last section a discussion of our approach is done.

2 Domain Characteristics

The domain we are concerned with is manufacturing planning for rotation-symmetrical workpieces to be machined on a lathe (?). A planning problem is given as a geometrical description of a workpiece and of the raw material that only can be cylindrical in our model (Fig. 1). The description of a workpiece is built up from geometrical primitives as cylinders, cones and toroids that describe monotone areas of the outline, possibly augmented by features as threads, grooves or special surface conditions. In most cases, the outline of a workpiece cannot be machined in one step, but repeated cutting operations are necessary to cut the difference between the raw material and the workpiece in thin layers. These layers are built up from atomic processing areas, that are automatically generated by extending the horizontal and vertical bounding lines of the geometrical primitives. Cutting an atomic area can be seen as an elementary cutting step.

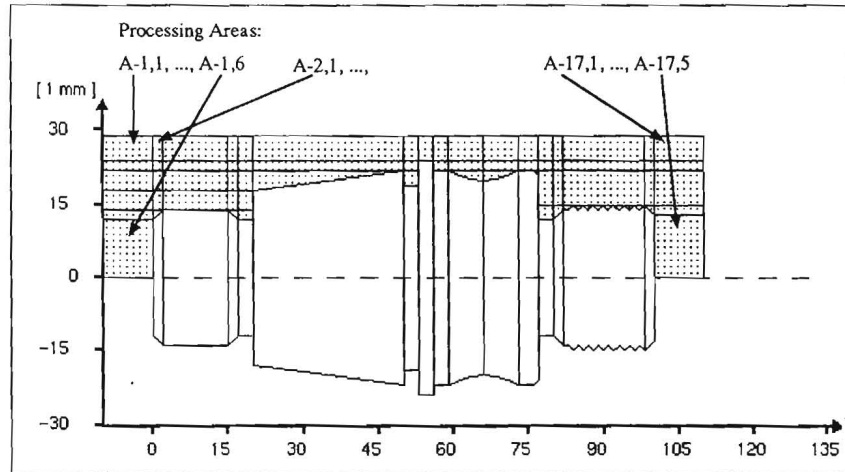


Figure 1: Atomic processing areas of a Workpiece

For machining the workpiece is clamped by a rotating clamping tool, while layers of material are removed by moving a cutting tool along the surface. Standard tools that are normally used for machining large areas have a fixed working direction, i.e., they can be used to cut off material only when being moved either to the left or to the right. Clamping a workpiece hides parts of its surface, therefore, after machining the

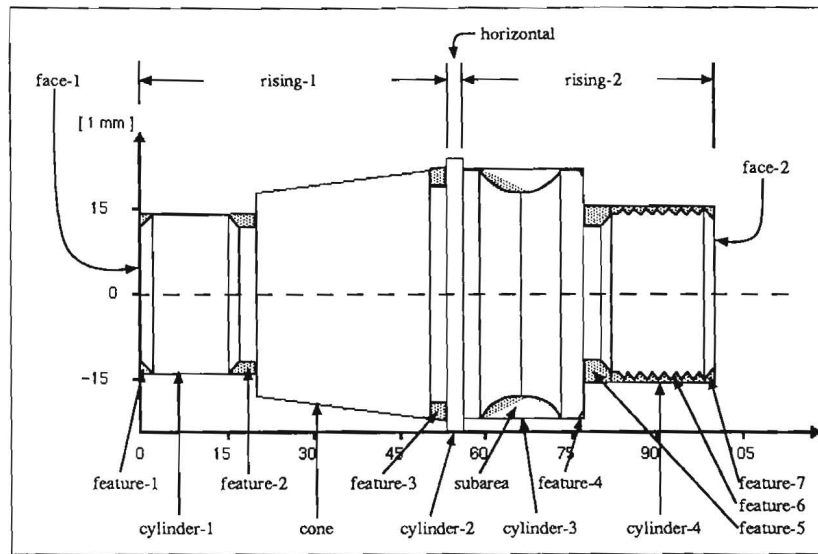


Figure 2: Complex processing areas of a workpiece

part not hidden, it must be turned and clamped on the other side. Additionally, caused by the geometry of standard cutting tools only horizontal outlines or outlines that are rising along a tool's moving direction can be machined. To machine a descending outline requires tool changes which increase the overall production costs. But after turning a workpiece and clamping it from the other side, e.g. necessary for machining an area hidden by a clamping tool, a descending outline is a rising one and can be machined with the same tool. For finding maximally monotonously rising areas, geometrical primitives are grouped to rising areas at both ends of the workpiece and a horizontal area between them (Fig. 2). Each of these compound areas can contain subareas that break the monotonous course of the outline. But from an abstract point of view they do not influence the construction because the monotonous outline necessarily has to be machined first before hidden subareas can be machined. For machining, the horizontal part can be added consistently to one of both rising areas, but choosing an alternative is one of the tasks of the planner because this choice can influence the plan's execution costs. Compound areas can be degenerated to consist of only one geometrical primitive, e.g. the horizontal area in Fig. 2.

The compound areas can be seen as an abstract description of the workpiece resulting in a hierarchical representation whose root node represents the whole workpiece with the compound areas as successors (Fig. 3). The successors of the compound areas are the geometrical primitives and the subareas of the workpiece. A geometrical primitive has successor nodes for its features and the atomic processing areas that are located above it. Subareas, again, can be hierarchically structured, defining their own subtrees. This hierarchical representation of the workpiece is the base for a hierarchically structured planning process.

3 Planning in CAPLAN

On an abstract level a planning problem is given by an initial situation, a set of goals, and a set of possible steps. The task of planning is computing a sequence of steps so that their execution starting from the initial situation results in a situation satisfying all goals. Planning in CAPLAN is based on the ideas of systematical non-linear planning (SNLP (McAllester & Rosenblitt, 1991)) and works on a set of partially ordered steps. A new planning problem is represented by two steps s_0 and s_∞ . The effects of step s_0 are the features that are valid in the initial situation and the preconditions of s_∞ represent the features that are the goals of the planning problem. Step s_0 has no preconditions and is ordered before all other steps of the plan. Step s_∞ has no effects and is ordered behind all other steps. A goal can be satisfied by every step which is not ordered behind the precondition's step and which has an effect that can be matched with the goal's feature. This can be a step already being contained in the plan or newly added to the plan. Both, the step that adds the effect and the step the precondition of which is satisfied, are connected by a causal

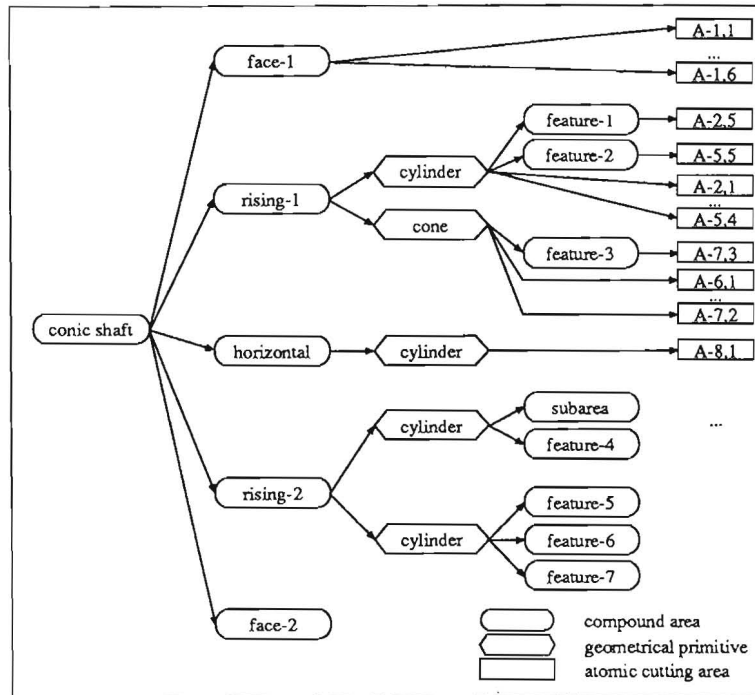


Figure 3: Hierarchical representation of a planning problem.

link that is annotated by the feature. A causal link is threatened by a step if it adds an effect or the negation of an effect that can be unified with the feature of the causal link. Threats must be resolved which can be done by ordering the step that threatens the causal link before the step that adds the effect or behind the step whose precondition has been satisfied or by adding constraints that make impossible the unification of effects.

Planning for machining a workpiece is preceded by transforming the set of compound areas and the set of features of the hierarchical description of the workpiece (Fig. 3) in a problem representation suitable for the planner (Fig. 4). Names, as `rising-1` used in the problem description, provide a link to the geometrical representation. They can be used to access further information not explicitly represented in the problem description but which are necessary for control decisions. Fig. 5 shows a part of the plan on the abstract planning level of the hierarchical planning process. First, the planner decided to work on goal `machined(rising-1)`. Therefore, the abstract operator `machine(rising-1)` is selected, which introduces subgoals to clamp the workpiece and to force the area `rising-1` to be free. Then, planning continues with the goal `machined(horizontal)`. Here, the planner can choose between introducing a new step to machine the horizontal component isolated from any other compound component or to utilize the side effects of another plan step. In this example the side effect of step `machine(rising-1)` is used. This is supported by the rule `combine(horizontal)` that introduces a goal that a connected area is machined. (Note: these rules are different from control rules; they are comparable to operators but do not represent actions and are not part of the resulting process plan.) The new goal `machined(?Area-1)` can be matched to one of the goals of the initial problem description and can be satisfied by a step that is already part of the plan. Using steps already existing is called a phantomization. As different matches are possible, this is a decision point that can strongly influence the execution costs of the plan.

By this phantomization, a decision is made that the compound areas `rising-1` and `horizontal` are processed by one step. This influences the expansion of the abstract plan step `machine(rising-1)` into a new planning problem on the concrete planning level (Fig. 6). Depending on this decision, the new planning problem is to select and to order cutting operations for the atomic processing areas of the compound areas `rising-1` and `horizontal`. Note that changing the abstraction level means changing the representation language, e.g., the complex processing area `rising-1` is replaced by the atomic processing areas `A-2.1`, ..., `A-7.2`.

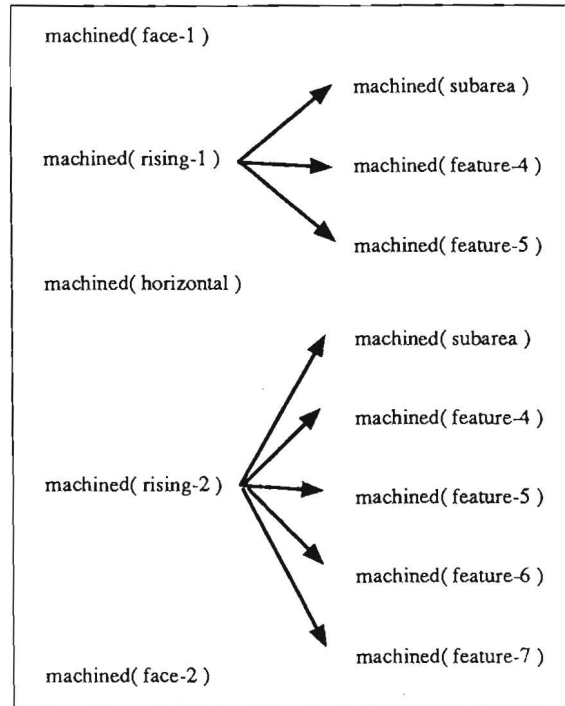


Figure 4: Set of initial goals and their ordering relations.

4 Case-Based Planning

Three characteristics make manufacturing planning a hard problem. 1) The production cost of a workpiece should be minimized. Optimization of plan execution costs, however, is not supported by AI planning techniques as described in the last section. Nevertheless, they are attractive because they allow an explicit representation of plan steps and reasoning about their interactions and effects. 2) The search space for a manufacturing plan is very large which makes impossible to generate all alternative solutions and select the best of them. And 3), there is no analytical model of the dependencies between the features of a workpiece, the possible manufacturing steps and the overall execution costs that could be used for choosing between alternatives during the planning process. Instead, human manufacturing planning is based on experience.

If no control knowledge is available, planning in CAPLAN is done by depth-first search or guided by user interactions. For support of planning in complex domains, the planning process can be controlled by a case, e.g., as in (Veloso, 1992). If a case is available the reuse of a decision that has shown to be successful in a similar situation is preferred to search when the planner has to choose at a choice point. If the rationals of a reused decision are satisfied there is a justification that it reduces backtracking and results in a better solution. In CAPLAN there are two kinds of choice points:

- the set of alternative steps to reach a goal, and
- the set of alternative constraints that can be added to a plan to resolve a threatened causal link (McAllester & Rosenblitt, 1991; Barrett & Weld, 1993).

An example for a choice point where the planner has to choose to reach a goal is `clamped(wp)` in Figure 5. There, the conflict set consists of the steps `clamp-with-jaw(?Jaw)` and `clamp-with-cl-class(?Cl-class)`.

In CAPLAN the reuse of episodic problem solving experience is done by CAPLAN/CBC. It organizes the case memory and supports case-based planning by retrieving the most similar case and by stepwise reusing the decisions stored in the case. After a problem has been successfully solved and if its solution is substantially different from that of the case it can be added to the case base. An important source for new problem solving experience are user interactions. Storing them as a new case adds control knowledge

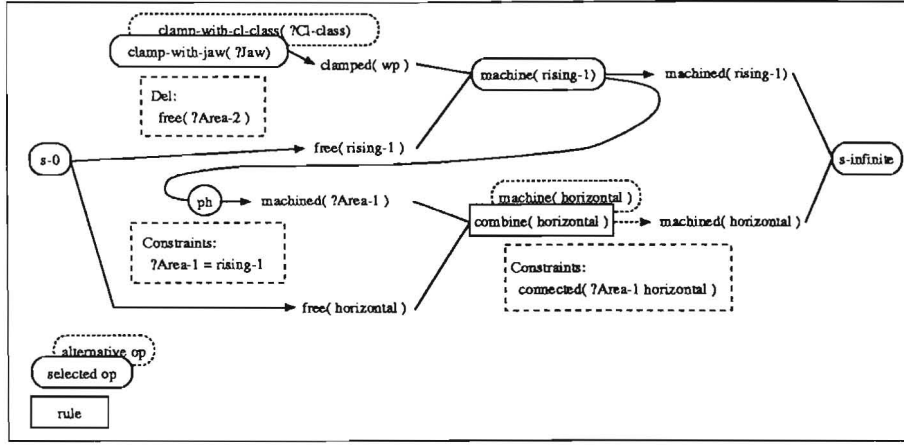


Figure 5: Partial plan on the abstract planning level.

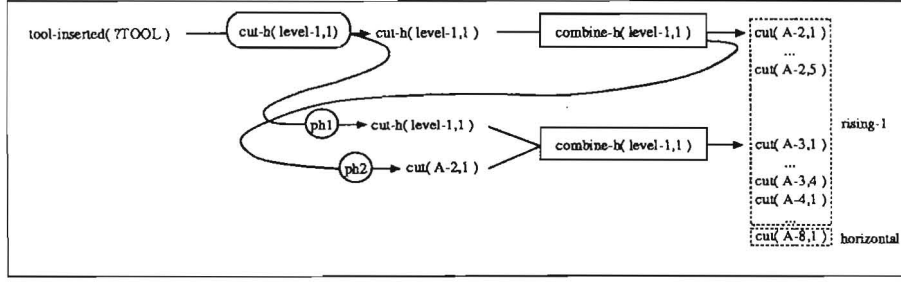


Figure 6: Partial plan on the concrete planning level for the abstract operator `machine(rising-1)` of Fig. 5.

to the system and increases its planning expertise because knowledge used in these interactions possibly complements the insufficiencies of the model. Adding a case to the case base is preceded by an analyzing step that extracts the relevant features from the problem description which are used for the organization of the case base and for efficient case retrieval.

4.1 Case Storage

The organization of the case memory highly influences the efficiency and possibly the result of case retrieval. In (Veloso, 1992) a domain independent architecture for automatic case storage has been described that is based on the assumptions that the set of goals of different problem descriptions are highly varying and that the solution of a problem can be decomposed into independent subplans. These assumptions are not valid in our domain. Machining plans for the considered class of workpieces are highly sequential so that such a decomposition is impossible. Further, the goals of planning problems are always similar, i.e., we always have to plan processing steps for one or two rising areas, a horizontal area and for the faces of the workpieces. Optionally, there can be a varying number of workpiece features. On the other hand, there are technological constraints that can be used to classify workpieces and, more important, that are reflected by the manufacturing plan, e.g., the kind of usable clamps depends on the ratio of the diameter of a workpiece to its length.

In our approach, these constraints are compiled into an initial, domain-specific structure of the case memory that spawns a decision tree. To add a new case to the case base its relevant category is determined by traversing the decision tree starting from its root to a leaf node. So, every new case is stored as a successor of a leaf node of the domain-specific decision tree which structures the case base. Further, the addition of a new case is preceded by footprinting (Veloso, 1992) the set of features of the initial state as a function of the goal statement and of the particular solution encountered. This process identifies the relevant features

of the initial situation with respect to a plan and filters out those features that are not relevant for the plan.

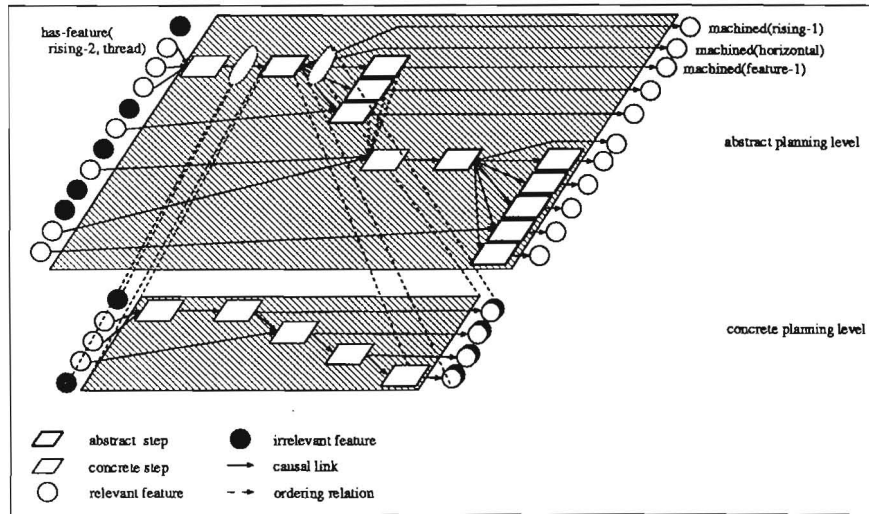


Figure 7: Interactions from a nonlinear hierarchical plan

Figure 7 shows the structure of a machining plan for the example of section 2. The footprint is independently computed for the abstract and for every concrete plan. In contrast to the approach in (Veloso, 1992) we don't use the set of goals to index a case, but the hierarchical problem representation described in Fig. 3. It compounds the set of goals and all features of the initial situation. This structure is annotated by the information on the relevance of a feature.

4.2 Case Retrieval

The purpose of the retrieval phase is to select a case from the case base that is useful to solve a new planning problem. A good criterion for judging the utility of cases is to consider the modification costs of the corresponding solutions. Unfortunately it is usually impossible to determine this measure without processing the modification (Paulokat et al., 1992; Smyth & Keane, 1993). Domain-dependent heuristics and domain-independent structural methods (like footprinting (Veloso & Carbonell, 1991)) have been developed to overcome this problem. CAPLAN/CBC combines both approaches by performing its retrieval phase in two steps:

1. A domain-dependent pre-selection using physical constraints and domain knowledge to obtain a small subset of cases, discarding most of the cases in the case base.
2. A domain-independent selection step using a footprint-like similarity measure (Veloso & Carbonell, 1991). Which computes structural information about previous generated plans and known problem descriptions (cf. section 4.1).

Given a CAD representation of a new workpiece (Fig. 1) the atomic and the complex processing areas (Fig. 2) are determined. In the next step the hierarchical representation of the planning problem (Fig. 3) and the set of initial goals and their ordering relations (Fig. 4) are computed. The initial goals and ordering relations represent a set of domain dependent technical constraints which must be satisfied by any case, e.g. the groove represented by *feature-2* in Fig. 2 can only be processed after the processing areas of *cylinder-1* and of the complex processing area *rising-1* have been processed.

In the current implementation, case retrieval in CAPLAN/CBC is made by comparing the hierarchical description (Fig. 3) of a new problem with the problem description of stored cases (cf. Déjà Vu (Smyth & Cunningham, 1992)). Using the tree-based problem representation the case match is made by a tree

matching algorithm (Tanaka & Tanaka, 1988). The similarity measure is computed as weighted sum of all adding and deleting operations necessary for transforming the hierarchical problem description of the case into the description of the current problem. Every transformation step is associated with domain-specific costs which are comparable to the effort that is necessary for the modification of the corresponding solutions (Humm et al., 1991).

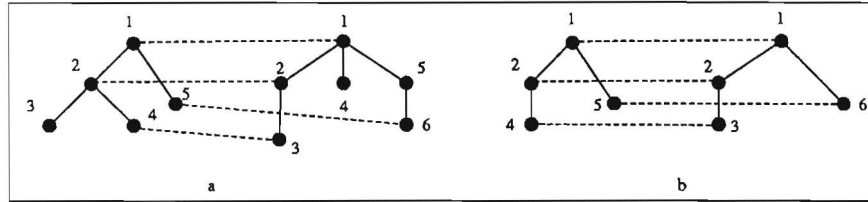


Figure 8: Mapping representations using a structural matching algorithm (a) and generating a reusable solution by deleting the respective nodes (b)

The distance between the case and the problem description is measured as follows: First, a *structure preserving match* (Tanaka & Tanaka, 1988) between the hierarchical structures of the workpieces is computed (Fig. 8a). Second, the difference between the two representations is calculated, i.e. the nodes that were not matched and the nodes that were matched, but that are of different type (for example a geometrical primitive of type torroid is matched with one of type cylinder) are counted (Fig. 8b). During this calculation different domain specific weights according to the type of the nodes are processed, since it is easier to replace an atomic processing area than a complex processing area e.g. counting a node which represents a primitive area has an smaller value than counting one which represents a complex processing area. A case qualifies for solving a new problem if the computed transformation costs are smaller than a given threshold.

To determine the relevance of features (Janetzko, Wess, & Melis, 1993) we compare the problem description with the annotated problem description (cf. section 4.1) of the case to determine in detail how many relevant features they have in common. This step corresponds to footprinting in PRODIGY/ANALOGY (Velooso & Carbonell, 1991). Concretely, the groups of features in the annotated problem description of the case that are completely included in the description of the current problem are determined. The relevance of these features is defined as the number of features that are entirely included in the problem description.

4.3 Case Replay

Although CAPLAN can autonomously solve problems by depth-first search, it is mainly intended as a planning assistant that provides a control interface for a human planner or external control components, such as CAPLAN/CBC. The input for the case-based control component consists of the problem description of a new planning problem, the case selected (cf. section 4.2) and an association between goals of the new planning problem and initial goals of the plan canned by the case. The associations are computed by tree matching of the retrieval phase. For solving a new problem, the decisions and their rationals of a case are used to choose from the set of alternative planning steps to satisfy a goal or to choose from the set of alternative constraints that can be added to resolve a threat of a causal link. This replaying of a decision of the case is done in four steps. First, the planner computes the set of alternative plan steps or conflict solving constraints that can be applied to the goal or to the conflict currently being worked on. In the second step a match of the alternative chosen by the decision of the case with the possible alternatives of the new problem is made. If there is a successful match, the decision's rationals are verified in the context of the new problem. Third, if the rationals are satisfied the step is chosen and the decision for choosing this alternative is provided with the verified rationals of the case. And fourth, the new subgoals resulting from the step chosen are matched against the subgoals resulting from the replayed decision. By every successful match a new association between a goal of the new planning problem and a goal of the case is generated. As soon as CAPLAN/CBC continues replaying for one of the prematched subgoals the associations to the case are already available, enabling the system to repeat the four steps described above.

Planning for goals that cannot be matched to a goal of the case are delayed until the replay process is completed. The remaining goals are solved by first-principle planning or possibly by user interactions.

Because replaying a case and planning by first-principles results in the same representation of a plan, all planning decisions made during case replay can be rejected if they don't allow a completion of a partial plan. The separation of the replay phase from solving unmatched goals is possible because of the partially ordered plan representation and the nonlinear planning paradigm. The first point is in contrast to PRODIGY/ANALOGY where a total-order planner is used which makes necessary interleaving of case replay and first-principle planning for unmatched goals.

5 Conclusions

We have described the hybrid planning architecture CAPLAN and its application in a process planning domain. A main characteristic of this domain is that planning is driven by minimizing plan execution costs. Human planning experts in this domain use a large number of heuristics and domain specific reasoning. Since this knowledge is extremely case sensitive its usage is not sufficiently supported by known planning techniques.

In our approach a general purpose planning system, CAPLAN, is combined with a case-driven control unit CAPLAN/CBC which allows to use previous experience in form of recorded cases to guide the problem solving process. If there exists a case which is useful to solve the current problem, its decisions are replayed (Veloso, 1992) in the current context, i.e., a planning decision of the case is only reused if its rationals are compatible with the problem description of the new problem. Thus, our approach is very similar to the work of Veloso (Veloso, 1992), but there are some important differences. First, planning in CAPLAN is based on the ideas of partial-order planning (SNLP (McAllester & Rosenblitt, 1991)), so the underlying planning architecture is different. Second, retrieval of useful cases in CAPLAN is realized as a combination of domain-independent and domain-dependent techniques which constrain the number of cases that have to be inspected during the retrieval phase. The annotated problem description is an extension of the concepts of footprinting (Veloso & Carbonell, 1991) and of interacting goals in PRODIGY. This extension is necessary due to the hierarchical representation of plans in CAPLAN. Third, the overall architecture of the system combines specialized domain dependent reasoners and a feature-based CAD system with general purpose planning (Kambhampati et al., 1991).

Contrary to other well-known case-based planning systems such as CHEF (Hammond, 1986), CAPLAN uses (like PRODIGY) an explicit domain model to describe the plan steps and the constraints that must be satisfied during the retrieval and adaptation phase. The use of domain specific knowledge and techniques for the retrieval of useful cases by calculating a *measure of adaptability* in CAPLAN is similar to the *adaptation guided retrieval* approach in Déjà Vu (Smyth & Keane, 1993).

References

- Barrett, A., & Weld, D. S. (1993). Partial-order planning. *Artificial Intelligence*, 67.
- Borrajo, D., & Veloso, M. (1994). Incremental learning of control knowledge for nonlinear problem solving. In *Proceedings of the European Conference on Machine Learning*.
- Chapman, D. (1987). Planning for Conjunctive Goals. *Artificial Intelligence*, 32, 333–377.
- Cheung, Y., & Dowd, A. L. (1988). Artificial intelligence in process planning. *Computer Aided Engineering*, 5(4), 153–156.
- Hammond, K. J. (1986). *Case-Based Planning: An Integrated Theory of Planning, Learning and Memory*. Ph.D. thesis, Yale University, New Haven, Connecticut.
- Humm, B., Schulz, C., Radtke, M., & Warnecke, G. (1991). A System for Case-Based Process Planning. In *Proceedings of the 1st CIRP Workshop on Learning in Intelligent Manufacturing Systems (IMS)*. CIRP. Budapest, Hungary.
- Janetzko, D., Wess, S., & Melis, E. (1993). Goal-Driven Similarity Assessment. In Ohlbach, H.-J. (Ed.), *GWAI-92: Advances in Artificial Intelligence*, pp. 283–298. Springer Verlag.

- Kambhampati, S., Cutkosky, M., Tenenbaum, M., & Lee, S. H. (1991). Combining specialized reasoners and general purpose planners: A case study. In *Proceedings of AAAI-91* Menlo Park, California. MIT Press.
- McAllester, D., & Rosenblitt, D. (1991). Systematic nonlinear planning. In *Proceedings of AAAI-91* Menlo Park, California. MIT Press.
- Minton, S. (1988). *Learning Search Control Knowledge — An Explanation-Based Approach*. Kluwer Academic Publishers.
- Optiz, H. (1970). *A Classification System to Describe Work Pieces*. Pergamon Press, Elmsford, N.Y.
- Paulokat, J., Praeger, R., & Wess, S. (1992). CABPLAN – fallbasierte Arbeitsplanung. In Messer, T., & Winklhofer, A. (Eds.), *Beiträge zum 6. Workshop Planen und Konfigurieren*, No. 166 in FR-1992-001, p. 169 Germany. Forwiss.
- Pérez, M. A., & Carbonell, J. (1993). Automated acquisition of control knowledge to improve the quality of plans. Tech. rep. CMU-CS-93-142, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213.
- Smyth, B., & Cunningham, P. (1992). Deja vu: A hierarchical case-based reasoning system for software design. In Neumann, B. (Ed.), *ECAI-92*, pp. 587–589.
- Smyth, B., & Keane, M. T. (1993). Retrieving adaptable cases: The role of adaptation knowledge in case retrieval. In Richter, M., Wess, S., Althoff, K., & Maurer, F. (Eds.), *Proceedings First European Workshop on Case-Based Reasoning*, pp. 76–82.
- Tanaka, E., & Tanaka, K. (1988). The tree-to-tree editing problem. *International Journal of Pattern Recognition and Artificial Intelligence*, 2.
- Veloso, M., & Carbonell, J. (1991). Variable-Precision Case-Retrieval in Analogical Problem Solving. In Bareiss, R. (Ed.), *Proceedings: Case-Based Reasoning Workshop*, pp. 93–106. Morgan Kaufmann Publishers.
- Veloso, M. M. (1992). *Learning by Analogical Reasoning in General Problem Solving*. CMU-CS-92-174, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213.
- Zhang, K. F., Wright, A. J., & Davies, B. J. (1988). A feature-recognition knowledge base for process planning of rotational mechanical components. *International Journal of Adv. Manufacturing Technology*, 4, 13–25.

PuK in der Theorie

Akquisition von Kontrollwissen zur Zielauswahl für Causal Link Planer

Reinhard Praeger, Vera Kettnaker, Frank Weberskirch
Fachbereich Informatik
Universität Kaiserslautern
67653 Kaiserslautern
e-mail: {praeger | kettnake | weberski}@informatik.uni-kl.de

Zusammenfassung

Causal Link Planer arbeiten auf partiell geordneten Plänen. Sie erlauben eine leichte Planrestrukturierung, so daß die Reihenfolge, in der Teilziele bearbeitet werden, für das Planungsergebnis nicht mehr relevant ist. Unter dem Performanzaspekt betrachtet, spielt die Zielreihenfolge dennoch eine wichtige Rolle. Aufgrund einer Analyse der Planungsdomäne lassen sich Zielinteraktionen vorhersehen und entsprechende Präferenzrelationen für die Zielauswahl definieren. Die vorliegende Arbeit stellt die Kontrollkomponente CbA (Control by Anticipation) vor, die solche Präferenzrelationen aufstellt und zur Steuerung der Zielauswahl im Causal Link Planer CAPlan benutzt.

1 Einführung

CAPlan [Weberskirch 1994] ist ein domänenunabhängiger Planer, der auf dem Algorithmus zur *systematischen, nichtlinearen Planung*, SNLP [McAllester, Rosenblitt 1991] basiert. Zur Lösung eines Planungsproblems sucht CAPlan in einem Raum von partiell geordneten Plänen (*PO-Planer*). [Barrett, Weld 1994] zeigen die Vorteile dieser Planer gegenüber Planungsansätzen, die mit total geordneten Plänen arbeiten (*TO-Planer*). Bei SNLP-Planern ist die Zielauswahl kein Backtrackingpunkt mehr. Es entsteht leicht der Eindruck, die Zielauswahl spiele hier eine untergeordnete Rolle. Es hat sich jedoch herausgestellt, daß die Zielauswahl die Planungsperformanz ganz wesentlich beeinflußt. Wir werden dies an einem einfachen Beispiel verdeutlichen. Es wird schnell klar, daß für Anwendungen in realen Domänen eine geschickte Kontrolle der Ziel- und Operatorauswahl notwendig ist.

Auf dem Gebiet der TO-Planer gibt es seit dem STRIPS-Ansatz bereits eine Tradition in der Entwicklung von Kontrollmechanismen für die Planung. Das Spektrum der eingesetzten Methoden zur Performanzsteigerung reicht von der Generierung von Makro-Operatoren [Korf 1985], dem Planen auf verschiedenen Abstraktionsniveaus [Bergmann 1993],[Knoblock 1994], der Extraktion von Kontrollregeln aufgrund einer Analyse des Plantrace mit EBL-Verfahren [Minton 1988], bis hin zu fallbasierten Ansätzen [Veloso 1992] zur Planungssteuerung.

Die im Bereich der TO-Planer entstandenen Kontrollverfahren lassen sich nicht ohne weiteres auf die Causal Link Planer übertragen. Der Grund hierfür liegt einerseits in den unterschiedlichen Suchräumen, und andererseits in den verschiedenen Arten von Entscheidungen, die zu treffen sind. Bei

TO-Planern werden die Entscheidungen in Abhängigkeit vom aktuell gültigen Zustand getroffen und die Zustandsänderungen mitprotokolliert. Causal Link Planer hingegen arbeiten auf einem partiell geordneten Plan, der durch Einführen neuer Planschritte, Constraints und Ordnungen zwischen Planschritten verfeinert wird. Die Planung erfolgt nach einer least-commitment Strategie, wonach der Plan nur partiell festgelegt und kein konkreter Weltzustand berechnet wird. Da außer im Startzustand während der Planung kein konkreter Zustand vorhanden ist, fehlt für die Kontrollmechanismen der TO-Planer die Situationsbeschreibung, auf die Bezug genommen wird. Eine Berechnung der gültigen Formeln während der Planung nach dem *Modal Truth Criterion (MTC)* wie es Chapman für den PO-Planer TWEAK [Chapman 1987] vornimmt, wäre für einen Causal Link Planer zu aufwendig.

Die in diesem Beitrag vorgestellten Kontrollmechanismen konzentrieren sich auf die Zielauswahl zur Steuerung des Causal Link Planers CAPlan. Sie sind Bestandteil der Kontrollkomponente CbA (Control by Anticipation). Die Idee dabei ist, durch ein analytisches Verfahren Zusammenhänge in der zugrundeliegenden Domäne bereits vor der eigentlichen Planung zu erkennen und das gewonnene Wissen zur Steuerung des Planers zu benutzen. Für die Analyse ist Hintergrundwissen erforderlich. Eine vergleichbare Idee wird in STATIC [Etzioni 1991] für den TO-Planer PRODIGY [Minton 88] verfolgt. Die durch STATIC gewonnenen Kontrollregeln formulieren jedoch Bedingungen an den Planungszustand, so daß sich diese ohne zusätzlichen Aufwand nicht durch einen Causal Link Planer verwerten lassen. CAPlan/CbA analysiert die Spezifikation der Domäne und erzeugt Präferenzordnungen auf den Zielen, die unabhängig von einem Planungszustand sind.

Neben CAPlan/CbA wird u. a. ein fallbasierter Ansatz in CAPlan/CbC [Muñoz, Paulokat, Wess 1994] zur Steuerung des Planers verfolgt. Beide Ansätze greifen auf unterschiedliche Wissensquellen zurück und sollen einander ergänzen.

2 Der Planer

CAPlan arbeitet nach dem SNLP-Algorithmus, der eine systematische Suche in einem Raum partiell geordneter Pläne durchführt [McAllester, Rosenblitt 1991]. Ein *partieller Plan* ist ein Tripel (S, O, B) mit:

- S ist eine Menge von *Planschritten*.
- O ist eine Menge von *Ordnungen* auf den Planschritten in S .
- B ist eine Menge von *Constraints* auf den Variablen, die in den Planschritten auftreten.

Planschritte werden durch *STRIPS-Operatoren* beschrieben, wobei die Vorbedingungen unterschieden werden in *Subgoals* und *Phantoms* und die Effekte eines Operators in *Purpose* und *Side Effects*. Phantoms sind Vorbedingungen, die ohne Anwendung eines Operators erreicht werden müssen. Subgoals entsprechen den üblichen Preconditions bei STRIPS-Operatoren. Purpose eines Operators sind Effekte, die der Benutzer mit der Spezifikation des Operators beabsichtigt, andere Effekte heißen Side Effects. Diese Unterscheidung bei der Operatorspezifikation wird vom Planer bei der Operatorauswahl genutzt und unterstützt eine effizientere Planung. Eine Operatordefinition ohne Purpose und Phantoms entspricht der üblichen Operatordefinition in STRIPS mit *Precondition*-, *Add*- und *Delete*-Listen.

Die Menge S enthält zwei ausgezeichnete Planschritte: *START* und *FINISH*. *START* hat als Effekte die initiale Situation und der Schritt *FINISH* besitzt die Zielsituation als Vorbedingung. Die Menge O beschreibt eine partielle Ordnung auf den Planschritten in S . B bezeichnet eine Menge von Constraints auf den Variablen, die durch die Planschritte eingeführt wurden. Der Planungsalgorithmus verwaltet eine Menge G von *Planungszielen*, die es zu erfüllen gilt. Planungsziele sind Paare (P, s) ,

wobei P eine Vorbedingung des Schritts s ist. Als Planungsziele treten zunächst die durch den Schritt *FINISH* eingeführten Ziele auf. Im Verlauf der Planung werden Planungsziele durch den Effekt eines neu in den Plan eingeführten Schritts oder den Effekt eines bereits vorhandenen Planschritts erfüllt. Die Vorbedingungen eines neu eingeführten Schritts s bilden wiederum Planungsziele, die in die Menge G aufgenommen werden. Wird ein Planungsziel (P,s) durch den Effekt eines Schrittes s' erfüllt, so erzeugt der Planer den *Causal Link* $s' \xrightarrow{P} s$. Anschließend wird das Paar (P,s) aus der Menge G gelöscht. Der Planungsalgorithmus verwaltet Causal Links in einer Menge L ; jeder dieser Links dokumentiert, weshalb ein Schritt in den Plan eingeführt wurde. Das Erfüllen von Planungszielen ist immer mit dem Erzeugen eines entsprechenden Causal Links verbunden.

Ein Causal Link garantiert, daß die Vorbedingung P zwischen den Schritten s' und s gültig ist. Wird ein neuer Schritt t in den Plan eingeführt, so ist zu prüfen, ob der neue Schritt eine Bedrohung (Threat) für den Causal Link darstellt. Ein *Threat* für den Causal Link $s' \xrightarrow{P} s$ liegt vor, wenn t einen Effekt besitzt, der P negiert (*negative Interaktion*) oder P einführt (*positive Interaktion*) und t parallel zu den Schritten s' und s in partiellen Plan liegt (d. h. zwischen s , s' und t bestehen noch keine Ordnungsbeziehungen). Durch die Berücksichtigung von positiven Interaktionen wird in der Threat-Definition die Systematik der Suche garantiert. Eine *systematische Suche* in Raum partiell geordneter Pläne besagt, daß jeder Planungszustand nur einmal erzeugt wird. Wird durch den Planer ein Threat eines Causal Links $s' \xrightarrow{P} s$ durch einen Schritt t entdeckt, so erzeugt der Planer ein Protectionziel $(t, s' \xrightarrow{P} s)$. Ein Protectionziel ist ein Paar bestehend aus einem Causal Link und dem Schritt, von dem die Bedrohung des Causal Links ausgeht. Ein Protectionziel kann entweder durch eine Verschärfung der Partialordnung auf den Plan aufgelöst werden oder durch Einführen von Constraints auf den Variablen des bedrohenden Schritts:

- Die *Separation*-Strategie führt Constraints auf den Variablen des bedrohenden Schrittes ein, so daß der Threat vermieden wird.
- Die *Promotion*-Strategie ordnet den bedrohenden Schritt hinter den Schritt, auf den der Causal Link zeigt.
- Die *Demotion*-Strategie ordnet den bedrohenden Schritt vor den Schritt, der den Causal Link einführt.

Während der Planung werden sukzessiv Planungsziele bearbeitet, Causal Links eingeführt und Bedrohungen (Protectionziele) aufgelöst. Die Planung terminiert, wenn alle Planungsziele bearbeitet sind und keine Bedrohungen von Causal Links auftreten, d.h. alle Protectionziele erfüllt sind. Planungsentscheidungen und deren Abhängigkeiten werden über einen TMS-Mechanismus organisiert. Ein *Dependency-directed Backtracking* bei Auftreten einer Inkonsistenz und die interaktive Planerstellung durch den Benutzer wird hierdurch unterstützt [Weberskirch 1994]. Zur interaktiven Planerstellung wird ein mächtiges Benutzerinterface zur Verfügung gestellt.

3 Problematik der Planungskontrolle

Zur Steuerung der Suche sieht CAPlan Kontrollkomponenten vor, mit deren Hilfe verschiedene Kontrollstrategien realisiert werden können (vgl. [Weberskirch 1994]). Kontrollkomponenten haben die Aufgabe aus einer Menge von Entscheidungsalternativen, die der Planungsalgorithmus erzeugt hat, eine Auswahl zu treffen. Generell lassen sich folgende *Entscheidungspunkte identifizieren*:

- Zielauswahl
Welches Planungsziel soll als nächstes bearbeitet werden?
Welches Protectionziel ist als nächstes zu behandeln?

- **Operatorauswahl**
Welcher Planungsschritt (Domänenoperator) soll gewählt werden?
Welcher Phantomisierungsoperator (d.h. ein schon vorhandener Schritt) soll genutzt werden?
Welcher Protectionoperator (Promotion, Demotion, Separation) soll verwendet werden?

Bei der Ziel- und Operatorauswahl müssen allerdings zuvor folgende Meta-Entscheidungen geklärt werden:

- **Auswahl des Zieltyps**
Wenn ein Threat aufgetreten ist, wann soll dieser aufgelöst werden? Protectionziele können sofort, d. h. sobald sie aufgetreten sind, oder verzögert bearbeitet werden. Damit verbunden ist gleichzeitig die Frage, in welcher Reihenfolge Protection- und Planungsziele abgearbeitet werden sollen. Hier ist eine Vielzahl von Varianten denkbar [Peot, Smith 1993].
- **Auswahl des Operatortyps**
Soll ein Planungsziel durch Einführung eines neuen Schritts oder durch einen bereits vorhandenen Schritt gelöst werden?

Tabelle1 faßt Entscheidungen, die eine Kontrollkomponente treffen kann, zusammen:

	Entscheidung	Meta-Entscheidung
Zielauswahl	Welches Planungsziel? Welches Protectionziel?	Planungsziel oder Protectionziel?
Operatorauswahl	Welcher Domänenoperator? Welcher Phantomisierungsoperator? Welcher Protectionoperator?	Domänenoperator oder Phantomisierungsoperator?

TABELLE 1. Kontrollentscheidungen in CAPlan

Die Entscheidungsstruktur in CAPlan ist grundsätzlich verschieden von der in TO-Planern. Über die Konzepte der Causal Links und Threats kommt zur Ziel- und Operatorauswahl noch eine Entscheidungsebene hinzu. Zu jedem Zeitpunkt wird stets der komplette Plan von initialer Situation bis zur Zielsituation mit allen Abhängigkeiten betrachtet. Es wird kein isolierter Zwischenzustand berechnet, der Abhängigkeiten im Plan in einer Situation einfriert. Die Planung erfolgt nach einer least-commitment Strategie, bei der die Variablenbelegungen und Ordnungsbeziehungen der Operatoren im Verlauf der Planung schrittweise eingeschränkt werden. Die Default-Kontrollkomponente in CAPlan arbeitet wie folgt:

- Die Auflösung von Threats erfolgt sofort, um die Suche auf den Raum der konsistenten Pläne zu beschränken. (Diese Strategie sieht wird auch in SNLP verfolgt.)
- Zur Auflösung von Threats wird die Separation-Strategie bevorzugt. Es handelt sich hier um eine least-commitment Strategie, weil die Einführung eines Constraints auf den Variablen im Vergleich zu einem Ordnungsconstraint unkritisch ist.
- Bevor ein Planungsziel durch einen neuen Planschritt erfüllt wird, versucht der Planer das Ziel durch einen im Plan vorhandenen Schritt zu erreichen (*Phantomisierung*), anstatt durch einen neuen Planschritt auch neue Planungsziele einzuführen.

Mit der letzten Heuristik versucht der Planer durch Ausnutzen von Seiteneffekten möglichst wenig neue Planungsziele zu erzeugen und somit schnell zu terminieren. Es findet also gleichzeitig eine Planoptimierung in bezug auf die Anzahl der benötigten Schritte statt.

Leider gibt es bereits sehr einfache Beispiele, in denen diese Strategie zu falschen Entscheidungen führt, die wiederum einen enormen Suchaufwand für den Planer zur Folge haben können. Wir gehen von der Problemdefinition aus, wie sie Abbildung 1 zugrunde liegt (siehe Problem Browser oben rechts):

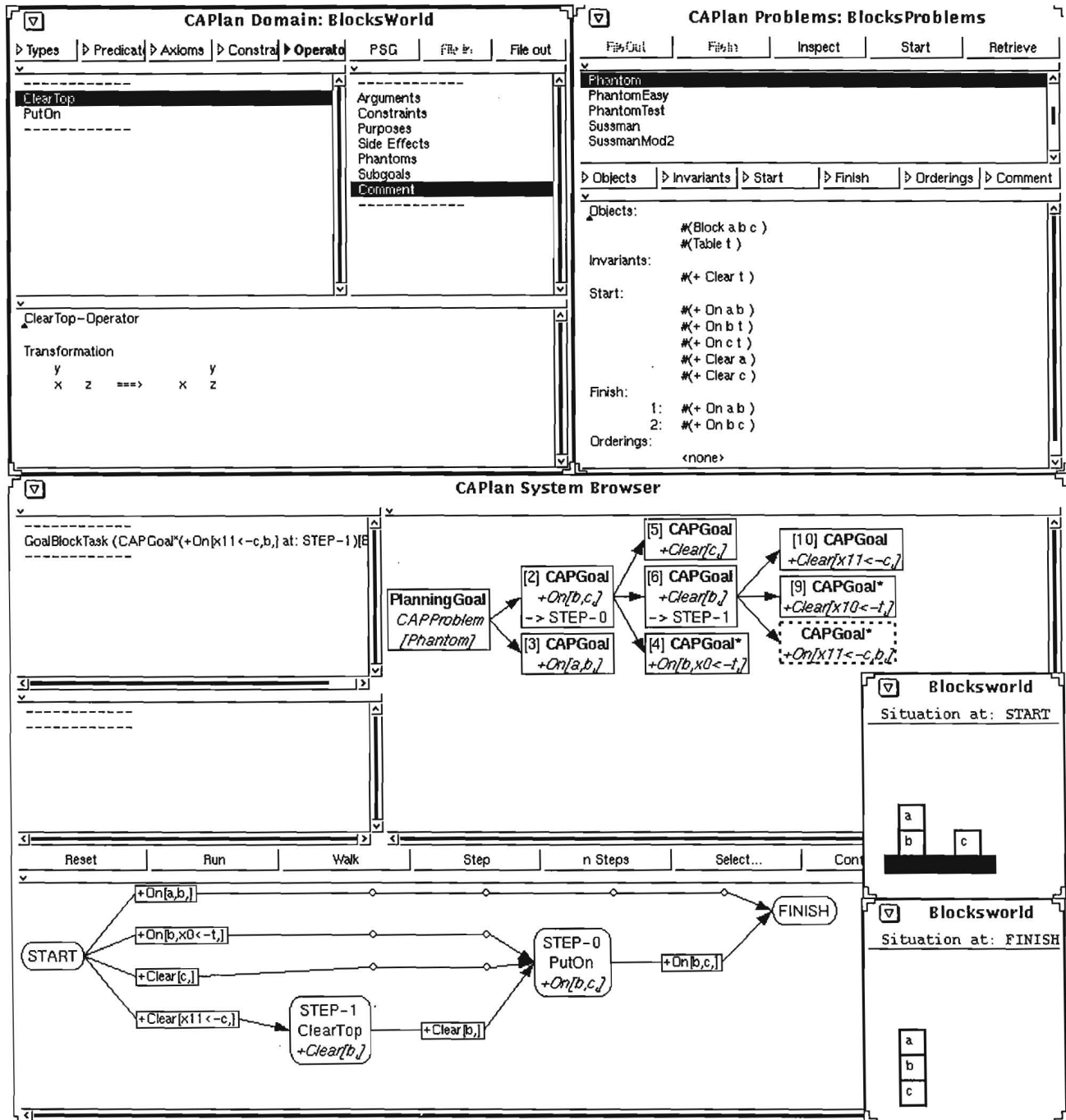


ABBILDUNG 1. Beispiel aus der Blocksworld Domäne mit 2 Operatoren

Ausgangspunkt für das Beispiel ist eine Modellierung der Blocksworld, die die beiden Operatoren *ClearTop* und *PutOn* zur Verfügung stellt. Das Problem für den Planer besteht darin, zu erkennen,

daß das in der Startsituation bereits erfüllte Teilziel $On(a,b)$ wieder zerstört werden muß, um einen Zielzustand zu erreichen. Entscheidet sich der Planer aber schon zu Beginn der Planung für das Ziel $On(a,b)$ und phantomisiert es mit der Startsituation (d.h. der Causal Link $START \rightarrow On(a,b) \rightarrow FINISH$ wird eingeführt) so wird für das 2. Teilziel $On(b,c)$ zunächst der komplette Suchraum generiert, bis der Planer schließlich den Causal Link wieder zurücknimmt und das Ziel $On(a,b)$ durch den Planschritt $PutOn(a,b)$ herstellt. Der CAPlan System Browser in Abbildung 1 zeigt den Planer bei dem Versuch, das Teilziel $On(b,c)$ zu erreichen.

Arbeitet der Planer dagegen zuerst an dem Teilziel $On(b,c)$, so werden durch den partiellen Plan für $On(b,c)$ die Möglichkeiten zur Erreichung des Ziels $On(a,b)$ stark eingeschränkt. Die Entscheidung, $On(a,b)$ mit der Startsituation zu phantomisieren, wird direkt verworfen und der Operator $PutOn(a,b)$ gewählt. Der erzeugte Suchraum ist wesentlich kleiner als beim 1. Lösungsversuch.

Anhand des Beispiels wird deutlich, daß die Heuristik, möglichst sparsam mit der Einführung von Planschritten umzugehen, nicht in jedem Fall zu guten Planungsergebnissen führt. Die Heuristik verallgemeinert in solchen Fällen zu stark. Es stellt sich die Frage, wie sich in einer konkreten Situation bestimmen läßt, ob es besser ist einen neuen Planschritt einzuführen oder den Effekt eines bereits vorhanden Schritts zu nutzen.

Die Idee, die in CAPlan/CbA realisiert wird, läßt sich folgendermaßen skizzieren: Um eine Entscheidung zwischen den beiden Alternativen neuer Planschritt oder Phantomisierung treffen zu können, ist zusätzliches Wissen über deren Auswirkungen und Abhängigkeiten zu anderen Schritten nötig. Solche Abhängigkeiten werden während der Planung durch CAPlan erkannt. Will man umfangreiches Backtracking aufgrund dieser Interaktionen jedoch vermeiden, so ist entscheidend, daß diese Informationen rechtzeitig vorliegen, d. h. zum Zeitpunkt der Entscheidung über die Bearbeitungsreihenfolge der Teilziele. Gelingt es, notwendige Zielinteraktionen vorher zu erkennen und die Ziele in einer hierdurch festgelegten Reihenfolge zu bearbeiten, so lassen sich voreilige Phantomisierungen schneller aufdecken. Aufwendiges Backtracking wird hierdurch vermieden.

Diese Überlegungen haben zunächst zu folgender Default-Strategie in CAPlan geführt: Ist ein Planungsproblem gegeben, so werden zuerst die Teilziele bearbeitet, die nur durch Einführen eines neuen Planschritts erreicht werden können. Die Bearbeitung aller Ziele, die durch Phantomisierung erfüllbar sind, wird zurückgestellt. Läßt man den Planer auf dem generierten Plangerüst weiterarbeiten, so findet er bereits in vielen Fällen deutlich schneller eine Lösung (Testbeispiele aus der Blocksworld Domäne) bzw. der Planer ist in der Lage Probleme zu lösen, die er zuvor in dem gegebenen Zeithorizont nicht lösen konnte.

Im folgenden Abschnitt wird die Analyse der Domäne beschrieben, wie sie in CAPlan/CbA erfolgt. Hintergrundwissen für die Analyse sind Domänenaxiome und die Operatorschemata. Diese Informationen dienen zum Aufbau von *Problem Space Graphs (PSGs)* wie sie Etzioni [Etzioni 1991] vorgeschlagen hat. Die Analyse entdeckt notwendige Zielkonflikte in der Problembeschreibung und legt eine Präferenzordnung (Partialordnung) auf den Teilzielen fest.

4 Analyse notwendiger Zielkonflikte

Es lassen sich zwei Arten von Zielkonflikten zwischen zwei Zielen G und P identifizieren:

- *Goal Clobbering*: Ein Goal Clobbering zwischen den Zielen G und P liegt vor, wenn der Plan, der das Ziel G erfüllt, als Seiteneffekt das geschützte Ziel P zerstört.

- **Precondition Violation:** Eine Precondition Violation zwischen den Zielen G und P liegt vor, wenn der Plan, der das Ziel G erfüllt, das Ziel P unerfüllbar macht, d.h. eine Vorbedingung, die für P nötig ist, irreparabel zerstört wird.

Ein *Goal Clobbering* zwischen den Zielen G und P tritt *notwendigerweise* auf, wenn alle Pläne, die das Ziel G erfüllen, gleichzeitig das Ziel P zerstören. Soll beispielsweise in der Blocksworld das Ziel $On(a,b)$ AND $On(b,c)$ erreicht werden, so wird jeder Plan, der $On(b,c)$ erfüllt, das bereits erfüllte Ziel $On(a,b)$ wieder zerstören. Für die Zielauswahl folgt daraus eine Präferenz des Teilziels $On(b,c)$ vor $On(a,b)$.

Eine *Precondition Violation* zwischen den Zielen G und P liegt *notwendigerweise* vor, wenn alle Pläne, die das Ziel G erfüllen, eine für P benötigte Vorbedingung zerstören. Um die Precondition Violation zu vermeiden, wird das Ziel P vor G bearbeitet.

Um ein Goal Clobbering zwischen zwei Zielen G und P zu erkennen, müssen die *notwendigen Effekte*, die ein Plan für das Ziel G verursacht, berechnet werden. Zerstört einer der Effekte das Ziel P , liegt ein Goal Clobbering vor. Für eine Precondition Violation müssen neben diesen Effekten zusätzlich die notwendigen Vorbedingungen für das Ziel P berechnet werden. Negiert ein Effekt eine notwendige Vorbedingung, tritt eine Precondition Violation auf.

Die Analyse notwendiger Effekte und notwendiger Vorbedingungen für ein Ziel erfolgt durch *Problem Space Graphen (PSG)* [Etzioni 1991]. PSGs sind grundsätzlich unabhängig von einer bestimmten Startsituation. Für jedes mögliche Teilziel einer Domäne (Literale, die als Effekte in den Operatordefinitionen auftreten) wird ein PSG aufgebaut. Ein PSG ist ein AND/OR Graph, der zu einem Zielliteral einen Teil des Lösungsraums angibt. Das Zielliteral ist die Wurzel des PSG. Als Knoten in einem PSG treten entlang eines Pfades abwechselnd Teilziele und Operatoren auf. Vom Zielliteral ausgehende OR-Verzweigungen geben Operatoralternativen an, die das Ziel erfüllen können. Von den Operatoren ausgehende AND-Verzweigungen fassen die Vorbedingungen eines Operators zusammen. Die PSGs sind keine Bäume, weil gemeinsame Vorbedingungen der Operatoralternativen nur einmal repräsentiert, jedoch mehrfach darauf verwiesen werden kann. Abbildung 2 gibt ein Beispiel für einen PSG an. Ein Algorithmus zur Konstruktion der PSGs ist in [Etzioni 1991] angegeben.

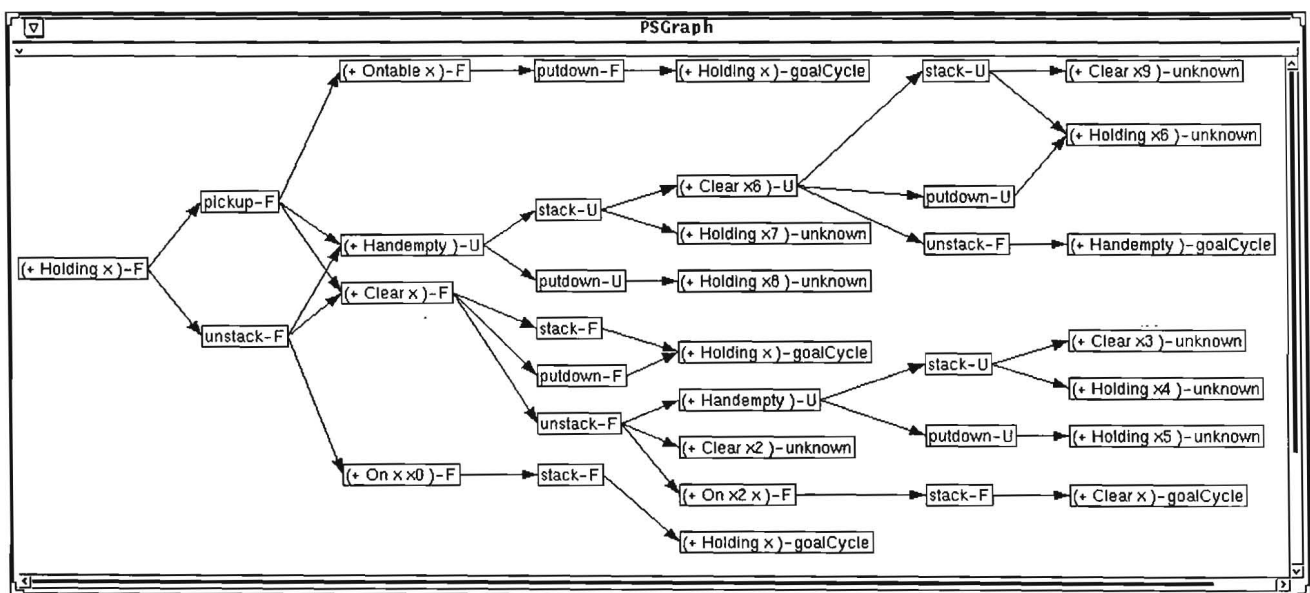


ABBILDUNG 2. Beispiel für ein PSG aus der Blocksworld Domäne mit 4 Operatoren

Um zu garantieren, daß ein PSG endlich bleibt, werden beim Aufbau eines PSG verschiedenen Terminierungskriterien überprüft. Das wichtigste ist der Test auf Teilzielrekursion (Goal Cycle). Tritt ein Teilziel in einem PSG auf dem Pfad zur Wurzel zum zweiten Mal auf, so wird die Expansion an dieser Stelle eingestellt. Das Teilziel wird zu einem Blatt im PSG und erhält die Markierung *goal-cycle*. Kann dagegen ein Teilziel eine Teilzielrekursion unter einer bestimmten Variablenbelegung verursachen, so wird das Teilziel zu einem Blatt und mit *unknown* markiert. Axiome, die auf den Relationen in der Domäne gelten, drücken weitere Terminierungsbedingungen aus. Beispielsweise gilt in der Blockworld für einen Klotz *a*: *Holding(a) OR Ontable(a) OR On(a, x)*, wobei *x* ein beliebiger Klotz sei. Ein Blatt in einem PSG, das mit *holds* markiert ist, bedeutet, daß aufgrund eines Axioms in einem Pfad des PSG alle möglichen Teilziele bereits vorhanden sind. Ein Blatt mit der Markierung *unachievable*, drückt aus, daß in der Domäne kein Operator existiert, der das Teilziel erreicht.

Um eine Präferenz zwischen zwei Zielen zu berechnen, werden zu beiden Zielen die PSGs betrachtet und notwendige Effekte bzw. notwendige Vorbedingungen extrahiert. Die PSGs können aufgrund ihrer Konstruktion (Terminierungsbedingungen) hierfür nur eine Annäherung geben, d.h. es kann nur eine Teilmenge berechnet werden. Die Pfade in einem PSG für ein Zielliteral geben die verschiedenen Möglichkeiten an, das Ziel zu erreichen. Effekte, die notwendigerweise mit dem Erreichen eines Ziels auftreten, sind gerade solche, die Operatoranwendungen entlang eines Pfades mit allen anderen Pfaden des PSG gemeinsam haben. Für das Ziel *Holding(x)*, dessen PSG in Abbildung 2 dargestellt ist, ergeben sich als notwendige Effekte *-Handempty* und *-Clear(x)*. Analog geht man bei der Berechnung notwendiger Vorbedingungen für ein Ziel vor. Zu den beiden Zielen *On(x,y)* und *On(y,z)* läßt sich die Präferenzordnung $On(y,z) < On(x,y)$ feststellen, denn *On(x,y)* hat als notwendigen Seiteneffekt *-Clear(y)* und *On(y,z)* besitzt die notwendige Vorbedingung *+Clear(y)*. Diese Ordnung vermeidet den Konflikt bezüglich der Relation *Clear(y)*.

Ein Vorteil der PSGs ist die Unabhängigkeit von einer konkreten Problembeschreibung sind. Sie beziehen sich lediglich auf die Operatordefinition und können generiert werden, bevor überhaupt eine Problemstellung gegeben ist. Die Analyse von Zielkonflikten bezieht sich stets auf Zielpaare. Für die gewonnene Präferenzordnung auf den Zielen muß anschließend die transitive Hülle berechnet werden.

5 Erkennen von Synergieeffekten

Neben den beschriebenen Zielkonflikten (Goal Clobbering und Precondition Violation) können zwischen zwei Zielen auch Synergieeffekte auftreten. Sie ergeben sich dadurch, daß die Bearbeitung eines Ziels Seiteneffekte zur Folge hat, die für das andere Ziel genutzt werden können. Das Ausnutzen von Seiteneffekten erspart den Einsatz von Domänenoperatoren, die wiederum neue Teilziele einführen würden. Das Planungsverfahren in CAPlan ist auf eine Berücksichtigung von Seiteneffekten ausgelegt, letztlich sogar darauf angewiesen (Terminierung). Das Erkennen von Synergieeffekten zwischen zwei Zielen *G* und *P* funktioniert ähnlich der Analyse von Zielkonflikten durch die PSGs.

Hierzu werden die Effekte betrachtet, die das Erreichen des Ziels *G* notwendigerweise zur Folge hat (siehe Abschnitt 4). Anschließend wird untersucht, ob Vorbedingungen im PSG des Ziels *P* durch die notwendigen Effekte von *G* erfüllt werden. Trifft dies zu, so wird beim gegenwärtigen Stand der Implementierung eine entsprechende Präferenzordnung zwischen *G* und *P* eingeführt, um zu garantieren, daß der zuvor erkannte Synergieeffekt während der Planung genutzt werden kann. Dabei geht allerdings die Information verloren, welcher Operator den Synergieeffekt herbeiführt. Künftige Arbeiten, die vor allen die Steuerung der Operatorauswahl zum Thema haben, sollen diesem Aspekt

Rechnung tragen und die Information nicht nur zur Zielauswahl, sondern auch zur Operatorauswahl nutzen.

Zwischen den Präferenzen, die aus der Analyse von Zielkonflikten gewonnen und denen, die aus Synergieeffekten abgeleitet werden, besteht ein qualitativer Unterschied. Bei den Präferenzrelationen aufgrund von Zielkonflikten handelt es sich um *harte* Vorgaben, die einen Zielkonflikt bei der Planung vermeiden. Nichtbeachtung würde notwendigerweise zu dem Zielkonflikt führen, der dann durch den Planer aufzulösen wäre. Bei den Präferenzen aufgrund von Synergieeffekten handelt es sich um *schwache* Vorgaben, die dafür sorgen, daß Entscheidungsalternativen dem Planer frühzeitig bereitstehen. Für die Operatorwahl hat das Ausnutzen von Synergieeffekten mehr heuristischen Charakter, weil nicht garantiert ist, daß der Synergieeffekt notwendig ist, um eine Lösung zu finden. Das Wissen um Synergieeffekte bei der Operatorauswahl trägt jedoch letztlich zu einer Optimierung der Pläne in bezug auf die Anzahl der benötigten Schritte bei.

6 Verwandte Arbeiten

Im Bereich der Kontrolle von TO-Planern hat die vorliegende Arbeit engen Bezug zu den Arbeiten im PRODIGY Projekt. Insbesondere zu STATIC [Etzioni 1990], das PSGs benutzt, um Zielkonflikte zu erkennen, und zu PRODIGY/EBL [Minton 1988], das Zielkonflikte durch einen EBL-Ansatz analysiert. Beide Arbeiten erzeugen vergleichbare Kontrollregeln, wobei sich STATIC ausschließlich auf die Domänenbeschreibung stützt und PRODIGY/EBL zusätzlich Beispiele benutzt. Die Kontrollregeln enthalten dabei Bedingungen, die sich auf einen konkreten Weltzustand beziehen. Das Ziel dieser Arbeit ist, solche Techniken auch für PO-Planer, genauer für Causal Link Planer, zur Verfügung zu stellen. Die Entscheidungsstruktur dieser Planer ist komplizierter, weil bei Ziel- und Operatorauswahl durch die Causal Links zusätzliche Aspekte berücksichtigt werden müssen. Der Hauptunterschied bei der Anwendung dieser Techniken in CAPlan liegt in dem Fehlen eines konkreten Weltzustands, der die bisher geleistete Planungsarbeit zusammenfaßt und als Bezugspunkt für Kontrollmechanismen dient. Kontrolle in CAPlan/CbA beschränkt sich bisher auf die Auswahl von Zielen. Hierzu werden die Ziele durch eine Präferenzordnung partiell geordnet. Ein MTC-Konzept wie es für TWEAK [Chapman 1987] realisiert wurde, um zu prüfen, welche Formeln zu einem Zeitpunkt im Plan gelten, wurde nicht benötigt. In STATIC werden neben Präferenzregeln auf den Zielen auch Entscheidungsregeln für die Operatorwahl aus den PSGs gewonnen. Dabei wird wiederum Bezug auf den jeweiligen Weltzustand genommen.

In [Katukam, Kambhampati 1994] wird ein EBL-Ansatz für den SNLP-Planer vorgestellt. Der Ansatz analysiert Planungsfehler, Überschreitungen der Tiefenbeschränkung, generiert Erklärungen dafür und erzeugt aus jeder Erklärung eine Kontrollregel. Hierzu werden Axiome über die Domäne herangezogen. Die Axiome sind vergleichbar zu denen, die beim Aufbau der PSGs eingesetzt werden.

In [Smith, Peot 1993] wird ein Ansatz beschrieben, der *Operator Graphs* für SNLP benutzt, um zu analysieren, welche Threats bis zum Ende der Planung aufgeschoben werden können. Die Operator Graphs werden bezüglich einer Problemstellung generiert und haben große Ähnlichkeit zu den PSGs.

7 Zusammenfassung und Ausblick

Wir haben eine Kontrollkomponente beschrieben, welche die Zielauswahl im domänenunabhängigen Planungssystem CAPlan steuert. Die Kontrolle basiert auf einer Analyse der Domänenbeschrei-

bung, mit der Absicht, Zielinteraktionen bereits im Vorfeld der Planung zu erkennen und dieses Wissen während der Planung bei der Zielauswahl einzusetzen. Durch die Analyse werden Zielpräferenzen erzeugt, die unabhängig von einem konkreten Zustand sind, und sich deshalb gut zur Kontrolle von Causal Link Planern eignen. Die Präferenzordnung auf den Teilzielen ermöglicht zum einen das Ausnutzen von Seiteneffekten und vermeidet zum anderen Backtracking aufgrund von Zielkonflikten.

Wichtig ist auch die Steuerung der Operatorauswahl. Für eine Auswahl unter den Domänenoperatoren können wieder die PSGs herangezogen werden. Sie enthalten Informationen darüber, unter welchen Bedingungen eine Operatoralternative nicht geeignet ist. Dies realisiert eine Art Look-ahead Funktion für die Operatorwahl.

Literatur

- [Barrett, Weld 1994] A. Barrett, D. Weld. Partial-order planning: evaluating possible efficiency gains. *Artificial Intelligence*, 67, No.1, 1994.
- [Bergmann 1993] R. Bergmann. Learning hierarchically clustered shared plan abstractions as problem solving knowledge with high utility for planning. in A. Horz, (hrsg.), *Beiträge zum 7. Workshop Planen und Konfigurieren*. Arbeitspapiere der GMD 723, St. Augustin, 1993
- [Chapman 1987] D. Chapman. Planning for Conjunctive Goals. *Artificial Intelligence*, 32: 1987.
- [Etzioni 1990] O. Etzioni. *A structured theory of explanation-based learning*. Dissertation, Carnegie Mellon University, CMU-CS-90-185, Pittsburgh, 1990.
- [Etzioni 1991] O. Etzioni. STATIC - A problem-space compiler for PRODIGY. In: *Proc. of AAAI-91*, 533-540, 1991.
- [Katukam, Kambhampati 1994] S. Katukam, S. Kambhampati. Learning explanation-based control rules for partial order planning. In: *Proc. of AAAI-94*, 582-587, 1994.
- [Knoblock 1994] C. A. Knoblock. Automatically generating abstractions for planning. *Artificial Intelligence* 68:243-302, 1994.
- [Korf 1985] R. E. Korf. Macro-Operators: A weak method for learning. *Artificial Intelligence* 32, 1987.
- [McAllester, Rosenblitt 1991] D. McAllester, D. Rosenblitt. Systematic nonlinear planning. In: *Proc. of AAAI-91*, 634-639, 1991.
- [Minton 1988] S. Minton. *Learning search control knowledge: An explanation-based approach*. Kluwer Academic Publishers, Boston, 1988.
- [Muñoz, Paulokat, Wess 1994] H. Muñoz, J. Paulokat, S. Wess. Controlling non-linear hierarchical planning by case replay. In M. Keane, J. Haton, M. Manago (Hrsg.), *Proc. of the 2nd European Workshop on Case-Based Reasoning EWCBR-94*, Paris, 1994.
- [Nilsson 1980] N. J. Nilsson. *Principles of Artificial Intelligence*. Tioga Publishing, 1980.
- [Peot, Smith 1993] M. Peot, D. Smith. Threat-removal strategies for partial-order planning. In: *Proc. of AAAI-93*, 492-499, 1993.
- [Smith, Peot 1993] D. Smith, M. Peot. Postponing threats in partial-order planning. In: *Proc. of AAAI-93*, 500-506, 1993.
- [Velooso 1992] M. Velooso. *Learning by analogical reasoning in general purpose problem solving*. Dissertation. Carnegie Mellon University, Pittsburgh, 1992.
- [Weberskirch 1994] F. Weberskirch. *Realisierung eines nichtlinearen Planungssystems zur Unterstützung der Arbeitsplanerstellung bei der computerintegrierten Fertigung (CIM)*. Diplomarbeit, Universität Kaiserslautern, 1994.

Innovation beim Konfigurieren

Lothar Hotz, Thomas Vietze¹

Universität Hamburg, FB Informatik

Vogt-Kölln Str. 30

D-22527 Hamburg

e-mail: lothar | thomas@ki6.informatik.uni-hamburg.de

Kurzfassung

Beim sogenannten Routine-Konfigurieren ist der Lösungsraum vollständig bekannt. Immer dann, wenn der vorgegebene Lösungsraum verlassen wird und Lösungen gefunden werden, die nicht vollständig vorformuliert sind, sprechen wir von einer innovativen Problemlösung. Wir erweitern die Verfahren zur Routine-Konfiguration, um Mechanismen zur innovativen Lösung von Teilproblemen zu finden, die ansonsten mit den Mitteln der Routine-Konfiguration nicht gelöst werden können.

0. Einleitung

Häufig wird suggeriert, daß allein durch die Verwendung von "neuen" Methoden im Konfigurationsbereich, wie fallbasiertes, optimierungsbasiertes, räumliches oder fuzzyfiziertes Konfigurieren, bereits innovatives Konfigurieren geleistet wird. Wir sind der Ansicht, daß innovatives Konfigurieren bestimmte, unabhängig von den genannten Methoden festlegbare Merkmale, aufweist. Es ist damit nicht als Oberbegriff anzusehen, sondern als eigenständige Methode, die alternativ oder zusätzlich eingesetzt werden kann. In diesem Papier soll der Anstoß zu einer Diskussion gegeben werden, an deren Ende Merkmale eines innovativen Konfigurationssystems stehen sollen. Dazu betrachten wir folgende Punkte:

1. Anwendungsbezug: Für komplexe Anwendungen ist die Betrachtung eines festen unveränderbaren Suchraums häufig zu eingeschränkt, so daß eine Aufweichung dieser Restriktion gefordert wird (Abschnitt 1).
2. Begriffsklärung: Der Begriff "innovatives Konfigurieren" steht stark im Zusammenhang mit dem Begriff des "innovativen Entwurfs" (innovative design). Dieser wird in [Gero 90] aus unterschiedlich komplexen Problemklassen abgeleitet. Diese Definition berücksichtigt nicht den relativen Charakter von Innovation - unserer Ansicht nach gibt es keine Innovation per se. Die etwas andere Sichtweise von Navinchandra beschreibt Innovation als das Überschreiten einer bestehenden Norm innerhalb einer gesellschaftsabhängigen Designkultur (design culture). Dabei wird der relative Aspekt von Innovation hervorgehoben. Da uns diese Definition der allgemeinen, normorientierten Innovation für technische Domänen zu vage erscheint, führen wir einen modellbezogenen Begriff der innovativen Konfigurierung ein. Dabei steht der relative und der konfliktbezogene Aspekt von Innovationen im Vordergrund (Abschnitt 2).
3. Quellen von Innovation: Innovationen kommen aus externen Quellen - in Bezug zu einem problemlösenden System. In Abschnitt 3 beschreiben wir mögliche Szenarien, welche externen Quellen genutzt werden können und in welcher Weise die Informationen aus diesen Quellen verwendet werden können.

1. Das diesem Papier zugrundeliegende Vorhaben wird mit Mitteln des Bundesministers für Forschung und Technologie gefördert (Förderkennzeichen ITW9101A6, Verbundvorhaben: PROKON). Die Verantwortung für den Inhalt dieses Papiers liegt bei den Autoren.

4. Merkmale eines innovativen Systems: Ein innovatives System hat bestimmte, über ein gängiges Konfigurationssystem hinausgehende Eigenschaften. Es muß z.B. vorsehen, ihm zunächst unbekannte Informationen zu bearbeiten. Dadurch wird ein hohes Integrationspotential in bestehende Datenverarbeitungswelten erreicht, da diese als wesentliche Quellen fremdartiger Information angesehen werden können. In Abschnitt 4 beschreiben wir die Anforderungen, die ein System erfüllen muß, um als innovativ bezeichnet werden zu können.

1. Warum benötigt man innovatives Konfigurieren?

Konfigurierungswerkzeuge werden seit längerem untersucht und entwickelt. Es gibt bereits Systeme, die erfolgreich im praktischen Einsatz sind (z.B. das COMIX System [Laudewein et al. 93]). Für viele Systeme ist ein vollständiges (Closed World Assumption) Domänenmodell Voraussetzung. Unter einem solchen Modell verstehen wir eine explizite oder implizite Aufzählung von Konfigurationen. Bei [Dörner 91] besteht das Modell des Konfigurierens aus Konfigurierungsaufgabe, Prozeßmodell und Begriffswelt. Während die ersten beiden Teile der Operationalisierung des Modells dienen, beschreibt die Begriffswelt den Raum der möglichen Lösungen.

Wir abstrahieren hier von der genauen Beschreibung der Konfigurationen in den verschiedenen Ansätzen (z.B. fallbasierter, schemabasierter (prototypischer), strukturorientierter, ressourcenorientierter oder interaktiver Ansatz). In jedem dieser Ansätze sind bestimmte konsistente und realisierbare Konfigurationen erzeugbar. Diese beschreiben das Modell eines Systems. In einem rein fallbasierten System besteht das Modell aus den Konfigurationen in der Falldatenbank und den Modifikationen, die aus diesen Fällen erzeugt werden können.

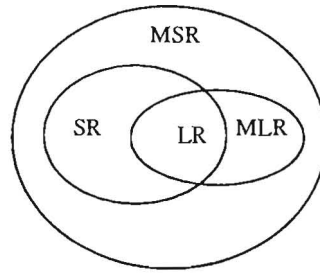
Bei einem strukturorientierten System besteht das Modell aus den aus den Strukturen ableitbaren Konfigurationen.² Beim sogenannten Routine-Konfigurieren ist der Lösungsraum vollständig bekannt [Brown 89]. Dies bedeutet, daß alle Konfigurationen, die als potentielle Lösungen in Betracht kommen, in dem Modell enthalten sein müssen. Der Konfigurierungsprozeß ist die Suche durch diesen häufig sehr großen oder unendlichen Suchraum.

Alle im Modell enthaltenen Lösungen sind per Definition korrekt. Eine Lösung, die in dem Modell enthalten ist und zur aktuellen Problemstellung paßt, ist somit ebenfalls korrekt. In vielen Fällen wird ein Modell nicht alle Lösungen enthalten, die in der Realität denkbar sind. Gründe dafür sind ein partiell unbekannter Lösungsraum (z.B. bei der Konfiguration von Flüssigkristallen), technischer Fortschritt, der neue Lösungsmöglichkeiten erschließt, mangelndes Wissen des Experten, der das Domänenmodell modelliert hat, etc. In Anlehnung an die Semantik in [Hein 90] und [Gero 90] lassen sich die folgenden Mengen identifizieren:

2. In einem strukturorientierten Ansatz, wie er z.B. durch das System Konwerk realisiert wird, sind Konfigurationen mittels einer Begriffshierarchie zur Beschreibung der zulässigen Komponenten, einer Zerlegungshierarchie zur Beschreibung der zulässigen Kombinationen von Komponenten (Konfigurationen), einem Constraintmodul zur Beschreibung der zu beachtenden Nebenbedingungen und Beschränkungen der Konfigurationen und einem Kontrollmodul zur Beschreibung des Vorgehens während der Konfigurierung erzeugbar. In diesem Ansatz wird das Modell implizit beschrieben; es enthält alle Konfigurationen, die aus "interessanten" Komponenten der Begriffshierarchie bestehen und mit der Zerlegungshierarchie und den Constraints konsistent sind. Als "interessante" Komponenten werden solche mit weitmöglichst eingeschränkten Eigenschaften betrachtet (die Blätter der Hierarchie).

Möglicher Suchraum (MSR),
Menge der theoretisch
möglichen Konfigurationen

Möglicher Lösungsraum
(MLR), Menge aller
theoretisch möglichen
Lösungen für ein
gegebenes Problem.



Suchraum (SR), Menge
aller im Domänenmodell
enthaltenen Konfigurationen

Lösungsraum (LR),
Menge der im Domänenmodell
enthaltenen Lösungen für ein
gegebenes Problem.

Die Menge MSR enthält alle in der realen Welt physikalisch möglichen Konfigurationen - ein eher philosophischer Raum. Das Modell eines Systems (d.h. alle durch das System erzeugbaren Konfigurationen) legt den Suchraum SR des Systems fest. Bei einem gegebenem Problem muß dieser Raum nach passenden Lösungen durchsucht werden. LR bezeichnet die Menge aller Lösungen eines gegebenen Problems, die durch das System gefunden werden können. Das System erreicht keine Lösungen, die zwar physikalisch möglich, jedoch nicht in SR enthalten sind; diese Lösungen liegen in MLR. Die Menge LR kann leer sein, obwohl MLR korrekte Lösungen enthält. Ob eine Konfiguration im Modell eines Systems enthalten ist, kann nicht per se bestimmt werden. Dies hängt letztendlich davon ab, ob die in einem System verwendeten Algorithmen entscheidbar sind. Z.B. ist es in einer endlichen Falldatenbank entscheidbar, ob eine gegebene Konfiguration enthalten ist oder nicht. Werden Konfigurationen z.B. mittels eines neuronalen Netzes ermittelt, ist es ungleich schwerer, die Zugehörigkeit einer Konfiguration zum Modell zu bestimmen.

Mit der Identifizierung einer korrekten Lösung ist ein Konfigurierungsproblem aber nicht unbedingt befriedigend gelöst. Die Qualität der ermittelten Lösung spielt eine wichtige Rolle. Die Konfigurierung einer möglichst guten Lösung wirft zwei Probleme auf: Wie können Lösungsalternativen ermittelt werden, wenn eine vollständige Suche über den gesamten Suchraum (SR) aus Komplexitätsgründen ausscheidet, und wie wird die Qualität einer Lösung bestimmt und vergleichbar gemacht? Um diese Probleme zu mildern, werden Heuristiken eingesetzt, die die Menge SR auf ein handhabbares Maß verkleinern sollen. An dieser Stelle zeigen sich die Schwächen eines vollständigen Domänenmodells. Gute Lösungen lassen sich häufig durch innovative Erweiterungen oder Anpassungen von vorgegebenen Domänenobjekten finden.

2. Was ist innovatives Konfigurieren?

Konfigurieren läßt sich in die beiden zunächst aufeinanderfolgenden Schritte Modellierung und Problemlösung trennen. Während der Modellierung wird die Domäne mit Hilfe einer allgemeinen Modellierungssprache beschrieben. Nach dem Modellierungsschritt wird zur Erstellung einer Konfiguration, die einer vorgegebenen Aufgabenstellung genügt, die Problemlösungsphase durchlaufen. Eine Einteilung von Problem in unterschiedliche Klassen erfolgt durch die Festlegung der Mischung dieser beiden Schritte.

In [Brown 83] und [Brown 89] werden Klassen von Konfigurationsproblemen beschrieben (classes of design). Danach wird Informationsverarbeitung beim Konfigurierung u.a. in die Beschreibung der möglichen Zerlegungen (decomposition) und des Vorgehens (design plans) unterteilt. Die umfassendste Klasse (Class 1, creative design) beinhaltet Probleme, die es notwendig machen, die beim Modellierungsprozeß beschriebene Zerlegung während des Problemlösungsprozesses zu ändern. Dadurch wird die Erstellung von Konfigurationen, die nicht

durch das Modell beschrieben werden, möglich. Bleibt die modellierte Zerlegung weitgehend erhalten und lediglich das Vorgehen muß angepaßt werden, handelt es sich nach Brown et al. um die 2. Klasse des Design (Class 2 design). Die 3. Klasse (class 3 design) wird als Routinedesign bezeichnet, da dort alle Zerlegungen und Vorgehensweisen bekannt sind und unverändert benutzt werden können, um Konfigurationen zu erstellen.

Auch nach [Gero 90] und [Coyne 87] können die einfachsten Konfigurationsprobleme (Routinekonfiguration) auf der Basis des jeweiligen Domänenmodells gelöst werden. Die vorgegebene Modellierung wird während der Problemlösung nicht verändert, das Vokabular des Modells reicht zur Lösung des Problems aus. Wird es notwendig, Kombinationen von Objekten (neue Zerlegungen), die nicht in der Modellierung vorgesehen sind, während der Problemlösung zu erzeugen, wird von innovativem Konfigurieren gesprochen. Hierunter fallen auch die möglicherweise notwendigen Änderungen von modellierten Wertebereichen von Parametern. Kennzeichnend für diesen Grad ist die Beibehaltung des durch das Modell festgelegten Vokabulars. Wird auch dieses Vokabular während der Problemlösung geändert, etwa durch Hinzunahme von neuen Konzepten oder neuen Parametern, spricht Gero von kreativer Konfiguration.

Nach [Günter 93] ist eine strenge Einteilung der Probleme in Klassen nicht möglich, es liegt vielmehr eine Skala von verschiedenen Graden von Problemen vor. Am einen Ende der Skala sind die Probleme angesiedelt, die mittels Routinekonfiguration gelöst werden können, am anderen Ende Probleme, die kreative Aktionen verlangen.

Navinchandra [Navinchandra 91] geht von einer Designkultur (design culture) aus, die durch die allgemeine Praxis, Entwurfsstile und Technologien bestimmt wird, die von den innerhalb der Designkultur arbeitenden Personen benutzt werden. Die Designkultur definiert den Kontext dieser Personen. Als Innovation wird das definiert, was von dem abweicht, was innerhalb einer Designkultur bekannt ist. Innovation erhält damit einen relativen Aspekt, hier in Bezug zu einer Designkultur. In Abhängigkeit von den drei Dimensionen "Spezifikation", "Prozess" und "Produkt" können unterschiedliche Grade von Innovation festgelegt werden. Dies führt auch hier zu einem Kontinuum von Routinedesign bis zu innovativem Design. Navinchandra beschreibt das System (CYCLOPS), das innovatives Design realisieren soll. Dabei steht die Exploration des Suchraums durch Relaxation von Constraints, die Bewertung einer Lösung durch neue Kriterien und die Anpassung alter Lösungen im Vordergrund. Unbefriedigende Lösungen können von dem System weiterverarbeitet, sowie Vor- und Nachteile einzelner Lösungen erfaßt werden. Hinter diesem Ansatz steht die implizite Annahme, daß die akzeptierte Norm der Designkultur in dem System modelliert wird. Komplexe Anwendungen zeigen, daß dies nicht immer möglich ist. Daher führen wir eine weitere Sichtweise auf innovatives Konfigurieren ein: Eine modellbezogene Definition von innovativem Konfigurieren.

Um brauchbare von unbrauchbaren Innovationen zu trennen, wird in der Psychologie eine Bewertungskomponente modelliert. Diese filtert Abweichungen von bisher bekannten Vorgehensweisen und Lösungen so, daß das Ergebnis des innovativen Vorgehens mit den gegebenen Zielen übereinstimmt. Eine Erweiterung hierzu ist die Entdeckung und Berücksichtigung zusätzlicher Ziele, die mit den bereits bekannten konkurrieren und diese entsprechend relativieren.

Wir verstehen unter innovativem Konfigurieren einen Vorstoß zu bisher unerreichbaren Elementen von MLR. Da der Raum LR durch die Modellbildung bzgl. einer Domäne festgelegt wird, ist innovatives Konfigurieren in unserem Sinne immer in Bezug zu einem Modell zu sehen. Innovativ ist eine Konfiguration, wenn sie von einem gegebenen Modell nicht beschrieben wird. Ein anderes Modell könnte diese Konfiguration enthalten, sie wäre damit zu diesem keine Innovation. Hat man z.B. ein Modell, welches genau 10 Konfigurationen beschreibt, so ist eine

weitere davon verschiedene Konfiguration eine Innovation bezüglich dieses Modells.

Als Innovationen bezüglich eines Modells betrachten wir die folgenden neuen, d.h. im Modell nicht beschriebenen, Entitäten:

- Neue Wertebereiche für Parameter und Relationen des Modells. Hierunter fallen u.a. neue Zerlegungen von Komponenten sowie Änderungen von vorhandenen Einschränkungen und Nebenbedingungen (etwa die Wertebereiche von Constraints).
- Neue Parameter, Relationen und Konzepte.
- Neue Nebenbedingungen und Abhängigkeiten zwischen Objekten.
- Neue Vorgehensweisen bzw. neue Problemlösungsprozeduren.

Die Festlegung auf das Modell eines Systems, welches die von ihm erstellbaren Konfigurationen beschreibt (siehe Abschnitt 1), erlaubt die genaue Einordnung einer neuen Konfiguration als innovativ oder nicht innovativ. Wenn eine Konfiguration innerhalb des Modells des Systems liegt, ist sie nicht innovativ, ansonsten ist sie innovativ (eine ähnliche Definition von kreativem Design wird in [Sun et al. 94] gegeben). Die Referenzierung auf eine allgemeingültige, jedoch nicht genau spezifizierbare Norm wird zugunsten der Bestimmbarkeit von Innovation in Bezug auf ein vorliegendes System aufgegeben. So kann es durchaus sein, daß eine Konfiguration, die in Bezug auf ein System innovativ ist, von einer Designkultur als nicht innovativ angesehen wird. Umgekehrt könnte bereits durch ein System eine Konfiguration beschrieben werden, die von einer Designkultur bisher nicht erkannt wurde - man erhält eine unverhoffte, jedoch nicht in unserem Sinne innovative Lösung.

Ein anderer Aspekt des relativen Charakters von Innovation liegt in dem darin enthaltenen Wandel. Eine bisher nicht im Modell enthaltene und damit innovative Konfiguration kann durch ein innovatives System in das Modell aufgenommen werden. Damit ist diese bislang innovative Konfiguration in den Bestand der modellierten Konfigurationen aufgenommen und ist damit nicht mehr innovativ in Bezug auf das Modell.³ Wir nennen ein System, welches es erlaubt neue Konfigurationen zur Lösung weiterer Aufgaben in das Modell miteinzubeziehen ein "wandelbares" System. Geeignete Innovationen können in einem wandelbaren System in das Modell integriert werden - das Modell wird fortgeschrieben (Wissensbasenevolution, KB evolution, siehe z.B. [Hinkelmann et al. 94]). Dabei werden u.a. Verfahren betrachtet, die es erlauben, ein neues Konzept an passender Stelle in ein bestehendes Modell einzuordnen.

Ein weiterer Aspekt von Innovation ist die Verbindung zu Konflikten. In Konfigurationssystemen wird meist angenommen, daß die durch das System beschriebene Menge von Konfigurationen die einzigen Kandidaten für die Lösung eines anstehenden Problems sind. Eine neue, nicht enthaltene Konfiguration, steht damit immer im Konflikt mit dieser Annahme. Dadurch begründet sich unsere - auf den ersten Blick provokante - These, daß jede Innovation mit Konflikten und Konfliktbewältigung zusammenhängt.⁴ Damit ist ein weiteres Merkmal einer innovativen Konfiguration der Grad der Konflikte mit den bekannten Konfigurationen. Eine Konfiguration, die für eine bekannte Variable einen neuen Werte festlegt (d.h. innovativ ist im Sinne von Gero), steht im Konflikt mit dem bisherigen Wertebereich der Variablen. Eine Kon-

3. Dieser Schritt von innovativen Lösungen zu Routinelösungen ist auch in anderen Bereichen erkennbar, so war früher ein Taschenrechner eine Innovation gegenüber einem Rechenschieber. Heute gehört der Taschenrechner längst zu den gängigen Hilfsmitteln.

4. Auch dieser Aspekt ist in anderen Bereichen bekannt. So wird in den Wirtschaftswissenschaften die Überwindung von Widerständen und Schwierigkeiten bei der Durchsetzung neuer Möglichkeiten (Innovationen) als eine wichtige Funktion des Unternehmers angesehen.

figuration, die eine neue Variable einführt, steht im Konflikt mit der bisherigen Menge von Variablen. Ein Maß für Konflikte kann durch die Abhängigkeiten der Variablen untereinander festgelegt werden. Ein neuer Wert für eine Variable, die in keiner Relation zu einer anderen steht, verursacht einen geringfügigen Konflikt. Eine Variable, die zu vielen anderen Variablen in Relation steht, verursacht einen entsprechend schwereren Konflikt. Kommen mehrere neue Variablen innerhalb einer innovativen Konfiguration vor, steht diese Konfiguration in einem größeren Konflikt zu den bisherigen, als eine Innovation mit einer geringeren Anzahl neuer Variablen. Durch den Grad der Konflikte wird das Kontinuum zwischen Routinekonfiguration und kreativer Konfiguration diskretisiert.⁵

In unseren Betrachtungen gehen wir von gewissen Basistechniken aus, welche das Modell eines Systems bestimmen. Solche Techniken können, wie bereits erwähnt, die bekannten KI-Methoden im Bereich der Konfigurierung sein (z.B. strukturorientierte, fallbasierte Konfigurierung) oder auch konnektionistische Ansätze, wie bei der Verwendung von neuronalen Netzen oder genetischen Algorithmen. Innovativ wird ein System erst dann, wenn es neben diesen Basistechniken, externe Quellen zur Gewinnung von Lösungen zuläßt.⁶ Ein System mit einer besonderen, "neuen" Basistechnik ist damit nicht innovativ. Die Kombination von Techniken allein ist ebenfalls nicht als innovativ zu bezeichnen. Werden z.B. strukturorientierte Techniken (Begriffs- und Zerlegungshierarchie etc.) mit fallbasierten Techniken gekoppelt, wobei in den Fällen jedoch nur Werte vorkommen, die durch die Struktur bestimmt sind, liegt keine Innovation vor. Dies wird etwa durch die Bibliothekslösung von Konwerk erreicht [Angelus et al. 94, Pfitzner 93]. Werden hingegen Fälle mit neuen Informationen berücksichtigt, sprechen wir von innovativ. Das in [Angelus et al. 94] beschriebene Generieren von Konzepten aus Datenbanken während der Wissenakquisition ist damit als innovativ gegenüber der Ausgangshierarchie zu bezeichnen. Bezüglich der resultierenden Hierarchie, die nach der Einbeziehung dieser Fälle in die Struktur entsteht (d.h. das System hat sich gewandelt), stellen diese Fälle keine Innovation mehr dar.

3. Woher kommen Innovationen?

Wir beschränken uns in diesem Papier auf die Behandlung von Innovationen während der Problemlösung und betrachten damit die Kopplung eines Konfigurationssystems an andere Systeme. Eine Lösung oder Konfiguration wird damit in unserem System nicht ausschließlich mit Hilfe des vorgegebenen Modells, sondern unter Zuhilfenahme anderer Quellen gewonnen. Wir betrachten damit die Frage: Wie geht ein Konfigurationssystem mit Information aus anderen Quellen während der Konfigurierung um?

Reale Domänen sind meist komplex und fortlaufenden Neuentwicklungen von Komponenten unterworfen. Eine Modellierung solcher Domänen ist dadurch natürlicherweise unvollständig. Ein Konfigurierungssystem muß flexibel mit dem dynamischen Charakter solcher Domänen umgehen können. Die kurzzeitige Einbindung von neuer Information während der Bearbeitung eines Problems kann damit ein wertvoller Schritt für die Integration eines wissensbasierten Systems in vorhandene Datenverarbeitungswelten sein. Ein Beispiel ist die Konfigurierung von Flüssigkristallen. Hier existiert ein umfangreiches Domänenmodell, das für viele Problemstellungen sinnvoll genutzt werden kann. Bestimmte Bereiche der Domäne sind jedoch nicht systematisch erforscht worden. Hier steht lediglich Wissen über bereits vorhandene Kristalle und

5. Eine ähnliche Klassifikation von Konflikten wird auch in [Bakhtari et al. 94] vorgeschlagen.

6. Als "primitiv-innovativ" kann man ein System bezeichnen, welches keinerlei Basistechniken beinhaltet, sondern lediglich externe Quellen zur Lösungsfindung anzapft.

ihre Eigenschaften zur Verfügung.

Innovationen können z.B. aus Falldaten, die bereits gelöste Probleme mit oder ohne Lösungsprozess enthalten, aus Datenbanken (etwa vorhandene Artikeldatenbanken oder Stoffdatenbanken), aus Simulationssystemen zur Ermittlung von Langzeiteigenschaften, aus neuronalen Netzen, die z.B. trainiert wurden, um bestimmte Eigenschaften abzuleiten, etc. stammen. Es ist auch denkbar, andere wissensbasierte Systeme anzukoppeln, einen menschlichen Experten als Quelle der Neuerungen miteinzubeziehen oder systematisch alle physikalisch möglichen Kombinationen (etwa bei der Positionierung einer Komponente) zu untersuchen. [Gero 90] spricht allgemein von Kombination, Analogie, Mutation und naturwissenschaftlichen Gesetzen (first principles) als Methoden zur Ermittlung von Innovationen. Das gemeinsame Ziel ist immer, weitere Eigenschaften der aktuellen Teillösung abzuleiten.

In [Sun et al. 94] werden in einem wissensbasierten CAD-System Lösungen auf der Basis von Prototypen gefunden. Es wird ein System beschrieben, welches Unterstützung bei der Entdeckung neuer Prototypen leistet. Dabei werden generative Techniken aus der qualitativen Physik angewandt.

In [Bhatta et al. 94] werden innovative Lösungen durch Analogieschluß zu anderen, problemfernen Domänen gefunden. Dort wird die Auffassung vertreten, daß Innovation immer analogiebasierte Schlüsse verlangt. Wir sehen in der Analogie eine mögliche Form der Entdeckung neuer Lösungen.

Folgende beispielhafte Szenarien sind mit Hilfe eines Systems für die innovative Konfigurierung denkbar:

- Der Experte erzwingt einen Wert für einen Parameter.
- Der Experte führt einen neuen Parameter während der Konfigurierung ein, um neue Konstellationen innerhalb vorgegebener Konfigurationen zu testen.
- Aus einer Datenbank werden per genauer Analyse der Datenzusammenhänge (database mining) neue Nebenbedingungen erkennbar.
- Übernahme eines Falles, der nicht durch das vorliegende Domänenmodell abgedeckt wird.

Um eine korrekte Integration solcher Systeme mit einem Konfigurationssystem zu gewährleisten, muß eine Anpassung der Modelle (Normierung der Eigenschaften und Wertebereiche) erfolgen, um jeweils gleiche Eigenschaften und Konzepte in den Systemen zu identifizieren. Probleme, die die angekoppelten Systeme mit sich bringen, z.B. ein problematisches Ähnlichkeitsmaß bei fallbasierten Systemen oder die zu kleine Trainingsmenge eines neuronalen Netzes werden durch die Kopplung nicht gelöst.

4. Berücksichtigung von Innovationen während der Konfigurierung

Wie die Integration von Innovationen während des normalen Konfigurierungsvorganges zu leisten ist, hängt in starkem Maße von den verwendeten Konfigurierungstechniken ab. In jedem Fall müssen Konflikte zwischen dem per Routine-Konfiguration erstellten Teil einer Lösung und den innovativen Erweiterungen behandelt werden.

Ein Merkmal eines innovativen Systems ist die Unterstützung, die das System bei der Bearbeitung und Einbeziehung neuer, unbekannter Informationen während der Bearbeitung eines Problems liefert. Das System kann dabei mittels der Closed World Assumption schließen, muß

diese jedoch, falls notwendig erweitern. Diese Erweiterung kann dadurch geschehen, daß eine zu einem Zeitpunkt gefundene Innovation, bei der weiteren Bearbeitung als Ausnahme betrachtet wird. Diese ist neben den vom System berechneten Lösungen gültig.

Wichtig ist eine konsistente Behandlung der Innovation, wenn sie mit dem der Routine-Konfigurierung zugrundeliegenden Modell im Widerspruch steht. Dies kann notfalls durch eine Ausnahmebehandlung geschehen. Denkbar ist aber auch die Verwaltung als neuer Fall für fallbasierte Verfahren oder durch die Einführung neuer Ressourcentypen beim ressourcenorientierten Konfigurieren. Das Aufweichen von Constraints (constraint relaxation) ist eine weitere Methode, die bei verschiedenen Ansätzen verwendet werden kann. Aufgrund der starken Abhängigkeiten zwischen einzelnen Parametern einer Innovation und der gesamten Konfigurierung ist davon auszugehen, daß eine interaktive, unterstützende Rolle des Systems eher zur Akzeptanz beim Benutzer führen wird, als eine automatische Generierung innovativer Konfigurationen.

Zusammenfassung

Wir schlagen eine relative Definition für innovatives Konfigurieren vor. Innovation ist das, was über das bereits bekannte und modellierte hinausgeht. Die Erweiterung des Suchraumes und des Lösungsraumes ist ein hinreichendes und notwendiges Kriterium für Innovation. Diese Sichtweise unterstützt eine evolutionäre Modellentwicklung, bei der Innovationen ausprobiert, überprüft und ggf. in das Modell integriert werden. Durch die Integration in das Modell verliert eine Innovation ihren externen Charakter. Das System an sich ist leistungsfähiger geworden.

Die Schritte Modellierung und Problemlösung werden miteinander verbunden, so daß ein wirklich flexibles System entsteht. Der Beweis, daß ein solches System für praktische Anwendungen geeignet ist, steht noch aus. Die automatische Verarbeitung wenig strukturierter Information ist ein schwieriges Problem. Wir gehen davon aus, daß ein innovatives Konfigurationssystem dem Anwender in erster Linie unterstützend zur Seite steht.

Literatur

[Angelus et al. 94]

A. Angelus, Ch. Bohley, A. Kathe und H. Reuter, *Integration von KONWERK in eine komplexe Anwendungsumgebung*, internes PROKON Papier, 1994.

[Bakhtari et al. 94]

S. Bakhtari und B. Bartsch-Spoerl, Bridging the Gap between AI Technology and Design Requirements, in: J.S Gero und F. Sudweeks (Hrsg.), *Artificial Intelligence in Design '94*, 753-768, Kluwer Academic Publishers, 1994

[Bhatta et al. 94]

S. Bhatta, A. Goel und S. Prabhakar, Innovation in Analogical Design: A Model-Based Approach, in: J.S Gero und F. Sudweeks (Hrsg.), *Artificial Intelligence in Design '94*, 57-74, Kluwer Academic Publishers, 1994

[Brown 83]

D.C. Brown, B. Chandrasekaran, An approach to expert systems for mechanical design, in: *Trends and Applications '83*, 173-180, IEEE Computer Society, NBS, Gaithersburg, Mai 1983

[Brown 89]

D. Brown, B. Chandrasekaran, *Design Problem Solving: Knowledge Structures and Control Strategies*, Morgan-Kaufmann, 1989

[Coyne 87]

R.D. Coyne, M.A. Rosenman, A.D. Radford und J.S. Gero, Innovation and Creativity in Knowledge-Based CAD, in: *Expert Systems in Computer-Aided Design*, ed. J.S. Gero, 435-465, Amsterdam: North-Holland, 1987

[Dörner 91]

H. Dörner, *Modelle für wissensbasiertes Konfigurieren*, Habilitationsschrift, Uni Halle, 1991

[Gero 90]

J.S. Gero, Design Prototypes: A Knowledge Representation Schema for Design, *AI Magazine*, Vol. 11, No. 4, 26-36, 1990

[Günter 93]

A. Günter, Modelle beim Konfigurieren, in: O. Herzog, Th. Christaller, D. Schütt (Hrsg.), *Grundlagen und Anwendungen der Künstlichen Intelligenz*, Springer, 1993

[Hein 90]

M. Hein, W. Tank, Assoziative Konfigurierung, in: *Proc. of the 14th German Workshop on Artificial Intelligence 1990*, Heinz Marburger (Hrsg.), 1990

[Hinkelmann et al. 94]

Hinkelmann, Knowledge-Base Evolution for Product and Production Planung, in: *AICOM* Vol. 7, Nr. 2, June 1994

[Laudewein et al. 93]

N. Laudewein, A. Brinkop, *Konfigurieren von Rührwerken mit COMIX*, in: *Expertensysteme 93*, pp. 112-124, Springer Verlag, 1993

[Navinchandra 91]

D. Navinchandra, *Exploration and innovation in Design*, Springer Verlag, 1991

[Pfitzner 93]

K. Pfitzner, *Fallbasiertes Konfigurieren technischer Systeme*, Dissertation, Uni Hamburg, 1993

[Sun et al. 94]

K. Sun und B. Faltings, Supporting Creative Mechaniccal Design, in: J.S Gero und F. Sudweeks (Hrsg.), *Artificial Intelligence in Design '94*, 39-56, Kluwer Academic Publishers, 1994

Räumliche Beziehungen und Constraint-Systeme:

Eine kurze Betrachtung über die Komplexität

Manfred Kopisch

Universität Hamburg, Fachbereich Informatik
Vogt-Kölln-Straße 30; 22527 Hamburg

Kurzfassung. Räumliche Beziehungen zwischen Objekten müssen bei bestimmten Konfigurierungsproblemen berücksichtigt werden, um eine korrekte Lösung zu finden. Die Verarbeitung solcher Beziehungen kann von einem Constraint-System geleistet werden. Insbesondere, wenn bereits andere, nicht-räumliche Beziehungen von einem einem Constraint-System bearbeitet werden, liegt der Vorteil der gemeinsamen Verarbeitung von räumlichen und nicht-räumlichen Beziehungen in einem Constraint-System auf der Hand. Es gibt aber gewisse Grenzen bei der Bearbeitung räumlicher Beziehungen mit Constraint-Systemen, die in diesem Beitrag beleuchtet werden sollen. Fragen nach der Komplexität, der Ausdrucksmächtigkeit, etc. stehen dabei im Vordergrund.¹

1 Einleitung

Einige Konfigurierungsaufgaben benötigen zum Auffinden einer korrekten Lösung räumliches Wissen. Ein Teil dieses räumlichen Wissens - das Wissen über räumliche Beziehungen zwischen Objekten - wird im folgenden aus zwei Blickwinkeln betrachtet:

- Welche Beziehungen müssen darstellbar sein?
- Wie können diese Beziehungen effizient verarbeitet werden?

Dafür werden zunächst zwei Domänen in Abschnitt 2 vorgestellt, die solches räumliches Wissen benötigen: Die Konfigurierung einer Küche und die Konfigurierung der Passagierkabine eines Verkehrsflugzeuges. Beispiele aus beiden Domänen geben einen Überblick über die benötigten räumlichen Beziehungen. In Abschnitt 3 wird deren Beschreibung und Verarbeitung mit Hilfe eines Constraint-Systems vorgestellt. Die Grenzen, die sich insbesondere im zeitlichen Verhalten aufzeigen, werden in Abschnitt 4 aufgezeigt. In Abschnitt 5 wird ein Vorschlag zur Verbesserung dieses Verhaltens gemacht und in einer Zusammenfassung eine Abschätzung der Einsetzbarkeit der vorgestellten Verfahren gegeben.

2 Räumliche Beziehungen in Konfigurierungsproblemen

Bei einer Konfigurierung soll aus einer Menge von Einzelteilen eine Lösung erstellt werden, die bestimmte Randbedingungen einhält. Diese Randbedingungen können unter Umständen räumlicher Natur sein. Im folgenden betrachten wir zwei Domänen, die deutlich machen, welche räumlichen Randbedingungen berücksichtigt werden müssen, damit eine konsistente Lö-

¹. Das diesem Bericht zugrundeliegende Vorhaben wird mit Mitteln des Bundesministers für Forschung und Technologie (Förderkennzeichen ITW9101A6, Verbundprojekt: PROKON) gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt beim Autor.

sung gefunden werden kann.

2.1 Konfigurierung einer Einbauküche

Für die Konfigurierung einer Einbauküche müssen mehrere Objekte, wie beispielsweise ein Herd, ein Kühlschrank, eine Spüle etc. auf einer vorgegebenen Grundfläche platziert werden. Dabei müssen Randbedingungen der Art "Ein Herd darf nicht neben einem Kühlschrank stehen." oder "Vor einem Fenster darf kein Schrank stehen." berücksichtigt werden, um zu einer konsistenten Lösung zu kommen.

An dieser sicherlich jedem geläufigen Domäne läßt sich gut zeigen, welche Art von Beziehungen benötigt wird. Eine typische Küche ist rechteckig und enthält nur rechteckige Gegenstände, die achsenparallel angeordnet sind.

Bisher haben wir die beiden Nachbarschaftsbeziehungen "neben" und "vor" aufgeführt. Bei genauerer Betrachtung zeigt sich, daß das hier genannte "neben" eine Berührungsrelation ist, während das "vor" auch bei einem Abstand, der größer als null ist, gelten soll. Wir müssen uns also darüber klar werden, was mit solchen Beziehungen in der jeweiligen Domäne gemeint ist und wie sie mit vordefinierten Konstrukten beschrieben werden können.

Die Küchendomäne wird von mehreren Autoren (u.a. [Baykan90] und [Schmitgen93]) benutzt, um die Fähigkeiten ihrer Systeme zur Layoutgenerierung zu erläutern. Beide benutzen ein ähnliches Repertoire an räumlichen Relationen, das wir in Abschnitt 2.3 vorstellen.

Bevor wir diese Relationen vorstellen, gehen wir noch auf eine weitere Domäne ein, in der ganz ähnliche Relationen verwendet werden.

2.2 Konfigurierung der Passagierkabine eines Verkehrsflugzeuges

Ein Kabinenlayout definiert die Passagierkabine eines Verkehrsflugzeuges, indem es die Anzahl, die Anordnung, die Art und die Lage der Einrichtungsgegenstände angibt. Einrichtungsgegenstände sind Objekte wie Passagiersitze, Küchen, Toiletten, Flugbegleitersitze, Stauschränke etc. Die Gesamtheit der Einrichtungsgegenstände definiert die Beförderungskapazität, den Komfort, den möglichen Service etc. der Passagierkabine.

Um das Layout einer Passagierkabine zu erstellen, müssen eine gewisse Anzahl von Einrichtungsgegenständen wie z. B. Sitze, Toiletten oder Küchen auf dem begrenztem Raum der Passagierkabine unter Berücksichtigung von gesetzlichen und technischen Randbedingungen so platziert werden, daß die Wünsche der auftraggebenden Fluggesellschaft bezüglich Klasseneinteilung, Sitzanzahl, Sitzabstand etc. möglichst optimal erfüllt werden. [Kopisch91], [Kopisch91a].

Sowohl unter den gesetzlichen und technischen Randbedingungen als auch unter den Kundenwünschen finden sich Restriktionen, die räumlicher Natur sind. Ein derartiger Kundenwunsch wäre z. B. der Abstand zwischen einzelnen Sitzen in einer bestimmten Klasse, der jeweils 34 Zoll betragen soll. Eine ähnliche Randbedingung, die gesetzlicher Natur ist, betrifft den Abstand eines Sitzes von einer festen Wand, der mindestens 42 Zoll betragen muß. Eine technische Randbedingung beschränkt die Anzahl der Küchen, die hinter Tür 4 der Passagierkabine liegen, auf maximal 4.

Diese Domäne läßt sich ebenfalls durch achsenparallele Rechtecke beschreiben, zwischen denen ähnliche Beziehungen gelten, wie in der Küchendomäne. Auch hier finden wir Nachbarschafts- und Abstandsbeziehungen. Zusätzlich haben wir hier das Enthaltensein in einem bestimmten Bereich und die Unterscheidung der Relation "vor" in eine objektzentrierte (vor dem Sitz) und eine globale, die Passagierkabine betreffende Relation (hinter Tür 4) gesehen. In [Kopisch93] findet sich eine Übersicht der für diese Domäne benötigten Relationen.

Im folgenden Abschnitt gehen wir auf die wichtigsten Arten von räumlichen Relationen ein, die in den beiden genannten Domänen benötigt werden.

2.3 Räumliche Relationen in Domänen mit achsenparallelen Rechtecken

Für die beiden aufgeführten Domänen gilt, daß achsenparallele Rechtecke auf einer geschlossenen achsenparallelen Grundfläche angeordnet werden müssen. Daher finden sich in beiden Domänen gleichartige Beziehungen, die von den unterschiedlichen Autoren ([Baykan91], [Kopisch93] und [Schmitgen93]) ähnlich gegliedert werden und sich jeweils in vier Ausprägungen für jede der möglichen Richtungen wiederfinden.

- Objektzentrierte Relationen
 - Lagerrelationen (vor, hinter, rechts, links)
 - Abstandrelationen (z. B. Vorderabstand)
 - Berührungsrelationen (z. B. berührt-links)
 - Überlappungsrelationen (z. B. überlappt-rechts)
 - Orientierungsrelation (z. B. gleich-ausgerichtet)
- Globale Relationen
 - Lagerrelationen (östlich, westlich, nördlich, südlich)
 - Abstandsrelationen
 - Berührungsrelationen
 - Überlappungsrelationen
 - Inhaltsrelationen (enthält, ist-enthalten-in)

Abbildung 1 macht einige der Relationen anschaulich. B liegt gleichzeitig östlich von A und vor A. B und E sind gleich-ausgerichtet. Der Hinterabstand von B zu D ist gleich dem Rechtsabstand von A zu D.

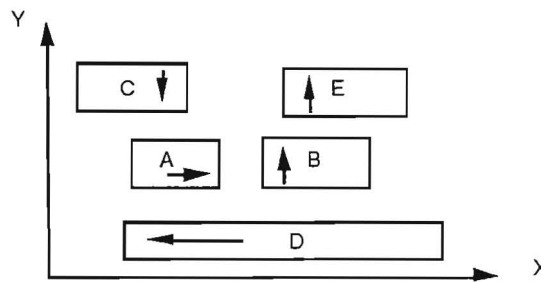


Abbildung 1: Räumliche Relationen

Verschiedene Autoren führen teilweise noch weitere Relationen, wie z. B. neben oder überlappt-nicht, ein, die sich aber leicht aus den vorhandenen Relationen ableiten lassen (z. B. neben := links oder rechts).

Es besteht also weitestgehender Konsens über die zur Beschreibung der Domänen benötigten Relationen. Im folgenden Abschnitt gehen wir auf die Beschreibung und Verarbeitung räumlicher Randbedingungen mit Hilfe solcher Relationen in einem Constraint-System ein.

3 Verarbeitung räumlicher Beziehungen mit einem Constraint-System

Räumliche Beziehungen zwischen Objekten stellen bei einer Konfigurierung Beschränkungen für ihre Platzierung und Dimensionierung dar. Viele Systeme, die sich mit räumlichen Beschränkungen beschäftigen, benutzen Constraint-Systeme, um die Einhaltung von Einschränkungen sicherzustellen (s. [Hernández90], [Baykan90], [Güsgen91] und [Kopisch93]).

Am Beispiel der 3-stufigen Constraint-Integration, die PLAKON [Cunis91] bietet, werden wir kurz erläutern, wie die Beschreibung und spätere Verarbeitung einer räumlichen Beziehung in einem Constraint-System, wie wir es beispielsweise in der Kabinenlayout-Domäne benutzen [Kopisch92], [Kopisch 93], funktioniert.

Die im vorigen Abschnitt aufgeführten räumlichen Relationen werden als sogenannte **domänenunabhängige Constraint-Relationen** definiert, die für die angegebenen Parameter sicherstellen, daß die jeweilige Relation erfüllbar ist. Diese vordefinierten Constraint-Relationen, wie z. B. "vor", werden zur Beschreibung von Randbedingungen, wie beispielsweise "Vor einer Küche müssen mindestens 36 Zoll Arbeitsfläche frei bleiben" als sogenannte **konzeptuelle Constraints** benutzt. Diese konzeptuellen Constraints werden während der Konfigurierung immer dann instantiiert, wenn ein betroffenes Objekt instantiiert wird. **Instantiierte Constraints** werden zu einem Constraint-Netz zusammengefügt, welches es erlaubt, bei Veränderung eines Parameters, mit ihm verbundene Parameter anzupassen.

Das folgende Beispiel zeigt die Benutzung der domänenunabhängigen Constraint-Relationen zur Beschreibung räumlicher Beziehungen in der Kabinenlayoutdomäne.

```
(def-konzeptuelles-constraints :name Freiraum-vor-Kueche
  :variablen-pattern-paare ( (?k :konzeptname kueche)
                             (?e :konzeptname einrichtungsgegenstand
                                :parameter ((vorderernachbar ?k))))
  :lokale-variablen        (abstand)
  :constraint-aufrufe       ( (vorderabstand ?k ?e abstand)
                             (groesser-gleich 36 abstand)))
```

Dieses konzeptuelle Constraint wird während der Konfigurierung immer dann instantiiert, wenn es zu einer instantiierten Küche einen Einrichtungsgegenstand gibt, der ihr "Vordernachbar" ist. Jedes Objekt hat dabei vier Slots in denen der bzw. die nächsten Nachbarn zu jeder Seite eingetragen sind.

Das folgende konzeptuelle Constraint stellt sicher, daß sich zwei Einrichtungsgegenstände nicht überlappen.

```
(def-konzeptuelles-constraints :name Ueberlappungsverbot
  :variablen-pattern-paare ( (?e1 :konzeptname einrichtungsgegenstand)
                             (?e2 :konzeptname einrichtungsgegenstand)))
  :constraint-aufrufe       ( (ueberlappt-nicht ?e1 ?e2)))
```

Diese deklarative Form der Beschreibung gewährleistet eine einfache Wartbarkeit der Wissensbasen. Der Einsatz solcher Constraint-Systeme findet aber seine Grenzen im Zeitverhalten.

4 Grenzen des Einsatzes von Constraint-Systemen für räumliche Probleme

Einleitend muß gesagt werden, daß Constraint-Satisfaction-Probleme (CSP) NP-vollständig sind. Allgemeine CSPs können also wahrscheinlich nicht in polynomialer Zeit gelöst werden. Daher werden Constraint-Systeme dazu benutzt, um aus gegebenen Wertemengen, die Werte zu entfernen, welche zu lokalen Inkonsistenzen führen und daher in einer globalen Lösung nicht enthalten sein können. Diese Verfahren, die Knoten-, Kanten- oder Pfadkonsistenz (zur Erklärung der Begriffe s. [Mackworth77]) herstellen, sind in polynomialer Zeit lösbar [Mackworth85].

Was aber bedeutet polynomielle Zeit in der Praxis? Nehmen wir einen der bekannten Algorithmen, die uns zur Herstellung von Kantenkonsistenz zur Verfügung stehen. AC-3 [Mackworth77] hat im schlimmsten Fall eine Zeitkomplexität von $O(d^3e)$, wobei d die Anzahl der Werte des größten Wertebereiches und e die Anzahl der Constraints ist, die im schlimmsten Fall von der Ordnung n^2 ist (n = Anzahl der Variablen). Andere Algorithmen, wie z. B. AC-4 [Mohr86] bieten für den schlimmsten Fall bessere Werte, aber verhalten sich u. U. in durchschnittlichen Fällen schlechter als AC-3. Vor der Auswahl eines Algorithmus sollte daher die Domäne bzgl. der darin enthaltenen Constraints und Wertemengen betrachtet werden, um Aussagen treffen zu können. Der bisher beste bekannte Algorithmus AC-6 [Bessière93] hat eine Worst-Case-Zeitkomplexität von $O(d^2e)$. Nehmen wir als Beispiel für eine praktische Betrachtung die Kabinenlayoutdomäne. Hier werden ca. 200 Objekte plaziert. Viele dieser Objekte sind dimensionierbar, so daß simple Constraints, für Beziehungen wie z. B. $(x_{\min} \text{ Obj}) + (x_{\max} \text{ Obj})$

ausdehnung Obj) = (xmax Obj) instantiiert werden müssen. Die Wertemenge für die Befestigung einer Sitzbank enthält u. U. 6000 Werte (die Passagierkabine ist ca. 2000 Zoll lang und bietet links, rechts und in der Mitte (s. Abb. 2) Sitzschienen mit Befestigungsmöglichkeiten in 1-Zoll-Abständen). Die Objekte haben 13 räumliche Variablen (Länge, Breite, Vorderseite, Drehung, Orientierung, x-min, x-ausdehnung, x-max, x-bereich, y-min, y-ausdehnung, y-max, y-bereich). Wir kommen also nur für die Sicherstellung der lokalen Konsistenz (nicht für das Finden der Lösung) im schlimmsten Fall mit $e = 0.5 * 200 * 13 * (200 * 13 - 1) = 3.378.700$ und $d = 6000$ auf $6000^2 * 3.378.700 \approx 1.2 \text{ E}14$.

A340 Cabin Layout

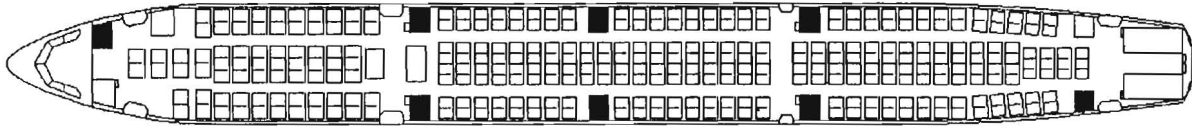


Abbildung 2: Schematisiertes Kabinenlayout eines A340.

Nutzt man einige Eigenschaften von räumlichen Constraints aus, so kann man Constraints-Checks sparen, indem man vorher Schlüsse über ihre Notwendigkeit zieht. Beispielsweise können Symmetrieeigenschaften oder die Ungerichtetheit von Constraints ausgenutzt werden. In [Freuder 94] finden sich allgemeine Vorschläge, die in AC-7 münden, die zum Teil auch bei räumlichen Constraints ausgenutzt werden können. Bei diesem Verfahren muß aber abgewogen werden zwischen dem Aufwand für die Inferenzen und dem Aufwand für die Constraint-Checks.

Wir schlagen für Domänen, wie sie in Abschnitt 2 vorgestellt wurden, ein zusätzliches Verfahren vor, das räumliche Nähe ausnutzt, um das Zeitverhalten dadurch zu verbessern, daß wesentlich weniger Constraints instantiiert werden.

5 Nutzung der Lokalität zur Verbesserung des Verhaltens

Rufen wir uns das Überlappungsverbot ins Gedächtnis, das fordert, daß sich kein Einrichtungsgegenstand mit einem anderen überlappt. So wie das konzeptuelle Constraint in Abschnitt 3 formuliert ist, führt es dazu, daß bei 200 Objekten $0,5 * 2600 * 2599 = 3.378.700$ Constraints instantiiert werden. Durch Einführung einer zusätzlichen Struktur, welche die Objekte in eine Matrix einordnet, läßt sich erreichen, daß die Anzahl der zu instantiierenden Constraints enorm sinkt. Die Zellen dieser Matrix sind dabei keineswegs quadratisch. Durch überprüfen der Wissensbasis wird ermittelt, wieviele Objekte in einer Konfiguration enthalten sein können, wie breit und wie lang die Grundfläche ist und entsprechend dieser Daten wird ein Array erzeugt, das so gestaltet ist, daß nach Möglichkeit nur ein Objekt in jeder Zelle enthalten ist. Diese Arrayzelle repräsentiert dabei keineswegs ein Quadrat, sondern ein Rechteck. In der Kabinenlayoutdomäne beispielsweise ist ein solches Rechteck 40 Zoll lang und 70 Zoll breit. Mit dieser Struktur ist es bei der Instantiierung eines Einrichtungsgegenstandes nicht mehr nötig ein Constraint für jeden vorhandenen Einrichtungsgegenstand zu instantiieren, sondern nur noch für jene Einrichtungsgegenstände, die eine Arrayzelle gemeinsam mit dem instantiierten Einrichtungsgegenstand teilen. In unserem Beispiel sind das ca. 4 Objekte - das hängt davon ab, welche Region der Passagierkabine, mit wievielen Möglichkeiten zur Plazierung eines Objektes wir betrachten. Dadurch müssen noch ungefähr $200 * (0,5 * 52 * 51) = 265.200$ Constraints instantiiert werden.

Bei anderen Constraints, die beispielsweise ein Nebeneinanderstehen verbieten, ergeben sich ähnliche Vorteile. Dürfen beispielsweise keine Toiletten neben einer Küche stehen, dann muß nicht mehr bei jeder Instantiierung einer Küche ein Constraint für jede vorhandene Toilette instantiiert werden, sondern nur noch für jede Toilette, die eine benachbarte Zelle belegt. So wird u. U. gar kein Constraint instantiiert, wo sonst ca. 10 instantiiert würden.

Indem die Lokalität von Objekten ausgenutzt wird und so die Menge der zu betrachtenden Objekte auf benachbarte Objekte in einem begrenztem Umkreis eingegrenzt wird, verringert sich die Anzahl der Constraints in dem Constraint-Netz je nach Aufteilung des Arrays um den Faktor 10 - 20.

In Abschnitt 3 ist die Lokalität in ähnlicher Weise für das Constraint *Freiraum-vor-der-Kueche* ausgenutzt worden, ohne das eine zusätzliche Struktur nötig war. Dort wurden vier Slots für den/die nächsten Nachbarn eines Objektes in jeder Richtung eingeführt. An diesem Punkt stellt sich die Frage, ob dieser Ansatz nicht ausreicht. Die Antwort ist: "Nein." Er ermöglicht es beispielsweise nicht, alle Objekte zu betrachten, die im Umkreis von 10m stehen. Das ist aber für vorgeschriebene Mindestabstände nötig. Beide Ansätze zielen in die selbe Richtung - Verringerung der Komplexität durch Verringerung der zu betrachtenden Objekte - und ergänzen sich. Der Eintrag z. B. des Vordernachbarn in den entsprechenden Slot kann nämlich mit Hilfe des Belegungsarrays wesentlich leichter geschehen als durch Betrachtung aller Objekte.

6 Zusammenfassung

Ausgehend von zwei Domänen haben wir ein Repertoire von räumlichen Relationen zusammengestellt, daß in Domänen mit achsenparallelen Rechtecken zur Beschreibung räumlicher Zusammenhänge benötigt wird. Durch Bereitstellung dieser Relationen als vorgefertigte Konstrukte für den Einsatz in einem Constraint-System ist es einem Wissensingenieur relativ leicht möglich, räumliche Zusammenhänge zu formulieren. Das ist aber nur der eine Teil der Aufgabe - das Wissen zu repräsentieren. Der andere Teil - nämlich dieses Wissen zu verarbeiten - erweist sich als der schwierigere. Bei Problemen, die nicht nur 8 Damen enthalten, treten Laufzeiten zutage, die solche Systeme in der Praxis schnell an ihre Grenzen stoßen lassen. Wir haben daher speziell für räumliche Constraints einen Vorschlag gemacht, der die Anzahl der zu bearbeitenden Constraints stark reduziert. Dafür wird ein zweistufiges Verfahren benutzt, daß mit Hilfe eines Belegungsarrays erst grob prüft, welche Objekte überhaupt in Frage kommen, bevor in der zweiten Stufe im eigentlichen, exakten Verfahren die benötigten Constraints instantiiert und ausgewertet werden. Durch dieses zweistufige Vorgehen von grob nach fein werden räumliche Probleme der aufgeführten Art handhabbarer. Der Preis der zusätzlichen Struktur und ihrer Verwaltung, insbesondere des Füllens und Löschsens von Zellen wird bei weitem aufgewogen durch die Ersparnis an zu instantiierenden Constraints. Allerdings ist der Fortschritt nur ein kleiner. Eine Verbesserung um ca. den Faktor 10 im Zeitverhalten bedeutet vereinfacht gesagt, daß wir statt mit 20 Objekten in einer Küche jetzt mit 200 Objekten in einer Passagierkabine umgehen können. Es bleibt aber festzuhalten, daß räumliche Beziehungen nach wie vor ein großes Problem darstellen. Wissensingenieure müssen sich beim Einsatz von Constraint-Systemen zur Bearbeitung von räumlichen Problemen die Grenzen solcher Systeme klar machen und jede Möglichkeit bei der Wissensmodellierung nutzen, um die zu berücksichtigende Zahl von Constraints zu drücken.

Literatur

- [Baykan90] Baykan, C. A., Fox, M. S: *Constraint Satisfaction Techniques for Spatial Planning* in: Hagen, P. ten, Veerkamp P. (eds.) *Intelligent CAD Systems 3: Practical Experience and Evaluation*
Springer, eurographics Seminar Series, 1990
- [Baykan91] Baykan, C. A., Fox, M. S: *WRIGHT: A Constraint Based Spatial Layout System* in: Tong, C., Sriram, D. (eds.) *AI in Engineering Design*
- [Cordier91] Cordier, M., Bessiere, C.: *Arc-Consistency and arc consistency again*, AAAI-93, 108 - 113
- Informatik-Fachberichte Nr.266, Springer, 1991
- [Cunis91] Cunis, R., Günter, A., Strecker, H. (Hrsg.): *Das PLAKON-Buch*
Informatik-Fachberichte Nr.266, Springer, 1991
- [Freuder94] Freuder, E. C.: *Using Metalevel Constraint Knowledge to Reduce Constraint Checking*
in: Workshopbeitrag auf der ECAI 94, 1994

- [Günter91] Günter, A., Dörner, H., Gläser, H., Neumann, B., Posthoff, C., Sebastian, H. J.: *Das Projekt PROKON -- Problemspezifische Werkzeuge für wissensbasierte Konfigurierung PROKON-Bericht Nr. 1, 1991*
- [Güsgen91] Güsgen, H. W.: Feasible Reasoning about Space
in: Proc. 5. Workshop Planen und Konfigurieren, 1-15, Hamburg 1991
- [Hernández90] Hernández, D.: Using Comparative Relations to Represent Spatial Knowledge
in: Hoepfner, W.: Workshop Räumliche Alltagsumgebungen des Menschen, 69-80, Universität Koblenz-Lindau, 1990
- [Kopisch91] Kopisch, M.: *Konfigurierung der Passagierkabine eines Verkehrsflugzeuges*
in Proc. 5. Workshop Planen und Konfigurieren, Hamburg, 176-179, Hamburg 1991
- [Kopisch91a] Kopisch, M.: *Expertensystemgestützte Konfigurierung der Passagierkabine eines Verkehrsflugzeuges*, Diplomarbeit, Universität Hamburg, Fachbereich Informatik, 1991
- [Kopisch92] Kopisch, M., Günter, A.: *Configuration of a Passenger Aircraft Cabin based on Conceptual Hierarchy, Constraints and flexible Control*
in: Belli, F., Radermacher, F. J (Eds.):
Industrial and Engineering Applications of Artificial Intelligence and Expert Systems
Proc. of 5th International Conference IEA/AIE-92, 421-430, Springer, 1992
auch deutsch in: Messer, T., Winklhofer, A. (Hrsg.): Proc. 6. Workshop Planen und Konfigurieren, 1-10, München 1992
- [Kopisch93] Kopisch, M.: *Spatial Relations for Configuration Tasks in General and for the Cabin Layout of the Airbus A340 in Particular*
in: Chung, P. W. H., Lovegrove, G., Ali, M.(Eds.):
Industrial and Engineering Applications of Artificial Intelligence and Expert Systems
Proc. of 6th International Conference IEA/AIE-93, 450-457, Gordon and Breach, 1993
- [Mackworth77] Mackworth, A. K.: *Consistency in Networks of Relations*
in: Artificial Intelligence 8, 95 - 118, 1977
- [Mackworth85] Mackworth, A. K., Freuder, E. C.: *The Complexity of Some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems*
in: Artificial Intelligence 25, 65 - 74, 1985
- [Mohr86] Mohr, R. Henderson, T.: *Arc and path consistency revisited*
in: Artificial Intelligence 28, 225-233, 1986
- [Schmitgen93] Schmitgen, S.: *Räumliche Fragestellungen bei der Konfigurierung*
Dissertation, Universität - GH - Paderborn, Fachbereich Mathematik/Informatik, 1993

PuK in der Praxis

Scheduling High-grade Steel Making

Jürgen Dorn and Reza Shams
Christian Doppler Laboratory for Expert Systems
Vienna University of Technology
Paniglgasse 16, A-1040 Vienna
email: dorn@dbai.tuwien.ac.at
Tel.: ++43-1-58801-6123
FAX: ++43-1-5055304

Abstract

The paper describes a scheduling system for a high-grade steel making plant. Because very specialized steel grades are produced the variety of the chemical analysis of the heats is very high and the lot sizes are small. One of the most constraining factors in the production is the infiltration of chemical elements between consecutive heats in the electric arc furnace. To find a schedule that violates no compatibility constraints is theoretically in the worst case an NP-hard problem.

An earlier attempt to solve the problem with conventional software failed due to this complexity. Some difficult to schedule jobs remained unscheduled. The expert system solves the problem because it first determines the criticality of jobs and applies then a heuristic based on the criticality values to guide the search for a feasible schedule. Important is further the system's capability to use and relax constraints locally. The expert system was fielded successfully in 1992 at Böhler Uddeholm in Kapfenberg.

1 Introduction

Scheduling of production processes involves the simultaneous consideration of jobs to be performed and the resources which they require. Usually a job is identified with a deliverable product which has to meet a certain quality. Associated with each job is some formal specification of the product to be produced. In the steel making shop the products are slabs or ingots to be cast. Their specification consists of a chemical analysis demanding certain amounts of alloying metals and some constraints on the maximal amount of other chemical elements. Furthermore, the size of the produced slabs and ingots and the casting procedure is specified.

A process plan describes how a certain product or a group of similar products can be processed. Typically a set of operations has to be performed with certain resources. Often technical constraints restrict the temporal sequence of these operations. If these operations always have to be processed in the same sequence, one speaks of a flow shop. Steel making is usually organized as a flow shop and the sequence on one unit implies the same sequence on other units because steel may not cool down too much between operations.

When a number of jobs are to be executed together, their resource requirements imply additional ordering constraints, which prohibit simultaneous demands on non-sharable resources. A scheduler must consider both the explicit ordering constraints imposed by the process plans and the implicit ordering constraints derived from the availability of resources. In our example two production lines share the continuous caster thus forcing synchronizing constraints. Finally, the scheduler must take into account release and due dates, working hours, set-up times, and maintenance intervals.

Sometimes scheduling problems are over-constrained by all these restrictions. In scheduling theory one differentiates therefore between hard and soft constraints¹. If all hard constraints are observed the resulting schedule is feasible. The satisfaction of the soft constraints like for example the due dates is then subject of optimization. Thus one differentiates all feasible schedules by their aggregated lateness and searches for the schedule with the minimal lateness. In many cases such evaluation functions are sequence dependent and there exist no simple constructive algorithm for more complex problems. In the worst case each possible sequence must be generated and evaluated to find the optimal solution. Already very simple problems are thus intractable as was shown by Garey and Johnson². Although flow shop problems seem to be easier than general job shop problems, the theoretical complexity is the same. The hard problem is always the sequencing and not the temporal allocation problem. However, in practice, problems are often easier because some sequences are already given by hard constraints. In our application some sequences are not allowed and consequently we try to find jobs that are constrained very strong by such constraints to sequence first these jobs.

Most scheduling theory (e.g. French¹) is dominated by approaches to find optimal solutions concerning the average flow time of jobs, the makespan or the aggregated lateness of all jobs. In practice such evaluations are of very limited use. Usually the difference between the best solutions is not so great and many durations that are assumed during planning are only vaguely known thus leading to a questionable optimum. Furthermore, unexpected events in the shop floor invalidate any such schedule regularly. Consequently many application-oriented researchers like e.g. Ow and Smith³ propose some kind of opportunistic scheduling. Here not theoretically optimal solutions are looked for, but good ones that are also robust in respect of possible disruptions in the shop floor. Moreover, in our application these quantitative eval-

uation criteria play only a secondary role. Steel making experts neglect the existence of one optimal solution. They give hard constraints what sequences are not allowed and they have rules (or soft constraints) that say what are good sequences. The knowledge on the quality of solutions only regards local decisions and cannot be applied so easily on the whole schedule.

Improvements in the organization of the steel making processes promise high returns of investment, because the capital expenditure in the steel making industry is very high. Consequently automation of scheduling these processes has attracted many solutions. Kerr⁴ gives a good survey on scheduling systems in the steel industry of the last decade. In this study it became clear that our application is unique in contrast to other scheduling systems in the steel industry because of the great diversity of steel qualities that are produced. For example the SCHEPLAN system described by Numao and Morishita⁵ optimizes the throughput of a steel making shop that produces with many casters. The compatibility of consecutive jobs plays a less dominant role because not so many different steel qualities are produced and different qualities can be cast on different casters. From an organizational point of view, one could ask to restrict the diversity of products in our application. Woodyatt *et al.*⁶ report on a solution at Bethlehem steel in which they try to find for a set of customer orders a minimal set of products that fulfill the customer requirements best. However, experts of our application deny such a solution for their company.

To our experience also many other technical and capacitive constraints are stemming from organizational issues. For example, the space for solidification of big forging grade ingots is restricted thus disabling the possibility to produce more than one of these ingots per day. However, it was not our task to improve this organization but to find a good solution under the given organizational constraints.

In a joint project between the *Alcatel Austria-ELIN Research Center* and the *Christian Doppler Laboratory for Expert Systems*, an expert system was developed for the *Böhler Uddeholm* in Kapfenberg (Austria). The system supports the technical staff in the steel making shop in generating schedules for steel heats for one week. The system was implemented with PAMELA a rule-based system developed by the *Alcatel Austria-ELIN Research Center*.

2 The Problem

Böhler Uddeholm is one of the most important European producers of high-grade steel. *High-grade steel* is crude steel refined with *alloying metals* like manganese, tungsten, chromium, or others. These alloying metals have desired effects like compression strength, impact strength, and many more. The company supplies steel to the aircraft industry, for turbine blades in power plants, and other high-tech applications. These customers demand a high quality and relative small amounts of steel goods.

A production planning system compounds orders to jobs and chooses required process plans. Furthermore, the system determines weeks when these jobs shall be processed by regarding the capacity of the steel making shop. It works out for each week two sets of jobs to be processed on both production lines. Usually the first shift in the steel making plant starts Sunday evening and the last shift ends Saturday. Sometimes fixed sequences for two or three jobs are given to facilitate the scheduling in the subsequent plant. The task of the scheduler is to find a possible sequence for all jobs with violating as few compatibility constraints between jobs as possible and to allocate resources over time without violating temporal and capacity constraints. The result of this scheduling process may be that some orders are reject-

ed and shifted to the next week. To reduce the number of rejected orders, general rules that explain what combinations of orders may be produced in one week are already regarded by the production planning system.

The steel making plant produces slabs or ingots of a certain quality for different subsequent plants like the forge, the rolling mill, or the foundry. The destination is important for the scheduling process, because the working hours of these plants must be considered. Sometimes products are stored for several days in intermediate stock yards, because the next plant cannot process the jobs in the same sequence as the steel making plant. The differing sequencing criteria of jobs cause considerable costs for the company. Moreover, since the steel cools down it must be warmed up again in the next plant. To reduce costs and to improve the quality, some orders have due dates.

2.1 The Steel Making Process

Pig iron produced in *blast furnaces* contains usually more than 4 % carbon and is therefore brittle. To get a deformable product, *steel* is produced by reducing carbon in pig iron down to 0.2 %. For many high-tech products the quality of steel must be even higher and reductions of different chemical elements that are still in the steel must be achieved. If the steel would be delivered immediately from the blast furnace this reduction process could be made in LD-converters. Since in Kapfenberg no blast furnace exists the crude steel and *scrap iron* are melted in electric arc furnaces. To reduce costs scrap iron with high percentages in desired alloying elements is used.

The steel making shop in Kapfenberg consists of three main production lines that share some aggregates. For every steel quality there is a process plan that describes which operations must be performed on which aggregates to produce the quality. These operations and sequences must be only replanned when a failure occurs.

The steel making process starts with the charge of crude steel and scrap iron in one of the three *electric arc furnaces* (EAF). The filling of one furnace is called a *heat* and it already contains the main alloying elements. The furnaces have different capacities from 17 t to 55 t. The duration of melting one heat depends on the size and the ingredients but also on external causes. Since the furnaces consume much electric energy, they are sometimes switched off due to voltage peaks. Up to five hours can be required for melting, but usually two to two and a half hour are enough. A fixed set-up and a maintenance interval of twenty minutes are included in this interval.

The liquid steel is poured into *ladles* that are transported by a crane to a *ladle furnace* (LF). If the preceding heat has a long processing time in the ladle furnace, the current heat must wait. This slack time may not exceed two hours. The next step is a *heat treatment* in the ladle furnace where the fine alloying takes place and whose duration is usually about the same as melting. Later a special treatment may be performed in the *vacuum oxygen decarburation* (VOD) unit or in the *vacuum decarburation* (VD) unit. The VD-unit can be converted into a VOD-unit. This conversion takes about three to four hours.

The last step in the steel making shop is the processing of the steel in a *horizontal continuous caster* (HCC) to form *slabs* or the casting of it into *moulds* to form *ingots*. The teeming rate for the HCC is about 50 t / h. If the casting format must be altered, a set-up time must be considered, too. For casting ingots, space in one of the four *teeming bays* (TB), where ingots can solidify in the moulds, is required. Normally the solidification time for ingots in hours is

half as much as the weight of the ingots in tons. Thus an 52 t ingot needs one day and for an ingot of 1 t half an hour is sufficient.

The Böhler-Electro-Slag-Topping (BEST)-technology is a special casting technology for big ingots. These ingots are treated additionally in the BEST-unit and for them only one place in the teeming bays (TB 1) exists. The ESU-unit is used for some extra-clean steel qualities. All aggregates and the routings for heats are shown in Figure 1. Since the third production line with EAF5 and ESU-unit is used only seldom it is not regarded further in scheduling.

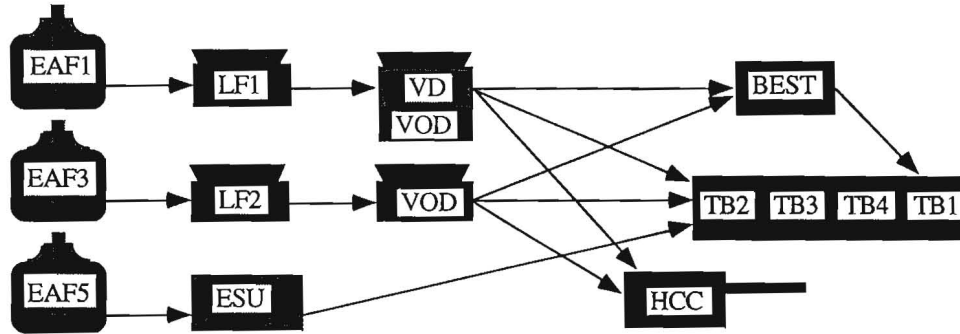


Figure 1: Aggregates in the Steel Making Shop

2.2 Constraints in Scheduling

The construction of schedule for the steel making shop underlies several constraints. These constraints are often vague, and they may conflict with each other. An engineer of the steel shop has no pretension to generate optimal schedules because he knows that the uncertainty in the shop floor execution would break the optimality of this plan. The engineer can decide that some schedules are better than others, but he cannot give an algorithm how to construct optimal schedules.

2.2.1 Compatibility Constraints

The main problem in scheduling is that residuals of one heat in the electric arc furnace may pollute the next heat. The engineers use as a rule of thumb that 3 % of a chemical element in a heat remain in the wall of the electric arc furnace and 3 % of the difference of the elements in two consecutive heats will be assimilated by the second heat. Of course, the 3 % are always on the safe side and in some cases the expert relaxes this factor.

Assume a heat h_1 contains 20 % nickel and a heat h_2 that is processed afterward in the furnace should have 4 % nickel. The second heat will take approximately

$$\frac{20 \% - 4 \%}{100 \%} \cdot 3 \% = .48 \% \text{ nickel from the wall.}$$

When scrap iron is inserted in the furnace this amount is considered and only 3.52 % of nickel are taken. If the second heat may contain only up to .25 % of nickel, then this sequence of heats is not allowed.

Although the rule is effective for 42 chemical elements, usually only eight main elements are considered. These elements are nickel (Ni), chromium (Cr), manganese (Mn), molybdenum

(Mo), tungsten (W), cobalt (Co), vanadium (V), and iron (Fe). The amount of the other 36 elements is usually very small and restrictions are no scheduling problem. Since not all elements react uniformly, exceptions exist that must be handled separately. Due to the diversity of qualities, these compatibility constraints often cannot be met and some heats are destroyed. They must be melted again and can be reused for other less critical orders. Although these compatibility criteria hold for every unit in the production process inclusive ladles, usually as a heuristic only the electric arc furnace constraints are observed. The ladles are no great problem since three of them are used in turn. On the other side, chemical constraints are even more stringent on the caster. However in practice this makes no problem since only few heats are cast on the caster.

Besides avoiding such a destruction by infiltration, it is expensive to waste rare elements. If one job demands a high percentage of an expensive element like cobalt, the next job should use as much of this residual as possible.

2.2.2 Temporal Constraints

In principle, the cast steel can be stored in intermediate stocks before it is further treated in one of the subsequent plants. However, then the steel must be reheated which causes considerable costs. For some qualities it is also better if it does not cool down. Consequently appointments are made between steel shop and subsequent plants that must be held within a tolerance of ± 2 hours. The average number of jobs with such due dates is about 10 %. Of course, these constraints are not really hard since they may be relaxed through negotiations with the subsequent plant. However, it is desirable to hold these due dates to reduce the time needed for negotiations. Furthermore, also these subsequent plants have to regard also compatibility constraints that are different to those of the steel shop. For some jobs no appointments are made, although their subsequent treatment should be done immediately after casting. These jobs should not be scheduled at the end of the week because usually the subsequent plants are not working then.

Some jobs with difficult treatments should be performed during day shifts, because an engineer should supervise these jobs. Usually the treatments in the aggregates behind the electric arc furnace are shorter than the duration of the melting. However, for very high qualities the duration in the ladle furnace is longer. Finally, the scheduler must guarantee that waiting intervals between operations may not exceed a certain limit. An objective for the production is to have these intervals as small as possible. This results also in a minimization of the flow time of jobs, which will reduce production costs. However, this objective is only a secondary goal function. The main objective is always to have as few destroyed heats as possible. Since this objective cannot be determined exactly, the secondary goal function is regarded seldom.

2.2.3 Capacity and Related Constraints

About 70 % of the jobs are cast into ingots and the remaining 30% are processed on the continuous caster. For both casting technologies some capacity constraints have to be regarded:

The continuous caster can be supplied with steel from both production lines. If two subsequent heats should be cast both on the continuous caster and have almost the same steel quality, the caster should process both jobs continuously without interruption. If the steel arrives

later a set-up is required which will be expensive. Jobs that are cast continuously are called *serial castings*. Such a serial casting is only possible if the steel is supplied from one production line. If on both production lines the caster is required then these jobs must be scheduled so that they do not conflict.

Big *forging grade ingots* (> 19 t) and BEST-ingots have a dedicated storage place that is very limited. If a 52 t ingot is cast the ingot occupies the whole space during solidification. Since such an ingot requires about one day for solidification only one ingot can be produced per day.

Casting steel in ingots is a very labor-intensive procedure. Workers first set up the moulds, then each mould is filled with a casting crane, and after solidification the moulds are stripped off. If a heat is cast in many small ingots (e.g. 50 ingots), the load for the workers that set up and strip off the moulds is larger than for few large ingots, because the handling of every mould takes approximately the same time. In principle two heats from the two production lines can be cast at the same time, because two cranes exist and the workers are able to do this. However, an objective in scheduling is to have a constant load distribution over the planning horizon for the workers. It should be avoided that at the same time two heats must be cast in small ingots and an equilibrium distribution of work between shifts should be accomplished.

If an amount of steel that does not make a full heat is ordered, it can be combined with another of compatible quality forming a *double-* or *triple-casting*. This means that only a part of the heat is poured from the arc furnace into the refinement ladle. Durations of treatments in the refinement and casting process will be shorter in this case.

2.3 Heuristics used by the Experts

The experts of the steel making shop apply heuristics to construct schedules. These are used to master the complexity of the construction and to control the uncertainties existent in the application. If no schedule is found, some constraints are relaxed since it is known that usually there will be enough freedom during execution to correct the violated constraints. Again, this relaxation is controlled by heuristics.

- **Bottleneck Resource**

Although steel making is in principle a flow shop it is not sufficient to schedule jobs only on one resource. However, to restrict the search heuristically it is promising to sequence first the operations on one bottleneck resource and to check then the constraints on the other resources. In most steel making applications this resource is the continuous caster, because it is very sensitive to compatibility constraints and must be delivered in time to cast without interruptions. In our application most heats are cast into ingots. For ingots there are no compatibility constraints and temporal constraints are not so hard. Since the infiltration of heats by predecessors is the most constraining issue, a sequence of heats in the electric arc furnaces is planned. Schedules constructed by the experts do not contain temporal slots for continuous casting although they regard also the constraints on the caster.

- **Alloying Cycles**

An important concept for the scheduling process is the *alloying cycle*. This is a series of heats in which the amount of a chemical element is slowly decreasing. For example, the concentration of chromium could decrease over several heats from 26 % to 10 %. Unfortunately due to other constraints it is not possible to order all jobs in one alloying cycle. Several alloying cycles for one element may occur in sequence and several alloying cycles for different elements may overlap in time. The following figure illustrates the alloying cycles for nickel and chromium based on real data of twelve jobs.

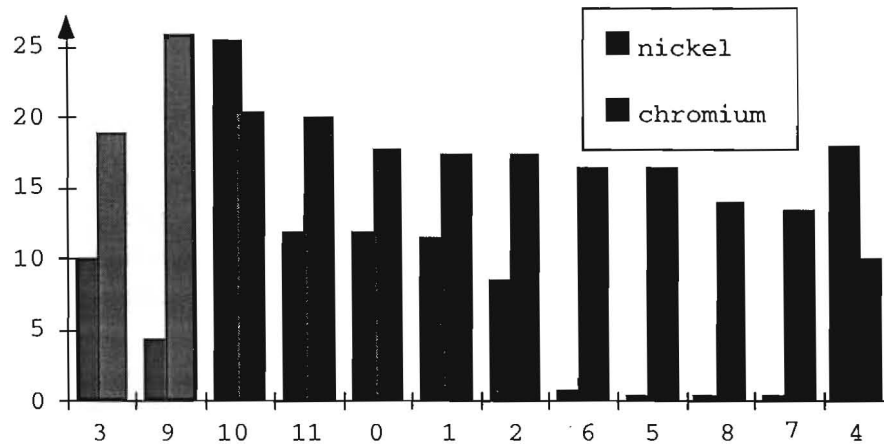


Figure 2: Overlapping Alloying Cycles

Heats should be scheduled in this order. Additionally, the quality that is produced at the end of a week affects the heats in the beginning of the next week. One important task driven by heuristics is the recognition of possible alloying cycles. This is supported by the experience of the engineers in the plant: the amount of an element should decrease slowly over a sequence of heats, but can increase very fast. Additionally, the number of peaks in this curve of a saw blade is kept to a minimum if possible.

- **Scheduling BEST-Ingots**

BEST-ingots introduce some problems. Typically there are groups of such jobs with all of them having the same chemical quality requirements. They are usually forging grade ingots, should be delivered hot to the forge, and are low-alloyed, which means their amount of alloying metals is very low. From the compatibility view point they should be produced in sequence. Unfortunately, they solidify slowly and there is only one place for them in the teeming bay.

Therefore BEST-ingots are scheduled at a rate of only one per day. Since they are low-alloyed, the furnaces will probably be high-alloyed in the intervening periods. Since they should be delivered hot, they cannot be scheduled before weekends or public holidays. Therefore at weekends the furnaces shall be “high-alloyed”, and as a consequence these orders contain many small ingots.

- **Conflicts on the Continuous Caster**

The HCC-unit is most economical if several heats with the same quality and format are cast continuously without breaks. In case that such a group of jobs exists, these jobs should be scheduled one after the other. However, all jobs of this group must be in time for their casting. After the processing of some jobs on the caster, a maintenance interval must be scheduled. Additionally, a set-up time of some hours must be reserved when a format conversion has to be performed.

Since the durations of the operations in the steelmaking process are very uncertain, the engineers prefer to charge the caster only with heats from one furnace. However, they sometimes change furnaces once per week. In this case, they decide when this change should take place before starting to schedule single jobs. Further, they accept only one format conversion. The expert schedules this conversion also a priori. Later, each job that requires the HCC-unit can be assigned to one part of the week.

Of course all these heuristics cannot be applied in a fixed sequence. It is very important to find good points to switch between them.

3 The Solution

The solution to the described problem is characterized by two design decisions. First, we have tried to model as far as possible the way how the human experts solve the problem. They do not consider any global evaluation function but only local decisions about good practice. Furthermore, they use heuristics to construct a schedule and they try to avoid backtracking. Second, the knowledge on compatibility of consecutive heats and temporal constraints are vague. Although it is better to have a difference in elements that is small sometimes it is also allowed that the difference is greater than 3%.

The expert system gets two list of jobs for two production lines from the production planning system. Each list contains about 35 jobs. The system needs between five and ten minutes to generate a solution for both production lines. Some control information tells the user what the system is doing and the user can break the search anytime. After a break or after completion the user can change manually the schedule and the system checks afterward the schedule again for any constraint violations. The problem solving process consists of three phases:

- in a pre-processing phase the system evaluates how difficult it is to schedule individual jobs in the actual planning horizon
- in the scheduling phase jobs are allocated starting with the most difficult one
- in a post-processing phase constraint violations are repaired by modifications of the schedule

3.1 Evaluation of Jobs

The given jobs are evaluated to determine any bottlenecks for the current week. If for example many jobs are to be cast on the continuous caster, the caster is a bottleneck for the current week. The system considers therefore these jobs as early as possible. If many big forging grade ingots are to be cast these jobs are the most critical. If for example orders for more than

five big ingots exist, it will be impossible to schedule all in one week. If exactly five exist the list is characterized as having *many* big ingots. We use symbolic variables like *many* and *few* so that a modification in the plant can be easily adapted in the system. For the scheduling strategy it is important whether many big ingots exist but not how many these are. The following data structure is build up and will be used later to characterize individual jobs.

```

JobList
  ContinousCasting      (too_many, many, normal, few, no)
  BigForgingGradeIngots (too_many, many, normal, few, no)
  DoubleCastings        (too_many, many, normal, few, no)
  SmallIngots           (too_many, many, normal, few, no)
  DeliveryDates         (too_many, many, normal, few, no)
  Runaways              (too_many, many, normal, few, no)
  ConstantElements      (Enumeration of Elements)
  VaryingElements       (Enumeration of Elements)
  AlloyingCycles        (Enumeration of Elements)

```

Runaways are jobs that demand one element in an unusual amount. For these jobs it is sometimes impossible to find a successor or a predecessor. For example, sometimes steel with a large percentage of cobalt is ordered. If all other heats do not allow cobalt, an additional heat must be scheduled after the heat that “cleans” the furnace. Apart from this, the heat can be useless.

A list of one furnace can not be evaluated independently of the other lists because the spatial constraints concerning the caster and the teeming bay effect both production lines.

After characterizing a given set of jobs, we assign a criticality value to each job that describes its relative difficulty. Thus a job whose allocation is constrained strongly gets a high criticality value and will be scheduled early. Whereby early does not mean that it is scheduled early or left in the schedule but we decide early where to schedule it. The best place in the schedule may be late. A job gets one of the attributes *very difficult*, *difficult*, *normal*, *easy*, or *very easy*.

Very important for the construction of good schedules is the compatibility of heats. To facilitate this, we construct a matrix of successor / predecessor entries that show how good two jobs fit after each other concerning their chemical elements. Now it is easy to see which jobs have many good predecessors and successors. Jobs that can be combined only with one other job are classified as *very difficult* to schedule. Thus we are able to schedule first *very difficult* jobs that would otherwise remain unscheduled, because no possible neighbor remains. The following graphic shows such a matrix for twelve jobs.

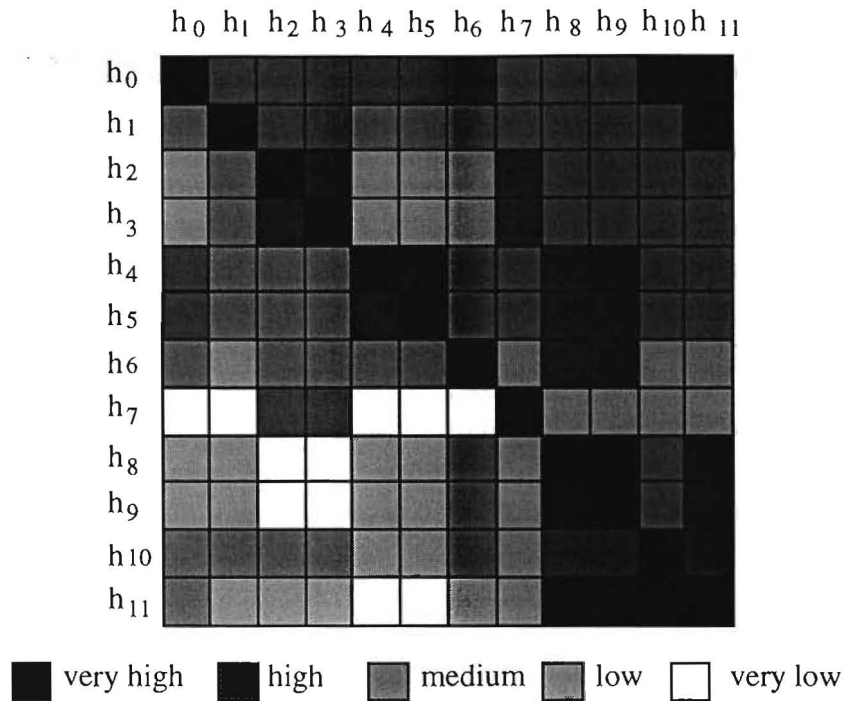


Figure 3: The Compatibility Matrix

As can be seen from the matrix the compatibility relation is not symmetric. The matrix is used to decide whether two jobs *can* be scheduled after each other and to decide if two jobs *must* be scheduled after each other. How this matrix is used will be seen in the next subsection.

3.2 Scheduling of Jobs

As described earlier, the experts primarily regard heats in the electric arc furnaces. Consequently also the expert system constructs a schedule for the furnaces. Although the scheduling problem is mainly a sequencing problem, a pure list is not sufficient as a representation for the schedule because temporal constraints must be checked. Although the users are not interested in exact time points, the system may not violate simple temporal constraints. Thus we represent a schedule as a list of temporal intervals with predefined length. As a general rule it is said that the loading of the furnaces start in a 3:3:4 cycle. This is a general assumption that will be refined later if the heats are allocated.

If we have a job with a due date we compute first its most likely flow time to get its optimal start time. If this start time is Thursday 1 pm, it will be allocated as 22 nd job at 12 am as it is sketched in Fig 4.

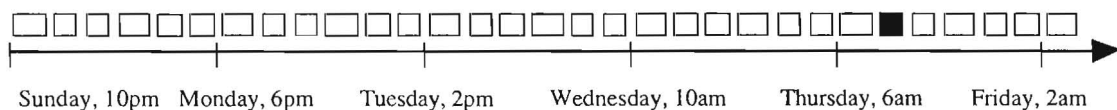


Figure 4: Representation of a Schedule

The scheduling of jobs is driven by the criticality value of jobs. Very difficult jobs are scheduled first. What the most difficult jobs are is dependent on all jobs that are to be scheduled. Typically, these are jobs that have a due date and are to be cast on the continuous caster

or heats that have a difficult chemical analysis and are to be cast into BEST-ingots. However, there is no order between the difficult orders and we select them randomly.

Besides the difficulty of jobs we also consider the difficulty to find a job for some interval. Of course in the beginning of the scheduling phase there are no difficulties for any intervals. However, after some jobs are scheduled it may become critical to fill the neighbors of the scheduled jobs since these must be filled with compatible heats. Most critical are usually gaps between two scheduled jobs because the heat that will be scheduled must fit to both neighbors. Thus if gaps appear the scheduling strategy is changed and it is looked for a job that can be inserted into the gap. The same happens if only one possible successor or predecessor exist. Finally easy jobs are scheduled in accordance with the alloying cycles.

The difficulty of jobs can change during the scheduling phase. A job that was first easy to schedule because it had many possible successors can become difficult because only one successor remains. This reasoning is explained in more detail in the next subsection.

3.2.1 Reasoning with Necessary Sequences

As mentioned earlier, a sequencing problem gets tractable if some sequences can be enforced. The list of orders contains already double castings which are predefined sequences that must be processed in this sequence. Serial castings on the continuous caster are not yet determined by the production planning system. Therefore one of the first tasks of the expert system is to decide whether serial castings can be found. The exact sequence of these heats can be open.

Furthermore, we can constrain the heat H_i to be an immediate predecessor of some heat H_j if H_i is the only allowed predecessor of H_j or if H_j is the only allowed successor of H_i . So if heat H_i will be scheduled, heat H_j will be scheduled too. If H_i is not yet scheduled and H_j shall be scheduled, then H_i must be automatically scheduled too. Moreover, if some heat H_k has only two possible successors H_j and H_l , then H_l can be associated to H_k because H_j is already given away.

Of course the usual case is that there is more than one possible successor as can be seen from the given matrix in Fig 3. Typically, there is one job that is highly compatible and one that is slightly worse. We choose heuristically the better one if no other constraints advise the second one.

These inferences are performed during the main scheduling phase. If one heat H_i is scheduled due to some other heuristic, the matrix is refreshed accordingly. If the scheduled heat is now successor of some other heat H_j , it is marked in the matrix that H_i cannot be anymore successor of another heat. Perhaps the consequence is that for a third heat H_k that had before two possible successors H_i and H_l only one possible successor H_l remains.

This kind of inference is performed concurrently with other inferences. Since no fixed deterministic course of inferences can be specified a realization with a procedural approach seem to be difficult. However, it can be realized very easily with a rule-based system.

3.2.2 Relaxing Constraints and Repairing Constraint Violations

If a job is introduced into the schedule several constraints have to be checked. If for example five big forging grade ingots are scheduled, these jobs will be scheduled early so that they can be distributed over the whole week with a distance that allows their solidification. Of course there may be other jobs yet scheduled so that compatibility constraints must be checked too.

The expert system tries to schedule jobs so that constraints are satisfied as good as possible. For the compatibility constraints this means that it tries to find neighbors that fit with *very high* or *high* compatibility. If for one heat exists only one successor with a high compatibility this heat is taken. However, if no *high* compatibility can be achieved, the expert system relaxes the constraint and a compatibility that is only *medium* is accepted. A further relaxation is possible after an acknowledgment of the human expert. If the expert system does not find a *medium* compatibility, it looks for a heat that is only *low* compatible. Then it asks the expert whether this sequence may be scheduled. If the expert denies the sequence, the job cannot be scheduled. The relaxation of temporal constraints is done in a similar way.

Sometimes the expert system decides unforced and allocates a job at an interval that is not so good. For example, if five big ingots are distributed over the whole schedule, only the distance but not their order is important. However, since these jobs are classified as difficult, their order will be fixed early. Later the schedule can be perhaps improved by exchanging two of these jobs. Thus the expert system after having scheduled all jobs, examines whether unscheduled jobs can be scheduled by first exchanging other jobs in the schedule. Furthermore, it examines constraint violations and tries to get rid of them again by exchanging again jobs.

3.3 Implementation

The expert system is implemented with PAMELA a rule-based forward-chaining expert system language especially suited for building real-time applications that was developed at the Alcatel-Elin Research Center⁷.

PAMELA-code is pre-compiled to pure C-code. Therefore the expert system can be executed on all machines that support C. PAMELA-code is also combinable with C-code. This facilitates the interface to the external environment and integration with procedural components. For example, the evaluation of jobs which is basically counting is realized with a C-program. It was also easy to integrate the expert system in the existent environment consisting of a Production Planning System and a Real-time Control System.

The user can interrupt the planning at any time to have a look at the schedule so far established or to propose a time for a job. This is due to PAMELA's ability to handle interrupts. The system answers whether this is possible resp. which constraints will be violated.

Although PAMELA is a forward-chaining programming language it is possible to freeze a state of the expert system and to return to this frozen state later on if the expert system or the user recognizes the chosen solution path as unfavorable. Thus, the user can investigate easily different schedules.

The rules of the expert system are split up in rule groups thus structuring the set of rules. Each group has its own task like selecting a job from the list, finding a possible place for a job in the schedule, or checking constraints. By activating or deactivating a rule group the system can be instructed dynamically whether this rule group should be used to contribute to the solution.

3.4 The User Interface

The user interface is very important for application where the user's reliance in the system is required. The user interface must show the result of the system in such a way that the user is

attracted by the main decisions of the system. In our case the system should show where problematic jobs are scheduled. Furthermore, the chemical compatibility is an important aspect that should be high-lighted.

Since the representation of the schedules for both furnaces would be too complex for the display, we have decided to show only one furnace. With a single button the user can switch between both furnaces. The following figure is a screen shot of the expert system.

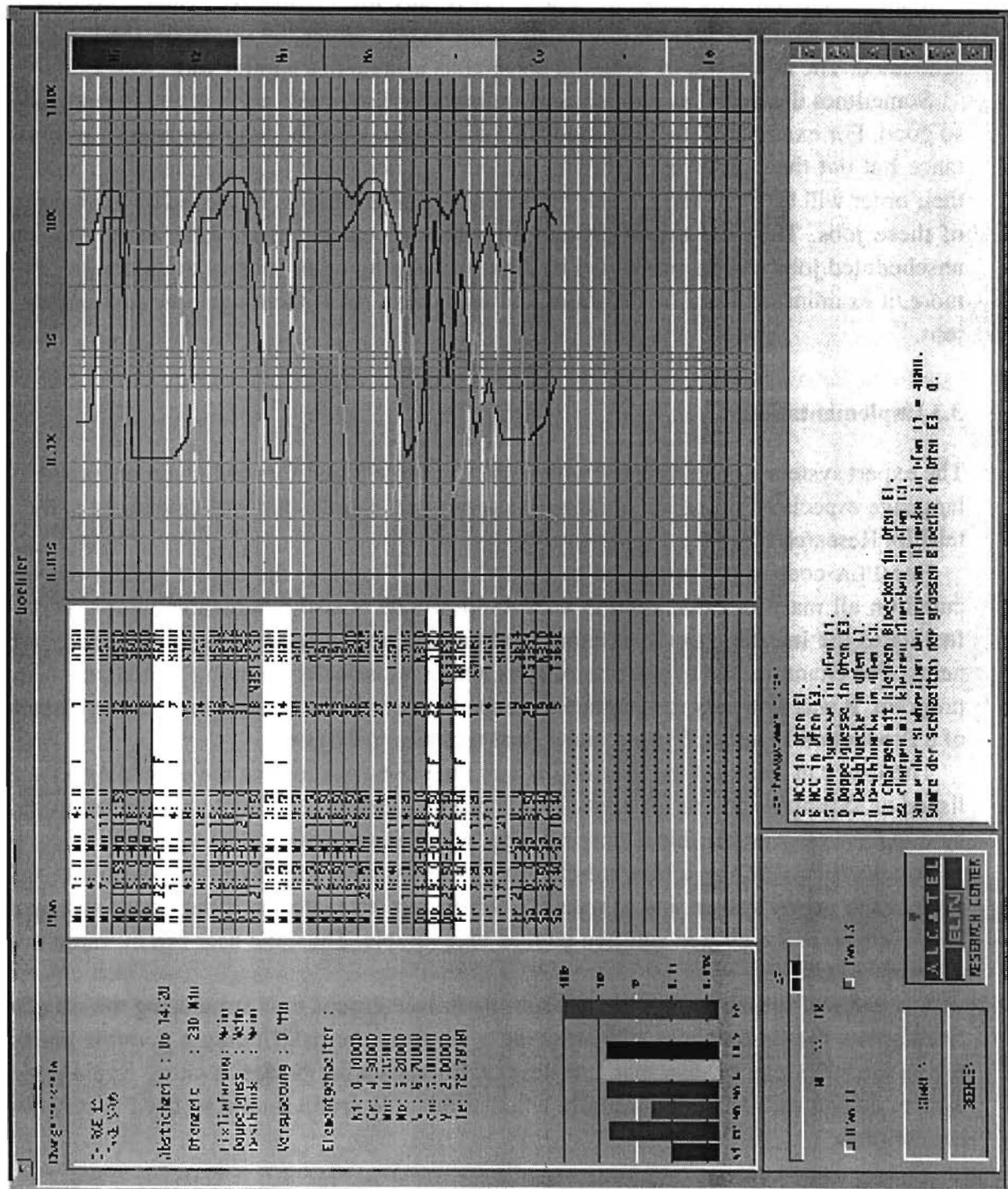


Figure 5: Screen Shot of the User Interface

In the middle column of the screen we see the list of all heats scheduled for one week. Each line describes the start time of a job, an abbreviation of its type, its number (identity) and the steel quality coded in a name. If the type is ‘‘C’’ the heat is cast on the caster. A ‘‘B’’ signifies a BEST-ingot, a ‘‘D’’ a double casting, and a ‘‘F’’ a job with a due date.

With the eight buttons on the right side we can choose chemical elements whose curves are then shown in the logarithmic diagram between buttons and job list. In the screen shot we have selected tungsten (yellow), nickel (red), and chromium (green).

If we select with the mouse device any single job in the job list its characteristics are shown in the left part of the system’s window. The attributes are its name, the time when the heat is poured into the ladle (Abstichzeit), its assumed time in the furnace (Ofenzeit), whether it has a due date (Fixlieferung), or it is a double casting (Doppelguss), or a BEST-ingot. A further line shows any lateness of a job in the case that it has a due date. Finally, all chemical elements are shown numerical and in a logarithmic diagram by columns.

On the bottom of the system’s window we see the button to switch between furnaces and a slide bar to switch between given data sets for different weeks. On the right side of the bottom is a small window for expert system messages that can be scrolled by buttons to look for old messages. This window is also used for textual answers of the user.

4 Evaluation

Before the expert system project was started, an attempt was made to schedule the heats with traditional software methods. This project was canceled because the program handled the constraints too rigidly. It was not able to relax constraints. It had scheduled a lot of jobs correctly that were easy to schedule, but eventually it stopped without having scheduled the difficult jobs. An exhaustive search is prohibitive since for 35 heats one has to examine 35! possible sequences.

Due to the complexity and uncertainty in the application, it was decided to try a solution with expert system technique. In 1990 a first prototype was developed. With this prototype the compatibility rules and the user interface were evaluated. In 1991 the actual expert system was developed and fielded successfully in the beginning of 1992.

The success is due to two reasons. First, and most important for the project was that the management and the experts were really interested in having such a tool. There was no problem of getting an expert or disbelief of the experts. The second, more technical reason is the ability to express knowledge more qualitatively and to neglect certain details that are not so important for the problem solution. We have defined classes of satisfaction degrees for constraints and have developed a problem-oriented solution method based on the concept to determine criticality values for jobs.

The main aims of the development were to relieve the plant engineers from work and to standardize the decisions. Before the system was installed the engineers were busy with planning for one day each week. However, their proper work is the supervision of the production. With the system a lot of annoying paper work is avoided and the planning time is reduced to half an hour. The solutions are generally acceptable for the engineers. Only in some specific cases where extraordinary steel qualities are to be produced the experts sometimes modify the solution of the expert system.

As a side effect, the experts can perform now more experiments or what-if-games to find good sequences. In any case, all schedules produced by the expert system must be acknowledged by the plant engineers because they are responsible.

The second aim seems to be more important for the company since a more deterministic decision framework supports quality assurance better. If a heat is destroyed due to infiltration this will be a reason to modify the appropriate rule. The rules of the expert system are thus a kind of law for the production.

With the knowledge acquisition some implicit knowledge of the experts was made explicit that was not known by the management of the plant. As a consequence of the project some rules were revised. However, a direct improvement of the production that can be measured was not recognized. Neither the flow time of jobs nor the percentage of destroyed jobs has been changed significantly. Perhaps such an improvement was achieved but due to organizational changes a valid comparison is impossible.

Almost two years have shown some drawbacks that should be overcome in a new version. One problem is modification of the production process. For example in 1992 a second strand for the continuous caster was erected. With this extension more jobs are cast on the continuous caster and with that also the scheduling bottleneck has changed slightly. Now also compatibility rules must be applied on the sequence of jobs on the caster. The adaptation of the software was possible by the insertion of a new rule group. However, the main problem was the strategy that must be adapted. Our first strategy was motivated by the experience of the experts, but the experts had no experience with the new situation. Thus we had to implement more common sense than experience.

A very important enhancement of our system would be the possibility to react on events in the shop floor that were not foreseen in the schedule. As described earlier there are many uncertainties in the production that invalidate any schedule regularly. At the moment the engineers could reuse the expert system, but they prefer to make small changes manually. The problem is that the expert system schedules from scratch and will produce a totally new schedule. However, this is not desired. What the engineers want is only a patch of the old schedule. In the actual practice the experts do not use the expert system for reactive scheduling. We have shown with some case data from this application how reactive scheduling with a representation of constraints by fuzzy sets and a repair with Tabu Search can be supported⁸. The second advantage of such a repair technique is that it is also a very general technique to optimize solutions. We have shown that such a search technique can start with any randomly generated schedule and will find a good one in reasonable time. In spite of its generality the strategy can be improved by rules of thumb.

A further improvement would be a support of the cooperation between the different plants. Often due dates are used as a mechanism to synchronize the production between plants. However, such data is very restrictive and the cooperation could be improved again through fuzzy sets that give a more flexible framework to negotiate. We have proposed to describe besides due dates also sequence relations by fuzzy sets to make the coordination between plants more flexible⁹.

5 References

- 1 French, S. Sequencing and Scheduling – An Introduction to the Mathematics of the Job-Shop, Chichester: Ellis Horwood, 1982.
- 2 Garey, M. R. and Johnson, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, New York: Freeman, 1979.
- 3 Ow, P.S. and Smith, S.F. Viewing Scheduling as an Opportunistic Problem Solving Process, *Annals of Operations Research* **12**, pp. 77–82, 1988.
- 4 Kerr, R.M. *Scheduling in the Steel Industry*, Study Christian Doppler Laboratory for Expert Systems CD-S 94/16, 1994
- 5 Numao, M. and Morishita, S. A Scheduling Environment for Steel-Making Processes, *Proceedings of the 5th International Conference on Artificial Intelligence for Industrial Applications*, Miami, pp. 279–286, 1989.
- 6 Woodyatt, L.R., Stott, K.L., Wolf, F.E., and Vasko, F.J. Using Fuzzy Sets to Assign Metallurgical Grades to Steel, *Journal of Metallurgy*, February 1992.
- 7 Barchini, F. and Theuretzbacher, N. The Challenge of Real-time Process Control for Production Systems, *Proceedings of the 7th AAAI Conference*, St. Paul, pp. 705–709, 1988.
- 8 Dorn, J, Kerr, R.M. and Thalhammer, G. Reactive Scheduling in a Fuzzy-Temporal Framework, in E. Szelke and R.M. Kerr *Knowledge-based Reactive Scheduling*, IFIP Transactions B-15, North Holland, pp. 39–55, 1993.
- 9 Dorn, J. and Kerr, R.M. Co-operating Scheduling Systems Communicating Through Fuzzy Sets, *Preprints of the 2nd IFAC/IFIP/IFORS-Workshop on Intelligent Manufacturing Systems (IMS'94)*, Vienna, pp. 367–373, 1994.

Erfahrungen mit einem CLP-basierten Ansatz zur Produktionsplanung *

Heinrich Petith
Astrid Schmücker-Schend
IBM Deutschland Informationssysteme GmbH
Wissenschaftliches Zentrum
Institut für Logik und Linguistik
Vangerowstr. 18
69115 Heidelberg
Tel.: 06221 / 594393 (-4230)
EMail: petith at vnet.ibm.com
schmueck at mazvm01.vnet.ibm.com

Zusammenfassung

Innerhalb des Projekts WISPRO (Wissensbasierte Produktionsplanung) werden Verwendbarkeit und Nutzen von Constraint Logic Programming zur Lösung von Produktionsplanungsproblemen untersucht. Basis für diese Untersuchungen bzw. die Implementierungen im Verlauf von WISPRO ist die Sprache PROTOS-L, ein Logik-Programmiersystem mit integriertem Constraint-Solver. PROTOS-L wurde im PROTOS-Projekt, dem Vorgänger von WISPRO, entwickelt. Im Verlauf von PROTOS wurden Produktionsplanungsprobleme mit einem rein logischen Ansatz angegangen, während das Hauptgewicht in WISPRO auf dem Einsatz von Constraints liegt. Dieser Übergang zu einem constraint-logischen Ansatz wurde im Wesentlichen durch Performanceprobleme motiviert, auf die man in PROTOS stieß. Innerhalb von WISPRO wurden unter Einsatz von Constraint Solving-Mechanismen verschiedene Prototypen entwickelt. Der erste, PPA (Produktionsplanungsassistent), führt die Verplanung einer Menge von zum Teil untereinander abhängigen (Teil-) Aufträgen durch, wobei jeder Teilauftrag eine Resource nur exklusiv belegen kann. Anhand des PPA wurden verschiedene Constraint- Propagationstechniken untersucht. Der zweite Prototyp, FCS (Finite Capacity Scheduler), realisiert Kapazitäts- bzw. Maschinengruppenplanung auf Basis von Cumulative Constraints. Zum Vergleich wurden mathematische Optimierungsmethoden implementiert. In Bezug auf Effizienz und Güte der generierten Pläne sind das CLP-basierte und das effizienteste mathematische System in etwa gleichauf; was Flexibilität und Effizienz der Programmerstellung anbelangt, ist der wissensbasierte Ansatz weit überlegen. Eine nähere Betrachtung beider Systeme eröffnete Möglichkeiten für eine sinnvolle Kombination der Ansätze. Die bisher entwickelten prototypischen Systeme wie z.B. das FCS werden im Rahmen der XPS95 demonstriert.

*Das diesem Bericht zugrundeliegende Vorhaben wurde mit Mitteln des Bundesministers für Forschung und Technologie unter Kennzeichen 01 IW 206 (Projekt WISPRO) gefördert. Die Verantwortung für den Inhalt dieses Berichts liegt bei den Autoren.

1 Einleitung

Das Projekt WISPRO (Wissensbasierte Produktionsplanung) ist ein BMFT-gefördertes Verbundprojekt mit den Partnern Wissenschaftliches Zentrum der IBM, Heidelberg, und GMD-FIRST, Berlin. Zentrales Thema ist, die Verwendbarkeit und den Nutzen wissensbasierter Methoden zur Lösung von Produktionsplanungssystemen zu untersuchen. Basis für die im Projekt durchzuführenden Implementierungen ist die logische Programmiersprache PROLOG, die im Vorgängerprojekt PROLOG, einem EUREKA-Projekt, entwickelt wurde (Beierle, Böttcher, & Meyer, 1994). PROLOG selbst ist eine PROLOG-ähnliche Sprache, erweitert um ein polymorphes Typkonzept, Modulkonzept, Datenbank-Anschluss u.a. an SQL-DB und einem Anschluss an OSF/Motif. PROLOG wurde entwickelt mit dem Ziel, die praktische Einsetzbarkeit logischer Programmierung zu verbessern. Während der Laufzeit des PROLOG-Projekts wurden u.a. in Zusammenarbeit mit Hoechst prototypische Planungssysteme im Bereich der Faserproduktion erstellt. Dabei zeichneten sich Performanceprobleme bei der Verwendung des rein logikbasierten PROLOG ab. Daher erweiterte man die Sprache um einen Constraint-Solver und setzte sich für WISPRO zum Ziel, einen constraint-logischen Ansatz zur Lösung von Produktionsplanungsproblemen zu verfolgen.

Eine wichtige Rolle sollten dabei Gespräche mit Anwendern spielen, um die Praxisorientierung der WISPRO-Forschung zu garantieren. In diesem Zusammenhang ist es auch wichtig zu überlegen, inwieweit Planungssysteme herkömmlicher Art bzw. Teilsysteme davon mit wissensbasierten Komponenten kombiniert werden können bzw. sollten.

2 Der rein logikbasierte Ansatz

Wie erwähnt, wurde innerhalb von PROLOG zunächst versucht, Produktionsplanungssysteme über einen rein logischen Ansatz zu behandeln. Dies bietet zunächst viele Vorteile gegenüber den herkömmlichen (mathematischen) Methoden. Anwender, mit denen man im Rahmen von PROLOG in Kontakt stand, beklagten die im folgenden erläuterten Probleme mit Standard-Planungstools.

Zum einen ist das Zustandekommen eines Plans völlig undurchschaubar. Dementsprechend schwierig ist es auch, einen automatisch generierten Plan nachträglich zu modifizieren. Das Ergebnis ist i.a. ein sehr schlechter Plan, auch wenn von einem ursprünglich fast optimalen Plan ausgegangen wurde. Ein Replanning ist daher mit Standardsystemen sehr schwer zu realisieren. Ein anderes genanntes Problem besteht darin, die für die Planung wichtigen Randbedingungen in die Berechnung des Plans einfließen zu lassen. Dies ist nur möglich durch eine Umsetzung dieser Bedingungen in eine entsprechende Belegung der Eingabeparameter des Standardsystems. Oft gibt es allerdings keine Möglichkeit, bestimmte Bedingungen bei der Plangenerierung zu berücksichtigen, ohne daß das Programm selbst modifiziert werden muß. Solche Änderungen, die eine Umsetzung von logischen Regeln und Heuristiken in eine mathematische Berechnungsstrategie bedeuten, sind oft nur mit großem Aufwand zu realisieren. Für all diese Probleme bietet die Verwendung eines logischen Programmiersystems als Basis für ein Planungstool Lösungsansätze. Insbesondere erlaubt eine Logik-Sprache eine wesentlich natürlichere Formulierung von Randbedingungen und Heuristiken im Programm. Dementsprechend leichter ist es auch, neue Bedingungen oder Planungsstrategien ins Planungssystem zu integrieren. Schließlich bietet ein

logischer Ansatz die Möglichkeit einer Erklärungskomponente, die z.B. Hinweise auf das Scheitern einer Plansuche liefert oder aber Erklärungen über das Zustandekommen eines generierten Plans abgibt.

Wie schon erwähnt, ging man aufgrund von Performance-Schwierigkeiten innerhalb von WISPRO dazu über, Produktionsplanungsprobleme auf Basis von Constraint Solving zu betrachten. Damit verliert man allerdings die Möglichkeit, auf relative einfache Weise Erklärungen generieren zu können, wie das bei den rein logischen Systemen der Fall ist. Was Performance, Flexibilität und Effizienz der Formulierung anbelangt, erfüllt der CLP-Ansatz aber bisher die Erwartungen.

3 Erfahrungen mit dem CLP-Ansatz

In WISPRO wurde zunächst ein Prototyp zur Terminierung auf Feinplanungsebene auf der Basis von PROTOS-L mit Constraints entwickelt. Das resultierende System PPA (Produktionsplanungsassistent) (Lock, 1994) geht von einem Auftragsnetz aus, aus dem jeder Teilauftrag unter Berücksichtigung seiner durchschnittlichen Laufzeit, vorgegebener Lieferdaten bzw. frühester Startzeitpunkte sowie bestehender Abhängigkeiten zu anderen Teilaufträgen auf eine Menge von Maschinen eingeplant werden sollte. Ziel dabei war, einen möglichst guten Plan zu erreichen, d.h. die Gesamtmenge der Aufträge in möglichst kurzer Zeit abzuarbeiten. Anhand dieses Prototyps wurden sowohl verschiedene Planungsheuristiken als auch verschiedene Propagationstechniken für Constraints auf ihren Einfluß auf Laufzeit und Güte des generierten Plans hin getestet.

In der Praxis spielen sowohl Kapazitäts- als auch Maschinengruppenplanung eine wichtige Rolle. Dem wurde durch die Entwicklung eines zweiten prototypischen Systems, des FCS (Finite Capacity Scheduler), Rechnung getragen, in das die Erfahrungen und Ergebnisse aus dem PPA einfließen. FCS berücksichtigt, daß ein (Teil-)Auftrag auf mehreren Ressourcen einer bestimmten Klasse ausführbar ist. Dies wurde über Cumulative Constraints (Aggoun & Beldiceanu, 1992) realisiert. Ausgehend von einer ursprünglichen Implementierung des Cumulative Constraint auf PROTOS-L-Ebene wurden nach und nach größere Anteile der Realisierung des Cumulative Constraint in den Constraint-Solver selbst verlagert, was u.a. zu besserer Performance führte. Damit war ein System entstanden, das die Kernanforderungen der Anwender aus dem Produktionsplanungsbereich erfüllte. Parallel zu den oben geschilderten Arbeiten wurden verschiedene mathematische Optimierungsmethoden in C implementiert, u.a. der als sehr effizient bekannte Sintflut-Algorithmus (Bosch, 1994). Dies hatte den Zweck, sowohl einen Vergleich mit herkömmlichen, mathematischen Optimierungsmethoden zu ermöglichen, als auch ein mögliches Zusammenwirken von mathematisch basierten und wissensbasierten Methoden untersuchen zu können. Ein anschließender Vergleich der Systeme FCS und Sintflut lieferte folgende Ergebnisse.

4 Vergleich mit mathematischen Methoden

Sowohl bzgl. Laufzeit als auch bzgl. Güte der generierten Pläne waren beide Systeme im Schnitt gleichauf, wobei zu beobachten war, daß bei einfachen Problemen der mathematische Ansatz schneller zu fast optimalen Lösungen führte, während der wissensbasierte Ansatz deutlich besser abschnitt, je komplexer das Planungsproblem und damit die zu-

gehörigen Constraints wurden. Der wichtigste Vorteil des CLP-Ansatzes lag jedoch in der wesentlich schnelleren Adaptierbarkeit des FCS an das jeweilige zu lösende Problem. Die dazu nötige knappe Formulierung der Planungsrandbedingungen durch Constraints resultierte in etwa 1/10 des C-Codes, der im mathematischen System zur Berücksichtigung dieser Bedingungen geschrieben werden mußte.

FCS und der mathematische Optimierer waren ursprünglich völlig unabhängig voneinander implementiert worden. Nach Abschluß der beschriebenen Tests wurden beide Systeme von ihrer Implementierung her miteinander verglichen. Bei näherer Betrachtung zeigte sich, daß Teile des mathematischen Optimierers eine Reimplementierung von Teilen des vom FCS benutzten Constraint-Solvers waren. Diese Teile spielen im mathematischen System zum einen vor dem eigentlichen Optimierungslauf eine Rolle, und zwar zur anfänglichen Eingrenzung des zunächst immensen Suchraums. Während der eigentlichen Optimierung überwachen sie zudem die Einhaltung der Randbedingungen. Dies geschieht im wesentlichen über eine Qualitäts- oder Bewertungsfunktion.

Dem Sintflut-Algorithmus liegt dabei folgendes Verfahren zugrunde: Ein anfänglicher Plan wird zufällig generiert. Ausgehend von diesem Plan werden kleine und zufällige Veränderungen probiert, die jedoch nur dann akzeptiert werden, wenn die Bewertung des veränderten Plans über einer bestimmten Akzeptanzschwelle, dem "Wasserstand", liegt. Diese Akzeptanzschwelle wird mit besser werdenden Plänen immer weiter angehoben. Beendet wird dieser Prozeß, wenn über längere Zeit keine besseren Pläne mehr gefunden worden sind. Durch Zulassen auch geringfügig schlechterer Pläne (in Relation zum bisherigen Wasserstand) während des beschriebenen Optimierungsprozesses wird gewährleistet, daß die Suche nicht gleich im ersten lokalen Optimum steckenbleibt.

Wie schon erwähnt, überprüft die Bewertungsfunktion typischerweise die Einhaltung der Randbedingungen. Je mehr und je wichtigere Randbedingungen verletzt sind, desto schlechter ist die Bewertung. Dazu müssen im klassischen Fall die Randbedingungen explizit (z.B. in C) in der Bewertungsfunktion codiert werden. Einem Constraint-Solver dagegen können diese Randbedingungen auf höherem Abstraktionsniveau beschrieben werden.

All dies legt nahe, eine Kopplung von mathematischem und wissensbasiertem Ansatz zu versuchen. Eine Verletzung der Randbedingungen könnte dann von der Bewertungsfunktion beim CLP-Teil des resultierenden Systems nur abgefragt und in eine entsprechende Bewertung umgesetzt werden. Auf diese Weise wäre es möglich, die guten Optimierungseigenschaften des mathematischen Ansatzes (s. auch 5) mit der Flexibilität und leichten Adaptierbarkeit des CLP-Ansatzes zu verbinden.

5 Ausblick

Die Gespräche mit Anwendern aus dem Produktionsplanungsbereich stellten ein zusätzliches Problem heraus, mit dessen Behandlung WISPRO sich in der kommenden Zeit beschäftigen wird. Die menschlichen Planer versuchen u.a., eine Vielzahl von Randbedingungen zu berücksichtigen, denen der von ihnen erstellte Plan nach Möglichkeit genügen soll. Dies sind Bedingungen wie die Garantie absoluter Termintreue für bestimmte Aufträge, minimale Lagerhaltungskosten, minimale Umrüstkosten usw. Dabei ist die Einhaltung der verschiedenen Randbedingungen vollständig zu berücksichtigen. Unter Umständen ist dies schon deshalb nicht möglich, weil sich verschiedene Bedingungen wider-

sprechen. So bedeutet z.B. absolute Termintreue i.a. gesteigerte Lagerhaltungskosten und umgekehrt. Dies macht es notwendig, verschieden priorisierte Constraints zu behandeln. Das bedeutet, daß eine Constrainthierarchie adäquat bearbeitet werden muß, in der sowohl harte Constraints als auch verschiedene Stufen weicher Constraints vorkommen. Ein weiteres Thema in diesem Zusammenhang ist die Plangenerierung trotz Verletzung auch harter Constraints, bzw. die Generierung von Plänen mit minimaler Constraintverletzung. Es hat sich nämlich gezeigt, daß in der Praxis beispielsweise Maschinenauslastungen von über 100% oder Terminüberschreitungen in geringem Umfang durchaus toleriert werden können. In solchen Fällen ist es erwünscht, einen Plan mit möglichst geringen Constraintverletzungen zu bekommen. Im vorhandenen System PPA sind Constraintverletzungen unzulässig, im System FCS sind Constraintverletzungen inzwischen in eingeschränktem Maß möglich. Mathematisch basierte Systeme erfüllen diese Anforderungen. Optimierer auf Basis des beschriebenen Sintflut-Algorithmus z.B. liefern oft Pläne mit Verletzungen der Randbedingungen, die aber in der praktischen Anwendung gut verwendbar sind. In WISPRO soll nun versucht werden, diese Möglichkeiten in den Rahmen der Constraint-Logischen Programmierung zu integrieren.

Reference

- Aggoun, A., & Beldiceanu, N. (1992). Extending CHIP in Order to Solve Complex Scheduling and Placement Problems. *Journées Francophone de la Programmation en Logique*, 52–66.
- Beierle, C., Böttcher, S., & Meyer, G. (1994). Report of the Logic Programming Language PROLOG-L (revised Version). Tech. rep., Institut für Logik und Linguistik, IBM Germany, Scientific Center Heidelberg.
- Bosch, R. (1994). Ein approximierender Algorithmus für die simultane Zeit- und Kapazitätsplanung. Master's thesis, Universität Karlsruhe.
- Lock, H. C. (1994). Dokumentation zum Produktionsplanungsassistenten PPA. Tech. rep., Institut für Logik und Linguistik, IBM Germany, Scientific Center Heidelberg.

Stücklistenverifikation mit **SPOCK**

*Wolfram Gieseke
Peter Küsell
Norbert Lenschow*

*Universität Osnabrück
Fachbereich Sprach- und Literaturwissenschaften
Institut für Semantische Informationsverarbeitung
Sedanstr. 4
49069 Osnabrück
email : crowley@hal.cl-ki.uni-osnabrueck.de*

Abstract

Als Antwort auf die Frage, inwieweit eine computergestützte Fehlerbehandlung von Stücklisten möglich ist, werden die Ergebnisse des Kooperationsprojektes zwischen dem Institut für Semantische Informationsverarbeitung der Universität Osnabrück und der Wilhelm KARMANN GmbH - **SPOCK** - vorgestellt.

Es wird erläutert, wie unter Zuhilfenahme des Konfigurationswerkzeuges PLAKON eine Vollständigkeitsüberprüfung der beim Automobilbau verwendeten Stücklisten mit Methoden des Konfigurierens durchgeführt werden kann.

Dabei werden nicht nur die allgemeinen Probleme bei der Überprüfung von Stücklisten diskutiert, sondern es wird auch über die Erfahrungen im Umgang mit dem System PLAKON berichtet.

Die als Abgleich einer Stückliste mit einem Referenzmodell realisierte Vollständigkeitsüberprüfung und zwei Ansätze zur Modellierung dieses Referenzmodells in Form von PLAKON-Wissensbasen werden vorgestellt. Weiterhin wird aufgezeigt, an welchen Stellen die Funktionalität PLAKONs für die Lösung der gestellten Aufgabe nicht ausreicht und wie sie durch Erweiterungen des Systems doch noch erzielt werden kann.

1 Einleitung

Im Rahmen des Projektes **SPOCK** (Stücklisten-Planung, -Organisation und computer-gestützte Konfiguration) wurde durch eine Kooperation zwischen dem Institut für Semantische Informationsverarbeitung (Universität Osnabrück) und der Wilhelm KARMANN GmbH in der Zeit von Okt. '93 bis Okt. '94 ein in der Industrie real existierendes Problem bearbeitet.

Die Wilhelm KARMANN GmbH arbeitet als Hersteller von PKW-Sondermodellen (u.a. dem VW-Golf-Cabrio) mit Stücklisten, welche nicht ausschließlich eigenständig erstellt und gewartet werden. Aufgrund der Dynamik der Stückliste (Abschnitt 2) ist dieses Unternehmen an einem System interessiert, welches vorhandene Listen zu einem beliebigen Zeitpunkt auf Vollständigkeit untersucht.

Wegen der Komplexität der Aufgabenstellung und der begrenzt zur Verfügung stehenden Zeit wurde zwischen den Kooperationspartnern vereinbart, die Aufgabe des Projekts auf die Erstellung einer Machbarkeitsstudie, die sich auf einen Teilumfang des Fahrzeuges - die Heckklappe - konzentriert, zu beschränken. In diesem Teilumfang treten alle das gesamte Fahrzeug betreffende Fehler exemplarisch auf.

Die konkrete Spezifikation des im Rahmen der Studie zu erstellenden Systems forderte die Überprüfung einer Variantenliste¹ mit anschließender Ausgabe und Interpretation von erkannten Fehlern.

Diese Aufgabe wurde mit dem Konfigurationswerkzeug PLAKON [Cunis et al., 1991], das schon mehrfach für realitätsnahe Anwendungen eingesetzt wurde [Kopisch und Günter, 1992], unter Verwendung der in ihm zur Verfügung gestellten Methoden des Konfigurierens gelöst.

Die Realisierung des Systems läßt sich anhand der Beantwortung folgender Fragen dokumentieren:

Warum ist die Stückliste so fehleranfällig?

Welche Fehler können in der Stückliste auftreten, wodurch werden sie verursacht und wie können sie erkannt werden? (Abschnitt 2)

Wie läßt sich das zur Überprüfung benötigte, komplexe Wissen adäquat und funktional repräsentieren? (Abschnitt 4)

Inwieweit ist PLAKON für die Aufgabe geeignet, welche Funktionen können genutzt werden, welche fehlen? (Abschnitt 6)

Wie und an welcher Stelle kann PLAKON um fehlende Funktionalität erweitert werden? (Abschnitt 5)

Durch eine Erweiterung des PLAKON-Systems und die Erstellung eines geeigneten Referenzmodells für Heckklappen wurde schließlich ein System geschaffen, welches die gewünschte Vollständigkeitsüberprüfung durchführt [SPOCK, 1994].

¹ Eine Variantenliste ist ein Auszug aus der Stückliste, der alle Teile für eine Ausprägung eines Fahrzeuges (z.B. VW-Golf-Cabrio mit Diebstahlsicherung, Airbag, japanische Version usw.) zu einem bestimmten Produktionstermin enthält.

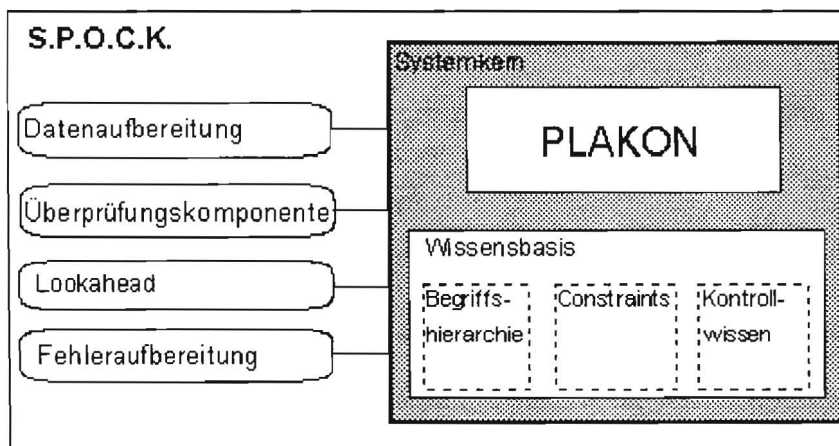
2 Stückliste - Einordnung und Probleme

Bei der hier beschriebenen Stückliste handelt es sich um eine Liste des Automobilherstellers Wilhelm KARMANN GmbH, welcher u.a. als Zulieferer und Hersteller für VW arbeitet. Die angesprochenen Funktionsweisen und Probleme treffen aber zu einem großen Teil auch auf alle anderen stücklistenorientierten Produktionsverfahren zu.

Die Stückliste ist die zentrale Datensammlung über alle Einzelteile eines Fahrzeugtyps (z.B. VW Golf). Hier werden unter anderem Qualitätseinstufungen, Bezugsarten, Strukturinformationen und Verweise auf Konstruktionszeichnungen gespeichert. Zusätzlich ist durch die Stückliste jederzeit ein Zugriff auf alle in einem Fahrzeugtyp jemals verwendeten Teile zu gewährleisten. Für die beschriebene Stückliste bedeutet dies, daß kein einmal aufgenommenes Teil je wieder daraus gelöscht wird. Die aktuell gültigen Teile werden über einen Einsatz- und Entfallsschlüssel aktiviert bzw. deaktiviert; dadurch enthält der gespeicherte Teileumfang ein Vielfaches der momentan verbauten Einzelteile. Die Komplexität dieses Mediums wird zusätzlich durch die codierte Speicherung von Informationen über Ausprägungen eines Fahrzeugtypes (Motorvarianten, Grundausstattungen, Zusatzausstattungen, Sondermodelle und länderspezifische Besonderheiten, usw.) erhöht. Die Codierung solcher Informationen erfolgt z.B. bei VW über sog. PR-Nummern (Schlüsselnummern für **P**rimäre Eigenschaften), die aus PR-Familien zu wählen sind, wobei eine PR-Nummern-Familie für jeweils ein Detail zuständig ist.

Momentan greifen die meisten Abteilungen des Unternehmens nur lesend auf die Stückliste zu. Stehen ihnen fehlerhafte Informationen zur Verfügung, kann dies hohe Kosten verursachen. So ist es u.a. möglich, daß es durch eine Bestellung von Einzelteilen auf der Grundlage einer inkorrekten Stückliste zur Lagerverwaltung und Bindung von Kapital in unbenötigten Einzelteilen kommt und evtl. sogar durch das Fehlen von Teilen zu einem Stillstand der Produktion. Änderungen an der Stückliste werden lediglich von einer speziellen Abteilung durchgeführt, müssen jedoch vom Auftraggeber bestätigt werden. Trotz dieser Beschränkungen entstehen (z.B. aufgrund der langen Laufzeit von Änderungen und unbeabsichtigtes, synchrones Bearbeiten sowohl beim Auftraggeber als auch bei KARMANN) immer wieder Fehler.

3 Architektur



Da PLAKON als Systemkern den größten Teil der benötigten Funktionalität zur Verfügung stellt, nimmt es zusammen mit den drei Komponenten der Wissensbasis (Begriffshierarchie, Constraints und Kontrollwissen) in der Systemarchitektur naturgemäß eine zentrale Position ein.

Um diesen Kern gruppieren sich die Erweiterungskomponenten :

- Die Datenaufbereitung
Diese Systemkomponente übersetzt eine Variantenliste in ein PLAKON- bzw. Lisp-Format.
- Die Überprüfungs-komponente
Sie übernimmt den eigentlichen Abgleich zwischen dem mit PLAKON konfigurierten Referenzobjekt und dem durch die Variantenliste beschriebenen Fahrzeug.
- Die Lookahead-Komponente
Durch sie wird der Konfigurierungsprozeß gesteuert.
- Die Fehleraufbereitung
Sie bereitet die ermittelten Daten auf und interpretiert sie.

Ein Systemzyklus verläuft folgendermaßen :

Die zu überprüfende Variantenliste wird durch die Datenaufbereitung in einen Lisp-Hashtable überführt, auf den später die Überprüfungs-komponente und die Fehleraufbereitung zugreifen können. Anschließend wird aus der Wissensbasis mit Hilfe eines Anforderungsprofils eine konkrete Heckklappe konfiguriert. Während dieses Vorgangs kommt zunächst die Lookahead-Komponente zur Wirkung, die zu einem bestimmten Zeitpunkt die Menge der zu überprüfenden Teile einschränkt. Während der endgültigen Spezialisierung der Einzelteile tritt dann die eigentliche Überprüfungs-komponente in Aktion und versucht, zu jedem in der Referenzheckklappe eingebauten Teil ein Pendant in der Variantenliste zu finden.

Die während dieser Phase ermittelten Daten werden abschließend von der Fehleraufbereitung verwendet, um ein ausführliches Fehlerprotokoll zu erstellen.

Das **SPOCK**-System ist - gebunden an seinen Kern PLAKON - in Common Lisp implementiert.

4 Die Wissensbasis

Eine wesentliche Aufgabe im Rahmen des Projektes bestand in der Erstellung eines allgemeingültigen Modells einer Heckklappe, welches als Referenz für die Überprüfung von Variantenstücklisten auf Vollständigkeit benutzt werden sollte. Weiterhin sollte es möglich sein, mit Hilfe dieser Referenzheckklappe den Aufbau korrekter Stücklisten zu unterstützen. Da diese Funktionen innerhalb des Systems PLAKON zu realisieren waren, bestand die konkrete Aufgabe in der Beschreibung des allgemeingültigen Wissens über die Struktur und die Bestandteile des Teileumfangs Heckklappe in Form einer PLAKON-Wissensbasis. Zu diesem Zweck mußten einerseits die tatsächlich möglichen Repräsentationsalternativen PLAKONs erarbeitet und zum anderen ein allen Anforderungen gerechtes Modell entwickelt werden. In der ersten Phase wurde die Modellierung einer den Repräsentationsfähigkeiten PLAKONs angepaßten Referenzheckklappe zur Vollständigkeitsüberprüfung einer beispielhaften Heckklappenstückliste durchgeführt. In der zweiten Modellierungsphase wurde untersucht, wie sich die erstellte Referenzheckklappe so verallgemeinern läßt, daß sie auch zur Überprüfung

weiterer Heckklappenstücklisten und anderen Anwendungen (z.B. Aufbau einer neuen Stückliste) benutzt werden kann.

Die Modellierung der den Repräsentationsfähigkeiten von PLAKON angepaßten Referenzheckklappe, geschah unter Zuhilfenahme der Stückliste eines Golf-Cabrios. Die für den Abgleich benötigten Informationen der Heckklappe wurden im Wesentlichen in PLAKON nachmodelliert, dabei jedoch darauf geachtet, daß das allgemeine Wissen über Heckklappen, welches dieser Stückliste entnommen werden konnte, möglichst von dem speziellen Wissen dieser Liste getrennt wurde.

192612250	136131	2	DICHTSTOPFEN	381K	T K54T K2H1419	030593	B	ST 00002	190892 120393
1E0827550	136210	2	GASDRUCKFEDER	380K	T R52T E29000	030593	N	ST 00002	130592 20393
	136500		DICHTUNG HECKKLAPPE						
TAB002766H	136520	---	T PROFIL DICHTUNG	380I	T K00 K1H8302	030593	B	ST 00000	221191 210493
1H5827609	136140	2	SCHUBSTANGE	381K	A K54T K1H38	030593 K3DL4 151193	V	ST 00001 DL4	230492
1E0827705	136530	2	DICHTUNG,HECKKL	380K	T K59T E29000	030593	N	ST 00001	221191 210493

Abb.1: Auszug aus der Stückliste

⇒

```
(
  (IST! (EIN ANBAUTEILE) (EIN ALLE_TEILE))
)

(IST! (EIN GASDRUCKFEDER_HK) (EIN ANBAUTEILE
  (PART-OF (EIN HECKKLAPPE))))

(IST! (EIN GASDRUCKFEDER_HK1) (EIN GASDRUCKFEDER_HK
  (TESTEN 'JA)
  (MENGE 2)
  (DATUM [19930503 40000000])
  (TEILENUMMER 'VW_1E0827550)))
```

Abb.2: Ein Konzept der PLAKON-Wissensbasis

Auch die Struktur des in der Stückliste enthaltenen Wissens wurde übernommen.

Zur Darstellung dieser Struktur stellt PLAKON im Wesentlichen zwei Relationen zur Verfügung :

- Die Relation 'is_a' ('ist_ein') zur Darstellung des klassifikatorischen Wissens, welches in der Stückliste eigentlich keine große Rolle spielt, in PLAKON jedoch u.a. dazu benutzt wird, jedes Objekt zu definieren und
- die Relation 'has_parts' ('hat_Teile') Umkehrrelation : 'part_of' ('Teil_von') zur Darstellung der Teil-Ganzes-Beziehung, über die die Elemente der Stückliste geordnet sind und die Liste selber eine hierarchische Struktur erhält.

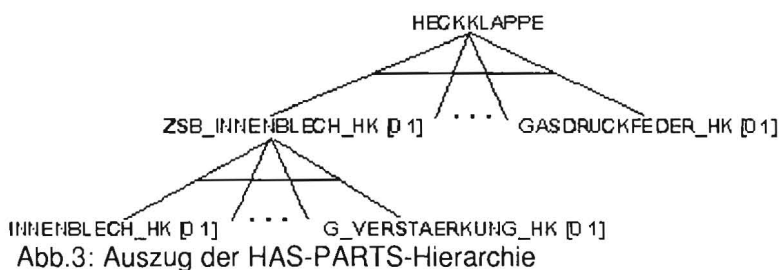


Abb.3: Auszug der HAS-PARTS-Hierarchie

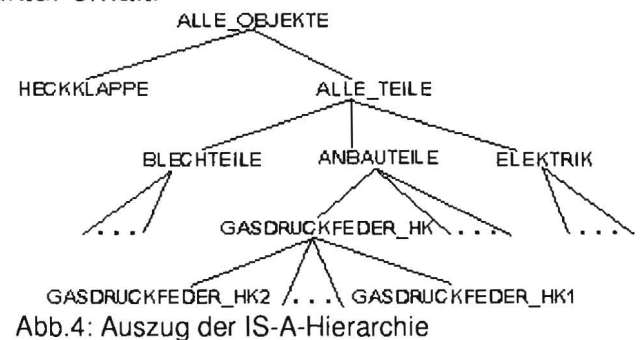


Abb.4: Auszug der IS-A-Hierarchie

Welche Heckklappenvarianten und Varianten der einzelnen Zusammenbauten mit Hilfe der Teile konfiguriert werden können, ist dabei nicht beliebig, sie werden durch die

möglichen PR-Nummern-Kombinationen eindeutig bestimmt. Jede einzelne PR-Nummern-Kombination legt fest, ob ein optionales Teil eingebaut wird oder nicht und welche Ausprägungen eines einzubauenden Teils, z.B.: Gasdruckfeder_Hk1 oder Gasdruckfeder_Hk3 (siehe dazu auch Abb.4) gewählt wird.

Das Modell der allgemeingültigen Heckklappe, welches in der zweiten Modellierungsphase entstand, unterscheidet sich von dem den Repräsentationsfähigkeiten PLAKONS angepaßten durch eine veränderte Gliederung des in ihm enthaltenen Wissens. Aufgrund der Erkenntnis, daß die Teile, die disjunkt zueinander sind, fast immer dem gleichen Zweck dienen und daß sie diesem Zweck erst dann gerecht werden, wenn sie in irgendeiner Weise mit dem Restsystem verbunden sind, wurde eine allgemeingültige zweckorientierte Struktur entwickelt [Reincke, 1995]. Sie wird durch das folgende Schema rekursiv aufgebaut :

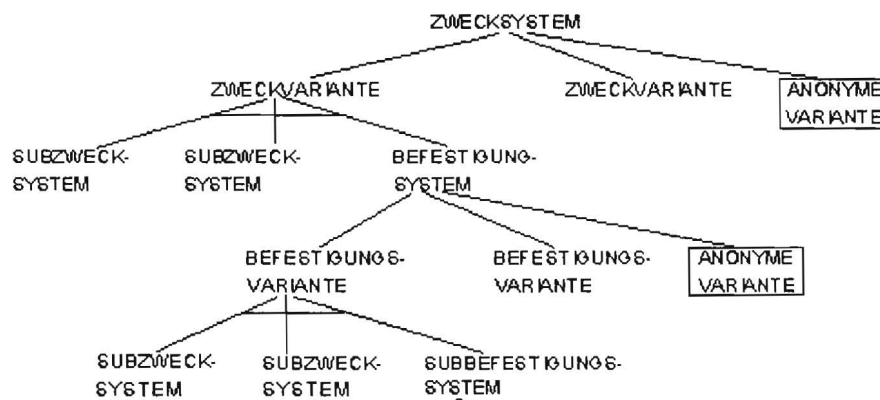


Abb.5: Zweckorientierte Modellierung²

Zur Entwicklung dieser Struktur wurden zusätzlich zur Stückliste sogenannte PDM (Produkt,Detail,Montage) -Blätter verwendet, um weitergehende Informationen über den Zweck der einzelnen Heckklappenteile zu erhalten. Der Vorteil dieser neuen Modellierungsart liegt hauptsächlich in der besseren Unterstützung der Erstellung neuer Stücklisten durch die den Bedürfnissen und der Denkweise des Konstrukteurs besser angepaßte Struktur der Wissensbasis.

5 Erweiterung des PLAKON-Systems

Neben dem Aufbau der Wissensbasis bestand eine andere Aufgabe in der Erweiterung des PLAKON-Systems um funktionale Elemente, die das System nicht zur Verfügung stellt.

Hauptproblem dabei war der Umstand, daß PLAKON so konzipiert ist, daß alle benötigten Informationen in einer Wissensbasis vorhanden sind. Bei dem angestrebten Vergleich liegen jedoch immer zwei Datenmengen vor, nämlich die Soll- und die Ist-Menge. Die Soll-Menge als PLAKON-Wissensbasis zu realisieren, bot sich an, da diese Menge auch Informationen enthält, die für die Kontrolle des Vergleichsprozesses benötigt werden.

² Anonyme Varianten bieten dem Erzeuger einer neuen Stückliste eine Auswahlmöglichkeit für den Fall, daß an einer Stelle der Konstruktion keine der alten Varianten verwendet, sondern eine völlig neue entwickelt werden soll.

Zusätzlich mußte die Möglichkeit, der Bearbeitung einer zweiten Wissensbasis (der Ist-Menge) in PLAKON geschaffen und darüberhinaus Wege zum Vergleich bzw. Datenaustausch zwischen beiden WBs erstellt werden.

5.1 Die Datenaufbereitung

Bei der Datenaufbereitung mußte zunächst das Problem der Zusammenführung der verschiedenen Datenformate gelöst werden. Die Wissensbasis - also das Soll-Wissen - liegt ohnehin als PLAKON-Wissensbasis vor, die Variantenliste, die das Ist-Wissen enthält, jedoch in Form einer ASCII-Datei mit recht spezieller Struktur. Diese Datei enthält darüberhinaus Informationen, die für den Vergleich nicht von Bedeutung sind. So waren die relevanten Informationen auszufiltern und in ein Lisp- und somit für PLAKON lesbares Format zu bringen.

Die Datenaufbereitungs-Komponente ist als modulare Schnittstelle konzipiert, die eine große Flexibilität hinsichtlich der vorhandenen Datenbestände gewährleisten soll. Es mußte also eine Repräsentationsform gefunden werden, die einen flexiblen und vor allem schnellen Zugriff zuläßt; hierzu wurde ein Hash-Table realisiert.

5.2 Die Überprüfung

Die Überprüfungs-komponente hat über ihren offensichtlichen Zweck der eigentlichen Vollständigkeitsüberprüfung hinaus noch zwei weitere wichtige Funktionen :

Als zentrales Modul der Erweiterung hat sie sowohl eine integrative Rolle als auch die Aufgabe, den Systemablauf außerhalb von PLAKON zu steuern. Die Überprüfungs-komponente selbst ist in PLAKON durch Berechnungsfunktionen integriert, die an verschiedenen Konzepten der Wissensbasis hängen. Je nachdem, von welcher Konzeptart die Komponente angestoßen wird, führt sie unterschiedliche Funktionen aus, so daß man drei Phasen unterscheiden kann :

Die 1. Phase wird von einem speziellen Konzept, nämlich dem "Startknoten", der grundsätzlich als erster evaluiert wird, ausgelöst. Diese Phase dient der Initialisierung der gesamten Erweiterung. Dazu gehört z.B. auch der Aufbau des Hash-Tables mit den Variantenlisteneinträgen durch die Datenaufbereitungskomponente.

Die 2.Phase wird durch jedes Konzept, das endgültig spezialisiert wurde, einmal angestoßen. In dieser Phase findet der eigentliche Vergleich statt, d.h. es wird im Hash-Table nach einem dem Konzept adäquaten Pendant gesucht und die Werte der beiden miteinander verglichen. Das Ergebnis dieses Vergleich wird ebenfalls im Hash-Table festgehalten.

Die 3.Phase wiederum wird ähnlich der ersten von einem speziellen Konzept initiiert, nämlich dem "Endknoten", welcher grundsätzlich als letztes evaluiert wird. Zu diesem Zeitpunkt ist die eigentliche Überprüfung beendet, folglich werden die ermittelten Daten, die während der Überprüfung auf den gesamten Hash-Table verstreut wurden, komprimiert und an die abschließende Fehleraufbereitung zur weiteren Verarbeitung übergeben.

5.3 Die Fehleraufbereitung

Die Fehleraufbereitungs-Komponente hat die Aufgabe, die durch den Überprüfungsprozeß entstandenen Daten über die Anwesenheit oder das Fehlen von Teilen in der Variantenliste zu analysieren, zu interpretieren und entsprechend aufbereitet auszugeben.

Für die Interpretation von Fehlern ist es wichtig, "Einfach-Teile" von "Mehrfach-Teilen" zu unterscheiden. Einfach-Teile sind Teile, die durch genau einen Eintrag in der Variantenliste definiert sind und Mehrfach-Teile Teile, die durch mehrere Einträge in der Variantenliste definiert sind. Eine Einteilung in diese beiden Fehlerarten ist dadurch motiviert, daß fehlende/zuviele Einfach-Teile nur eine Ursache haben können, fehlende/zuviele Mehrfach-Teile jedoch verschiedene. Diese Fehlerkategorien beinhalten Teile, die aufgrund eines fehlenden Eintrags in der Stückliste nicht oder beispielsweise durch einen versehentlichen Doppeleintrag zuviel vorhanden sind. Aber ein Eintrag kann durch eine falsche Mengenangabe auch in sich fehlerhaft sein. Für diesen Fall wird als weitere Fehlerkategorie der Mengenfehler bei Einfach- bzw. Mehrfach-Teilen eingeführt.

Die Fehleraufbereitungs-Komponente ordnet zuerst alle aufgetretenen Fehler einer dieser Kategorien zu und ist somit in der Lage, alle für die Vollständigkeitsüberprüfung relevanten Fehler zu erkennen. Allerdings ist eine Statistik über fehlende oder zuviel vorhandene Teile noch nicht besonders hilfreich, da die eigentliche Arbeit, nämlich die Suche nach den Ursachen der Fehler - die ja eigentlich nur Symptome sind -, erst jetzt beginnt. Deshalb ist es der Fehleraufbereitung weiterhin möglich, die ermittelten Fehler zu interpretieren. Dazu werden bestimmte Fehler miteinander in Verbindung gebracht und auf diese Weise Konstellationen ermittelt, die Rückschlüsse über Fehlerursachen zulassen.

Diese Interpretation wurde exemplarisch für zwei Fehlerkonstellationen realisiert :

Das doppelte Auftreten von Teilen - i.d.R. durch einen versehentlich doppelten Eintrag in der Stückliste verursacht - kann durch zwei fast identische Einträge für dasselbe Teil in der Variantenliste nachgewiesen werden. Aber auch zwischen fehlenden und zuvielen Teilen kann aufgrund der Teilnummernstruktur u.U. eine Verbindung hergestellt werden, wenn nämlich zu einem fehlenden Teil ein überflüssiges Pendant existiert, dessen Teilnummer sich nur in einem zusätzlichen Index von der des fehlenden unterscheidet. Dies deutet daraufhin, daß durch ein fehlerhaftes Datum oder eine fehlerhafte PR-Nummer ein falsches Teil eingebaut wurde.

Diese Interpretationen werden zusammen mit den aufgetretenen Fehlerkategorien abschließend von der Fehleraufbereitungskomponente ausgegeben und stellen den eigentlichen Output des **SPOCK**-Systems dar.

5.4 Der Lookahead

Eine weitere besondere Erweiterung stellt die Lookahead-Komponente dar. Sie ist nötig, um eine Schwäche von PLAKON, nämlich den Umgang mit optionalen Teilen, auszugleichen. Optionale Teile sind solche, die - im Gegensatz zu obligatorischen - nicht immer eingebaut werden müssen, wohl aber eingebaut werden können. Als Beispiel mögen die Scheibenwischer eines Fahrzeuges dienen; die Front-Scheibenwischer sind obligatorisch, die Heckscheibenwischer hingegen optional, da sie nur bei

einigen Fahrzeugvarianten (z.B. Kombi- oder Stufenheckversionen) eingebaut werden. Für PLAKON besteht die Schwierigkeit darin, daß ein Konzept "Heckscheibenwischer" existiert, für das es u.U. keine korrekte Instantiierung gibt (wenn nämlich eine Varianten ohne Heckscheibenwischer gebaut werden soll). Zur Lösung dieses Problems ist es erforderlich, massiv in den Konstruktionsprozeß von PLAKON einzugreifen. Zu einem bestimmten Zeitpunkt des Ablaufs, zu dem PLAKON eine Liste aufgebaut hat, die alle Teile enthält, die im weiteren spezialisiert werden sollen, wird der Prozeß unterbrochen und diese Liste manipuliert, indem alle die optionalen Teile entfernt werden, für die es keine korrekte Instantiierung gibt. Demnach würde zum Beispiel bei dem Bau einer Limousine, das Konzept "Heckscheibenwischer" aus der Liste der noch zu spezialisierenden Teile gestrichen werden.

Genau dies leistet die Lookahead-Komponente, allerdings stellt sie eine sehr spezifische Lösung des Problems dar, die sich nicht ohne weiteres auf ähnliche Problemstellungen übertragen läßt.

6 Ergebnisse

Das wichtigste Ergebnis besteht sicherlich in dem erstellten System. Es ist in der Lage, verschiedenste Fehler in Variantenlisten von Heckklappen zu erkennen und dem Anwender Hinweise auf die Anzahl, Art und eventuelle Ursache dieser Fehler zur Verfügung zu stellen. Das System **SPOCK** beantwortet somit die Frage, inwiefern eine computergestützte Fehlerbehandlung von Stücklisten möglich ist. Darüber hinaus besitzt es als weitere Funktion die Möglichkeit, den Aufbau von neuen korrekten Stücklisten zu unterstützen. Als weiteres Resultat des Projektes ist der Wissenstransfer zwischen der Wilhelm KARMANN GmbH und der Universität Osnabrück im Hinblick auf wissensbasierte Konfigurationswerkzeuge anzusehen, mit dessen Hilfe nicht nur bei KARMANN Know-how aufgebaut werden konnte, sondern - gewissermaßen im Gegenzug - bei den Mitarbeitern des Projektes ein großes Verständnis der Probleme um eine Stückliste entstand. Dieses Verständnis wiederum führte zu der wichtigen Einsicht, daß aufgrund der Dynamik von Stücklisten die Maßnahmen zur Sicherstellung ihrer Korrektheit nicht erst bei der Überprüfung von fertigen Listen, sondern viel eher bei ihrer Erzeugung und Änderung ansetzen sollte.

Auch über das System PLAKON wurden wichtige Erkenntnisse gewonnen :

Als Kern des **SPOCK**-Systems hat PLAKON durchaus seinen Zweck erfüllt. Viele Komponenten konnten verwendet werden, dazu zählen insbesondere die Funktionen zur Wissensrepräsentation, wobei die bereits vorhandenen Relationen für die Modellierung völlig und die Möglichkeiten zur Steuerung des Programmablaufs ausreichten, die für unser System sehr wichtig waren. Andere relevante Komponenten wurden kaum (z.B. Constraints) oder gar nicht (z.B. Konfliktlösung) benutzt. Andererseits mußte PLAKON um eine Reihe von Komponenten erweitert werden, um fehlende Funktionalitäten auszugleichen (Abschnitt 5).

Für die Entwicklung eines Prototypen war PLAKON also geeignet, für ein Anwendungssystem sollte aber eine Reimplementierung erwogen werden, die sich auf die verwendeten Komponenten sowie die Erweiterung beschränkt.

Literaturverzeichnis

[Cunis et al., 1991] R.Cunis, A.Günter, H.Strecker. Das PLAKON-Buch. Springer, Informatik Fachberichte 266,1991

[Kopisch und Günter, 1992] M.Kopisch und A.Günter. Configuration of a Passenger Aircraft Cabin based on Conceptual Hierarchy, Constraints and Flexible Control. In Proc. 5th IEA/AIE. Springer, 1992.

[SPOCK, 1994] **SPOCK**-Projekt Abschlußbericht, 1994

[Reincke, 1995] K. Reincke. Prinzipienbasiertes Modellieren, Arbeitsbericht des Instituts für Semantische Informationsverarbeitung, Universität Osnabrück.

Integration von KONWERK in eine komplexe Anwendungsumgebung¹

A. Angelus, Ch. Bohley, A. Kathe, H. Reuter

1. Nutzung externer Datenquellen für Konfigurierungsprozesse

Reale Konfigurierungsaufgaben erfordern oft eine sehr große Anzahl von Objektbeschreibungen und umfassende Darstellungen von Beziehungen zwischen den einzelnen Objekten. Auch bei der Anwendung des Konfigurierungssystems KONWERK liegen Erfahrungen mit sehr komplexen Anwendungsdomänen vor (siehe z.B. [Kopisch92] und [Schädler93]). Für reale, routinemäßige Nutzungen muß dann genau untersucht werden, ob die vom Konfigurierungssystem benötigten Daten und Informationen auch von diesem System verwaltet werden. Unterliegen diese Daten einer häufigen Änderung und Weiterentwicklung oder werden die gleichen Daten auch für andere Anwendungen benötigt, so kann die Verwaltung und Pflege durch spezielle Systeme effektiver und sogar notwendig sein. Im folgenden soll die Konzeption für eine allgemeine Datenschnittstelle vorgestellt werden, die einerseits unterschiedliche Darstellungsformen wie objektorientierte und relationale Datenbanken sowie sequentielle Dateien berücksichtigt und andererseits auf einige typische Konfigurierungsabläufe orientiert ist, wie sie auch durch KONWERK realisiert werden.

Die Arbeit knüpft an diesbezügliche Erfahrungen an, die mit dem System PLAKON beim Einbeziehen von Simulationsergebnissen [Strecker91] und bei der Konzeption einer Fallbibliothek [Pfitzner93] gesammelt wurden. Darüber hinaus spielen externe Datenquellen eine entscheidende Rolle für innovative Konfigurierungsprozesse. In [Vietze94] werden Vorschläge zur Erweiterung von KONWERK für solche Prozesse entwickelt, wobei folgende Aktionen unterschieden werden:

- Neue Wertebereiche für Parameter und Relationen des Modells.
- Neue Parameter, Relationen und Konzepte.
- Neue Nebenbedingungen und Abhängigkeiten zwischen Objekten.
- Neue Vorgehensweisen bzw. neue Problemlösungsprozeduren.

Die Behandlung von Innovationen bezieht sich dabei ausschließlich auf den Problemlösungsprozeß. Bei der Entwicklung einer allgemeinen Datenschnittstelle sollte jedoch der Wissensakquisitionsprozeß in gleicher Weise einbezogen werden. Die oben genannten Aktionen bilden auch die Hauptbestandteile während des Modellierungsprozesses, so daß mit Hilfe der Datenschnittstelle und spezieller Zugriffsmethoden eine effektive Generierung von Teilen der Wissensbasis realisiert werden kann.

Im folgenden soll die Konfigurierung hydrogeologischer Modelle als eine Anwendungsdomäne vorgestellt werden, die durch eine starke Heterogenität der einzelnen Konstruktionskomponenten und eine große Komplexität der zu erstellenden Modelle charakterisiert ist. Für die Konfigurierung der Modelle wird eine Vielzahl sehr unterschiedlicher Daten benötigt, die durch eigenständige Systeme verwaltet werden:

- geographische Informationssysteme für geographische Beschreibungen, Karten und Luftbilder
- Datenbanken für Meßdaten und Stoffkonstanten
- Simulationssysteme und spezielle Programmbibliotheken für Strömungs- und Transportprozesse, sowie chemische und biologische Stoffumwandlungen
- Expertensysteme und Fallbibliotheken für allgemeine Zusammenhänge, gesetzliche Vorschriften und heuristische Informationen

An einfachen Beispielen soll demonstriert werden, wie aus einem Datenbankzugriff neue Konzepte erzeugt oder Auswahlmengen für einzelne Parameter generiert werden. Aus einer Fallbibliothek werden externe Daten in die Konfigurierung einbezogen. Die einzelnen Fälle können sowohl in einer objektorientierten Datenbank als auch in sequentiellen Dateien abgelegt sein. Zunächst beziehen sich die Fälle auf schon realisierte Konfigurationen. Perspektivisch ist auch an eine Nutzung allgemeiner Fallbibliotheken in Anlehnung an den in [Fabel93] beschriebenen Ansatz denkbar.

¹Diese Arbeit wurde mit Mitteln des Bundesministeriums für Forschung, Technologie, Wissenschaft und Bildung (Verbundvorhaben PROKON) und des Kultusministeriums des Landes Sachsen-Anhalt (Projekt MIMOSE) gefördert.

2. Konfigurierung hydrogeologischer Modelle mit KONWERK²

2.1 Entwurf einer Begriffshierarchie für hydrogeologische Modelle

Im Umweltbereich gibt es zahlreiche Untersuchungs- und Planungsaufgaben, die eine Anwendung hydrogeologischer Methoden erfordern. Dazu zählen z.B. die Rekultivierung von Bergbaufolgelandschaften, die Verhinderung von Gefahrstoffausbreitungen im Boden und im Grundwasser sowie die Erschließung von Trinkwasserressourcen. Die Lösung derartig komplexer Problemstellungen erfordert intensive Forschungs- und Untersuchungsaktivitäten zur Aufklärung hydrogeologischer Gesetzmäßigkeiten wie auch eine schnelle Umsetzung neuer Erkenntnisse in ingenieurtechnische Verfahren und landschaftsplanerische Maßnahmen.

Eine wichtige Methode bei der Gestaltung dieses Prozesses besteht in der Entwicklung geeigneter Modelle als Abbilder realer Gesetzmäßigkeiten, Zusammenhänge und Prozesse.

Für die überwiegende Mehrheit hydrogeologischer Prozesse lassen sich die folgenden zwei Grundaufgaben der Hydrogeologie angeben:

- Beschreibung von Strömungsproblemen im Untergrund (Geofiltration),
- Beschreibung von Stofftransport, -austausch und -umsetzung (Migration).

Die Lösung beider Aufgaben setzt die Kenntnis räumlicher, zeitlicher und stofflicher Eigenschaften des Untersuchungsraumes und der untersuchten Prozesse voraus. In Abhängigkeit von der Aufgabenstellung kommt eine große Zahl unterschiedlicher Methoden und Verfahren zum Einsatz, die auf Erkenntnissen aus sehr unterschiedlichen Wissenschaftsgebieten basieren.

Die Projektkonzeption basiert auf der Annahme, daß sich solche hydrogeologischen Modelle aus einzelnen Modellkomponenten konfigurieren lassen. Der Entwurf der Begriffshierarchie wurde durch folgende Tatsachen bestimmt:

- Das Konzept *Geo-Modell* dient der Spezifikation der Aufgabenstellung. Mit Hilfe der darin angegebenen *has-parts Relationen* wird das Gesamtmodell zusammengesetzt.
- Die Beschreibung der geographischen und geologischen Verhältnisse wird durch Instanzen des Konzeptes *Teilgebiet* realisiert. Dieses Konzept ermöglicht u.a. die Darstellung von Grundwasserleitern und der stofflichen Zusammensetzung einzelner Gesteinsschichten.
- Für die Modellierung von Migrationsprozessen spielen neben der Beschreibung von Strömungsprozessen im Untergrund geochemische, geophysikalische und evtl. mikrobiologische Gesetzmäßigkeiten eine wichtige Rolle.
- Das Konzept *Material* dient der Beschreibung geophysikalischer und geochemischer Eigenschaften des Untergrundes.
- Mit Hilfe einer *Aufgabenstellung* können typische Modellierungsprobleme charakterisiert werden, so z.B. die *Erkundung* von Grundwasser- oder Schadstoffausbreitungen, die *Erschließung* neuer Trinkwasserreservoirs, die *Sanierung* von Altlasten oder die *Rekultivierung* von Tagebaufolgelandschaften.

Die Hauptbestandteile hydrogeologischer Modelle werden nun als Unterkonzepte von *Modellierungsobjekt* spezifiziert. Außer den Modellierungszielen sind das die Beschreibung geologischer Gebiete, die Charakterisierung von Materialeigenschaften, die Angabe von Meßmethoden und die Einbeziehung von Berechnungsverfahren.

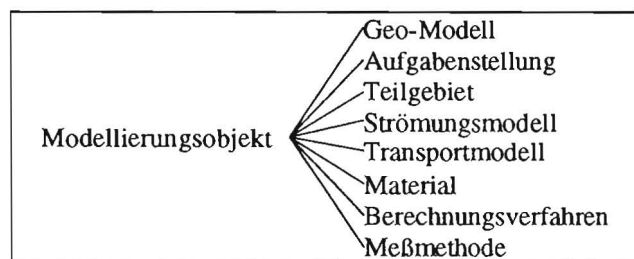


Abb. 2.1: Hauptkomponenten hydrogeologischer Modelle

Die Bezeichnungen auf dieser Abstraktionsebene sollen den Charakter als *Modellierungsobjekt* unterstreichen. Im folgenden Abschnitt ist ein Beispiel für eine Verfeinerung der Begriffshierarchie angegeben. Diese Hierarchie ist in jeder Beziehung offen, so daß weitere Abstraktionen, Zerlegungen, Spezialisierungen und Modifikationen vorgenommen werden können.

²Eine ausführlichere Darstellung ist in [Angelus94] enthalten.

2.2 Charakterisierung von Teilgebieten

Ziel jeder hydrogeologischen Modellierung ist es, bestimmte hydrogeologische Sachverhalte oder Prozesse in ihrer räumlichen und zeitlichen Dynamik für ein spezielles Untersuchungsgebiet darzustellen. Dieses Gesamtgebiet gilt es in Subgebiete zu unterteilen, die in einem gewissen Sinne als homogen zu betrachten sind. Eine derartige räumliche Diskretisierung orientiert sich in der Regel stark an den im Gesamtgebiet vorkommenden Substraten, ihren Phasen sowie ihren hydrogeologisch relevanten Eigenschaften, insbesondere der Grundwasserleitfähigkeit. Je nach Untersuchungsziel werden daher Gebiete ausgewiesen, die ähnliche geochemische Prozesse (i.a. Transport-, Umwandlungs-, Austausch- oder Speicherprozesse) oder geohydraulische Kennwerte (z.B. Durchlässigkeitsbeiwert oder Transmissivität, Porosität oder Speicherkoeffizient) erwarten lassen.

Für die komplette nähere Charakterisierung der Subgebiete ist in der Modellhierarchie das Konzept *Teilgebiet* vorgesehen. Ein Subgebiet soll als *Mehrphasensystem* verstanden werden. Die naturräumlich vorgegebene Trennung von Wasser und Gestein an der Erdoberfläche bedingt die weiterführende Unterteilung des *Mehrphasensystems*, wobei neben dem Unterkonzept *Gewässer* das Gestein entsprechend seinem Verhalten als potentieller Grundwasserleiter in *Festgestein* (Kluftgrundwasserleiter), *Lockergestein* (Porengrundwasserleiter) und den Spezialfall *Karst* (Karstgrundwasserleiter) unterteilt wird.

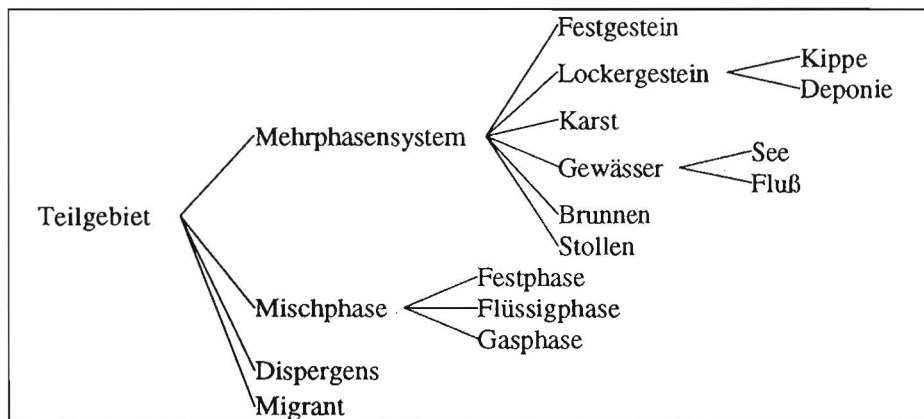


Abb. 2.2: Modellierung von Teilgebieten

Über die Bereichskomponenten *Mischphase*, *Dispergens* und *Migrant* werden die *Mehrphasensysteme* bzw. Subgebiete näher spezifiziert, während der eigentliche Bezug zu konkret gemessenen bzw. ermittelten Meß- und Stoffdaten über das Konzept *Material* realisiert wird.

2.3 Modellierung von Migrationsprozessen

Migrationsprozesse charakterisieren Stoffausbreitungen im Boden und im Grundwasser. Sie werden bestimmt durch die Bewegung der strömenden fluiden Phasen im Untergrund und durch Speicher-, Umwandlungs- und Austauschprozesse, denen die transportierten Migranten unterworfen sind.

Strömungs- und Transportprozesse

Die quantitative Beschreibung der unterschiedlichen Strömungs- und Transportvorgänge wird im allgemeinen durch Systeme gewöhnlicher und partieller Differentialgleichungen realisiert. So werden z.B. Strömungsprozesse auf der Grundlage des *Darcy-Gesetzes* und des Massenerhaltungssatzes beschrieben. Die wichtigsten den Stofftransport im Untergrund bewirkenden Vorgänge sind Konvektion, Dispersion und Diffusion.

Für die Lösung der entsprechenden Differentialgleichungssysteme mit Hilfe numerischer Näherungsverfahren existiert eine ganze Reihe effektiver Computerprogramme. Eine Unterstützung der Modellierung mit solchen Programmen bezieht sich unter anderem auf die Erstellung und Modifikation sachgerechter Eingabedateien durch heuristische und methodische Erfahrungen, die in Form von Fakten und Regeln abgelegt werden. Darüber hinaus soll im Rahmen der Projektarbeiten versucht werden, aus Ergebnisdateien Strategien für die Entwicklung neuer Modelle und Planungsvarianten abzuleiten.

Geochemische Reaktionen

Die Migranten in der Boden- und Grundwasserzone sind vielfältigen chemischen Wechselwirkungen und Umsetzungen unterworfen. Eine für die mathematische Modellierung nützliche Klassifikation unterteilt chemische Reaktionen bzgl.

- homogen bzw. heterogen und
- schnell und reversibel bzw. langsam und/oder irreversibel.

Bei homogenen Reaktionen befinden sich die Reaktionsteilnehmer innerhalb einer Phase. Heterogene Reaktionen berücksichtigen phasenübergreifende Wechselwirkungen zwischen den Reaktionspartnern.

Die Begriffe „schnell“ und „langsam“ stellen den Bezug des Zeitmaßstabes der Reaktion auf den Zeitmaßstab der Transportprozesse her. Zur Beschreibung von schnellen und reversiblen Reaktionen finden thermodynamische Gleichgewichtsmodelle Anwendung, deren Ansätze auf dem Massenwirkungsgesetz beruhen. Gleichgewichtsprozesse werden im allgemeinen als System algebraischer Gleichungen dargestellt. Langsame und/oder irreversible, also nicht an die Einstellung von chemischen Gleichgewichten gebundene Reaktionen, müssen über kinetische Ansätze erfaßt werden, beispielsweise mit Hilfe von Umsatzraten oder Halbwertszeiten. Typischerweise werden solche Prozesse durch Differentialgleichungen bezüglich der Zeit beschrieben.

3. Einbeziehung externer Informationen

3.1 Externe Daten im Konfigurationsprozeß

Bei der Entwicklung des Konfigurierungssystems KONWERK wurde von einer strikten Trennung folgender Teilschritte ausgegangen:

- Wissensakquisition zur Bearbeitung von Domänen- und Kontrollwissen,
- Anforderungsmodellierung bzw. Charakterisierung der Aufgabenstellung und
- Konfigurierung einer Lösung.

Die Einbeziehung externer Informationen wurde (zum Beispiel im Zusammenhang mit der Anwendung fall-basierter Methoden und der Einbeziehung von Simulationsergebnissen) meist nur für den Konfigurierungsprozeß betrachtet. Die Abgeschlossenheit und Unveränderlichkeit der Wissensbasis ist eine wichtige Voraussetzung für die Ableitung korrekter Ergebnisse.

Demgegenüber ist es für praktische Konfigurierungsaufgaben oft notwendig, auch während der Wissensakquisition und der Anforderungsmodellierung einen Zugriff auf externe Dateien zu ermöglichen. Im folgenden soll eine allgemeine Datenschnittstelle vorgestellt werden, die die Verbindungsaufnahme mit unterschiedlichen Datenquellen für das gesamte Konfigurierungssystem in einheitlicher Weise realisiert. Damit wird dann auch eine Mehrfachnutzung spezieller Dateizugriffsmethoden für unterschiedliche Bearbeitungsverfahren ermöglicht.

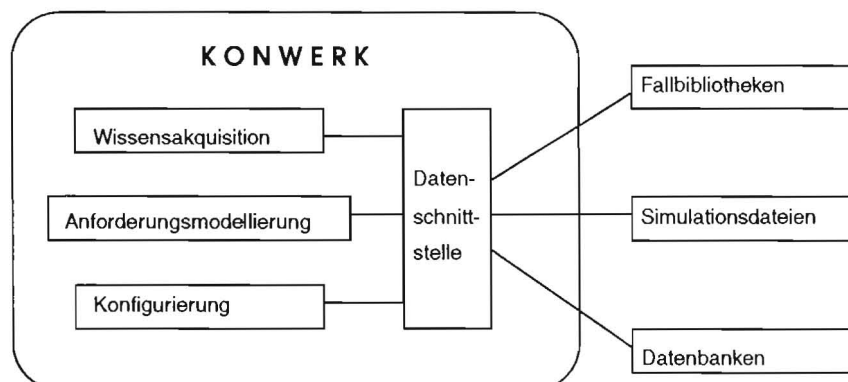


Abb. 3.1.: Nutzung externer Datenquellen für die Konfigurierung

Der Zugriff auf eine Datenbank kann somit für die Generierung von Konzepten und die Festlegung von Parameterbeschränkungen während der Bearbeitung der Wissensbasis genutzt werden. Darüber hinaus kann auch die Anforderungsmodellierung durch die Modifikation externer Daten oder die Generalisierung von Fällen gestaltet werden.

3.2 Entwurf einer allgemeinen Datenschnittstelle

Bei der Konzipierung einer allgemeinen Datenschnittstelle für KONWERK wurde zunächst der Schwerpunkt auf die Einbeziehung von Datenbanksystemen in den Konfigurierungsprozeß gelegt. Ziel ist die Unterstützung objektorientierter als auch relationaler Datenbanken.

Die objektorientierte Realisierung KONWERK's erlaubt die Einführung der Klasse *datenbank-system*:

```
(defclass datenbank-system ()
  ((server      :accessor db-server      :initarg :db-server)
   (user        :accessor db-user        :initarg :db-user)
   (pwd         :accessor db-pwd         :initarg :db-pwd)
   (appname     :accessor db-appname     :initarg :db-appname)
   (db-name     :accessor db-name       :initarg :db-name)
   (is-login    :accessor db-is-login    :initform nil)
   (query-result :accessor db-query-result :initform nil))
)
```

Die Klasse *datenbank-system* soll alle Informationen vereinigen, die sowohl objektorientierten als auch relationalen Datenbanken eigen sind. Es handelt sich dabei um Informationen, die das Anmelden an den Datenbankserver betreffen, den Namen der benutzten Datenbank, sowie das Ergebnis einer Datenbankanfrage. Entsprechende generische Funktionen (*db-login*, *db-use*, *db-get-db-name* ...) werden durch diese Klasse zur Verfügung gestellt. Die Generierung und Ausführung einer Datenbankanfrage übernimmt die Methode *db_select*:

```
(defmethod db_select ( (db datenbank-system) &key) )
```

Das Ergebnis der Anfrage wird dann in dem Slot *query-result* abgelegt.

Von der Klasse *datenbank-system* sind die Klassen *SQL-datenbank-system*, *OO-datenbank-system* und *FILE-daten-system* abgeleitet (s.a. Abb. 3.1). Die *SQL-datenbank-system*-Klasse repräsentiert relationale Datenbanksysteme mit SQL-Unterstützung. Sie enthält zusätzliche Slots für die in der aktuellen Datenbank enthaltenen Tabellen.

```
(defclass SQL-datenbank-system (datenbank-system)
  ((db-table-name-lis :accessor db-table-name-list :initform nil)
   (db-table-list     :accessor db-table-list      :initform nil))
)
```

Die geerbten Methoden (*initialize-instance*, *db-login* ...) wurden angepaßt. Weitere Methoden zur Bestimmung der Slot-Inhalte von *db-table-name-list* und *db-table-list* sind hinzugekommen. Analog zur *SQL-datenbank-system*-Klasse wird mit der Klasse *OO-datenbank-system* verfahren.

Die Klasse *FILE-daten-system* wurde eingeführt, um auch Daten aus einfachen Dateien in gleicher Weise wie Daten aus Datenbanksystemen verarbeiten zu können. Voraussetzung ist, daß die Daten in strukturierter Form abgelegt wurden. Dateien eines Verzeichnisses werden in diesem Fall als Datenbank aufgefaßt.

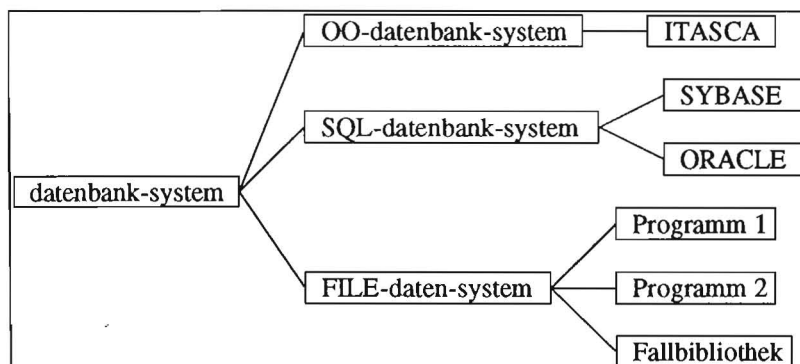


Abb. 3.1: Interne Struktur der Datenschnittstelle

Mit Hilfe der Makros *def-SQL-datenbank-system*, *def-QO-datenbank-system* und *def-FILE-daten-system* besteht die Möglichkeit, sich eigene Datenschnittstellen zu definieren. So generiert der folgende Makroaufruf

```
(def-SQL-datenbank-system SYBASE
  :user      "mimose"
  :dbname    "minerale"
  :dbappname "konwerk Version 0.5"
  :dbserver  "my-sybase")
```

eine Klasse *SYBASE* als Unterklasse von *SQL-datenbank-system*. Von der Klasse *Sybase* wird dann eine Instanz mit den angegebenen Slot-Inhalten erzeugt. Bereits vorhandene Methoden können angepaßt oder neue hinzugefügt werden.

3.3 Nutzung von externen Informationen für die Wissensakquisition

Im folgenden soll an einigen Beispielen die Nutzung der Datenschnittstelle demonstriert werden.

Generierung von Konzepten

Für die Untersuchung von Migrationsprozessen spielt die Beschreibung der stofflichen Eigenschaften des Untergrundes eine wesentliche Rolle. Diese Beschreibung geschieht meist in zwei Schritten:

- die Bestimmung der Gesteinsarten und
- die Charakterisierung der im Gestein enthaltenen Minerale und chemischen Verbindungen.

Für solche Objekte gibt es Tabellen mit normierten Bezeichnungen und Parametern. Leider werden in den einzelnen Behörden und Einrichtungen unterschiedliche Tabellen verwendet, in denen regionale und historische Besonderheiten berücksichtigt werden. Weiterhin werden diese Tabellen für sehr unterschiedliche Anwendungen eingesetzt und unterliegen einer häufigen Änderung. Da den Stoffen verschiedene Eigenschaften zugeordnet sind, ist es sinnvoll, sie als Konzepte darzustellen. Unter diesen Bedingungen ist es notwendig, diese Konzepte mit Hilfe einer Datenbankanfrage während der Wissensakquisition zu generieren. Für die angegebene Wissensbasis wurden Funktionen realisiert, die für die Konzepte *Gestein* und *Mineral* entsprechende Unterkonzepte durch Zugriff auf eine SYBASE-Datenbank oder eine spezielle Eingabedatei für die Berechnung chemischer Gleichgewichte liefern.

Generierung von Wertebereichen für Parameter

Einen wesentlichen Bestandteil von hydrogeologischen Modellierungen bilden umfangreiche Simulationsstudien, für die meist komfortable Spezialprogramme existieren. Ein Datenaustausch ist meist nur über die Ein- und Ausgabedateien dieser Programme möglich. Zur Charakterisierung dieser Programme wurden eigene Konfigurierungskonzepte definiert, für die der Name der Eingabedatei ein eigenständiger Slotwert ist. Während der Wissensakquisition muß für diesen Slot eine Auswahlmenge mit den möglichen Dateinamen generiert werden. Mit der Anweisung

```
(def-do :name      SIMULA
        :oberkonzept COMPUTERPROGRAMM
        :parameter  ((file (:file-liste  "/home/data")))
  ...
)
```

wird ein Konzept *Simula* definiert, für dessen Slot *file* eine Auswahlmenge aller Dateien im Verzeichnis */home/data* generiert wird.

3.4 Datenbankzugriff als Bearbeitungsverfahren

Um Daten während der Konfigurierung über die Datenschnittstelle zur Verfügung zu stellen, wurde das Bearbeitungsverfahren *datenbankanfrage* implementiert. Dieses Bearbeitungsverfahren ruft im wesentlichen die Methode *db-query* (*object slot ks-typ* azw *vorherige-werte*) auf, wobei

<i>object</i>	ein Domänenobjekt aus der Begriffshierarchie,
<i>slot</i>	der zu belegende Slot des Domänenobjektes,
<i>ks-typ</i>	der Konstruktionsschritttyp,

azw der aktuell zulässige Wertebereich,
 vorherige-werte nil oder eine Liste der bereits erfolglos versuchten Werte.

Somit besteht die Möglichkeit, für jeden einzelnen Slot die entsprechende Anfrage an die Datenschnittstelle zu erstellen. In der statischen Wissensbasis ist dem entsprechenden Slot der Name des Bearbeitungsverfahrens *datenbankanfrage* anzuhängen und die Methode *db-query* zu definieren (s. Beispiel). Der Rückgabewert der Methode *db-query* wird als Vorschlag für die Belegung des Slots *slot* betrachtet. Wird der Wert nil zurückgegeben, so gilt das Bearbeitungsverfahren *datenbankanfrage* als fehlgeschlagen und das nächste Bearbeitungsverfahren wird aufgerufen.

Beispiel:

```
(def-do :name MEHRPHASENSYSTEM
      :oberkonzept TEILGEBIET
      :parameter
        ((porositaet [float 0 1] ('datenbankanfrage)))

  (defmethod db-query ((instanz mehrphasensystem)
    (slot (eql 'porositaet))
    (ks ks-param-klasse)
    (azw t)
    (vorherige-werte t))

  ...)
```

Die obigen Beispiele haben Möglichkeiten aufgezeigt, wie mit Hilfe einer allgemeinen Datenschnittstelle eine Kommunikation mit anderen Programmen und Dateien organisiert werden kann. Damit sind dann auch Möglichkeiten geschaffen, Daten zu integrieren, die im Ausgangsmodell nicht enthalten sind. In [Vietze94] wurde eine Konzeption entwickelt, die durch Erweiterung der Kontrollmechanismen solche innovativen Konfigurationsschritte ermöglicht.

4. Fallbasiertes Konfigurieren

4.1 Interaktives Konfigurieren mit einem Bibliotheksbrowser

Eine naheliegende Möglichkeit des fallbasierten Konfigurierens ist die Einbeziehung von Informationen aus ehemaligen Konfigurationen auf interaktivem Wege. Im folgenden wird eine für KONWERK realisierte Variante beschrieben. Der dazu geschaffene „Bibliotheksbrowser“ lädt Fälle aus Dateien, die vorher gespeicherte Konfiguration enthalten. Diese sind im Stil der in KONWERK üblichen Konzeptdefinitionen abgelegt. Nach dem Laden einer Fallbibliothek existieren die Konfigurationen als Instanzen der Klasse „Bibliothekslösung“ und sind für die aktuelle Konfiguration mit Hilfe des Bibliotheksrowsers verfügbar.

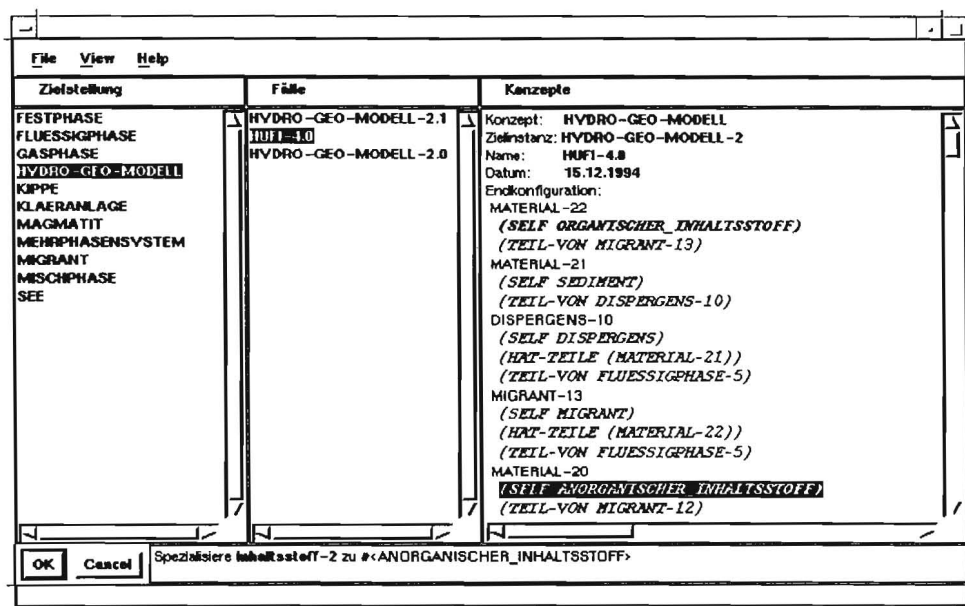


Abb. 4.1: Der interaktive Bibliotheksbrowser

Im Browser werden die aktuell verfügbaren Bibliothekslösungen entsprechend der jeweiligen Zielstellung geordnet. (s. Abb. 4.1). Die dem aktuellen Konstruktionsschritttyp entsprechenden Slots mit den zugehörigen Slotwerten werden hier graphisch hervorgehoben. Nach der interaktiven Auswahl kann der Slotwert an die Konfigurierung übergeben werden.

Der Konstruktionsschritt wird entweder mit diesem Slotwert aus der Bibliothekslösung vollzogen oder es erscheint eine Fehlermeldung über den für diesen Schritt per aktuell zulässigem Wertebereich nicht erlaubten Slotwert. Slotwertübernahmen mit Hilfe des Bibliotheksbrowsers sind möglich für Parametrierungen, Spezialisierungen und Zerlegungen. Eine graphische Hervorhebung der SELF-Slots und zugehöriger Slotwerte bei Spezialisierungen erfolgt, wenn die Begriffshierarchie der aktuellen Konfigurierung eine Spezialisierung auf jenen Slotwert erlaubt.

Eine Bibliothekslösung kann nach dem Ende einer erfolgreichen Konfigurierung auf Wunsch in eine Fallbibliothek abgespeichert werden.

4.2. Einbindung fallbasierter Methoden in die Kontrolle des Konfigurierungsprozesses

Der Kontroll-Modul von KONWERK bietet dem Anwender die Möglichkeit, den Konfigurierungsprozeß individuell zu steuern. Dies geschieht über Strategiedefinitionen, die Bestandteil der statischen Wissensbasis sind und in denen anwender- bzw. domänenspezifische Forderungen eingebettet werden können. Insbesondere enthält jede Strategie eine nach Prioritäten geordnete Liste von Bearbeitungsverfahren, die während des Konfigurierungsprozesses zur Anwendung kommen. In jedem Konfigurierungsschritt wird die Liste der Bearbeitungsverfahren, beginnend mit demjenigen der höchsten Priorität, systematisch durchlaufen. Die Anwendung eines Bearbeitungsverfahrens liefert einen für den jeweiligen Konfigurierungsschritt zulässigen „Wert“, der das Ergebnis des aktuellen Konfigurierungsschrittes repräsentiert, oder wird mit dem Rückgabewert NIL abgeschlossen. In ersterem Fall bleiben die nachfolgenden Bearbeitungsverfahren (für den aktuellen Schritt) unberücksichtigt, in letzterem Fall wird mit dem nächsten Bearbeitungsverfahren fortgesetzt.

Neben der im vorangegangenen Abschnitt beschriebenen interaktiven Möglichkeit des fallbasierten Konfigurierens kann über die Definition geeigneter Bearbeitungsverfahren und zugehöriger Methoden die fallbasierte Arbeit auch ohne Interaktion durchgeführt werden. Dies ist insbesondere für stark strukturierte Domänen von Interesse, wo vorhandenes Fallwissen ohne nochmalige Prüfung direkt in den aktuellen Konfigurierungsprozeß übernommen werden soll.

Im CBR-Modul (CBR = case based reasoning) von KONWERK sind z.Z. zwei fallbasierte Techniken realisiert. Beide gehen von einem Referenzfall aus, an dem sich der aktuelle Konfigurierungsprozeß orientiert. Ein Referenzfall ist ein Tripel (**ini-tk**, **elab-netz**, **end-tk**), wobei

ini-tk	eine initiale Teilkonstruktion (=Aufgabenstellung),	
end-tk	eine aus ini-tk in einem früheren Konfigurierungsprozeß hervorgegangene fertige (vollständig instantiierte) Endkonstruktion	und
elab-netz	das zugehörige Elaborationsnetz	

repräsentieren. Referenzfälle sind in einer objektorientierten Datenbank abgelegt, die über eine KONWERK-eigene Datenbankschnittstelle an das System angebunden ist.

Nach jedem Konfigurierungsprozeß erhält der Anwender die Möglichkeit, die von ihm generierte Konfiguration als Fall in die Falldatenbank einzuspeichern, so daß diese für spätere Konfigurierungsprozesse als möglicher Referenzfall zur Verfügung steht.

Die Bereitstellung des Referenzfalles erfolgt außerhalb des Konfigurierungsprozesses. Referenzfälle können vor und während des Konfigurierungsprozesses ausgewählt werden. Insbesondere kann während der Konfigurierung der Referenzfall gewechselt und die fallbasierte Arbeit jederzeit aktiviert bzw. deaktiviert werden.

Im folgenden sollen die beiden bisher implementierten CBR-Verfahren kurz skizziert werden.

Das Bearbeitungsverfahren **Werteheuristik** übernimmt aus dem jeweils aktuellen Referenzfall den „Wert“ des aktuellen Konfigurierungsschrittes. Hierzu wird im Elaborationsnetz des Falles eine zum aktuellen Konfigurierungsschritt adäquate Stelle gesucht. D. h., es wird im Elaborationsnetz des Falles ein Konfigurierungsschritt gesucht, bei dem Objekt, Slot und Typ des Konfigurierungsschrittes mit dem aktuell durchzuführenden Schritt übereinstimmen. Dessen „Wert“ wird dann im aktuellen Schritt wiederverwendet.

Ist die Suche im Referenzfall nach einem adäquaten Schritt erfolglos, wird mit dem nächsten in der Strategie festgelegten Bearbeitungsverfahren fortgesetzt. Werden mehrere adäquate Stellen gefunden, erhält der Anwender die Möglichkeit, den von ihm gewünschten Referenzschritt interaktiv zu bestimmen. Ist kein Referenzfall aktiv, so liefert das Bearbeitungsverfahren Werteheuristik NIL als Rückgabewert, d.h., in diesem Fall bleibt das Bearbeitungsverfahren wirkungslos.

Das Bearbeitungsverfahren **Verfahrensheuristik** sucht analog zur Werteheuristik im aktuellen Fall einen adäquaten Referenzschritt, übernimmt jedoch nicht den „Wert“ des Referenzschrittes in den aktuellen Konfigurierungsprozeß, sondern übergibt lediglich das im Fall verwendete Bearbeitungsverfahren. Der aktuelle Konfigurierungsschritt wird dann mit diesem Bearbeitungsverfahren durchgeführt.

Wie bei der Werteheuristik wird bei erfolgloser Suche nach einem adäquaten Referenzschritt oder wenn kein Referenzfall aktiv ist mit dem nächsten Bearbeitungsverfahren der aktuellen Strategie fortgesetzt. Bei mehreren in Frage kommenden Referenzschritten erfolgt wiederum eine interaktive Auswahl des gewünschten Referenzschrittes.

4.3 Konfigurieren mit Teillösungen

Um die Auswahl eines Referenzfalles aus einer Fallbibliothek effektiv zu gestalten, ist es sinnvoll, vor der eigentlichen Fallauswahl die potentielle Fallmenge in geeigneter Weise einzuschränken.

Auf der Grundlage einer Bewertung aller Fälle läßt sich z.B. durch Einführung eines Abstandsmaßes die Ähnlichkeit der vorhandenen Fälle mit der gegebenen Aufgabenstellung beschreiben (s. [Pfitzner93] und [Reuter93]). Durch die Angabe einer Toleranzgrenze für das Abstandsmaß kann zunächst eine Grobauswahl vorgenommen werden. Eine wichtige Charakterisierungsmöglichkeit geeigneter Fälle besteht in der Angabe einer Auswahl von Teillösungen, die die Endkonfiguration enthalten sollte. Da beim Konfigurieren mit KONWERK jedes Konzept der Begriffshierarchie auch als Zielobjekt verwendet werden kann, kommen als Teillösungen sowohl Teil- als auch Endkonfigurationen in Betracht. Komplexe Teillösungen können somit auch einzeln konfiguriert werden.

Die im fallbasierten Modul von KONWERK realisierte Grobauswahl basiert auf der Vorgabe einer Liste von Konzepten, auf deren Grundlage relevante Fälle bereitgestellt werden. Durch geeignete Strategien können damit einzelne Teillösungen aus der Fallbibliothek komplett übernommen werden. Da der Referenzfall während der Konfigurierung gewechselt werden kann, müssen die jeweiligen Teillösungen nicht alle aus einem einzigen Fall entnommen sein.

An der Implementation weiterer Auswahlkriterien und Verfahren zur Integration von Wissen aus externen Datenquellen wird gearbeitet.

5. Literatur

- [Angelus94] A.Angelus, Ch.Bohley, J.-P.Grell, A.Kathe, D.Lazik, I.Senf: Rechnergestützte Konfigurierung hydrogeologischer Modelle; 4. Arbeitstreffen „Werkzeuge für Simulation und Modellbildung im Umweltbereich“, 1994
- [Günter91] A.Günter, H.Dörner, H.Gläser, B.Neumann, C.Posthoff, H.-J.Sebastian: Das Projekt PROKON im Überblick; PROKON-Bericht Nr. 1, 1991
- [Fabel93] C.-H.Coulon, H.Dürschke, W.Gräther, B.Linowski, W.Oertel: Fallverwaltung und Fallretrieval - Implementierung und Tests; FABEL Report No. 7, 1993
- [Kopisch92] M.Kopisch, A.Günter: Konfigurierung der Passagierkabine des AIRBUS A340 basierend auf einer Begriffshierarchie , einem Constraint-System und einer flexiblen Kontrolle; PROKON-Bericht Nr 2, 1992
- [Pfitzner93] K.Pfitzner: Fallbasiertes Konfigurieren technischer Systeme; Diss. Hamburg 1993
- [Reuter93] H.Reuter: Abstrakte Räume als theoretische Basis für die Beschreibung von Konfigurierungsproblemen; PROKON-Memo Nr. 44, 1993
- [Schädler93] K.Schädler, B.Kampa: Chemisches Konfigurieren mit Plakon; Beiträge zum 7. Workshop Planen und Konfigurieren, 1993
- [Strecker91] H.Strecker: Simulationsgestützte Systemkonfigurierung am Beispiel automatischer Röntgenprüfsysteme. Diss. Hamburg 1991
- [Vietze94] L.Hotz, Th.Vietze: Innovatives Konfigurieren in technischen Domänen; Beiträge zum 9. Workshop Planen und Konfigurieren, 1995

Konfigurieren bei der Angebots- und Auftragsklärung

Konfigurieren bei der Angebots- und Auftragsklärung¹

Wolfgang Tank

MAZ Hamburg GmbH
Harburger Schloßstraße 6-12
21079 Hamburg

Kurzfassung

Konfigurieren ist gerade für die Angebots- und Auftragsbearbeitung interessant, weil die Vollständigkeit und Konsistenz von Angeboten bzw. Aufträgen gewährleistet, der Prozeß der technischen Klärung beschleunigt und die Wirtschaftlichkeit des Einsatz von Konfigurierungstechnologie in diesem Kontext plausibel dargestellt werden kann. Im Rahmen der Erläuterung von spezifischen Fragestellungen aus dem Anwendungskontext werden insbesondere unternehmensstrategische Wirkungen von Konfigurationssystemen diskutiert.

Einleitung

Im Zuge der Globalisierung von Märkten und des gestiegenen Wettbewerbsdrucks kommt es gerade für exportorientierte Unternehmen nicht mehr auf die breiteste Produktpalette sondern auf die Qualität der Vertriebskanäle und die Einschätzung der Kundenwünsche an.

Gerade Fertigungsunternehmen in Deutschland, deren Unternehmensphilosophie sehr stark durch Ingenieurskunst geprägt ist, stehen hier vor einem schmerzlichen Umorientierungsprozeß, um ihre weltweite Wettbewerbsposition zu behaupten. Dieser Prozeß läßt sich auf den einfachen Nenner bringen: Produkte statt Sonderlösungen.

Konfigurierungssysteme können diesen Prozeß in vielfältiger Hinsicht fördern und treffen auf einen wachsenden Markt. Die Nische zwischen Standarderzeugnissen und Sonderanfertigungen wird immer breiter:

- Standarderzeugnisse werden auf Kundendruck zunehmend variantenreicher.
- Sonderanfertigungen basieren aus Kostengründen mehr und mehr auf standardisierten Teilen.

¹ Der Beitrag gibt eine erläuternde Einführung zur gleichnamigen Spezialsektion des 9. Workshops Planen und Konfigurieren (PuK) im Rahmen der 3. Deutschen Expertensystemtagung (XPS'95).

Die meisten praktischen Konfigurierungssysteme werden im Rahmen der Angebots- und Auftragsklärung angewendet. Die im Workshop vorgestellten Anwendungen sollen zielgerichtet anhand nachfolgender Fragen diskutiert werden.

Warum wird Konfigurierungstechnologie gerade bei der Angebots- und Auftragsklärung angewendet?

Methoden und Verfahren der KI zur Lösung von Konfigurierungsproblemen gehen von einem abstrakten strukturwissenschaftlichem Problemverständnis aus: Ziel ist es, ein komplexes Gebilde aus Einzelteilen nach gewissen Regeln zusammenzustellen (vgl. u.a. [1], [2]). Mit diesem Verständnis lassen sich innerbetrieblich in den unterschiedlichsten Bereichen Konfigurierungsprobleme finden.

Es liegt an den besonderen Nutzeffekten, daß die meisten praktischen Konfigurierungssysteme für die Angebots- und Auftragsklärung verwendet werden. Der Nutzeffekt läßt sich sowohl in Zeit, Kosten und Qualität ausdrücken.

Das Erstellen von Angeboten über variantenreiche Produkte dauert zu lange. Dies gilt insbesondere dann, wenn die Variantenvielfalt aufgrund vielfältiger Restriktionen nur unzureichend beherrscht wird. Die Folge davon:

Viele Angebote sind technisch unzureichend geklärt, und verursachen im Auftragsfall nicht kalkulierte Folgekosten. Hinzu kommt, daß die Umwandlungsrate von Angeboten zu Aufträgen gering ist. Nicht erfolgreiche Angebote erhöhen die Gemeinkosten.

Konfigurierungssysteme minimieren das technologische und finanzielle Risiko durch fundiertere Angebots- und Auftragsklärung. Sie senken erwiesenermaßen die Fehlerrate von Angeboten und Aufträgen und sorgen damit für einen reibungslosen, termingerechten Ablauf der Produktion ohne unkalkulierte Folgekosten.

In Verbindung mit der Annahme bekannter Einzelteile und vorausgedachter Aufbaustrukturen, die den meisten Konfigurierungsmethoden zugrunde liegt, wirken Konfigurierungssysteme als unternehmensstrategisches Instrument in folgender Hinsicht:

Konfigurierungssysteme fördern die Produktorientierung eines Unternehmens durch

- Straffung der Sortiments und der Variantenvielfalt sowie durch
- Erhöhung des Grads der Wiederverwendung von Lösungen

Dabei setzen sie einen Gruppenarbeitsprozeß in Gang, der weite Teile des Unternehmens erfaßt (Entwicklung, Konstruktion, Vertrieb, Einkauf, Arbeitsvorbereitung). Als Folge reduziert sich der Abstimmungsaufwand zwischen den an Angebots- und Auf-

tragsabwicklung beteiligten Stellen. In der Produktion ergibt sich ein Rationalisierungseffekt durch Standardisierung und höhere Losgrößen.

Konfigurierungssysteme fördern die Entscheidung für Baukastenentwicklungen. Viele Unternehmer scheuen eine solche Entscheidung angesichts a priori höherer Entwicklungskosten und einer unzureichenden Gewähr, daß sich eine Baukastenentwicklung anfrage- und auftragsgebunden tatsächlich durch geringeren Aufwand rechnet. Ein Hauptgrund liegt in der mangelnden Wissenssicherung aus dem Entwicklungsprozeß aufgrund unzureichender Arbeitsmittel. Konfiguratoren können hier Abhilfe schaffen.

Konfigurierungssysteme erschließen neue Vertriebswege durch personenungebundene Verfügbarkeit technischer Entscheidungskompetenz und ermöglichen einen aktiven Vertrieb. Für variantenreiche Standardprodukte wird die Angebotserstellung im Außendienst möglich.

Konfigurierungsarbeiten erhöhen die Marktpräsenz eines Unternehmens durch eine höhere Anzahl umlaufender Angebote. Bei einer allgemein steigenden Zahl von Anfragen schaffen Konfigurierungssysteme die Bearbeitungskapazitäten, um unabhängig von der Werdewahrscheinlichkeit eines Auftrags mehr Kundenanfragen qualifiziert zu bearbeiten.

Stellen Angebots- und Auftragsklärung unterschiedliche Anforderungen an die Konfigurierungstechnologie und macht es aus Sicht der Technologie noch Sinn, die beiden Aufgaben zu unterscheiden?

Die Unterscheidung zwischen Angebots- und Auftragsklärung ist bewährte betriebliche Praxis. Im Rahmen der Angebotserstellung wird ein Kundenproblem in Abwägung des Aufwands zum erwarteten Nutzen durch einen ungewissen Auftragseingang nur soweit geklärt, wie es zur Minimierung des unternehmerischen Risikos notwendig ist. So pragmatisch die Unterscheidung auch sein mag, so liegt in ihr doch auch die besondere Herausforderung, unabhängig ob Anfrage oder Auftrag dahinzukommen, in jedem Fall die Klärung mit der gleichen Gründlichkeit durchzuführen, und das möglichst ohne Mehraufwand.

Ergebnis der Angebotsklärung ist eine Angebotsstückliste aus vertrieblicher Sicht. Die Auftragsklärung liefert eine Auftragsstückliste aus konstruktiver und produktionstechnischer Sicht. Es liegt auf der Hand, daß Angebotsstücklisten eine wesentlich geringere Tiefe als Auftragsstücklisten haben und ein Erzeugnis nicht bis zur "letzten Schraube" auflösen.

Für eine Angebotsklärung mit der Gründlichkeit einer Auftragsklärung müßte man jedoch eine Auflösung bis auf Bauteilebene vornehmen, und das Angebot dann durch Abstraktion von der Auftragsstückliste gewinnen.

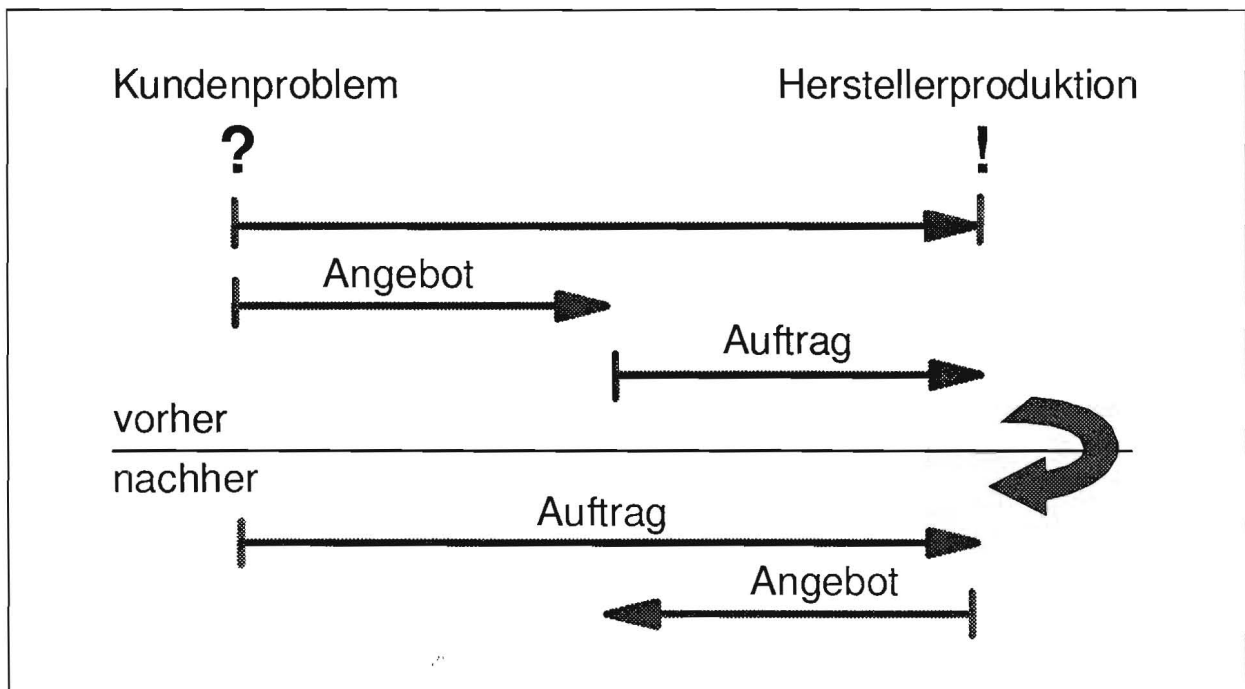


Abb. 1: Konkretisierungsgrad von Angebots- und Auftragsklärung

Dies ist bei hinreichender Abstimmung zwischen Angebots- und Auftragsstücklisten sicherlich zu leisten. Eine besondere Schwierigkeit erwächst jedoch aus der Tatsache, daß Vertrieb und Entwicklung typischerweise unterschiedliche Sichten auf eine Erzeugnisgliederung haben und diese nicht unbedingt strukturerhaltend sein müssen.

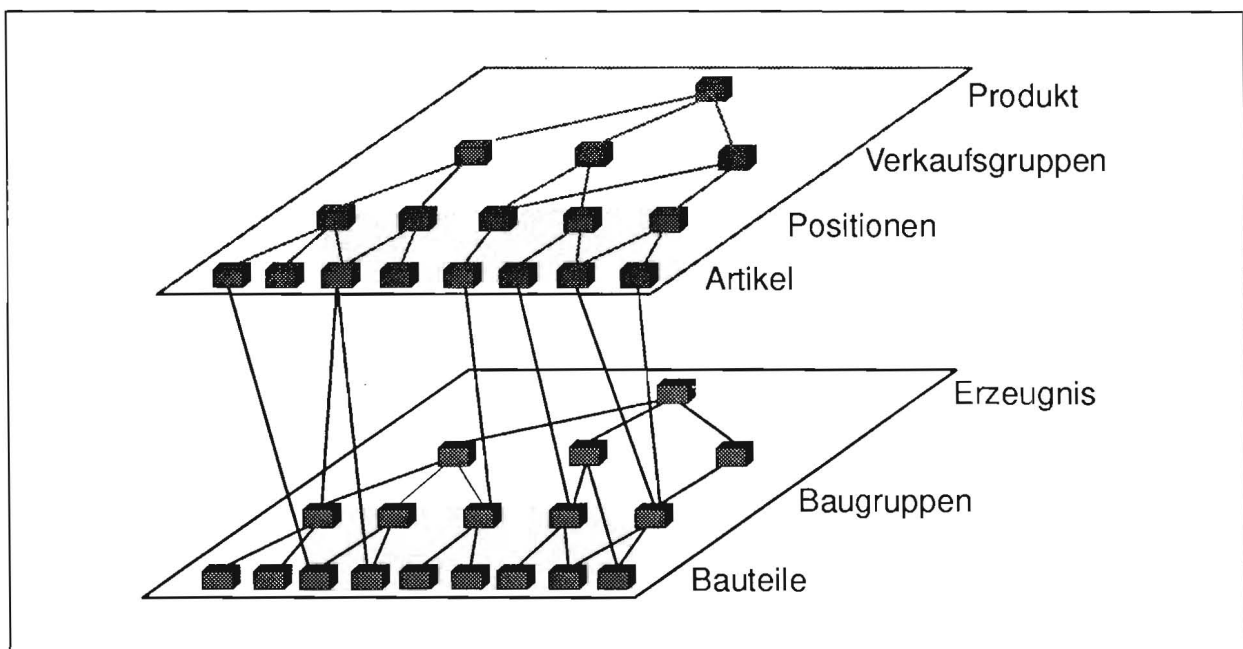


Abb. 1: Erzeugnisgliederung aus technischer und vertrieblicher Sicht

Zwar werden Verkaufsartikel in der Regel zu einem oder mehreren Bauteilen bzw. Baugruppen aufgelöst. Es kann aber auch der umgekehrte Fall auftreten, daß ein Multifunktionsbauteil aus strategischen Gründen vertrieblich differenziert wird.

Bei der Verfeinerung von Angebotsstücklisten bzw. bei der Abstraktion von Auftragsstücklisten müssen so komplexe n:m-Beziehungen gehandhabt werden, die es gilt deklarativ zu beschreiben, um ein integriertes vertriebliches und technisches Produktmodell dauerhaft konsistent zu halten.

Welche Auswirkungen hat der Einsatz von Konfigurierungssystemen auf betriebliche Abläufe in diesem Kontext?

Konfigurierungssysteme sind das Ergebnis einer Wissenssicherung aus dem Entwicklungszyklus und nehmen die Rolle eines Bindegliedes zwischen Entwicklung und Produktion ein. In diesem Sinne fördern sie die Trennung von Entwicklungs- und Produktionszyklen, wobei mit Produktionszyklus die Abwicklung von Anfragen und Aufträgen eingeschlossen ist. Durch den Einsatz von Konfigurierungssystemen wird bei entsprechender Produktvorbereitung die Konstruktion immer seltener in die Auftragsabwicklung einbezogen und dadurch die Durchlaufzeiten von Aufträgen erheblich verkürzt.

Für variantenreiche Standardprodukte, die nach Katalog und Preisliste verkauft werden, wird es durch Konfigurierungssysteme möglich, die Angebotserstellung in den Außendienst zu verlagern mit der Konsequenz der Konzentration des gesamten Geschäftsprozesses auf den Verkäufer und Freisetzungen im Bereich des klassischen Verkaufsinnendienstes.

Wie werden Konfigurierungssysteme in die DV-Infrastruktur integriert?

Die Integration von Konfigurierungssystemen läuft bisher in den meisten Fällen auf teure Einzellösungen hinaus. Zur Integration werden typischerweise Schnittstellen zu CAD-Systemen, Konstruktions- und Produktdatenverwaltungssystemen (EDM und PDM) PPS-Systemen und auch zu Vertriebsinformations- und -steuerungssystemen (VIS) geschaffen.

Es bleibt zu hoffen, daß im Zuge der Bestrebung eines normierten Datenaustausches zwischen CAD-Systemen auf der Basis von STEP/EXPRESS auch die Integration von Konfigurierungssystemen über diese systemneutrale Schnittstelle gehandhabt werden kann.

Welche Erfahrungen mit kommerziell verfügbaren Konfigurierungswerkzeugen wurden dabei gemacht?

Mittlerweile existieren eine Reihe von kommerziell verfügbaren Konfigurierungswerkzeuge. Zu nennen wären beispielhaft die Systeme SAP-Konfigurator, Object Engine (Camos), XCON (Dr. Schade und Partner), Configuration Master (ISE), INKOS (GPS) oder KONFEX (Planware).

Auch haben viele PPS- und EDM-Systeme mittlerweile Module für eine wissensbasierte Variantengenerierung. Viele dieser Systeme sind Entscheidungstabellenorientiert und orientieren sich an der ISO-Norm 5806 [2].

Interessant für eine anwendungsorientierte Forschung auf dem Gebiet wissensbasierter Konfigurierung sind Erfahrungen, die aus Anwendungen mit diesen Systemen gemacht worden sind.

Schlußwort

Konfigurierungssysteme haben Ihren betrieblichen Nutzen nachweislich unter Beweis gestellt. So konnte in Einzelfällen der auftragsgebundene Entwicklungs- und Konstruktionszyklus verkürzt werden. Eine Senkung der Fehlerquote in Produktionsstücklisten hat den reibungslosen Durchlauf von Aufträgen durch Fertigung und Montage gefördert. Ebenso haben wissensbasierte Konfigurierungssysteme die Qualität von Angeboten verbessert. Verbunden mit einer beschleunigten Angebotsabgabe konnte in verschiedenen Unternehmen die Umwandlungsrate von Angeboten zu Aufträgen nachweislich erhöht und damit die Wettbewerbsposition der Unternehmen gestärkt werden (vgl. [4]).

In der unternehmensstrategischen Bedeutung von Konfigurierungssystemen liegt Chance und Hindernis zu gleich. Konfigurierungsprojekte sind ganzheitlich zu bearbeiten (vgl. [4]). Sie erfassen dabei notwendigerweise weite Teile des Unternehmens. Für ein solches Projekt ist der erklärte Wille der Geschäftsleitung und das klare Bekenntnis für eine Produktstrategie notwendig.

Unter diesen Voraussetzungen ist die Akquisition eines Konfigurierungsprojektes mit langer Überzeugungsarbeit verbunden. Das beste Argument würde dabei die Zusage liefern, daß einmal mühsam aufbereitetes Wissen dauerhaft von Anwenderseite gepflegt werden kann. In Anbetracht der Dynamik des Wissens um ein Produkt bestehen hier noch große Defizite den Bereich der Wissensakquisitionsunterstützung betreffend.

Literatur

- [1] **S. Biundo, A. Günter, J. Hertzberg, J. Schneeberger und W. Tank:** *Planen und Konfigurieren*. In: Einführung in die Künstliche Intelligenz (Hrsg. G. Görz), Addison-Wesley Publishing, S. 767-828, 1993
- [2] **W. Tank:** *Wissensbasiertes Konfigurieren - Ein Überblick*. In: KI 1/93, FBO-Verlag, S. 7-10, 1993
- [3] **ISO 5806:** *Informationsverarbeitung; Spezifikation für Entscheidungstabellen mit je einer Antwort*, 12.84
- [4] **W. Tank:** *Konfigurierung in der Angebotserstellung - Technologietransfer als ganzheitliches Beratungsthema*. In: Beiträge zum Workshop "Modelle in der Konfigurierung" (Hrsg. A. Günter), KI'93, Berlin, 1993

angebotsgerechten Konfiguration von Investitionsgütern

ANDREAS KEBBEL UND MATHIAS KRÖMKER

BREMER INSTITUT FÜR BETRIEBSTECHNIK UND ANGEWANDTE ARBEITSWISSENSCHAFT (BIBA)
HOCHSCHULRING 20 28359 BREMEN E-MAIL: kro@biba.uni-bremen.de

Über den Inhalt:

Im Beitrag wird ein Windows-basierter Lösungsansatz eines Systems zur integrierten Angebotserstellung für komplexe Produkte vorgestellt. Im System, das alle angebotsrelevanten Tätigkeiten unterstützt, ist ein Thesaurus zur Konfiguration der technischen Lösung enthalten. Es handelt sich dabei um einen praxisorientierten Ansatz, der z. Zt. bei zwei Entwicklern von Sondermaschinen auf deren spezifische Bedürfnisse angepaßt wird. Neben der Beschreibung des Systemkonzepts schildern die Autoren die Umsetzung in die Praxis.

Über die Autoren:

Dipl.-Ing. Andreas Kebbel, Jahrgang 1966, Studium der Produktionstechnik an der Universität Bremen von 1986 bis 1993. Seit April 1993 wissenschaftlicher Mitarbeiter am BIBA im Bereich „Angebotserstellung“.

Dipl.-Ing. Mathias Krömer, Jahrgang 1966, studierte Produktionstechnik an der Universität Bremen. Nach erfolgreichem Abschluß im Jahre 1991 als wissenschaftlicher Mitarbeiter am BIBA. Er ist verantwortlich für das Aufgabengebiet „Angebotserstellung“.

1. Problematik der Angebotserstellung für Investitionsgüter

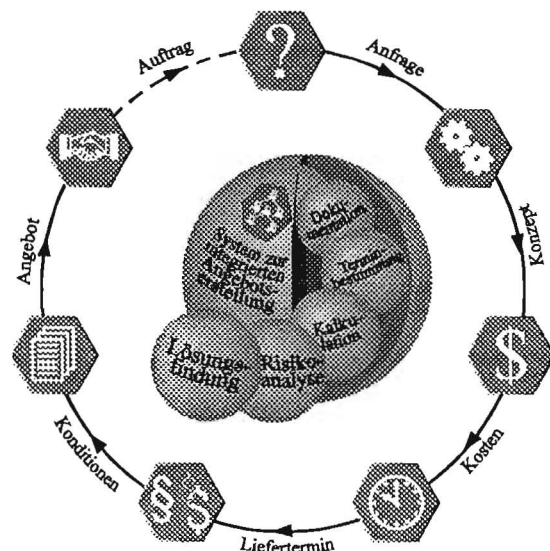
Komplexe Produkte wie Anlagen oder Sondermaschinen, die nach Kundenspezifikationen entwickelt werden, verkaufen sich nicht von selbst - zunächst muß ein überzeugendes, ansprechendes Angebot ausgearbeitet werden. Dadurch entstehen bereits im Vorfeld eines Auftrags erhebliche Kosten, die auf wenige gewonnene Aufträge umgelegt werden müssen. Je komplexer das angebotene Produkt, desto größer wird der Aufwand bei der Angebotserstellung.

Ein Angebot unterliegt verschiedenen, teilweise gegensätzlichen Anforderungen. So steht die Verlässlichkeit der im Angebot gemachten Angaben im Widerspruch zum Zeitdruck, der durch kurzfristige Änderungswünsche seitens des Kunden entsteht. Die technische Machbarkeit kann oft aufgrund des unvollständigen Anforderungsprofils und der Neuheit des Produktes nicht abgesichert werden. Überdies erschweren eng bemessene Abgabetermine die Arbeit.

2. Entwicklung eines Systems zur integrierten Angebotsbearbeitung (SINA)

Effiziente Methoden und innovative Werkzeuge können die verschiedenen Phasen der Angebotsbearbeitung (Anfrageerfassung und -bewertung, Lösungsfindung, Kalkulation, Terminierung und Dokumenterstellung) wesentlich erleichtern (s. *Abb. 1* rechts). Anliegen des Bremer Instituts für Betriebstechnik und angewandte Arbeitswissenschaft (BIBA) ist es, durch Entwicklung eines PC-basierten Systems unter der Benutzungsoberfläche „Windows“ eine praxisgerechte Lösung für klein- und mittelständische Unternehmen anbieten zu können.

Um den Entwicklungsaufwand so gering wie möglich zu halten und eine hohe Ausfallsicherheit zu gewährleisten, wird das System durch Anpassung von kommerzieller Standardsoftware erstellt. Abb. 2 auf der folgenden Seite stellt das Prinzip dar und nennt die wesentlichen Applikationen. Von entscheidender Bedeutung ist der Integrationsaspekt, d. h. die Beantwortung der Fragestellung, wie die Einzelkomponenten anwenderfreundlich zu einem Gesamtsystem integriert werden können.



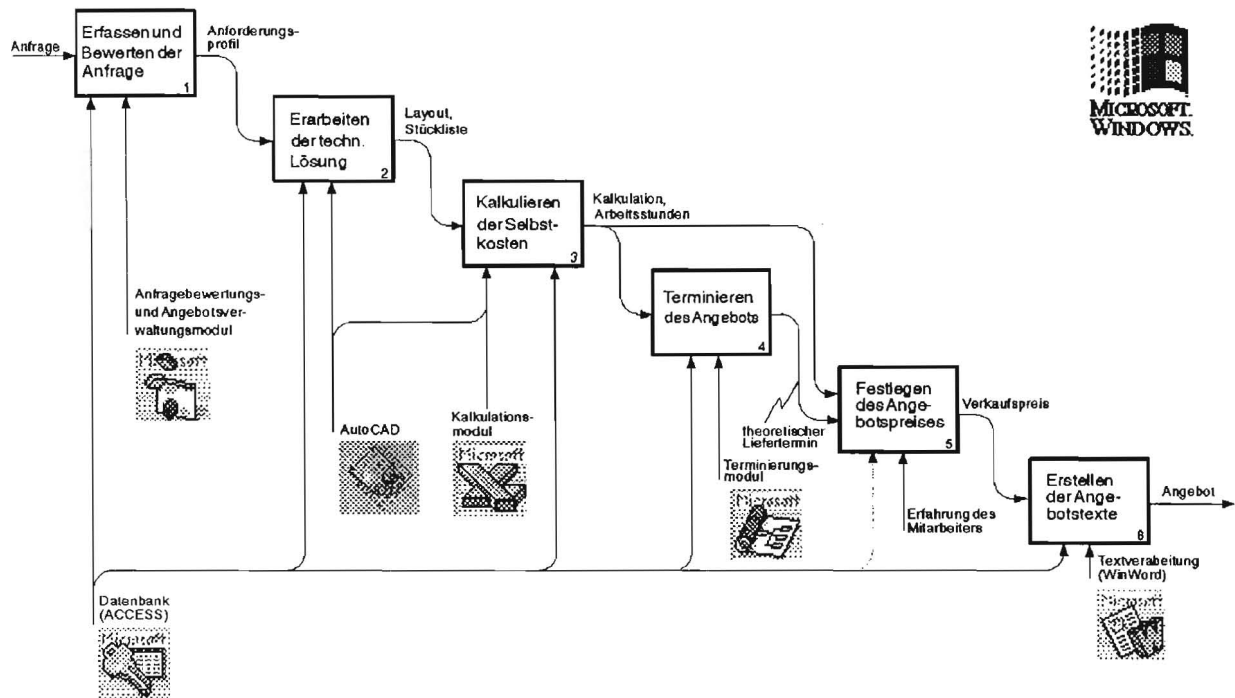


Abb. 2 Unterstützung der Angebotserstellung durch EDV-Applikationen

Das Basissystem wurde zunächst in Zusammenarbeit mit einem Entwickler von flexiblen Montagelinien (s. Produktbeispiel in Abb. 3) entwickelt. Anschließend erfolgte eine Anpassung für einen Anbieter von Hochgeschwindigkeitsschleifmaschinen.

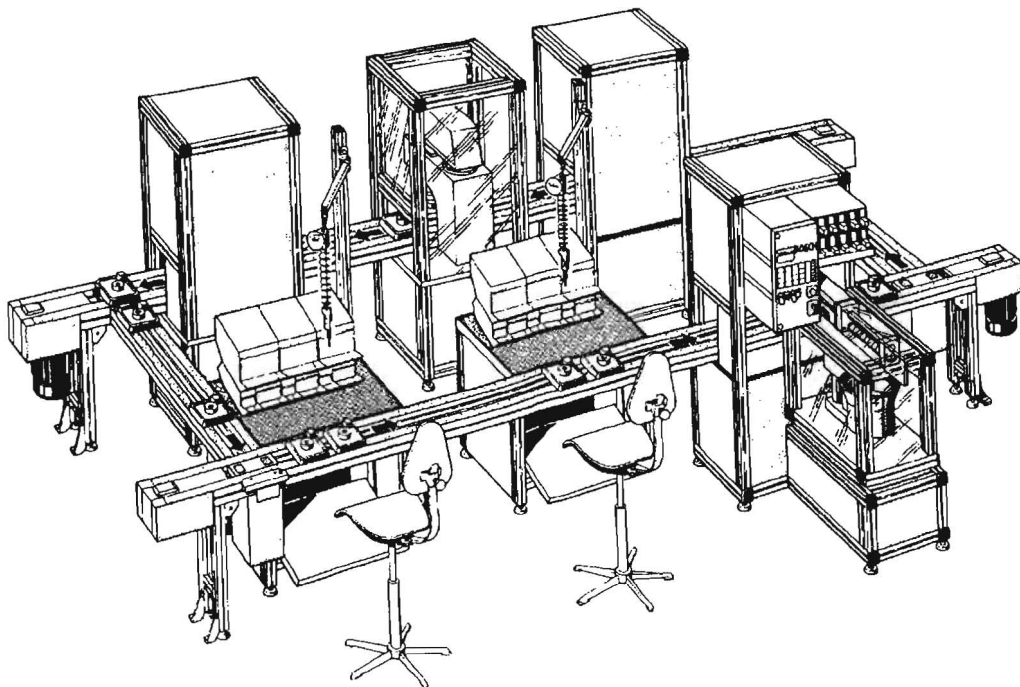


Abb. 3 Beispiel einer Montagelinie (Quelle: Robert Bosch GmbH)

3. Beschreibung der Systemfunktionalitäten

Das System zur integrierten Angebotserstellung unterstützt alle wesentlichen Aktivitäten bei der Angebotserstellung. Der Anwender gelangt zunächst in eine Hauptübersicht, in der er bestehende bzw. in Bearbeitung befindliche Angebote abrufen sowie neue Angebotsprojekte erstellen kann.

Ausgehend von dieser Bildschirmmaske können angebotsspezifisch Anfrageerfassungsdaten abgefragt werden. Hierzu wurde eine produktspezifische Checkliste elektronisch.

Ein wichtige Voraussetzung einer erfolgreichen Akquisition stellt die kontinuierliche Kommunikation mit dem Kunden dar. Da sowohl auf Kundenseite als auch auf Seite des Anbieters eine Vielzahl von Mitarbeitern im Rahmen der Angebotsbearbeitung einbezogen werden, kommt es zum Austausch einer Fülle von Unterlagen. Das SINA-System ermöglicht die automatische Verwaltung aller beim Anbieter generierten Dokumente, d. h., der Angebotsbearbeiter braucht sich nicht um die physikalische Speicherung und Verwaltung der von ihm erstellten Unterlagen zu kümmern. Durch Anwählen der entsprechenden Dateisymbole wird automatisch das Dokument mit der dazugehörigen Anwendung (hier: die entsprechende Textverarbeitung) gestartet. Ist noch kein Dokument vorhanden (weil beispielsweise ein neues Telefax erstellt werden soll), so wird dieses neu angelegt.

Die entscheidende Unterstützung der Angebotsbearbeitung liefert das System jedoch bei der technischen Lösungsfindung und der dazugehörigen Kalkulation.

4. Konfiguration im Rahmen der technischen Lösungsfindung

Die eigentliche Hauptaufgabe der Angebotsbearbeitung stellt die technische Lösungsfindung dar. Beispielsweise umfaßt ein Angebot für eine Montageanlage zur Verdeutlichung des Funktionsprinzips stets ein Anlagenlayout. Auch für eine zuverlässige Preisbestimmung ist Konstruktionsarbeit in der Angebotsphase unumgänglich. Der Konstruktionsprozeß im Rahmen einer Angebotsbearbeitung läßt sich nach [1] in drei bzw. vier Phasen aufgliedern (siehe auch Abb. 4):

- Funktionsfindung: Unter dem Begriff "Funktionsfindung" ist die Festlegung der Gesamt- und Teilfunktionen und damit eines funktionalen Ablaufschemas zu verstehen. Bezogen auf eine Montageanlage bedeutet dies die Untergliederung der gesamten Montageaufgabe in einzelne Funktionen. Zur Darstellung des Ablaufes eignet sich beispielsweise ein Vorranggraph.
- Prinzipielerarbeitung: Im Rahmen der Prinzipielerarbeitung werden Lösungsprinzipien zur Erfüllung der einzelnen Funktionen gesucht. Für die Bewertung der Prinzipien bietet sich ein Morphologischer Kasten an. Als Lösungsprinzip für die Sortierung von Elementen sei beispielsweise ein Sortiertopf genannt.
- Gestaltung: Nachdem das Lösungsprinzip für den jeweiligen Montage-ablaufschritt festgelegt wurde, erfolgt die Umsetzung in eine reale Lösung. Es wird ein maßstäblicher Entwurf erstellt. Der Rückgriff auf eine Bibliothek bereits realisierter bzw. standardisierter Lösungen reduziert den Bearbeitungsaufwand und minimiert das technische Risiko.
- Detaillierung: Die Detaillierung umfaßt die Vervollständigung und "Verfeinerung" der entwickelten Lösung. Es werden Fertigungsunterlagen, d. h. Einzelteil- und Montagezeichnungen sowie Stücklisten erstellt. Beispielsweise sind, bezogen auf die Montageanlage, die Bandstützen für die Aufstellung der Transportbänder zu berücksichtigen. Die Detaillierung der Angebotslösung sollte abgebrochen werden, wenn das angebotene Produkt hinreichend beschrieben und eine Basis für die nachfolgende Kalkulation gegeben ist.

Die Phasen Gestaltung und Detaillierung fallen oftmals zusammen. Abb. 4 auf der folgenden Seite verdeutlicht die Prozeßkette:

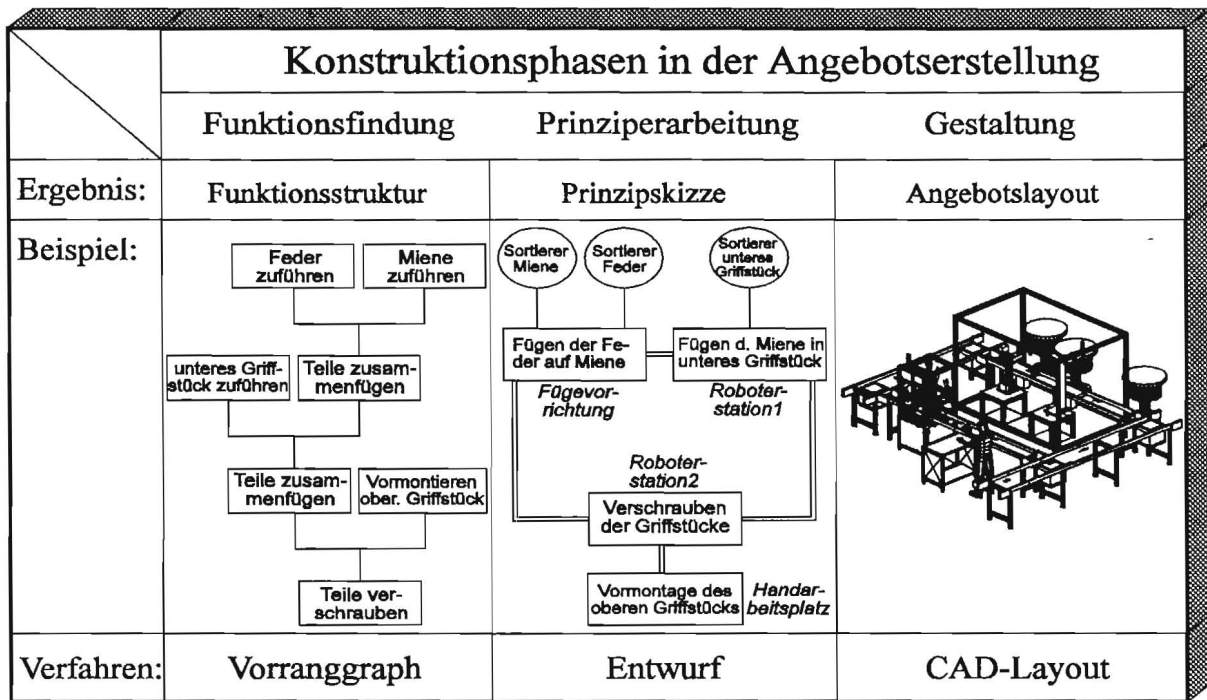


Abb. 4 Konstruktion im Rahmen der Angebotserstellung

Die konsequente Wiederverwendung vorhandener Lösungen durch die Konfiguration, d. h., die Anwendung von Anpassungs- und Variantenkonstruktionen anstelle von Neukonstruktionen, soll durch Methoden zum Auffinden ähnlicher Lösungen in bereits erstellten Angeboten bzw. Aufträgen erleichtert werden, da es in der Regel nicht erforderlich ist, völlig neue konstruktive Lösungen zu erarbeiten. Man kann zwischen der Wiederverwendung von Aufträgen und Angeboten wie folgt unterscheiden:

- Aufträge: Wird auf vorhandene Lösungen zurückgegriffen, ergibt sich außer dem ersparten Konstruktionsaufwand der Vorteil, daß sowohl technische Daten als auch zuverlässige Aussagen über Fertigungsmöglichkeiten und Herstellkosten vorliegen. Weiterhin liegen die Funktionsbeschreibungen, die ebenfalls im Rahmen der Angebotskonstruktion erstellt worden sind, vor.
- Angebote: Wird auf ein Angebot zurückgegriffen, liegen mehr oder weniger detaillierte Lösungen und dazugehörige Beschreibungen vor. Hauptinformationen sind in diesem Fall die Angabe der Arbeitsprinzipien, grobe geometrische Formen sowie eine Funktionsbeschreibung im Rahmen des Angebotsdokumentes.

Um auf die Informationen dieser bereits durchgeführten Angebote bzw. Aufträge gezielt zurückgreifen zu können, ist es notwendig, das Produktspektrum zu standardisieren. Im Anschluß daran kann mit Hilfe von Klassifizierungssystemen auf die erarbeiteten Lösungen zurückgegriffen werden. In der Angebotsphase muß auf ein standardisiertes Produktspektrum zurückgegriffen werden, um Angebote mit vertretbarem wirtschaftlichen und zeitlichen Aufwand erstellen zu können.

4. Standardisierung als Grundlage der Konfiguration

Somit ist die Standardisierung des Produktspektrums die Voraussetzung für die Wiederverwendung von technischen Lösungen. Unterbleibt dieses, dann ist es lediglich möglich über das Erfahrungswissen einzelner Mitarbeiter evtl. vorhandene Lösungen erneut zu verwenden. Dieses bedeutet beispielsweise, daß sich ein Projektteur an ein spezifisches Angebot, bei dem eine ähnliche Lösung verwendet wurde, "erinnern" muß, um diese verwenden zu können.

Die Möglichkeiten zur Standardisierung der Lösungselemente reichen von der Definition der Funktionsbeschreibung eines Lösungselementes über die Festlegung des physikalischen Wirkprinzips bis zur Standardisierung

der Lösungselemente über die Festlegung von Fertigungstechnologie, Werkstoff usw. Dieses kann beispielsweise für Einzelteile oder Baugruppen durchgeführt werden.

Die einzelnen standardisierten Elemente können dann verschiedene Informationen, wie beispielsweise technische Merkmale, Herstell- und Beschaffungskosten sowie Lieferanten, zugeordnet werden. Nach erfolgter Standardisierung des Produktspektrums, stellt der Zugriff auf die erarbeiteten Lösungen ein weiteres Kernproblem der technischen Lösungsfindung und damit der Konfiguration eines Produktes dar.

Zur Wiederverwendung und damit zur Nutzung von vorhandenen technischen Lösungen (Know-how) sind grundsätzlich Klassifizierungssysteme geeignet. Sie ermöglichen eine Strukturierung unterschiedlicher Produkte eines Betriebes nach bestimmten Kriterien. Demzufolge bieten sie auch die Möglichkeit, nachdem die Kriterien eines gesuchten Gegenstandes bestimmt wurden, nach ihnen zu suchen. Systeme zur Strukturierung des Produktspektrums werden wie folgt unterschieden:

- *Art der beschriebenen Bauteilmerkmale (fertigungs-, funktionsgerecht usw.)*

- *Art des Klassifizierungscodes (numerisch, sprechend):*

Es wird auch von Codiersystemen gesprochen. In diesem Fall besteht die Beschreibung der technischen Lösung und der Kundenprobleme aus Schlüsselnummern. Den jeweiligen Kundenproblemen werden eine Verschlüsselung (Sortierkriterium) und der Nummerncode der technischen Lösung zugeordnet. Dadurch ist es möglich für ein Problem mehrere Lösungen anzubieten sowie eine Lösung für mehrere Probleme zu nutzen. Diese Systeme eignen sich jedoch nur, solange eine überschaubare Anzahl von Parametern zur Darstellung der Probleme und Lösungen ausreichen.

- *Aufbau des Klassifizierungscodes:*

Bei komplexeren Produkten, wie sie häufig in der Investitionsgüterindustrie anzutreffen sind, besteht die Möglichkeit, Klassifizierungssysteme einzusetzen. Dabei werden die Merkmale eines Erzeugnisses numerisch verschlüsselt und ähnliche Erzeugnisse in einer Klasse zusammengefaßt. Der Aufbau dieses Nummernsystems kann

- hierarchisch: strenge Hierarchie als Bäume oder Netze,
- teilhierarchisch: die Objekte sind bedingt miteinander verknüpft, d. h. sie können nach Eigenschaften ausgewählt werden (z. B. Thesaurus),
- ahierarchisch: voneinander unabhängige Informationen (z. B. Schlagwortkataloge)

sein.

Ein Klassifizierungssystem kann erst dann sinnvoll genutzt werden, wenn ein größerer Vorrat an klassifizierten Lösungen aufgebaut wurde. Eine neue Lösung ist jeweils soweit zu entwickeln, daß sie in einer Klasse des System unterzubringen ist.

Um eine entsprechende Grobgruppierung von Objekten vorzunehmen bzw. Gruppen mit den gesuchten Merkmalen auszuwählen, bietet sich ein sog. Thesaurus an. Er ist ein Begriffssystem, in dem eine Schlagwortliste mit der Darstellung hierarchischer und anderer Beziehungen zwischen Begriffen und Synonymen usw. gekoppelt wird. Abhängig vom Aufbau kann ein Thesaurus ein bestimmtes Fachgebiet ganz oder teilweise systematisch und umfassend abdecken. Die Schlagworte eines Thesaurus, wie beispielsweise Baugruppe oder Roboter, können sich in

- Deskriptoren, die der Indexierung (beispielsweise Strukturierung eines Produktspektrums in Anlagen, Module, Baugruppen usw.) dienen und
- Nicht-Deskriptoren, die ergänzt im Thesaurus enthalten sind (Synonyme)

aufteilen.

Zwischen den einzelnen Schlagworten eines Thesaurus existieren Abhängigkeiten in Form von Relationen, mit deren Hilfe der Thesaurus die Suchfunktionen eines Informationssystems unterstützt. Die Eingrenzung eines gesuchten Objektes erfolgt dann über die gewählten, charakteristischen Merkmale.

Die geeigneten Hilfsmittel und Methoden zur Weiterverarbeitung der gefundenen Lösungen sind dem jeweiligen Anwender ebenfalls zur Verfügung zu stellen, da er in der Regel ähnliche Lösungen oder Lösungselemente vorfindet, die überarbeitet und daraufhin ebenfalls dokumentiert werden müssen.

5. Umsetzung der Prinzipien im Rahmen des Systems zur integrierten Angebotserstellung

Die automatisierte Konfiguration der technischen Lösung ist im System zur integrierten Angebotserstellung (SINA) über ein Tool zur Auffindung standardisierter Lösungskomponenten realisiert. Dieses Modul erlaubt sowohl eine manuelle als auch eine automatisierte Zusammenstellung der entsprechenden Bauteile, die zur Realisierung einer bestimmten Montageaufgabe notwendig sind. Über eine Maske können vordefinierte Baugruppen bzw. Einzelteile aus der Datenbank abgerufen werden und mit den zugehörigen Daten, d. h. beispielsweise Kostensätze und Materialkosten, in ein unternehmensspezifisches Kalkulationsschema übertragen werden.

Weiterhin werden neben diesen kalkulatorischen Daten auch Beschreibungen und technische Zeichnungen zu einzelnen Baugruppen oder Einzelteilen in der Datenbank gehalten, die dem Anwender beispielsweise während der Dokumenterstellung zur Verfügung stehen.

Am Beispiel der bereits in Abb. 4 beschriebenen Kugelschreibermontage wird die Konfiguration und Kalkulation dargestellt: Nachdem die entsprechenden Projektdaten angelegt sind, hat der Projektteur die Möglichkeit mit Hilfe des Thesaurus, der lediglich dem Datenzugriff und nicht der Datenhaltung dient, auf bereits standardisierte Lösungskomponenten zurückzugreifen. Über diesen klarsprachlichen Thesaurus wird ein entsprechender Klassifizierungsschlüssel erzeugt. Es werden je nach Tiefe der Schlüsselpezifizierung Einträge allgemeiner oder spezieller Art gesucht. In Abb. 5 ist der grundsätzliche Aufbau für einen Hersteller von Montageanlagen dargestellt. Es besteht beispielsweise die Möglichkeit nach einer Anlage, einem Modul, einer Baugruppe oder einem Einzelteil zu suchen. In dem dargestellten Beispiel wird nach einem standardisierten Handarbeitsplatz zum Einlegen der entsprechenden Bauteile in den Werkstückträger gesucht. In Abb. 5 kann man erkennen, daß der Anwender zunächst nach einem Modul sucht. Der weitere Verlauf ist in Abb. 6 zu erkennen, d. h., es wird das Modul Handarbeitsplatz nach dem BMW-Standard (stehend) gesucht.

The screenshot shows a window titled "Thesaurus". On the left, there is a vertical list of search categories: "Anlage", "Modul" (highlighted with a dark background), "Baugruppe", and "Einzelteil". To the right of this list, there is a list of options: "Werkstücktransport", "Handarbeitsplatz" (highlighted with a dark background), "Automatikstation", and "Zuführeinrichtung". Below these lists, there is a small empty rectangular box for input.

Abb. 5 Grundsätzlicher Aufbau des Thesaurus

This screenshot shows the "Thesaurus" window with more specific selections. The "Modul" category is still selected on the left. In the options list, "Handarbeitsplatz" is selected. To the right of "Handarbeitsplatz", there is a sub-list: "BMW-Standard" (highlighted with a dark background), "Bosch-Standard", "Taktunabhängig", and "Taktabhängig". Further to the right, there is another list with "Sitzend" and "Stehend" (highlighted with a dark background). Below these lists, there are three small empty rectangular boxes for input.

Abb. 6 Auswahl des Modul Handarbeitsplatz

Der Schlagwortkatalog dient somit nur der Klassifizierungsschlüsselgenerierung und ist als separate Datenbank eingerichtet. Die eigentlichen Kalkulationsdaten werden in einer zentralen Datenbank gehalten

Nachdem ein entsprechender Handarbeitsplatz in der Teilebibliothek mit Hilfe des Thesaurus gefunden wurde, besteht die Möglichkeit die relevanten Kalkulationsdaten der Datenbank einzusehen. In Abb. 7 ist das dazugehörige Datenblatt dargestellt. Gut zu erkennen sind die Kalkulationswerte für den Konstruktions-, Fertigungs-, Kaufteileaufwand sowie Daten für Steuerungsaufwände. So ist beispielsweise ersichtlich, daß die mechanische Konstruktion in 10 Stunden abgeschlossen ist. Dieser relativ geringe Wert ist damit zu erklären, daß lediglich eine Anpassungskonstruktion vorzunehmen ist, da der Handarbeitsplatz bereits in der Teilebibliothek abgelegt ist und damit die entsprechenden Unterlagen vorhanden sind.

Identnummer: E			
Bezeichnung: Handarbeitsplatz		Hersteller: BWM	
Techn Daten: h=1000, b=800, t=800		Best.Nr.:	
Konstruktion	Fertigung	Kaufteile	Steuerung
ma-k: 10	tef: 17	Fremd-KT: 2750	steuer-a: 0
el-k: 5	ma-w: 12	Mech-KT: 0	steuer-e: 0
	el-w: 0	Elek-KT: 0	Ventile: 0
	ING: 0	Pn/Hy: 0	Motoren: 0
	Mo: 0	MST: 0	

Abb. 7 Kalkulationsdaten der Teilebibliothek

Weiterhin besteht die Möglichkeit im Rahmen dieses Datenblattes Zeichnungs- und Textinformationen zu einem entsprechenden Bauteil zu speichern. In Abb. 8 ist dieses am Beispiel eines Scara-Roboters vom Typ Bosch SR 80 dargestellt. Die Zeichnungsinformationen können direkt aus diesem Formular mit Hilfe der OLE-Verknüpfung bearbeitet werden und stehen, ebenso wie die textuellen Beschreibungen bei der Erstellung des Angebotsdokumentes zur Verfügung.

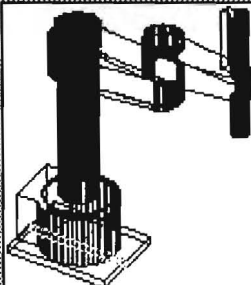
Identnummer: E1			
Bezeichnung: Scara Bosch SR 80 mit IQ 140		Hersteller: BOSCH	
Techn Daten: Armlänge 800 mm, z-Hub 200 mm		Best.Nr.: 3842999661	
Konstruktion	Fertigung	Kaufteile	Steuerung
ma-k: 0	tef: 0	Fremd-KT: 0	steuer-a: 0
el-k: 0	ma-w: 0	Mech-KT: 49900	steuer-e: 0
	el-w: 0	Elek-KT: 0	Ventile: 0
	ING: 0	Pn/Hy: 0	Motoren: 0
	Mo: 0	MST: 0	
Beschreibung: (Angebotstext)	Scara-Roboter, 4 Achsen Armlänge 800 mm, z-Hub 200 mm Stativlänge 500 mm, 6 m Verb.-Kabel Handhabungsgewicht 2 -5 kg		
von: MS			
aktualisiert: 94,5			

Abb. 8 Verwaltung von Zeichnungs- und Textinformationen

Nachdem ein Bauteil mit Hilfe des Thesaurus ausgewählt wurde, wird es in ein entsprechendes Kalkulationsschema eingefügt (vgl. Abb. 9). Die dazugehörige Auflistung der ausgewählten Einzelteile ist erkennbar.

Angebots-Kalkulation

Angebots-Nr.: 33052-3

Lfd. Nr.	Stck.	Bezeichnung	Best. Nr.	Konstr. (h)		Fertigung (h)			IBS (h)		Fremd
				ma-k	el-k	tef	ma-w	el-w	Ing	Mo	
4	1	Automatikstation		10	0	20	15	0		0	
5	1	Transfersystem		15	15	15	20	10		0	
13	2	BWM-AUTO-1		20			80	152		0	
16	50	Werkstückträger		115		500	200			0	
17	1	Handarbeitsplatz		10	5	17	12	0	0	0	2750
18	1	Scara Bosch SR 80 mit IQ	3842999661	0	0	0	0	0	0	0	0
Summe				170	20	552	327	162	0	0	2750

Abb. 9 Auflistung der ausgewählten Einzelteile

Bei der Angebotskalkulation sollte generell ein möglichst hoher Genauigkeitsgrad erreicht werden, um so eine Fehlkalkulation zu vermeiden. Diese kann zu einem Verlustgeschäft, bei zu gering angesetzten Kosten oder zu einer Nichterteilung des Auftrages, bei zu hohen Kosten führen. Auf der anderen Seite muß der Tatsache Rechnung getragen werden, daß nicht jedes Angebot zu einem Auftrag wird. Die Kosten für die Erstellung eines Angebotes müssen daher in der Regel auf mehrere Aufträge verteilt werden und sollten so gering wie möglich ausfallen.

Eine präzise Preisfindung für eine anzubietende Montageanlage ist nur auf Basis einer Einzelteilkalkulation möglich, da alle anderen Kalkulationsmethoden auf empirischen Verfahren basieren und somit zu einer bestimmten Ungenauigkeit führen. Die konsequente Anwendung der Einzelteilkalkulation setzt das Vorgeben von (Standard-) Lösungen voraus, die, wie hier exemplarisch dargestellt ist, mit einem Thesaurus verwaltet werden können.

Im Anschluß an die Preisfindung folgt die Preisbildung, bei der es darum geht, den entgültigen Angebotspreis unter Abwägung von Kundenverhältnis, Auftragschancen, momentaner Auslastung etc. festzulegen. Sie stellt somit stets eine unternehmenspolitische Entscheidung dar, für die als wichtigste Voraussetzung die genaue Vorkalkulation des möglichen Auftrags vorliegen muß. Abb. 10 stellt einen Ausschnitt aus einer Preisbildung dar, in der beispielsweise die Summen der Einzelteile für die einzelnen Kostenstellen (z. B. Konstruktion Mechanik) mit dem entsprechenden Stundensatz multipliziert werden.

Bremer Werkzeug- und Maschinenbau 		Angebotskalkulation		Datum: 10.11.1994	
Auftrag-Nr.:				Angebots-Nr.: 33052-3	
Kunde:		BPM		Antragseingang: 07.10.1993	
Anspruchspartner:		Herr Cohen		Angebotsabg.: 12.10.1993	
				Projektant: BA	
Angebotsumfang			Termine	Soll	Ist
Konstruktion:	☐		Kundenw. Abnahme		
Anfertigung:	☐	MONTAGEANLAGE-G40	Kundenw. Lieferung		
Änderung:	☐	ANGEBOT-NR.3 3502-3	Konstruktion		
Erweiterung:	☐		Planung		
Teilefertigung:	☐		Erprobung		
			Abnahme		
			Lieferung		
Konstruktionskosten					
	Vorg.-Std.	Davon BWM Erpr.	Std.-Satz	Summe	
Konstruktion Mechanik	170		95,00	16.150,00	Gesamt Konstruktions- kosten (DM)
Konstruktion Elektrik	20		95,00	1.900,00	
Projektleiter	20		95,00	4.750,00	
Zwischensumme	240			22.800,00	
Verwaltung und Vertrieb (10 % von Konstruktionskosten)				2.280,00	25.080,00
Materialkosten und Fremdleistung					
Lagermaterial	Mechanik			5.000,00	

Abb. 10 Preisbildung

Nachdem die Gesamtstundenzahlen ermittelt wurden, können sie mit Hilfe eines Terminierungsmoduls zur Bestimmung eines realistischen Liefertermins verwendet werden. Ein solches Modul sollte die Grobterminierung in der Angebotsphase unterstützen und die entsprechende Terminstrecke beispielsweise in Form eines Gantt-Diagrammes darstellen (vgl. Abb. 11). Die Information, die hieraus hervorgehen sollte, ist eine eindeutige Darstellung des kritischen Pfades, um über die Vergabe von Prioritäten Einfluß auf die einzelnen Vorgänge nehmen zu können.

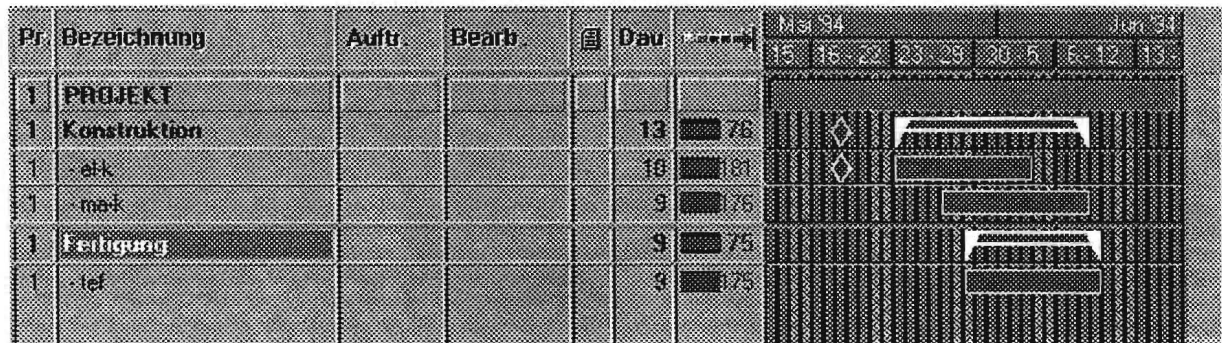


Abb. 11 Das Terminierungsmodul

Durch das Terminierungsmodul wird es möglich, im Rahmen der Angebotsbearbeitung eine grobe Kapazitätsplanung unter Berücksichtigung bereits erhaltener Aufträge und weiterer Angebote durchzuführen. Für das konkrete Angebot stellen die bisher ermittelten Daten die Grundlage für die Dokumenterstellung dar, die den Abschluß der Angebotserstellung bildet.

6. Zusammenfassung und Ausblick

Die Teilkonzepte bzw. funktionalen Anforderungen an ein integriertes System zur Angebotserstellung wurden durch eine modular aufgebaute Software erfüllt, die zum einen benutzerfreundlich gestaltet ist und zum anderen die in vielen Unternehmen verfügbaren PC-Systeme unterstützt. Die Entscheidung fiel aus diesem Grund auf die Oberfläche „Windows“, die weite Verbreitung besitzt. Um den Entwicklungsaufwand gering zu halten, wurde größtenteils Standardsoftware verwendet. Das Kernstück des Systems stellt eine relationale Datenbank „ACCESS“ dar.

Das System bietet einen Übersichtsmodus, der über den aktuellen Bestand an Anfragen, Daten (Kunde, Produkt, Auftragswert etc.) sowie über die Bedeutung der Anfrage für das Unternehmen als Ergebnis der Anfragebewertung informiert. Desweiteren bietet ein integriertes Berichtswesen die Möglichkeit diverse angebots- und auftragsspezifische Statistiken und Übersichten in graphisch aufbereiteter Form darzustellen. Die allgemeinen Angebotsdaten, wie beispielsweise Kosten, werden in einer Projektverwaltung zur Verfügung gestellt.

Zur Konfiguration der technischen Lösung stellt das System einen Theasaurus zur Verfügung, in dem standardisierte Lösungskomponenten abgelegt sind. Während des Konfigurationsvorgangs führt das System automatisch die entsprechende Kalkulation durch. Ebenfalls werden die erforderlichen Kapazitäten errechnet und der frühestmögliche Liefertermin ermittelt. Die wesentlichen Daten einschließlich Beschreibungen der Lösungskomponenten werden an das Dokumenterstellungsmodule übergeben, welches dem Bearbeiter die Angebotsschreibung erleichtert.

SINA stellt mit den beschriebenen Funktionalitäten einen praxisorientierten Ansatz zur integrierten Angebotserstellung dar. „Integriert“ bedeutet hier konkret, daß der Anwender nicht zwischen verschiedenen Systemen wechseln muß, sondern mit einem Gesamtsystem arbeitet, in dem die Daten zwischen den einzelnen integrierten Applikationen ohne Zutun des Anwenders ausgetauscht werden.

Für die Zukunft ist beabsichtigt, das System in den Vertriebsbereichen weiterer Unternehmen im Bereich der Investitionsgüterindustrie einzuführen. Dabei ist auch eine Erweiterung der Systemfunktionalitäten angedacht. Beispielsweise steht die Anbindung an verschiedene CAD-, PPS- und Simulationssysteme [2] an.

Literaturverzeichnis

- [1] Grabowski, H. u. a.: „Angebotserstellung in der Investitionsgüterindustrie“, VDI-Verlag, Düsseldorf 1983
- [2] Krömker, M.; Meyer, R. und Thoben, K.-D.: „Generierung von Simulationsmodellen aus Baugruppen-spezifischen Teilmodellen zur Unterstützung des Entwurfs kundenspezifischer Investitionsgüter“, Konferenzband zum 8. ASIM Symposium der GI, Stuttgart, Oktober 1994
- [3] Krömker, M.; Thoben, K.-D. und Wickner, A.: „Kooperative Produktspezifikation in der Angebots-erstellung“, Beitrag zur SYSTEC 94, München, Oktober 1994, abgedruckt im VDI-Bericht Nr. 1148, VDI-Verlag, Düsseldorf 1994

Regelbasierte Angebotskalkulation auf Prozeßkostenbasis

Prof. Dr. rer.pol. Joachim Fischer
Dipl.-Wirt.Ing Klaus Hauschulte
Dipl.-Wirt.Ing. Karsten Jakuschona
Prof. Dr.-Ing. Rainer Koch

Schwerpunkt Wirtschaftsinformatik 1 & Operations Research
Fachgebiet Computeranwendung und Integration in Konstruktion und Planung
Universität-GH Paderborn, 33095 Paderborn
E-Mail: jjaku1@cik.uni-paderborn.de

1 Problematik der Angebotskalkulation

Die Aufgabe der Angebotskalkulation ist es, in einer vorausschauenden Rechnung die zukünftigen Preise für anzubietende Leistungen (Produkte, Projekte) so festzulegen, daß einerseits der Auftrag nicht durch überhöhte Preisforderungen an die Konkurrenz verlorenggeht und andererseits im Falle einer späteren Auftragsabwicklung die anfallenden Kosten gedeckt werden und ein Gewinn erzielt wird.

Nach - teilweise bereits während - der technischen Lösungsfindung ist es erforderlich, den Liefertermin festzusetzen und den Preis für diese technische Lösung zu kalkulieren.

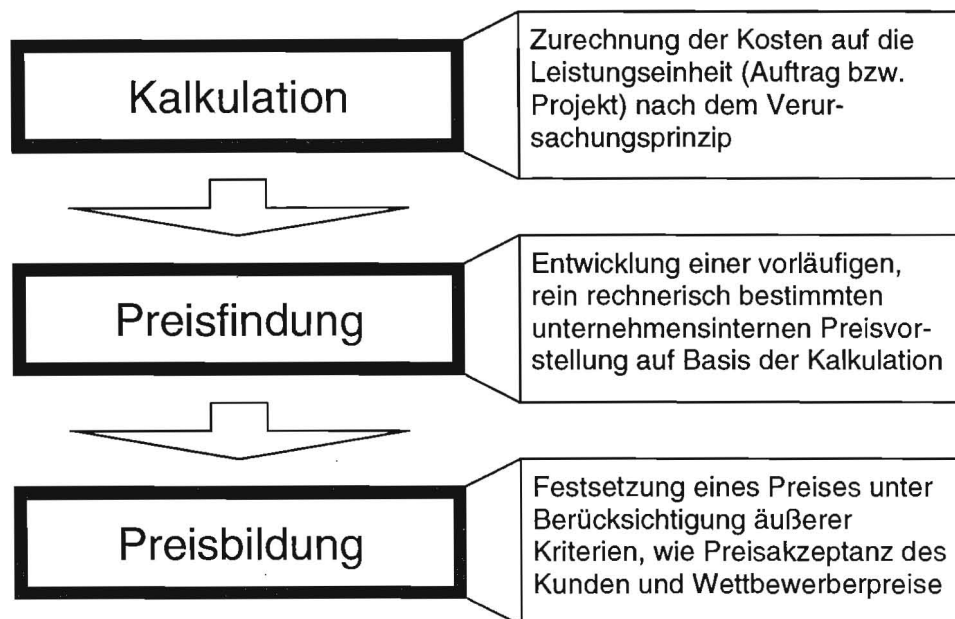


Abb. 1: Aufgaben der Angebotsbearbeitung

Die besondere Problematik liegt dabei häufig in dem Zeitdruck, unter dem das Angebot abgegeben werden muß, obwohl die Lösungsfindung noch nicht vollständig abgeschlossen ist und nicht vorhersehbare Mehraufwendungen bei der Auftragsabwicklung zu besonderen Risiken führt.

Die Angebotskalkulation umfaßt neben der Kalkulation, die auch als Kostenträgerstückrechnung bezeichnet wird, die Aufgaben der Preisfindung und Preisbildung. Bei der **Kalkulation** erfolgt die Zurechnung der Kosten auf die jeweils betrachtete Leistungseinheit (Auftrag, Projekt, Produkt oder Dienstleistung) nach dem Verursachungsprinzip. Bei der Preisfindung erfolgt zuerst die Entwicklung eines rein rechnerisch bestimmten, unternehmensinternen Preises auf Basis der Selbstkosten. Dabei werden die Rendite-Anforderungen des Unternehmens in bezug auf die Leistungseinheit berücksichtigt. Oftmals wird es aber der Fall sein, daß dieser erwartete Preis über dem am Markt erzielbarem liegt. In einem dritten Schritt erfolgt daraufhin die Preisbildung, die den Preis unter Berücksichtigung äußerer Kriterien, wie Preisakzeptanz des Kunden, Wettbewerberpreise, Marktanteile, Marktsituation etc. festlegt. Die Preisbildung kann aber auch Rückwirkungen auf die technische Lösungsfindung haben, um die Kundenanforderungen und Kundennutzen durch ein anderes Lösungskonzept effektiver und kostengünstiger zu realisieren.

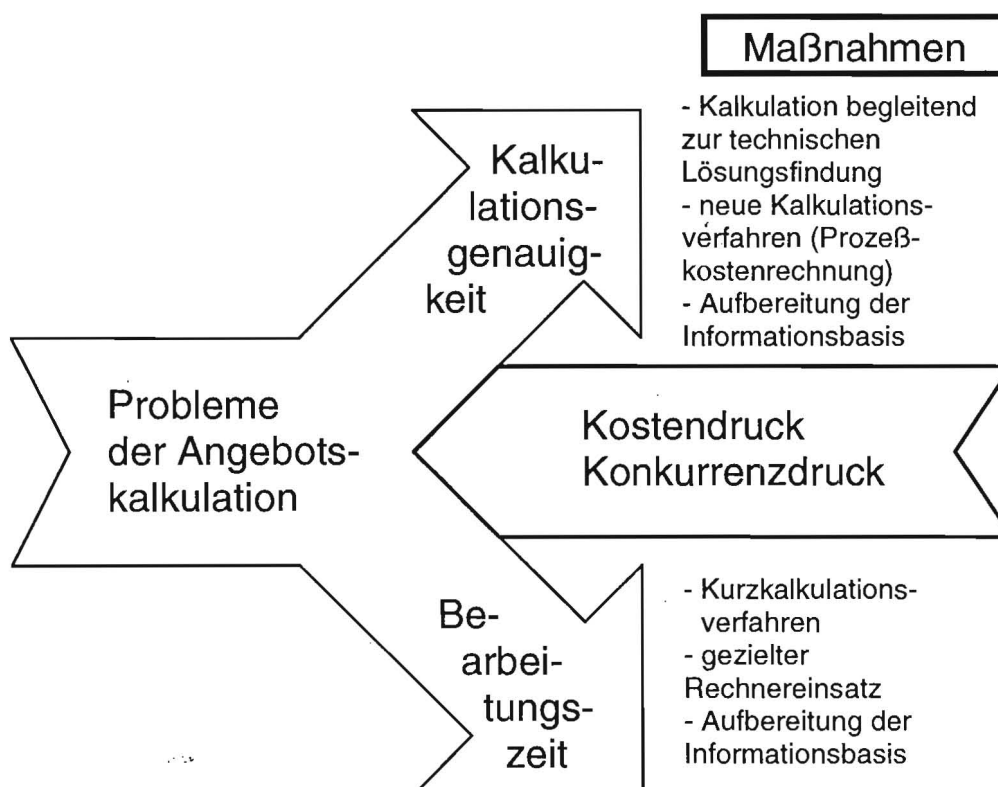


Abb. 2: Probleme der Angebotskalkulation

Durch die zunehmende Marktdynamik und den damit verbundenen Konkurrenz- und Kostendruck ist es erforderlich, die gesamte Angebotsbearbeitung - und somit auch die Angebotskalkulation - zu beschleunigen. Die Bearbeitungszeit läßt sich insbesondere durch den Einsatz der entwicklungsbegleitenden Kalkulation und durch EDV-Einsatz verbessern. Grundlage dazu bildet aber die vorausschauende Aufbereitung von Informationen, um in der Angebotsphase einen schnellen und gezielten Zugriff auf diese Dokumente zu ermöglichen. Weiterhin können Fehlkalkulationen und damit verbundene Verluste immer seltener durch hohe Renditen anderer Projekte kompensiert werden. Die Forderung nach einer höheren Kalkulationsgenauigkeit steht im engen Zusammenhang mit der technischen Lösungsfindung. Da aber die Umwandlungsrate von Angeboten zu Aufträgen gering ist und eine immer schnellere Angebotsbearbeitung gefordert wird, wird eine Detaillierung der technischen Lösung nicht in dem Maße vorgenommen, wie dies aus kalkulatorischen Gründen erforderlich wäre. Lösung bietet

hier eine Kalkulation begleitend zur technischen Lösungsfindung. Grundlage ist hierbei wiederum die Aufbereitung der Informationsbasis sowie die Sicherstellung der verursachungsgerechten Kostenzuordnung, die bei traditionellen Kalkulationssystemen häufig nicht mehr gegeben ist.

Die heute vorliegende flexibel automatisierte Fertigung zeichnet sich aber durch hohe Gemeinkostenanteile (bis zu 75 % der Gesamtkosten) aus. Teilweise wird heute in Unternehmen mit Zuschlagssätzen von bis zu 1000 % zu den Einzelkosten gerechnet. Eine Anwendung der Zuschlagskalkulation ist somit nicht mehr zeitgemäß.

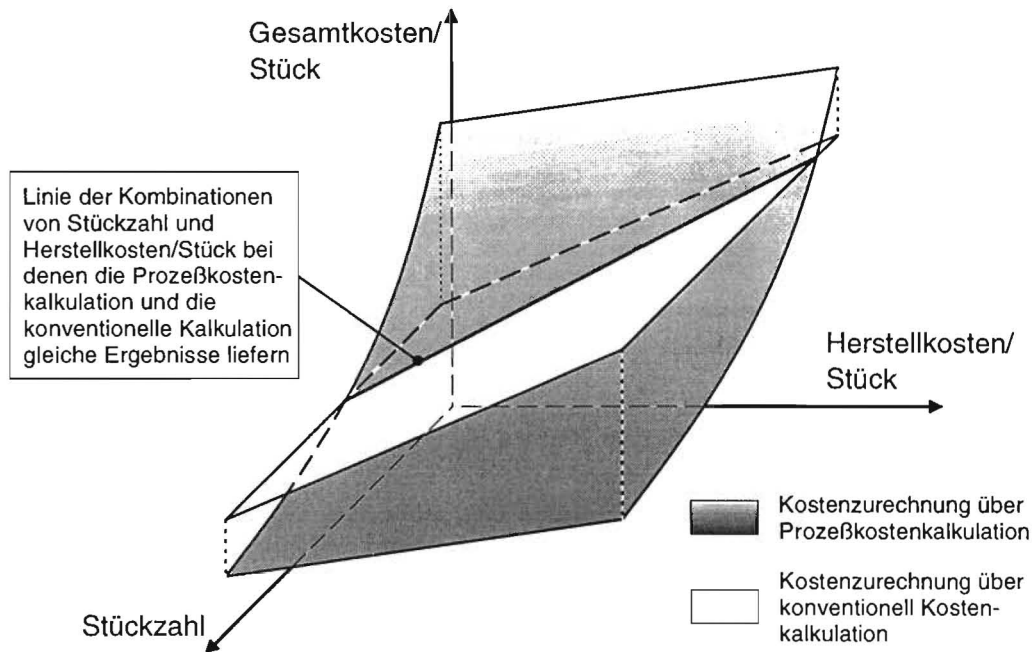


Abb. 3: *Prozeßkostenrechnung zur Verbesserung der verursachungsgerechten Zuordnung der Produktkosten*

Dem Fehler der Zuschlagskalkulation liegen zwei wesentliche Ursachenquellen zugrunde. Die **Allokationsfehler** entstehen durch die ausschließliche Nutzung volumenorientierter Zuschlagsbasen (z. B. Material-, Lohneinzelkosten oder Maschinenstunden), um die Gemeinkosten in Abhängigkeit von der Höhe der jeweiligen Zuschlagsbasis auf die Produkte zu verrechnen. Der alleinige Gebrauch volumenorientierter Bezugsgrößen verzerrt Produktkosten, wenn einige der für das Produkt notwendigen Aktivitäten nicht mit dem Produktionsvolumen und der Produkt- und Prozeßkomplexität korrelieren. Der **Degressionsfehler** kommt durch die Verrechnung eines konstanten Gemeinkostensatzes pro Stück zustande. Die Prozeßkosten pro Stück für die interne Abwicklung von Materialbestellungen, Fertigungslosen, Kundenaufträgen etc. verringern sich jedoch mit steigenden Stückzahlen.

2 Prozeßorientierter Ansatz der Kalkulation

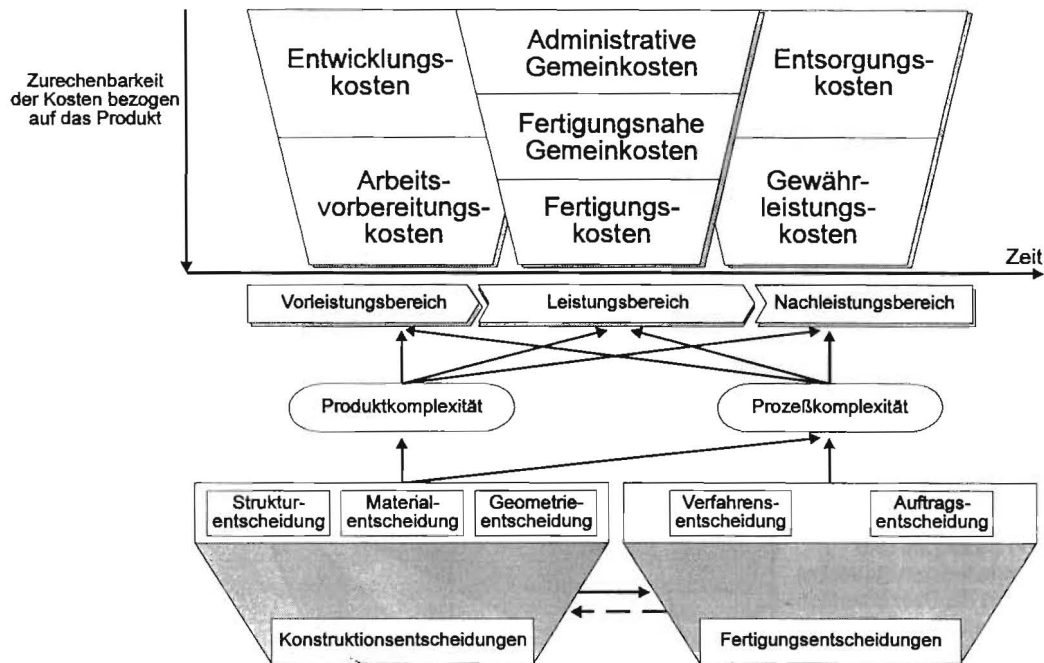


Abb. 4: Prozeßkostenprognose aller Phasen des Produktlebenszyklus

Für ein Kosteninformationssystem muß es daher Ziel sein, alle während der Phasen des Produktlebenszyklus (von der Entwicklung bis zur Entsorgung) anfallenden Kosten für das Produkt zu prognostizieren.

Vor der Fertigung liegen Vorleistungsvorgänge, die den Bereichen Entwicklung und Arbeitsvorbereitung zugeordnet werden. Nach Fertigung und Auslieferung entstehen Nachleistungsvorgänge, die z. B. aufgrund der gesetzlichen Anforderungen auf Rücknahme der Produkte oder auf Gewährleistung schon in der Konstruktion berücksichtigt werden müssen.

Die Konstruktions- und Fertigungsentscheidungen haben erhebliche Auswirkungen auf die Produkt- und Prozeßkomplexität und induzieren somit in den verschiedenen Phasen des Produktlebenszyklus Kosten. Beispielsweise wirkt die Anzahl unterschiedlicher Einzelteile oder Materialien auf die erforderlichen Vorgänge in der Arbeitsvorbereitung und auf die notwendigen Entsorgungsvorgänge.

Neben der reinen Kostenprognose ist es daher notwendig, den Zusammenhang zwischen den Entscheidungen und den Kosten transparent zu machen.

Das Kosteninformationssystem generiert für alle Phasen des Produktlebenszyklus Prozeßstrukturen, die eine Kostenzuordnung insbesondere auch für die Vor- und Nachleistungsbereiche ermöglichen. Ausgehend von den Konstruktions- und Fertigungsentscheidungen erfolgt die Bildung von Vorgangsobjekten. In drei Schritten werden für die Vorgangsobjekte die Kosten berechnet:

Zunächst werden den Vorgangsobjekten die erforderlichen Vorgangsketten aus den Vorleistungs-, Leistungs- und Nachleistungsbereichen zugeordnet. Ein Vorgang beschreibt dabei die erforderlichen Bearbeitungsaktivitäten eines Objektes, läßt aber deren Menge und Wert zu-

nächst offen. Für die Vorgänge werden durch ein regelbasiertes System die geeigneten Ressourcen und deren erforderlichen Mengen oder Zeiten bestimmt und damit die Verfahren spezifiziert. Sind für die Vorgangsobjekte die Verfahren bestimmt, werden nach Zuordnung des Mengen- und Wertgerüsts die Prozesse abgeleitet.

Innerhalb des Lebenszyklus eines Produktes werden Vorgänge unterschiedlich oft durchgeführt. So wird z. B. jedes Produkt verpackt, aber nur einmal konstruiert. Die Häufigkeit ist somit von der Art des Vorgangs und der herzustellenden Stückzahl abhängig. Weiteren Einfluß nehmen Entscheidungen über den Produktionsablauf. So treten Vorgänge wie "Maschine rüsten" in Abhängigkeit von der Losgröße auf. Die Fertigungsentscheidung bedingt somit die Verteilung der Kosten auf das Produkt bzw. die Konstruktionsobjekte. Aufgrund derer lassen sich Fertigungsobjekte (Stückzahl, Los, Gruppe identischer Objekte und Objekt) generieren, die eine Ordnung der Vorgänge nach ihrer Häufigkeit erlauben. Vorteil dieser Vorgehensweise ist die Erhöhung der Kostentransparenz, die sich später bei der Kostenverteilung und -darstellung zeigt.

3 Umsetzung des Regelwerkes

Nach Spezifizierung der Vorgänge erfolgt eine detaillierte Verfahrensplanung. Unter einem Verfahren versteht man einen Vorgang, dem Fertigungsprinzip und Fertigungsressourcen zugeordnet sind. Daraus läßt sich dann ein Mengengerüst ableiten, das die notwendigen Ausführungszeiten der Verfahren festlegt. Zum Beispiel kommen beim Vorgang "Lochen" die Prinzipien "Stanzen, Nibbeln und Laserschneiden" in Betracht. Wird das Prinzip "Stanzen" gewählt, so beschränkt sich die Ressourcenmenge auf Stanzmaschinen, von denen eine ausgewählt wird. Abschließend wird die Zeit für den Vorgang "Lochen" bzw. das Verfahren "Loch auf ausgewählter Maschine stanzen" ermittelt. Neben Maschinendaten beinhalten die Ressourcendaten auch Informationen über Werkzeug-, Material- und Personalalternativen.

Auf die Verfahrensbildung wirken bereits Kostenaspekte, wenn mehrere Alternativen von Prinzipien und Ressourcen möglich sind. Die Entscheidung erfolgt dann entsprechend der kostenoptimalen Ressource.

Nachstehende Abbildung zeigt am Beispiel des Vorgangs „Transformieren“ beispielhaft den Regelzusammenhang der Ressourcenzuordnung im Rahmen der Verfahrensbildung.

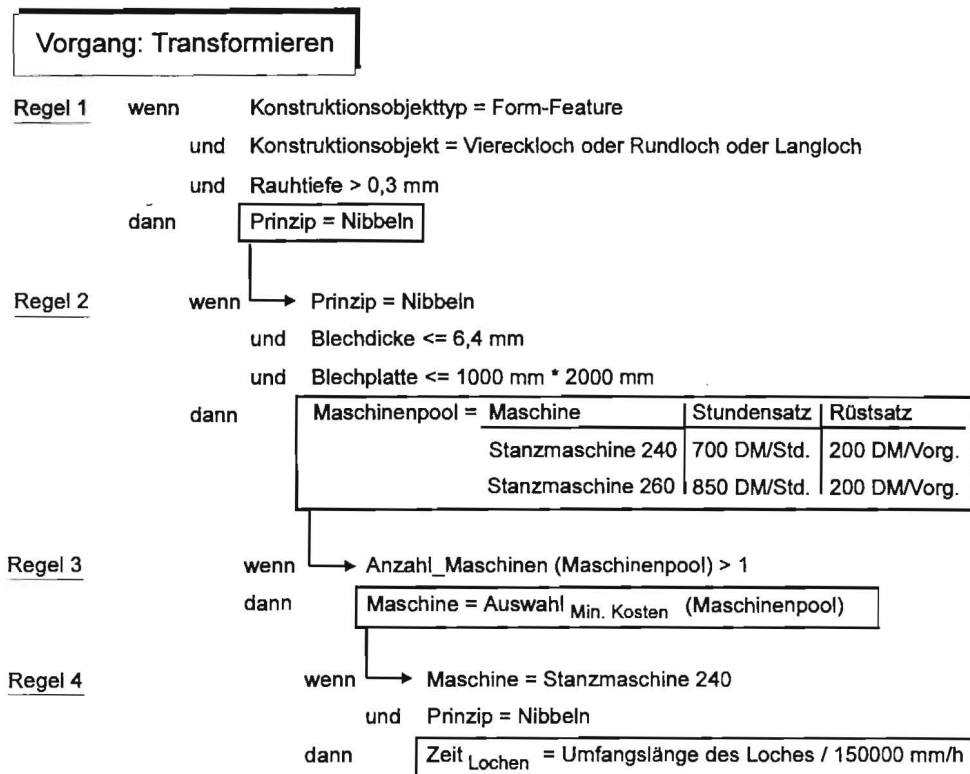


Abb.5: Regelverkettung

Auf Basis der ermittelten Verfahren wird der Ressourcenverzehr eines jeden Prozesses mit Hilfe eines Kostentreibers auf die Prozesse verrechnet. Die Kostensätze, wie z. B. Maschinenstundensatz oder Stundensatz für Personal, sind bei den jeweiligen Ressourcen hinterlegt. Bei der Kostenanalyse ist im Kostenberechnungsmodul die Auswertung in Objektsicht (z. B. alle Kosten für ein Einzelteil und die zugehörigen Form-Features) und Prozeßsicht (Kosten nach Leistungsbereichen aufgeteilt) möglich.

Die automatische Ableitung von konkreten betrieblichen Ressourcen aus allgemeinen Vorgängen im Rahmen der Verfahrensplanung ist, wie zuvor erläutert, mit Problemen behaftet. Zur Lösung dieser Schwierigkeiten werden im folgenden die Möglichkeiten einer DV-gestützten Regelverarbeitung vorgestellt. Die zur Umsetzung eines Regelwerks verfügbaren Werkzeuge lassen sich in drei Kategorien einteilen.

Konventionelle Programme

Konventionelle Programme werden eigens für einen speziellen Anwendungsfall erstellt. Diese sind im allgemeinen in einer höheren Programmiersprache (z. B. FORTRAN, COBOL, C) geschrieben und erfordern bei der Erstellung den Einsatz eines Entwicklungssystems. Dabei wird vorausgesetzt, daß die mit der Implementierung der Regeln betraute Person über ausreichende Programmierkenntnisse verfügt. Da dieses oftmals nicht der Fall ist, werden meist EDV-Spezialisten mit der Regelumsetzung beauftragt. Bei der Kommunikation zwischen Programmierer und Auftraggeber können wichtige Informationen verloren gehen. Obwohl konventionelle Programme innerhalb eines bestimmten Rahmens sehr flexibel sind, lassen sie

sich in der Regel nur mit einem hohen Aufwand auf andere betriebliche Anwendungssituationen übertragen.

Expertensysteme

Der Einsatz von Expertensystemen eignet sich besonders auf Gebieten, in denen Planungswissen nur schwer strukturiert werden kann, bzw. nicht immer vollständig vorhanden ist. Den Vorteilen der Verarbeitung unvollständiger Strukturen stehen auch Nachteile gegenüber. Während bei konventionellen Programmen der Programmablauf vorgegeben ist, wird bei einem Expertensystem mit Hilfe des sogenannten Inferenzmechanismus nach der anzuwendenden Regel gesucht. Dadurch entstehen gegenüber anderen Systemen höhere Laufzeiten.

Entscheidungstabellen

Systeme auf Basis von Entscheidungstabellen bieten dem Anwender ein Werkzeug, mit dem er seine spezifische Planungslogik ohne spezielle Programmierkenntnisse EDV-gerecht und übersichtlich in Form von Tabellen erfassen kann. Eines der Haupteinsatzgebiete der Entscheidungstabellentechnik liegt im Bereich der Arbeitsplanung. Der grundsätzliche Aufbau einer Entscheidungstabelle besteht aus Bedingungen und Maßnahmen, die über sogenannte Anzeiger (Regeln) miteinander verknüpft werden. Entscheidungssituationen können flexibel beschrieben und an komplexe Sachverhalte sowie betriebsspezifische Gegebenheiten angepaßt werden. Ein weiterer Vorteil ist die leichte Erlernbarkeit des Umgangs mit Entscheidungstabellen. Aufgrund ihres stark strukturierten Aufbaus sind Entscheidungstabellen in der Lage, Ablaufstrukturen übersichtlich abzubilden. Damit wird jedoch nicht die Strukturierung des eigentlichen Problems erleichtert. Diese Aufgabe verbleibt weiterhin im Bereich des Anwenders. Lediglich die Implementierung der Regeln wird beschleunigt.

Eine generelle Aussage für oder gegen eine der geschilderten Verarbeitungsformen ist aufgrund der vielfältigen Unterschiede nicht möglich. Auch können an dieser Stelle nicht alle Argumente zur Beurteilung der einzelnen Möglichkeiten aufgezeigt werden. Eine Bewertung ist zudem von der jeweiligen Einsatzsituation abhängig. Nachstehende Tabelle faßt die wesentlichen Vor- und Nachteile zusammen.

	Vorteile	Nachteile
Konventionelle Programme	☐ Regeln können sehr flexibel abgebildet werden ☐ Relativ geringe Laufzeiten gegenüber Expertensystemen und Entscheidungstabellen	☐ EDV-Spezialist (Programmierer) wird benötigt ☐ Sind schwer übertragbar auf andere Situationen ☐ Hoher Aufwand bei der Pflege von Regeln
Expertensysteme	☐ Verarbeitung von nur schwer strukturierbarem oder unvollständigem Planungswissen	☐ EDV-Spezialist (Wissensingenieur) wird benötigt ☐ Sehr hohe Laufzeiten
Entscheidungstabellen	☐ Hohe Übersichtlichkeit ☐ Leichte und flexible Wartbarkeit	☐ Strukturierung des Problems wird dem Anwender überlassen

Abb. 6: Vergleich DV-gestützter Möglichkeiten zur Regelverarbeitung

3.1 Einsatz einer regelbasierten Komponente in einem Kosteninformationssystem

Sowohl der Einsatz eines Expertensystems als auch die Verwendung eines konventionellen Programms sind bereits im Rahmen des KICK-Projektes zur konstruktionsbegleitenden Kalkulation getestet worden. Da die zuerst durchgeführte Realisierung mit einem Expertensystem zu unverhältnismäßig hohen Laufzeiten führte, wurde im weiteren ein erster Prototyp zur Ressourcenzuordnung in Form eines C-Programms erstellt. Der dadurch erreichte Performance-Gewinn führte jedoch zu den bereits im Zusammenhang mit konventionellen Programmen erläuterten Nachteilen.

Der hier vorgestellte Ansatz ist, dem Kosteninformationssystem KICK ein auf Entscheidungstabellen (Abk.: ET) basierendes Regelwerk zur Verfügung zu stellen, da nur mit einem solchen ET-System die notwendige Flexibilität bei der Pflege der Regeln sowie gleichzeitig ein zufriedenstellendes Laufzeitverhalten zu erwarten sind.

Ausdrücke in Entscheidungstabellen

Damit Bedingungen und Aktionen abgespeichert und interpretiert werden können, sind Datenformate festzulegen, die allen darauf zugreifenden Funktionen bekannt sind. Die Definition

dieser Datenformate basiert auf dem Konzept von Ausdrücken. Jeder im Entscheidungstablensystem verwendete Ausdruck hat folgenden einfachen Aufbau:

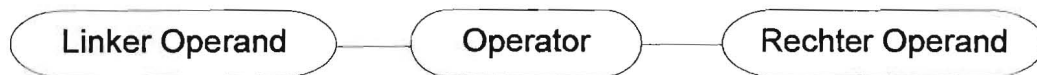


Abb. 7: Grundaufbau eines einfachen Ausdrucks

Je nach verwendetem Operator werden folgende Arten von Ausdrücken unterschieden:

Ausdruck	Operator	Ergebnis	Verwendung ET	Beispiel
Vergleich	< ≤ = < > ≥	Logischer Wert	Bedingung	Rauhtiefe > 3 mm
Formel	+ - * /	Numerischer Wert	Bedingung / Aktion	Länge * Breite * Höhe
Zuweisung	= bzw. :=	<undefiniert>	Aktion	Menge := 1,5 Std.

Abb. 8: Ausdrucksarten in einer Entscheidungstabelle

Die logischen Operatoren ("UND", "ODER") werden nicht benötigt, da sie bereits durch den Grundaufbau von Entscheidungstabellen durch Anordnung von Bedingungen (Konjunktion) und Regeln (Disjunktion) abgebildet werden. Ein Ausdruck mit einem logischen Operator kann, wie nachstehendes Beispiel zeigt, in einfache Ausdrücke zerlegt werden.

Beispiel:

Blechdicke < 5 mm UND
Blechdicke < 6 mm

Blechdicke = 5 mm ODER
Blechdicke = 6 mm

		R1
B1	Blechdicke < 5 mm	J
B2	Blechdicke < 6 mm	J

		R1	R2
B1	Blechdicke = 5 mm	J	N
B2	Blechdicke = 10 mm	N	J

Abb. 9: Komplexe Ausdrücke in Entscheidungstabellen

Durch die oben beschriebenen Operatoren werden in den Ausdrücken die unterschiedlichsten Operanden miteinander verknüpft. Operanden stellen dabei einen numerischen Wert dar bzw. können einen solchen Wert annehmen. Für das Entscheidungstablensystem werden drei Gruppen von Operanden unterschieden:

☐ Allgemein

Konstanten (z. B. PI, spezifisches Gewicht von Stahl)

Lokale Variablen (Speicherung von Zwischenergebnissen)

☐ KICK-spezifisch

Lesende Zugriffe auf Datenbankfelder des KICK-Systems (abhängig von aktuell zu bewertenden Konstruktionsobjekten)

Funktionen zum Analysieren des Datenbestands (mit Hilfe programmierter Routinen können komplexe Auswertungen vorgenommen werden. Das Ergebnis dieser Auswertungen wird über den Returncode der Funktion in Bedingungen bzw. Aktionen aufgenommen.)

Verweise auf Vorgänge und Konstanten (Ein Verweis repräsentiert unabhängig vom aktuell zu bewertenden Konstruktionsobjekt einen konkreten Datensatz.)

☐ Formeln

4 DV-System

Die nachstehende Abbildung veranschaulicht die Architektur des ET-Systems, das aus folgenden drei Komponenten besteht:

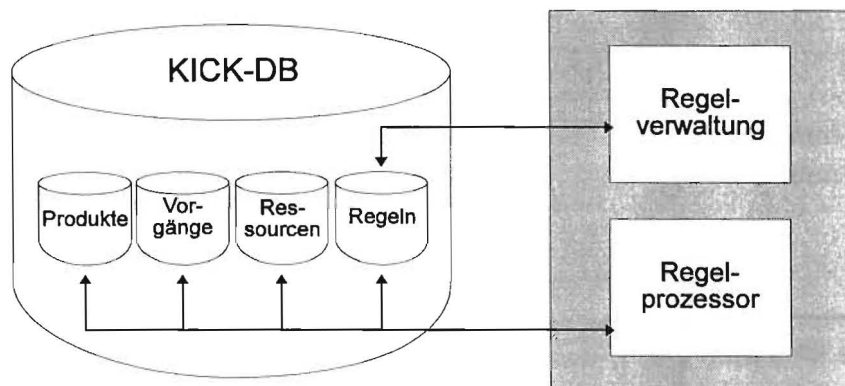


Abb. 10: Architektur des datenbankgestützten Entscheidungstabellensystems

Datenbasis

Die für Entscheidungstabellen relevanten Daten werden zusammen mit den Daten zur konstruktionsbegleitenden Kalkulation in einer gemeinsamen SQL-Datenbank gespeichert. Dadurch ist es möglich, die referentielle Integrität zwischen Regeln und den KICK-Daten mit Hilfe des Datenbankmanagementsystems sicherzustellen.

Regelverwaltung

Die Regelverwaltung stellt dem Regelprozessor Funktionen zum Einfügen von Regeln in der Datenbasis bereit. Der Regelprozessor greift auf die in der Datenbasis verwalteten Regeln zurück.

Regelprozessor

Der Regelprozessor arbeitet die abgespeicherten Entscheidungsregeln zur Ressourcenzuordnung anhand der eingehenden Vorgänge zu einem Konstruktionsobjekt (Produkt, Baugruppe oder Einzelteil) ab und ordnet diesen Vorgängen die entsprechenden Ressourcen zu.

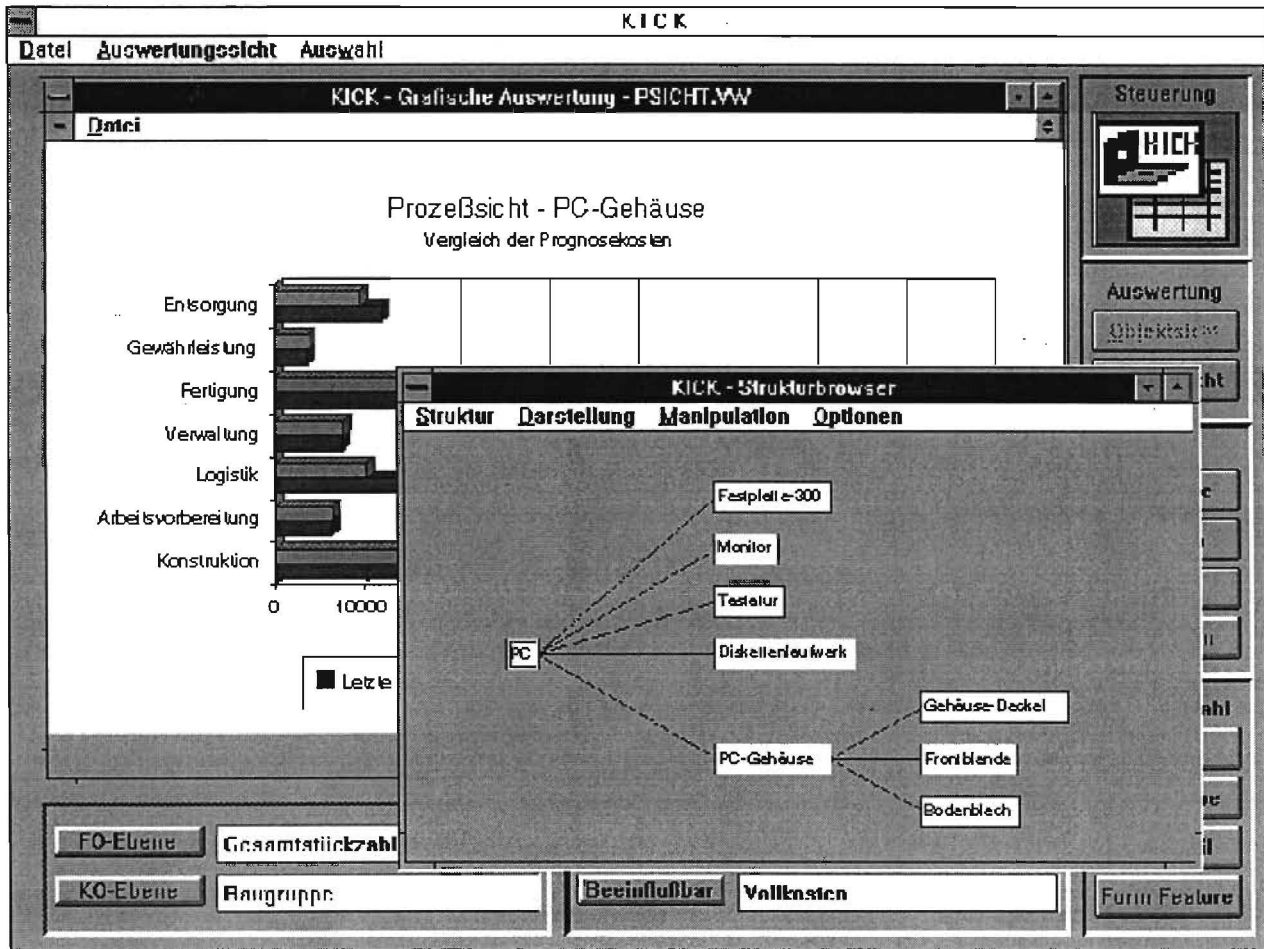


Abb. 8: Benutzungsoberfläche des Kalkulationsmoduls

Das realisierte DV-System KICK besteht insgesamt aus folgenden Programmodulen:

- ☐ Objekt-Struktur-Browser
- ☐ Sachmerkmal-Modul
- ☐ CAD-Modul
- ☐ Regelwerk zur Vorgangs- und Ressourcenzuordnung

☐ Kalkulationsmodul

Diese Module sind in eine Oberfläche integriert und greifen auf eine gemeinsame Datenbank (SQL-Server) zu. Das System ist als Client/Server-Architektur realisiert und ermöglicht es, verschiedene relationale Datenbanksysteme zu verwenden. Die Datenversorgung erfolgt aus Produktionsplanungs- und Rechnungswesenssystemen.

Zum Aufbau der Objektstruktur wird der Browser eingesetzt. Zur Detaillierung werden dem Konstruktionsobjekt im Sachmerkmalsmodul die aktuell geforderten Ausprägungen zugewiesen. In späteren Phasen können die Bauteile mit Hilfe eines CAD-Moduls detailliert werden. Das Regelwerkmodul legt aufgrund der Sachmerkmalsausprägungen Vorgangsstrukturen für die Prozeßbereiche fest und bestimmt die zur Ausführung der Vorgänge notwendigen Zeiten und Ressourcen. Das Kalkulationsmodul übernimmt Kostenrechnungsdaten und ermittelt daraus Prozeßkostensätze. Zur Darstellung der Kosten kann der Anwender zwischen zwei Aggregationsebenen (Objekt- und Prozeßsicht) wählen.

Wissensbasierte Angebotsbearbeitung von Materialflußanlagen am Beispiel von Recyclinganlagen¹

Dr.-Ing. Klaus Richter, Dipl.-Ing. Olaf Wolter
Institut für Förder- und Baumaschinentechnik, Stahlbau, Logistik
Universitätsplatz 2, 39106 Magdeburg
E-mail: wolter@masch-bau.uni-magdeburg.de

1. Anliegen

Mit der verstärkten Orientierung mittelständischer Komponentenhersteller auf die Entwicklung und den Vertrieb kompletter Anlagen verändern sich neben organisatorischen Notwendigkeiten auch die Anforderungen an die CAX-Werkzeuge in den produktionsvorbereitenden Abteilungen des Unternehmens. Das betrifft sowohl die Komplexität in der Verwaltung der Anlagenkomponenten als auch die Wahl der Planungs- und Konfigurierungsmethoden. Die produktspezifische Arbeitsweise des Serienfertigers wird durch das projektbezogene Systemdenken des Anlagenprojektanten ergänzt. Die Unternehmen müssen mittelfristig zu rechen-technischen Unternehmenslösungen kommen, die Erzeugniskonstruktion und Anlagenplanung unter einem ganzheitlichen Datenbankkonzept durchgängig und methodisch ähnlich bedienen, um z. B. ohne Schnittstellenverluste die Erstellung von Baugruppenstücklisten aus einem Anlagenlayout zu ermöglichen.

Die Angebotsphase im Anlagenbaugeschäft ist gegenüber der Auftragsphase durch einige spezifische Besonderheiten gekennzeichnet, die sich aus fehlenden Daten im Tender der Ausschreibung, Unsicherheiten in der Berücksichtigung von Alternativen und Informationsdefiziten in Parametern der zu nutzenden Ausrüstungen und Baugruppen ergeben. Diese „Nachteile“ können durch die Integration aggregierter Daten- und Methodensammlungen in einen ganzheitlichen Modellansatz effizient rechentechnisch gemindert werden.

Es wird ein Konfigurierungsansatz vorgestellt, der über die verschiedenen Phasen des Planungsprozesses unter Nutzung strukturierter Lösungskataloge die funktionsorientierte Gestaltung von Materialflußanlagen rechentechnisch unterstützen kann.

¹ Die im Rahmen eines Stipendiums der Stiftung Industrieforschung zum Thema „CAX-Methoden zur wissensbasierten Planung von Materialflußanlagen“ gewonnenen Erkenntnisse wurden am Institut für Förder- und Baumaschinentechnik, Stahlbau, Logistik der Universität Magdeburg für die Fa. Förderanlagen Calbe GmbH in eine Anwenderlösung umgesetzt, die eine Konfiguration von Recyclinganlagen in frühen Phasen der Angebotsbearbeitung ermöglicht.

2. Der Konfigurierungsgegenstand Recyclinganlage

Materialflußtechnische Anlagen zur Realisierung von Schüttgutprozessen beinhalten förder- technische und verfahrenstechnische Komponenten verschiedenster Hersteller, die nur durch ihr Zusammenwirken die Funktionsfähigkeit der Anlage gewährleisten und deshalb exakt aufeinander abzustimmen sind. Die große Variabilität in der Anlagengestaltung zwingt den Planer, rationell bei der Komponentenauswahl vorzugehen. Dem tragen die Komponentenhersteller Rechnung, indem sie Baureihen von maschinenbautechnischen Baugruppen und Elementen anbieten, die schnell und flexibel verfügbar sind (Bild 1).

Schüttgut-Recyclinganlagen als junge Branche in der Materialflußtechnik werden bestimmt durch eine begrenzte Anzahl kompakter verfahrenstechnischer Ausrüstungen zur Gutbehandlung, die mit Stetigförderern und Rutschen für den Guttransport komplettiert werden (Konfigurierungsmerkmal „Closed-World-Assumption“). Verfahrensstrukturen sind für definierte Zielstellungen vorhanden (Konfigurierungsmerkmal Strukturiertheit).

Für einige Komponenten wie den Gurtförderer sind Dimensionierungsvorschriften vorhanden, andere - insbesondere verfahrenstechnische Ausrüstungen - können auch auf Grund unvollständiger Berechnungsgrundlagen nur durch Tests und vergleichende Betrachtungen ausgelegt werden /LOCH90/. Die Gestaltung von Gurtförderern gehört dennoch zu den kompliziertesten Themen in der Materialflußplanung, da die Integration der Daten verschiedenster Komponentenhersteller und die Variation von Erfahrungswerten für sensible Punkte des Gurtförderers ein manuell kaum beherrschbares Lösungsfeld erzeugen.

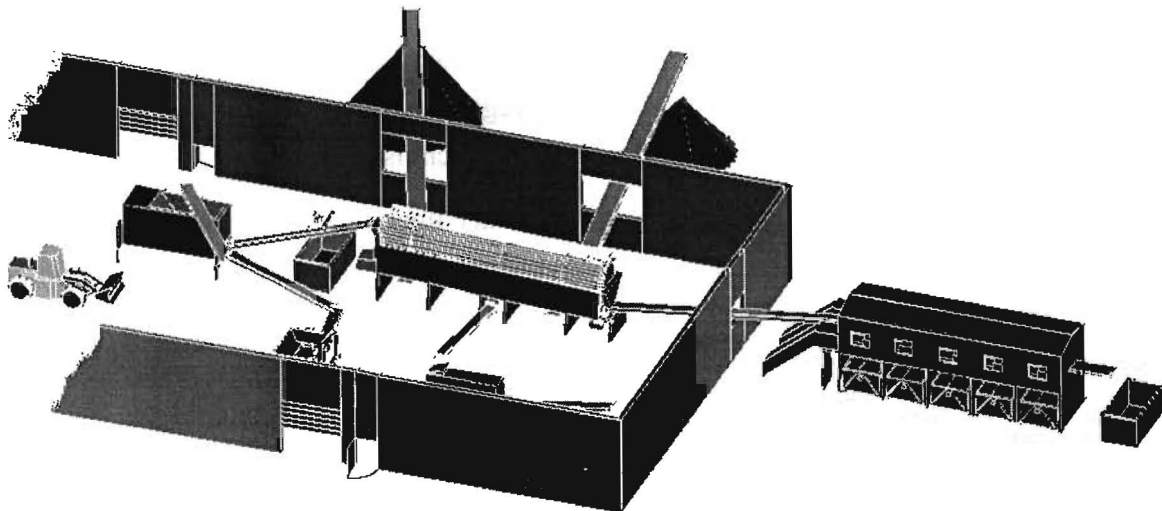


Bild 1 Recyclinganlage - Übersichtsdarstellung

3. Konfigurierung materialflußtechnischer Anlagen

3.1. Der Konfigurierungsansatz

Die Konfigurierung von materialflußtechnischen Anlagen gliedert sich grob in konzeptionelle Phasen (Problembeschreibung, Entwicklung der Funktionsstruktur und Funktionsträgerstruktur) und in die gestaltende Phase /RICH87/. Die Strukturen bauen aufeinander auf und sind durch eine spezielle Knoten-Kanten-Kopplung über die Konfigurierungsphasen hinweg ineinander überführbar, um die stark iterativen Planungsvorgänge zur Variantenbildung unterstützen zu können.

Für die Strukturen gelten folgende Vereinbarungen:

- a) Funktionsstruktur: Abbildung der wichtigsten Systemfunktionen in Form von gerichteten Graphen mit Knoten (Funktion) und Kanten (Materialflußbeziehung und -richtung).
„In den Funktionsstrukturen wird keine Aussage über das Aggregat gemacht, das eine einzelne Funktion erfüllen soll“ /LOCH91, S. 51/.
- b) Funktionsträgerstruktur: Abstrakte Beschreibung konkreter technischer Realisierungen durch Instrumentierung.
„Durch Abstraktion der geometrischen Daten können wesentliche technologische Informationen über Anlagen dargestellt und verglichen werden“ /LOCH91, S. 56/.
- c) Gestaltsstruktur: Geometrische Beschreibung der präzisierten und vollständig bewerteten Instrumentarien.

Eine derartige 3-Sichtenbeschreibung ermöglicht ein vertikales und horizontales Konfigurieren von Materialflußanlagen. Das horizontale Konfigurieren ist durch den Prozeß der Anordnung und Kombination der einzelnen Konfigurationsobjekte gekennzeichnet, während das vertikale Konfigurieren die Überführung der Strukturen in andere Konfigurierungsphasen übernimmt.

3.2. Methoden und Vorgehensweise bei der Anlagenkonfiguration

In den Konfigurierungsstufen bedient sich der Anlagenplaner verschiedener typischer Methoden:

- Auswahl von (katalogisierten) Konfigurierungsobjekten (Selektionsproblem),
- Festlegung quantitativer und qualitativer Objekteigenschaften (Dimensionierungsproblem),
- Erzeugung einer Systemstruktur (Kompositionsproblem) und
- Bewertung und Vergleich von Gesamt- und Teillösungen (Evaluierungsproblem).

Diese Tätigkeiten kehren in allen Stufen des Projektierungsprozesses wieder (Bild 2).

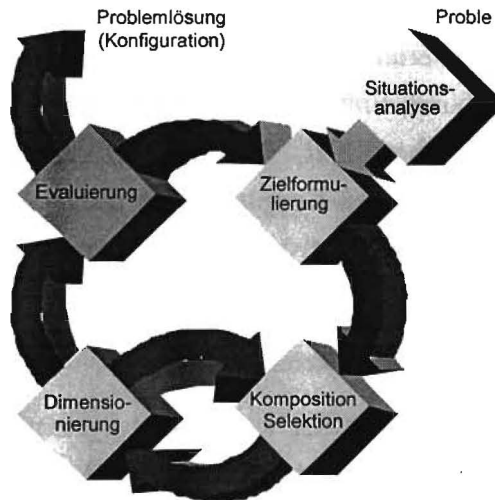


Bild 2 Tätigkeiten im Konfigurierungszyklus

Die Vor- und Rückwärtsverketzung der Strukturen über die Konfigurierungsphasen hinweg soll es auch erlauben, aus einem reinen Gestaltentwurf nachträglich konzeptionelle Objektstrukturen erzeugen zu können (Bild 3).

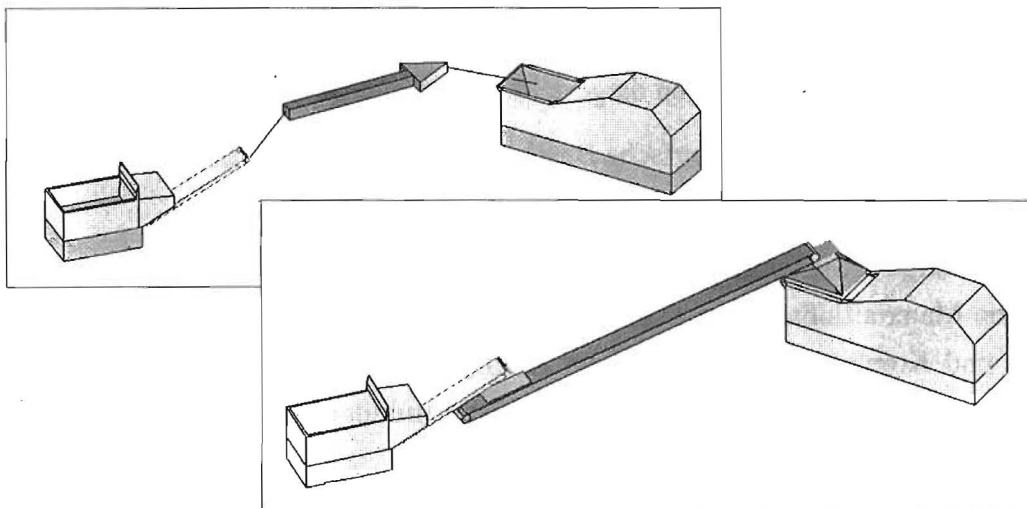


Bild 3 Überführung von Konfigurierungsphasen (Beispiel Transportfunktion - Gurtförderer)

Die Durchsetzung dieses Zieles hat folgende Konsequenzen:

- Hinter jedem (katalogisierten) Gestaltmodell muß der gesamte Produktentstehungsprozeß einschließlich der Funktions- und Funktionsträgerstruktur abgelegt sein.
- Die symbolische Darstellung konzeptioneller Strukturen muß in das 3D-Modell der Anlage integrierbar sein. Das bedeutet die Entwicklung von 3D-Symbolen.
- Die Kopplungsstruktur muß konsistent bleiben, um die den Systemknoten betreffenden Stoffflüsse für neue Auswahl- und Dimensionierungsüberlegungen nutzen zu können.

4. Konfiguration von Recyclinganlagen

4.1. Der Strukturentwurf

Unter den Konfigurationsobjekten werden alle Basiselemente eines Konfigurationssystems zusammengefaßt. Sie enthalten die für die Beschreibung der Erzeugnisse notwendigen technischen, technologischen und kommerziellen Merkmale und Gestaltdefinitionen für alle Phasen des Produktentstehungsprozesses einschließlich der Methoden zur Exemplarisierung.

Der Spezifikation von Produktdatenmodellen und Schnittstellen für den Datenaustausch gehen umfangreiche Analysen zum Einsatzgebiet voraus, um branchenspezifische Datenstrukturen abstrahieren zu können. Sie umfassen die Klassifikation und Strukturierung der Erzeugnisstruktur, die Definition von Sachmerkmaleisten und Modellen für alle Planungsphasen sowie die Beschreibung der Auswahl-, Dimensionierungs- und Anordnungsmethoden. Der unternehmensübergreifende Abgleich von Produktdatenstrukturen ermöglicht es, Planungsmethoden zu verallgemeinern und abstrahierte Konfigurationsobjekte in allen Planungsphasen zur Verfügung zu stellen /TOME92/.

Die Methoden für die Festlegung der genauen Gestalt (Exemplarisierung) erfordern projektierungsspezifisches Wissen zur funktionsbedingten

- Gestaltung der einzelnen Komponenten (Baugrößen usw.),
- Anordnung der Komponenten untereinander (Kontaktierungen, Bediengänge, Sympathie- und Antipathiebeziehungen usw.),
- Einpassung in eine gegebene Einbausituation (Schnittstellen, Anschlüsse, Einbauten, Sperrflächen usw.).

Die Gestalt des einzelnen Konfigurationsobjektes kann mit unterschiedlichen Strategien erzeugt werden, die von dem eingesetzten CAD-System, aber auch von dem übergebenen Modell des Erzeugnisherstellers abhängen. Neben der einfachen Skalierung und Streckung von branchenspezifischen Grobmodellen, die für Grobplanungsphasen bzw. für eine effektive Darstellung von umgebenden Einrichtungen vollkommen ausreichen, bieten parametrisierte, formfeature-basierte Modelle mit Zugriff auf technische Datenbanken eine elegante Möglichkeit, Erzeugnisse realitätsnah abzubilden. Aus Sicht des Anlagenplaners gilt die Strategie „so genau wie nötig“, in jedem Fall aber inklusive einer exakten Abbildung der funktionsbestimmenden Flächen, die als Bezugsflächen für Anordnungsroutinen genutzt werden.

Da im Strukturentwurf alle phasenspezifischen Ausprägungen des Konfigurationsobjektes gehalten werden, ist ein Wechsel zwischen den verschiedenen Darstellungsebenen - auch auf Teilstrukturen beschränkt - zur Modifikation von Objekten möglich. Das bedingt neben gut strukturierten Lösungskatalogen die Verfügbarkeit graphischer Darstellungen von allen Funktionen, Wirkprinzipien, Funktionsträgern und Gestaltmodellen für das Anwendungsgebiet (Bild 4).

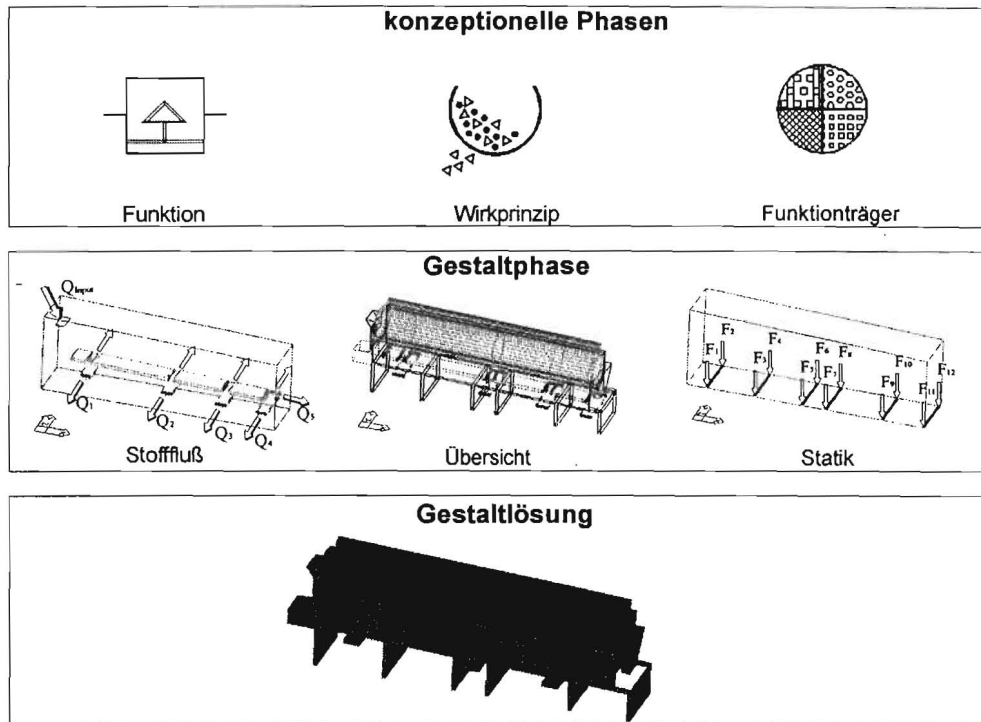


Bild 4 Modellsichten einer Mischtrommel in den verschiedenen Konfigurierungsphasen

Multifunktionale Sichten auf Technologische Schemata, hier in Form der gleichzeitigen Anzeige von Funktion und Verfahren in einem Systemknoten, können Abhängigkeiten zwischen konzeptionellen Phasen besser aufzeigen (Bild 5).

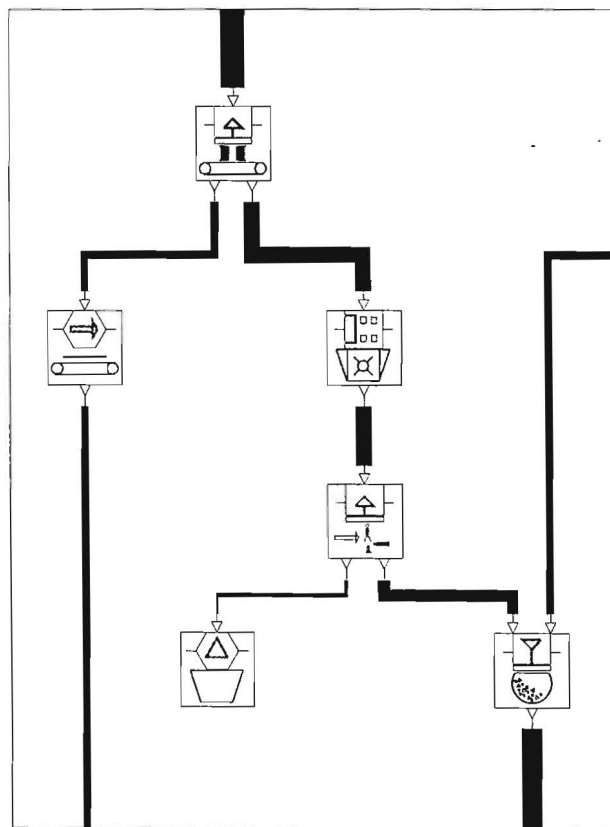


Bild 5 Verfahrens- und Funktionssymbole in einem Technologischen Schema

4.2. Methoden zur Anordnung/Gestaltanpassung

Im Gestaltungsprozeß unter besonderer Berücksichtigung der konfigurierenden (projektierenden) Arbeitsweise wird zwischen

- a) Methoden zur wirkflächenbezogenen Anordnung von Gestaltmodellen und
 - b) Methoden zur geometrischen Anpassung von Modellen an die Umgebung
- unterschieden.

zu a) Für den generativen Aufbau komplexer Strukturen (auch von Reihenfolgen) aus einfachen Gestaltmodellen mit einer einheitlichen, baugrößenunabhängigen Beschreibungsmethode erweisen sich statik- und stoffflußbezogene Wirkflächen als vorteilhaft.

Diese Methode bildet die Denkweise des Projektanten nach und wird deshalb auch als Verständigungsmittel mit dem Projektanten genutzt. Sie ermöglicht eine einfache Beschreibung projektierungsspezifischen Wissens und ist leicht programmtechnisch umsetzbar, sofern der Geometriemodellierer bzw. die Zeichnungsdatenbank den wirkflächenbezogenen Zugriff auf geometrische Strukturen zulassen. Die Anordnungsmethoden sind produktneutral. Die Gestaltmodelle können unter Einbeziehung besonders referenzierter Funktionsflächen, z. B. in Form von Layern, vom Projektanten selbst erstellt werden.

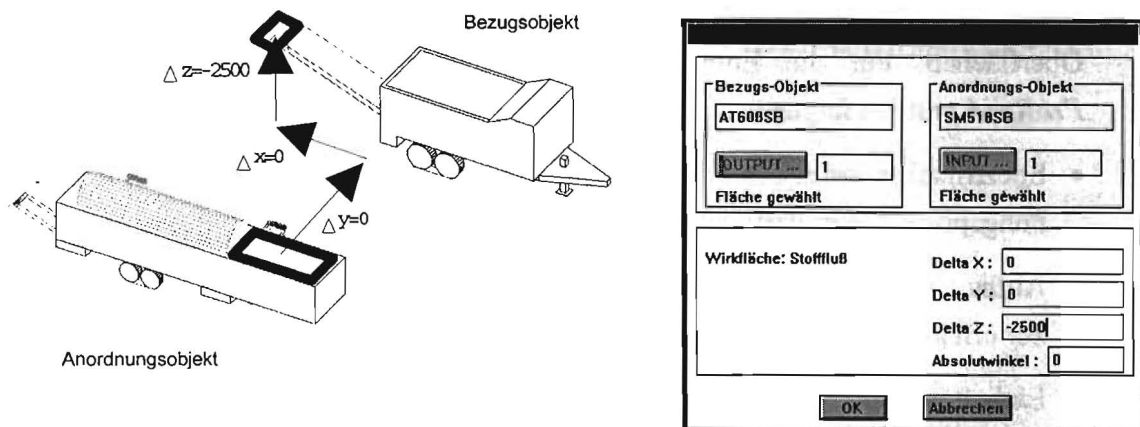


Bild 6 Wirkflächenorientierte Anordnung von Geometriemodellen

zu b) Für die gestaltbezogene Anpassung von Modellen an die Umgebung sind Methoden notwendig, die die Gestaltparameter zum Zeitpunkt der Einbindung in das Anlagenlayout neu berechnen. Die Modelle müssen u. U. hochgradig parametrisiert sein, um möglichst flexibel Gestaltänderungen zuzulassen. Abhängig vom speziell verwendeten CAD-System kann dadurch ein hoher spezifischer Programmieraufwand entstehen, der nur vom CAD-Spezialisten zu realisieren ist. (Bild 7)

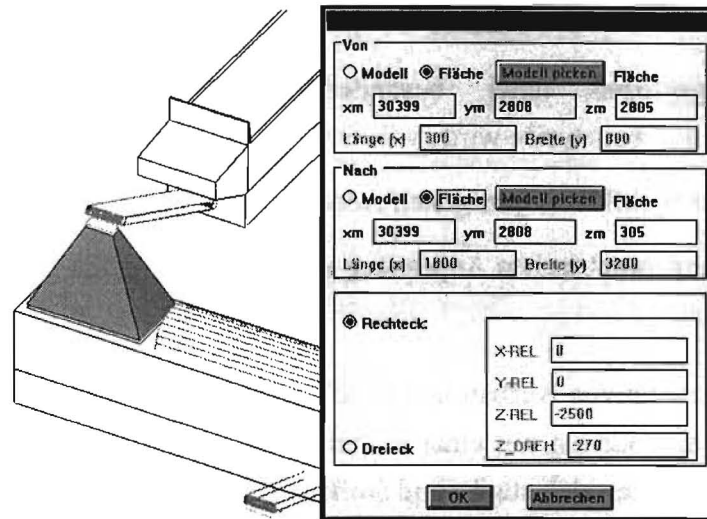


Bild 7 Projektierungsspezifische Kopplung von Ausrüstungen (Beispiel Rutsche)

5. Einbettung der Verfahren in ein integriertes Ingenieursystem

Um den Forderungen nach einem einheitlichen Rahmensystem für die rechnergestützten Konfigurierungs- und Planungsmethoden nachzukommen, gewinnt das „Integrierte Ingenieursystem“ als koordinierendes Konzept an Bedeutung /GAUS94, BAST90/. In der Anlagenplanung wird auf Grund des stark wechselnden Aufgabencharakters der Verständlichkeit von branchenspezifischen Dialogstrategien große Aufmerksamkeit gewidmet. Die vorwiegend gestaltende Tätigkeit des Anlagenplaners bindet Konfigurierungsabläufe eng an CAD-Oberflächen. Für die Entwicklung eines Konfigurationssystems für Recyclinganlagen *ProRecA* wurden folgende Schwerpunkte anwendungsorientiert bearbeitet (Bild 8):

- Spezifikation eines Produktmodells Recyclinganlage, das alle Phasen des Produktentstehungsprozesses in Partialmodellen abbildet,
- Aufbau von branchenspezifischen Lösungselementebibliotheken für alle Planungsphasen zur effektiven Konfiguration komplexer Produkte,
- Einbettung von Methoden zur gestaltbezogenen Anordnung und Anpassung von Modellstrukturen mit dem Schwerpunkt „Verbindungselemente Gurtförderer und Rutsche“,
- Definition branchenspezifischer Schnittstellen für den Produktdatenaustausch auf Basis eines relationalen Datenbankkonzeptes.

Die anforderungsgerechte, branchenweite Bereitstellung der Konfigurationsobjekte in Form von Lösungskatalogen kann nur über standardisierte Datenschnittstellen erfolgen. Datenaustauschformate für Katalog- und Normteile sind seit einigen Jahren genormt (/DINF14/), werden aber mit den laufenden Entwicklungen zum Standard für den Produktmodell datenaustausch STEP ("Standard for the Exchange of Product Model Data" /ANDE94/) Veränderungen erfahren. Die vorliegenden Aktivitäten werden in den Entwurf für ein STEP-Anwenderprotokoll "Fabrikplanung/Materialflußtechnik" einfließen.

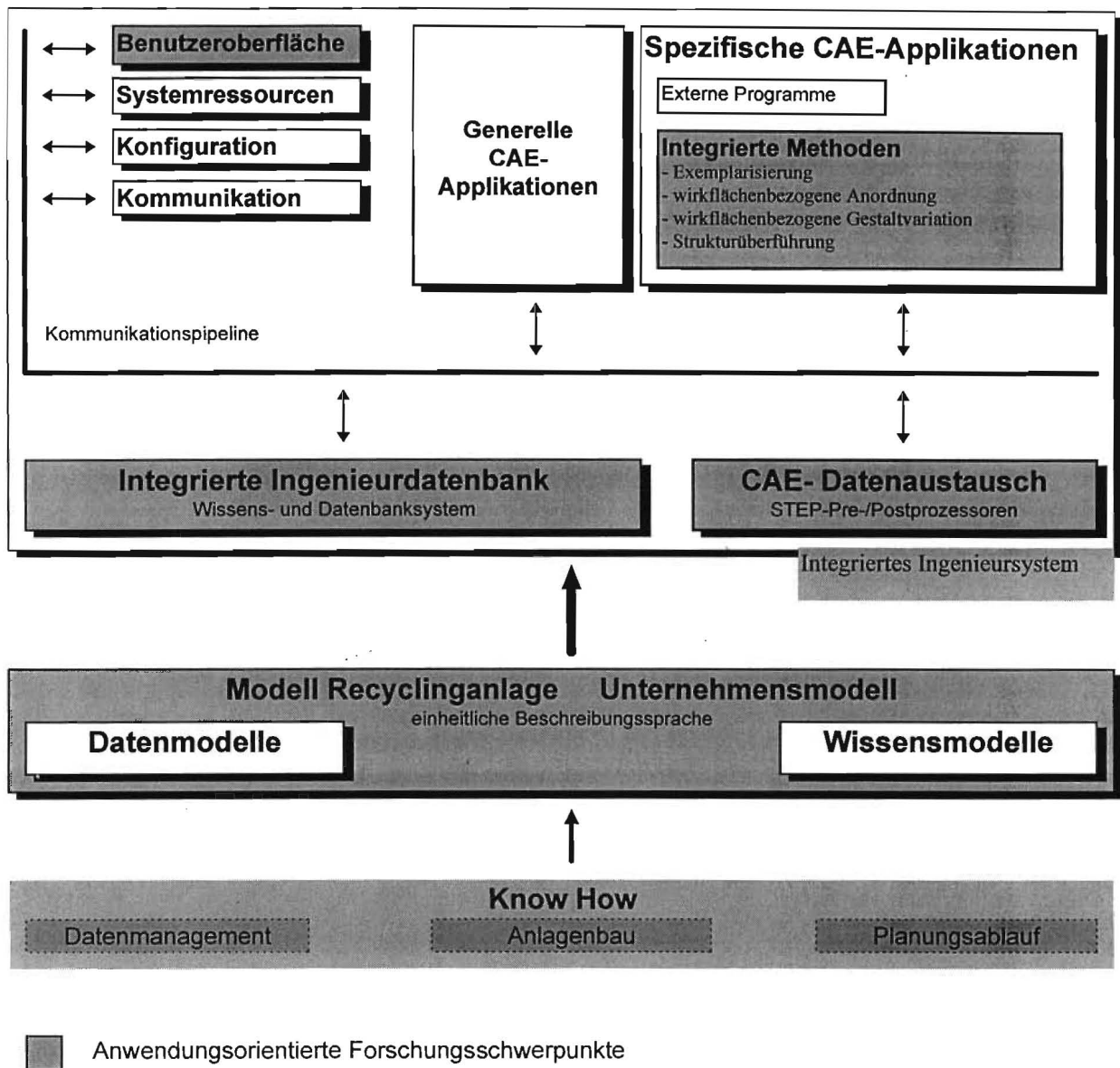


Bild 8 Aufbau eines integrierten Ingenieursystems für die Gestaltung von Recyclinganlagen

6. Literatur

- /ANDE94/ Anderl, R.; Wasmer, A.: Chancen und Risiken der Produktdatentechnologie (PDT). In: Gausemeier, J. [Hrsg.] CAD'94. München, Wien: Hanser, 1994, S. 57-72.
- /BAST90/ Bastek, A.; Deh, U.; Richter, K.; Scholz, E; Schütze, B; Ziems, D.: Universelles CAD-System für die Konstruktion und Projektierung. In: Maschinenbautechnik 39 (1990) 4, S. 148 - 153.
- /DINF14/ N.;N.: DIN-Fachbericht 14: CAD-Normteile nach DIN. Beuth -Verlag 1990.
- /GAUS94/ Gausemeier, J., Frank, Th., Genderka, M.: Erfolgspotentiale integrierter Ingenieursysteme. In: Gausemeier, J. [Hrsg.] CAD'94. München, Wien: Hanser, 1994, S. 3-28.
- /LOCH90/ Lochter, F.; Richter, K.; Rauschenbach, W.; Rempt, D: Rechnergestützte Dimensionierung von Aufbereitungsmaschinen im Prozeß der Projektierung von Schüttgutförderanlagen. In: Neue Bergbautechnik 20 (1990) 7, S. 260 - 264.
- /LOCH91/ Lochter, F.: Rechnergestützte Strukturierung und Dimensionierung in der Angebotsphase, aufgezeigt am Beispiel von Schüttgutförderanlagen. Magdeburg, TU Magdeburg, Dissertation, 1991
- /RICH87/ Richter, K.: Zur Methodik und Algorithmierung der Projektierung von Gurtbandförderanlagen in der Angebotsphase. Magdeburg, Technische Hochschule, Dissertation, 1987
- /TOME92/ Tomek, A.: Anwendung von Ordnungssystemen für Materialflußkomponenten. Diplomarbeit an der Fakultät für Allgemeinen Maschinenbau der Techn. Univ. Magdeburg, Magdeburg 1992.



Konfigurierung, Auswahl und Auslegung von statisch belasteten, hydrodynamisch geschmierten Gleitlagern in KONWERK¹

Zusammenfassung

Innerhalb des BMFT-Verbundprojektes PROKON - „**Problemspezifische Werkzeuge** für die wissensbasierte **Konfigurierung**“ wird ein Expertensystem zur Auswahl und Auslegung von Gleitlagern entwickelt. Aufgabe des XPS ist die Unterstützung des Konstrukteurs, der meist kein Gleitlagerexperte ist, bei der fachgerechten Auslegung eines Gleitlagers.

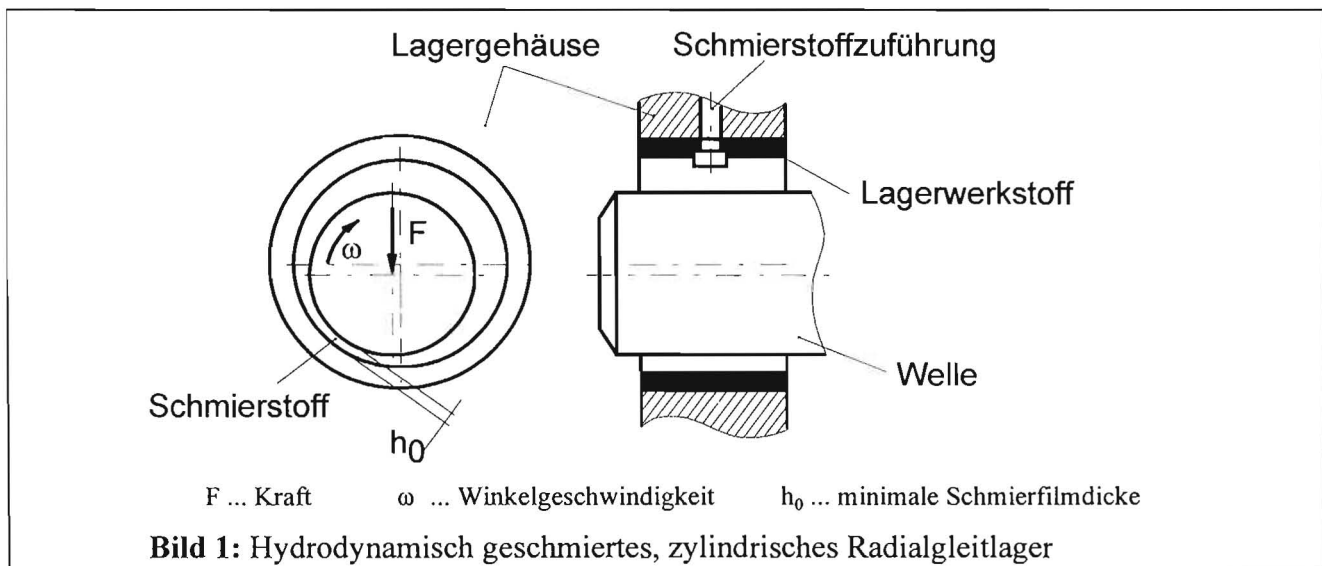
Dazu werden mit Hilfe des modularen Werkzeugsystems KONWERK für verschiedene Gleitlagertypen Gleitlager aus den Bestandteilen Lagertyp, Schmierstoff, Schmierstoffzuführung, Lagergehäuse, Lagerwerkstoff und Welle konfiguriert, jeweils geeignete Komponenten ausgewählt, ausgelegt (berechnet) und optimiert.

Der besondere Reiz aus Sicht der Konfigurierung liegt in der Integration von Optimierungsmethoden, Fuzzyverfahren und relativ komplizierten, weil iterativen Berechnungsabläufen.

1. Hydrodynamisch geschmierte Gleitlager

Gleitlager sind ein Maschinenelement zur Abstützung einer rotierenden Welle in einem Gehäuse (oder umgekehrt). Die auftretende Relativbewegung führt zum Gleiten der Welle in der Lagerschale und damit zu mehr oder weniger großer Reibung. Bei hydrodynamisch geschmierten Gleitlagern wird eine vollständige Trennung der beiden Teile der Gleitpaarung durch einen Schmierstoff realisiert und damit eine starke Reduzierung der Reibleistung (=Verlustleistung) erreicht.

Den prinzipiellen Aufbau eines (Radial-)Gleitlagers zeigt Bild 1.



¹ Das diesem Beitrag zugrundeliegende Vorhaben wird mit Mitteln des Bundesministers für Forschung und Technologie unter dem Förderkennzeichen 01 IW 202 E5 - Verbundvorhaben PROKON gefördert. Die Verantwortung für den Inhalt liegt beim Autor.

2. Auslegung hydrodynamisch geschmierte Gleitlager

Ziel der Gleitlagerauslegung ist es, durch die Auswahl geeigneter Komponenten und die richtige Festlegung der Auslegungsparameter eine ausreichende Betriebssicherheit (Schmierfilmdicke) bei möglichst geringer Verlustleistung zu gewährleisten.

Die Besonderheiten der Gleitlagerauslegung sind:

- ein hoher Anteil von "Expertenwissen" bei Festlegung der Auslegungsparameter und der Ergebnisinterpretation einschließlich
- Unschärfe und Unsicherheit
- relativ komplizierte Berechnungen, relativ hohe Rechengenauigkeiten (bzgl. wissenschaftlicher Systeme)
- iterative Berechnungsabläufe
- die Notwendigkeit der Optimierung nach verschiedenen Kriterien

Dabei ist von einem Expertensystem zur Auswahl und Auslegung von Gleitlagern die Lösung folgender Aufgaben zu unterstützen:

- Konfigurierung des Gleitlagers aus Lagertyp, Schmierstoff, Schmierstoffzuführung, Gehäuse, Lagerwerkstoff, Welle (Bild 1)
- Auswahl geeigneter Komponenten (Lagertyp, Schmierstoff, Lagerwerkstoff, ...)
- Festlegung geeigneter Auslegungsparameter
- Nachrechnung der ausgewählten Variante

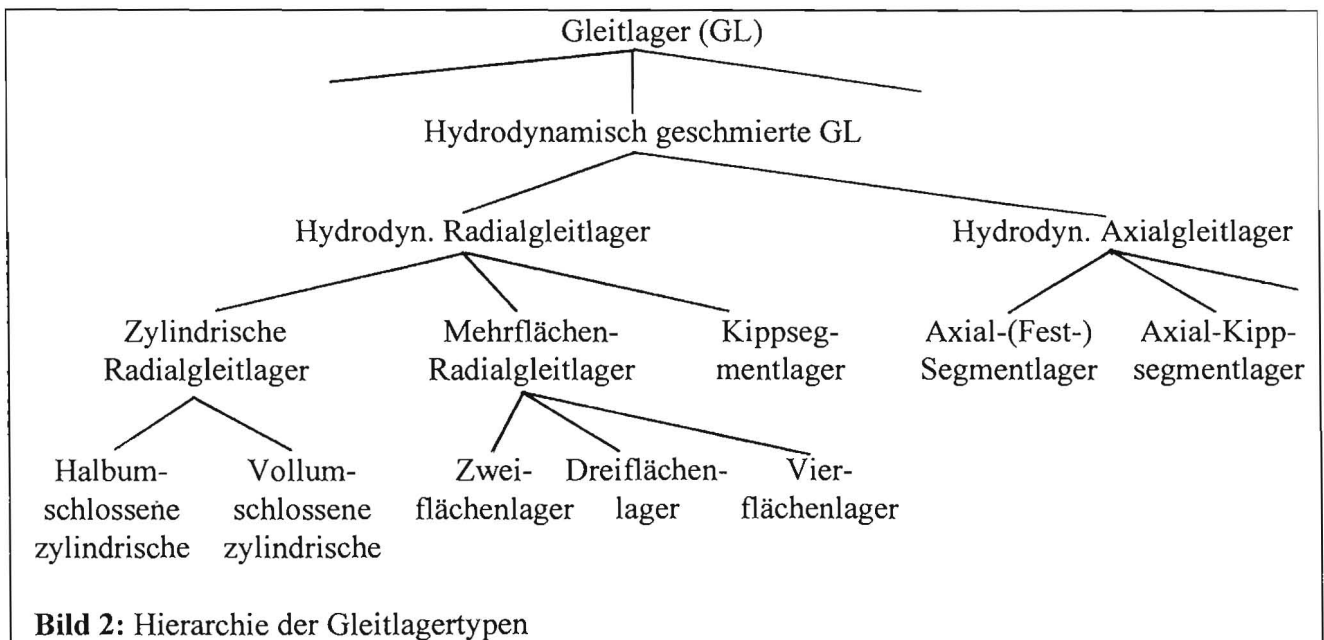
2.1. Konfigurierung

Die Konfiguration des Gleitlagers im Sinne seiner Struktur, seines Aufbaus stellt kein Problem dar. Sie liegt im Prinzip für jeden Lagertyp fest. Unterschiedlich sind lediglich die Ausprägungen einzelner Komponenten. Aufgrund der begrenzten Anzahl verschiedener Lagertypen könnten alle Konfigurationen explizit definiert werden.

2.2. Auswahl geeigneter Komponenten

Problematischer gestaltet sich die Auswahl geeigneter Komponenten in Abhängigkeit vom jeweiligen Einsatzfall. Hierfür sind einerseits Berechnungen erforderlich und andererseits Expertenwissen einzubeziehen, wobei aufgrund der iterativen Berechnungsabläufe manche Entscheidungen eigentlich erst nach erfolgter Auslegung getroffen werden könnten.

Bild 2 zeigt als Beispiel einen Ausschnitt aus der Hierarchie der Gleitlagertypen, aus der letztlich ein Typ auszuwählen ist.



Auswahlkriterien sind hierbei u.a.:

- Krafrichtung, Kraftgröße
- Drehzahlen
- Abmessungen
- Belastungscharakteristik (Störkräfte, ...)
- zulässige Verlustleistung/Reibung/Erwärmung
- ...

Zur Unterstützung von Auswahlentscheidungen bieten sich u.a. **Fuzzyverfahren** an, mit denen das vorhanden Expertenwissen repräsentiert werden kann. Als Beispiel soll hier die Auswahl eines Axialgleitlagertyps anhand der spezifischen Lagerbelastung dienen (Bild 3).

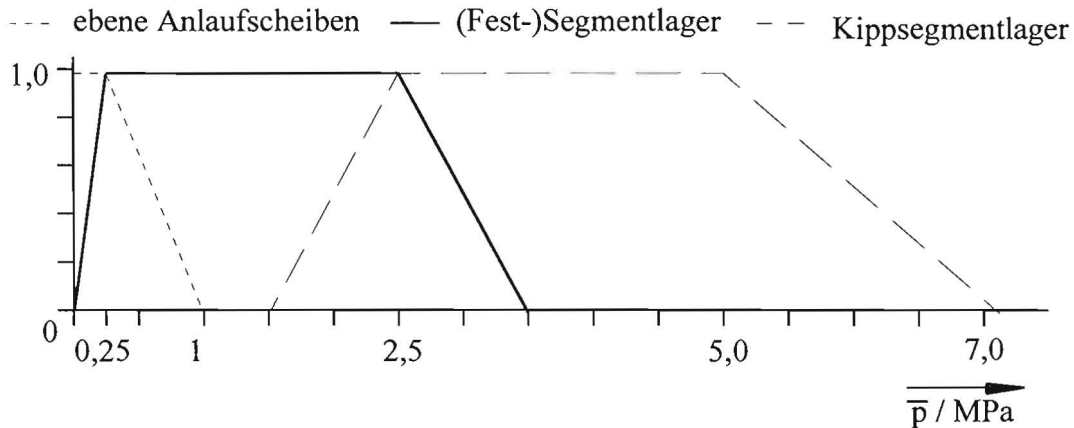


Bild 3: Fuzzybewertung der Einsatzgrenzen verschiedener Axialgleitlagertypen anhand der spezifischen Lagerbelastung

Eine scharfe Trennung der Einsatzbereiche von Festsegmentlagern und Kippsegmentlagern bei 2,5 MPa würde die tatsächlichen Verhältnisse unbefriedigend repräsentieren. Vielmehr gibt es einen Bereich um 2,5 MPa, in dem beide Lagertypen eingesetzt werden können. Anhand der Fuzzybewertung beider Alternativen kann sich der Konstrukteur letztlich für einen Lagertyp entscheiden.

2.3. Festlegung geeigneter Auslegungsparameter

Bei der Auslegung von Gleitlagern muß der Konstrukteur Parameter angeben, die seinen Einsatzfall beschreiben und eine Berechnung des Lagers gestatten. Einige dieser Parameter sind in gewissen Grenzen frei wählbar. Der Gleitlagerexperte legt aufgrund seines Wissens und seiner Erfahrungen die Parameter so, daß gute, dem Optimum nahekommende Lösungen entstehen und unterscheidet sich dadurch vom weniger spezialisierten Konstrukteur. Dieses Defizit sollte ein Expertensystem zumindest teilweise ausgleichen.

Eine Möglichkeit besteht in der Nutzung von **Optimierungsverfahren** zur Parameterfestlegung. Interessant sind bei den betrachteten Gleitlagern im wesentlichen 4 Optimierungsziele:

- große Schmierfilmdicke (Betriebssicherheit)
- geringe Reibleistung (Energieverluste)
- geringer Schmierstoffdurchsatz (technischer Aufwand)
- geringe Lager-/Schmierfilmtemperatur (Ölalterung, ...)

Diese Ziele sind z.T. gegenläufig, was die Problematik zusätzlich verschärft. Die einfachste Lösung ist, die für den konkreten Einsatzfall wichtigste Größe zu optimieren, besser wäre jedoch eine multi-kriterielle Optimierung auf der Basis einer (einsatzfallabhängigen) Wichtung der vier Kriterien. Diese sollte die komplexen Entscheidungen eines Gleitlagerexperten möglichst gut nachvollziehen. Als Beispiel soll die Wahl des relativen Lagerspiels ψ bei der Auslegung zylindrischer Radialgleitlager betrachtet werden (Bild 4).

Optimal bezüglich der Reibleistung und des Öldurchsatzes wäre hier ein möglichst kleines relatives Lagerspiel von 0,3 ‰, was aber zu hohen Lagertemperaturen und geringer Schmierfilmdicke führen würde. Bei Betrachtung aller Kriterien könnte das Optimum z.B. bei 1 ‰ liegen.

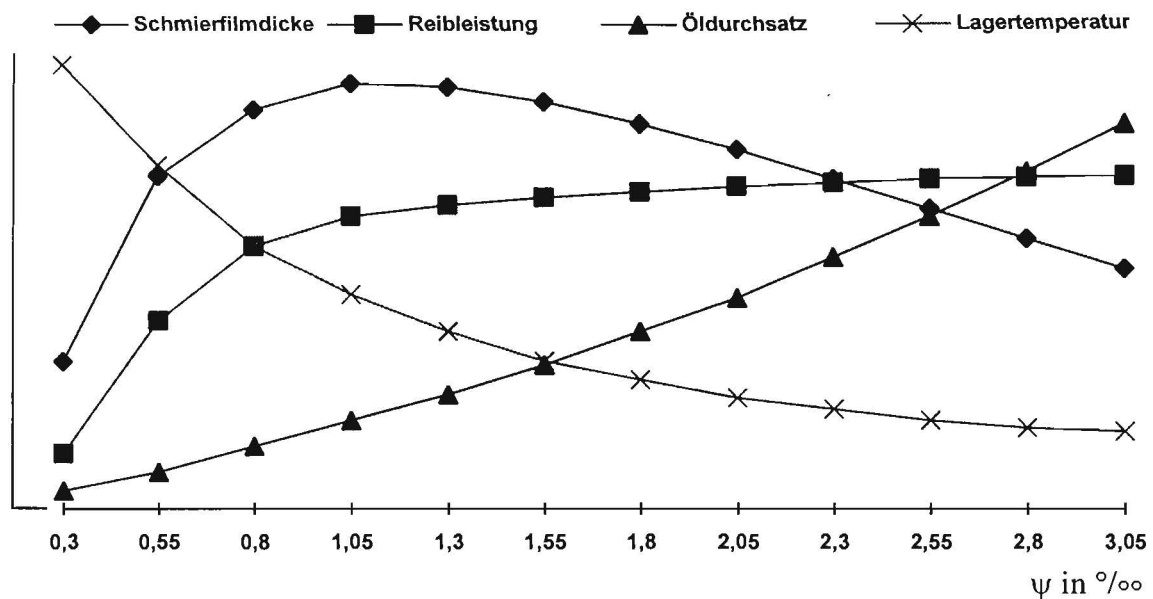


Bild 4: Abhängigkeit verschiedener Parameter zyl. Radialgleitlager vom relativen Lagerspiel ψ

2.4. Nachrechnung der ausgewählten Variante

Die Berechnung von Gleitlagern ist iterativ und nicht ganz trivial. Die Berechnung zylindrischer Radialgleitlager soll dies verdeutlichen.

Gegeben sind: Belastung, Drehzahl, Abmessungen, ein Schmierstoff und andere Größen

Gesucht sind: minimale Schmierfilmdicke, Schmierfilmtemperatur, Reibleistung und Öldurchsatz

Iteration 1: Iteration der effektiven Schmierfilmtemperatur

Vorgabe einer (geschätzten) effektiven Schmierfilmtemperatur

⇒ Dichte des Schmierstoffs für die effektive Schmierfilmtemperatur

$$\rho_{eff} = \rho_{20^\circ C} \cdot \left[1 - \alpha_v (\vartheta_{eff} - 20^\circ C) \right]$$

⇒ Viskosität des Schmierstoffs für die errechnete Dichte

$$\eta_{eff} = 0,18 \cdot e^{\left[\left(\frac{159,56}{\vartheta_{eff} + 95} \right) - 0,181913 \ln \left(\frac{\eta_{40^\circ C} \cdot \rho_{eff}}{0,18} \right) \right]}$$

⇒ Sommerfeldzahl für die gegebene Belastung und die errechnete Viskosität

$$So = \frac{F \cdot \psi_{eff}^2}{D \cdot B \cdot \eta_{eff} \cdot \omega_{eff}}$$

⇒ **Iteration 2:** Iteration der relativen Exzentrizität ε aus der Sommerfeldzahl

$$So = \left(\frac{B}{D} \right)^2 \cdot \frac{\varepsilon}{2 \cdot (1 - \varepsilon^2)^2} \cdot \sqrt{\pi^2 \cdot (1 - \varepsilon^2) + 16 \cdot \varepsilon^2} \cdot \frac{\alpha_1 (\varepsilon - 1)}{\alpha_2 + \varepsilon}$$

⇒ Berechnung des Öldurchsatzes für die berechnete Viskosität und relative Exzentrizität, ...

⇒ Berechnung der Wärmeabgabemenge für den errechneten Öldurchsatz und die angenommene, effektive Schmierfilmtemperatur, ...

⇒ Berechnung der Reibleistung für die errechnete Sommerfeldzahl, ...

⇒ Vergleich der Reibleistung mit der Wärmeabgabemenge und

Korrektur der Annahme für die effektive Schmierfilmtemperatur

⇒ Berechnung der minimalen Schmierfilmdicke

$$h_0 = 0,5 \cdot D \cdot \psi_{eff} \cdot (1 - \varepsilon)$$

Bild 5: Iterativer Berechnungsablauf für zylindrische Radialgleitlager

3. Realisierung in KONWERK

KONWERK ist ein modulares Werkzeugsystem für Konfigurierungsaufgaben im weitesten Sinne und wird derzeit im Rahmen des BMFT-Verbundprojektes PROKON (Projektende Juni 1995) entwickelt. KONWERK stellt in sogenannten „Basismodulen“ eine gewisse Grundfunktionalität zur Verfügung, die durch die Integration von „Erweiterungsmodulen“ in verschiedene Richtungen, z.B. Fuzzyverfahren, Optimierungsverfahren, ausgebaut - also „erweitert“ werden kann.

Zur Realisierung eines Expertensystems zur Auswahl und Auslegung von Gleitlagern wird eine Wissensbasis erstellt, die das im vorangegangenen Abschnitt beschriebene Problem bearbeiten soll. Bestandteile der Wissensbasis sind:

- **eine Begriffshierarchie:**

Hier werden die verschiedenen Gleitlagertypen, Schmierstoffe (Öle), Schmierstoffzuführungen, Lagergehäuse und Werkstoffe in einer *Spezialisierungshierarchie* als „Domänenobjekte“ (DO) verwaltet. Für sie werden in Slots Eigenschaften definiert, die auf untergeordnete Objekte vererbt werden.

Die für die eigentliche Konfigurierung benötigte *kompositionelle Hierarchie* wird durch Relationen („has-parts“, „part-of“) aufgebaut, die ebenfalls Bestandteil der Definition eines DO sind.

- **Constraints:**

Mit Hilfe von Constraints werden verschiedene Abhängigkeiten zwischen DO und zwischen Eigenschaften von DO definiert. Z.B. wird über ein Constraint sichergestellt, daß sich das Verhältnis von Lagerbreite zu Lagerdurchmesser im Bereich zwischen 0,3 und 1 einstellt. Für den (häufigen) Fall, daß der Lagerdurchmesser vorgegeben ist, führt das Constraint zu einer Einschränkung des Wertebereiches für die Lagerbreite.

- **Strategien zur Steuerung** der Konfigurierung und Auslegung:

Durch die Definition von Strategien wird sichergestellt, daß bei der Problemlösung nicht der zweite Schritt vor dem ersten getan wird. Bei der Gleitlagerauslegung sind zunächst Einflußgrößen wie Belastung, Drehzahl, Abmessungen u.a. zu ermitteln, bevor Schmierstoff, Gehäuse und Schmierstoffzuführung festgelegt werden. Anschließend kann die eigentliche Berechnung des Gleitlagers erfolgen und der Werkstoff festgelegt werden.

- **Berechnungsverfahren** (Berechnungsfunktionen):

Wie bereits erwähnt, sind bei der Gleitlagerauslegung iterative, nicht ganz triviale Berechnungen anzustellen. Diese lassen sich zum größten Teil nicht über Constraints abbilden. Deshalb werden in KONWERK Methoden (sogenannte „Berechnungsfunktionen“) definiert, mit denen sich beliebige Funktionalität realisieren läßt. Diese Methoden „hängen“ an einem Slot eines DO und werden dann aktiv, wenn die Steuerung diesen Slot zur Bearbeitung freigibt. Einzige Bedingung für die erfolgreiche Ausführung einer solchen Berechnungsfunktion ist, daß der zurückgelieferte Wert von KONWERK als zulässiger Wert dieses Slots akzeptiert wird.

Die in 2.4. dargestellte Iteration der effektiven Schmierfilmtemperatur wird auf diese Art und Weise gelöst.

Zur Nutzung von Fuzzy-Verfahren stehen die Erweiterungsmodule „Fuzzy-Begriffshierarchie“ und „unscharfe Constraints“ zur Verfügung, welche die Definition von Zugehörigkeitsfunktionen und das Rechnen mit unscharfen Werten gestatten.

Auf die weiteren KONWERK-Konzepte (Optimierung, fallbasiertes Arbeiten, ...) und die Problematik der Bedienoberfläche kann hier leider nicht eingegangen werden.

Zusammenfassend ist festzustellen, daß KONWERK für einen großen Bereich von Konstruktions- und Konfigurierungsaufgaben tragfähige Konzepte bereitstellt. Für einen Teilbereich konnte das durch den realisierten Prototyp zur Auswahl und Auslegung von Gleitlagern nachgewiesen werden. Es ist jedoch anzumerken, daß die Nutzung der verschiedenen Konzepte einen gewissen Einarbeitungsaufwand erfordert, der nicht zu unterschätzen ist.



Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

-Bibliothek, Information
und Dokumentation (BID)-
PF 2080
67608 Kaiserslautern
FRG

Telefon (0631) 205-3506
Telefax (0631) 205-3210

e-mail
dfkibib@dfki.uni-kl.de
WWW
<http://www.dfki.uni-sb.de/dfkibib>

Veröffentlichungen des DFKI

Die folgenden DFKI Veröffentlichungen sowie die aktuelle Liste von allen bisher erschienenen Publikationen können von der oben angegebenen Adresse oder per anonymous ftp von <ftp.dfki.uni-kl.de> (131.246.241.100) im Verzeichnis `pub/Publications` bezogen werden. Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

DFKI Publications

The following DFKI publications or the list of all published papers so far are obtainable from the above address or by anonymous ftp from <ftp.dfki.uni-kl.de> (131.246.241.100) in the directory `pub/Publications`. The reports are distributed free of charge except where otherwise noted.

DFKI Research Reports

RR-94-39

Hans-Ulrich Krieger

Typed Feature Formalisms as a Common Basis for Linguistic Specification.
21 pages

RR-94-38

Hans Uszkoreit, Rolf Backofen, Stephan Busemann, Abdel Kader Diagne, Elizabeth A. Hinkelman, Walter Kasper, Bernd Kiefer, Hans-Ulrich Krieger, Klaus Netter, Günter Neumann, Stephan Oepen, Stephen P. Spackman

DISCO-An HPSG-based NLP System and its Application for Appointment Scheduling.
13 pages

RR-94-37

Hans-Ulrich Krieger, Ulrich Schaefer

TDL - A Type Description Language for HPSG, Part 1: Overview.
54 pages

RR-94-36

Manfred Meyer

Issues in Concurrent Knowledge Engineering. Knowledge Base and Knowledge Share Evolution.
17 pages

RR-94-35

Rolf Backofen

A Complete Axiomatization of a Theory with Feature and Arity Constraints
49 pages

RR-94-34

Stephan Busemann, Stephan Oepen, Elizabeth A. Hinkelman, Günter Neumann, Hans Uszkoreit

COSMA - Multi-Participant NL Interaction for Appointment Scheduling
80 pages

RR-94-33

Franz Baader, Armin Laux

Terminological Logics with Modal Operators
29 pages

RR-94-31

Otto Kühn, Volker Becker, Georg Lohse, Philipp Neumann

Integrated Knowledge Utilization and Evolution for the Conservation of Corporate Know-How
17 pages

RR-94-23

Gert Smolka

The Definition of Kernel Oz
53 pages

RR-94-20

Christian Schulte, Gert Smolka, Jörg Würtz
 Encapsulated Search and Constraint Programming
 in Oz
 21 pages

RR-94-18

Rolf Backofen, Ralf Treinen
 How to Win a Game with Features
 18 pages

RR-94-17

Georg Struth
 Philosophical Logics—A Survey and a Bibliography
 58 pages

RR-94-16

Gert Smolka
 A Foundation for Higher-order Concurrent Constraint Programming
 26 pages

RR-94-15

Winfried H. Graf, Stefan Neurohr
 Using Graphical Style and Visibility Constraints for a Meaningful Layout in Visual Programming Interfaces
 20 pages

RR-94-14

Harold Boley, Ulrich Buhrmann, Christof Kremer
 Towards a Sharable Knowledge Base on Recyclable Plastics
 14 pages

RR-94-13

Jana Koehler
 Planning from Second Principles—A Logic-based Approach
 49 pages

RR-94-12

Hubert Comon, Ralf Treinen
 Ordering Constraints on Trees
 34 pages

RR-94-11

Knut Hinkelmann
 A Consequence Finding Approach for Feature Recognition in CAPP
 18 pages

RR-94-10

Knut Hinkelmann, Helge Hintze
 Computing Cost Estimates for Proof Strategies
 22 pages

RR-94-08

Otto Kühn, Björn Höfling
 Conserving Corporate Knowledge for Crankshaft Design
 17 pages

RR-94-07

Harold Boley
 Finite Domains and Exclusions as First-Class Citizens
 25 pages

RR-94-06

Dietmar Dengler
 An Adaptive Deductive Planning System
 17 pages

RR-94-05

Franz Schmalhofer, J. Stuart Aitken, Lyle E. Bourne jr.
 Beyond the Knowledge Level: Descriptions of Rational Behavior for Sharing and Reuse
 81 pages

RR-94-03

Gert Smolka
 A Calculus for Higher-Order Concurrent Constraint Programming with Deep Guards
 34 pages

RR-94-02

Elisabeth André, Thomas Rist
 Von Textgeneratoren zu Intellimedia-Präsentationssystemen
 22 Seiten

RR-94-01

Elisabeth André, Thomas Rist
 Multimedia Presentations: The Support of Passive and Active Viewing
 15 pages

RR-93-48

Franz Baader, Martin Buchheit, Bernhard Hollunder
 Cardinality Restrictions on Concepts
 20 pages

RR-93-46

Philipp Hanschke
 A Declarative Integration of Terminological, Constraint-based, Data-driven, and Goal-directed Reasoning
 81 pages

RR-93-45*Rainer Hoch*

On Virtual Partitioning of Large Dictionaries for
Contextual Post-Processing to Improve Character
Recognition

21 pages

RR-93-44

*Martin Buchheit, Manfred A. Jeusfeld, Werner
Nutt, Martin Staudt*

Subsumption between Queries to Object-Oriented
Databases

36 pages

RR-93-43*M. Bauer, G. Paul*

Logic-based Plan Recognition for Intelligent Help
Systems

15 pages

RR-93-42*Hubert Comon, Ralf Treinen*

The First-Order Theory of Lexicographic Path Or-
derings is Undecidable

9 pages

RR-93-41*Winfried H. Graf*

LAYLAB: A Constraint-Based Layout Manager for
Multimedia Presentations

9 pages

RR-93-40

*Francesco M. Donini, Maurizio Lenzerini, Daniele
Nardi, Werner Nutt, Andrea Schaerf*

Queries, Rules and Definitions as Epistemic State-
ments in Concept Languages

23 pages

RR-93-38*Stephan Baumann*

Document Recognition of Printed Scores and Trans-
formation into MIDI

24 pages

RR-93-36

*Michael M. Richter, Bernd Bachmann, Ansgar Ber-
nardi, Christoph Klauck, Ralf Legleitner, Gabriele
Schmidt*

Von IDA bis IMCOD: Expertensysteme im CIM-
Umfeld

13 Seiten

RR-93-35*Harold Boley, François Bry, Ulrich Geske (Eds.)*

Neuere Entwicklungen der deklarativen KI-
Programmierung — *Proceedings*

150 Seiten

Note: This document is available for a nominal char-
ge of 25 DM (or 15 US-\$).

RR-93-34*Wolfgang Wahlster*

Verbmobil Translation of Face-To-Face Dialogs

10 pages

RR-93-33*Bernhard Nebel, Jana Koehler*

Plan Reuse versus Plan Generation: A Theoretical
and Empirical Analysis

33 pages

RR-93-32*David R. Traum, Elizabeth A. Hinkelman*

Conversation Acts in Task-Oriented Spoken Dia-
logue

28 pages

RR-93-31*Elizabeth A. Hinkelman, Stephen P. Spackman*

Abductive Speech Act Recognition, Corporate
Agents and the COSMA System

34 pages

RR-93-30*Stephen P. Spackman, Elizabeth A. Hinkelman*

Corporate Agents

14 pages

RR-93-29*Armin Laux*

Representing Belief in Multi-Agent Worlds via Ter-
minological Logics

35 pages

RR-93-28

*Hans-Ulrich Krieger, John Nerbonne, Hannes Pir-
ker*

Feature-Based Allomorphy

8 pages

RR-93-27*Hans-Ulrich Krieger*

Derivation Without Lexical Rules

33 pages

RR-93-26*Jörg P. Müller, Markus Pischel*

The Agent Architecture InterRRaP: Concept and
Application

99 pages

RR-93-25*Klaus Fischer, Norbert Kuhn*

A DAI Approach to Modeling the Transportation Domain
93 pages

RR-93-24*Rainer Hoch, Andreas Dengel*

Document Highlighting — Message Classification in Printed Business Letters
17 pages

RR-93-23*Andreas Dengel, Ottmar Lutz*

Comparative Study of Connectionist Simulators
20 pages

RR-93-22*Manfred Meyer, Jörg Müller*

Weak Looking-Ahead and its Application in Computer-Aided Process Planning
17 pages

RR-93-20*Franz Baader, Bernhard Hollunder*

Embedding Defaults into Terminological Knowledge Representation Formalisms
34 pages

RR-93-18*Klaus Schild*

Terminological Cycles and the Propositional μ -Calculus
32 pages

RR-93-17*Rolf Backofen*

Regular Path Expressions in Feature Logic
37 pages

RR-93-16*Gert Smolka, Martin Heuz, Jörg Würtz*

Object-Oriented Concurrent Constraint Programming in Oz
17 pages

RR-93-15*Frank Berger, Thomas Fehrle, Kristof Klöckner, Volker Schölles, Markus A. Thies, Wolfgang Wahlster*

PLUS - Plan-based User Support Final Project Report
33 pages

RR-93-14*Joachim Niehren, Andreas Podelski, Ralf Treinen*

Equational and Membership Constraints for Infinite Trees
33 pages

RR-93-13*Franz Baader, Karl Schlechta*

A Semantics for Open Normal Defaults via a Modified Preferential Approach
25 pages

RR-93-12*Pierre Sablayrolles*

A Two-Level Semantics for French Expressions of Motion
51 pages

RR-93-11*Bernhard Nebel, Hans-Jürgen Bürckert*

Reasoning about Temporal Relations: A Maximal Tractable Subclass of Allen's Interval Algebra
28 pages

RR-93-10*Martin Buchheit, Francesco M. Donini, Andrea Schaerf*

Decidable Reasoning in Terminological Knowledge Representation Systems
35 pages

RR-93-09*Philipp Hanschke, Jörg Würtz*

Satisfiability of the Smallest Binary Program
8 pages

RR-93-08*Harold Boley, Philipp Hanschke, Knut Hinkelmann, Manfred Meyer*

COLAB: A Hybrid Knowledge Representation and Compilation Laboratory
64 pages

RR-93-07*Hans-Jürgen Bürckert, Bernhard Hollunder, Armin Laux*

Concept Logics with Function Symbols
36 pages

RR-93-06*Hans-Jürgen Bürckert, Bernhard Hollunder, Armin Laux*

On Skolemization in Constrained Logics
40 pages

RR-93-05*Franz Baader, Klaus Schulz*Combination Techniques and Decision Problems for Disunification
29 pages**RR-93-04***Christoph Klauck, Johannes Schwagereit*GGD: Graph Grammar Developer for features in CAD/CAM
13 pages**RR-93-03***Franz Baader, Berhard Hollunder, Bernhard Nebel, Hans-Jürgen Profitlich, Enrico Franconi*An Empirical Analysis of Optimization Techniques for Terminological Representation Systems
28 pages**RR-93-02***Wolfgang Wahlster, Elisabeth André, Wolfgang Finkler, Hans-Jürgen Profitlich, Thomas Rist*Plan-based Integration of Natural Language and Graphics Generation
50 pages**RR-93-01***Bernhard Hollunder*An Alternative Proof Method for Possibilistic Logic and its Application to Terminological Logics
25 pages

DFKI Technical Memos**TM-94-04***Cornelia Fischer*PantUDE – An Anti-Unification Algorithm for Expressing Refined Generalizations
22 pages**TM-94-03***Victoria Hall*Uncertainty-Valued Horn Clauses
31 pages**TM-94-02***Rainer Bleisinger, Berthold Kröll*Representation of Non-Convex Time Intervals and Propagation of Non-Convex Relations
11 pages**TM-94-01***Rainer Bleisinger, Klaus-Peter Gores*Text Skimming as a Part in Paper Document Understanding
14 pages**TM-93-05***Michael Sintek*Indexing PROLOG Procedures into DAGs by Heuristic Classification
64 pages**TM-93-04***Hans-Günther Hein*Propagation Techniques in WAM-based Architectures — The FIDO-III Approach
105 pages**TM-93-03***Harold Boley, Ulrich Buhmann, Christof Kremer*Konzeption einer deklarativen Wissensbasis über recyclingrelevante Materialien
11 pages**TM-93-02***Pierre Sablayrolles, Achim Schupeta*Conflict Resolving Negotiation for COoperative Schedule Management Agents (COSMA)
21 pages**TM-93-01***Otto Kühn, Andreas Birk*Reconstructive Integrated Explanation of Lathe Production Plans
20 pages

DFKI Documents**D-94-15***Stephan Oepen*German Nominal Syntax in HPSG — On Syntactic Categories and Syntagmatic Relations —
80 pages

D-94-14*Hans-Ulrich Krieger, Ulrich Schaefer.*TDL - A Type Description Language for HPSG,
Part 2: User Guide.

72 pages

D-94-12*Arthur Sehn, Serge Autexier (Hrsg.)*Proceedings des Studentenprogramms der 18. Deutschen Jahrestagung für Künstliche Intelligenz KI-94
69 Seiten**D-94-11***F. Baader, M. Buchheit, M. A. Jeusfeld, W. Nutt (Eds.)*Working Notes of the KI'94 Workshop: KRDB'94
- Reasoning about Structured Objects: Knowledge Representation Meets Databases

65 pages

Note: This document is no longer available in printed form.

D-94-10*F. Baader, M. Lenzerini, W. Nutt, P. F. Patel-Schneider (Eds.)*

Working Notes of the 1994 International Workshop on Description Logics

118 pages

Note: This document is available for a nominal charge of 25 DM (or 15 US-\$).

D-94-09DFKI Wissenschaftlich-Technischer Jahresbericht
1993

145 Seiten

D-94-08*Harald Feibel*

IGLOO 1.0 - Eine grafikunterstützte Beweisentwicklungsumgebung

58 Seiten

D-94-07*Claudia Wenzel, Rainer Hoch*

Eine Übersicht über Information Retrieval (IR) und NLP-Verfahren zur Klassifikation von Texten

25 Seiten

D-94-06*Ulrich Buhrmann*

Erstellung einer deklarativen Wissensbasis über recyclingrelevante Materialien

117 Seiten

D-94-04*Franz Schmalhofer, Ludger van Elst*

Entwicklung von Expertensystemen: Prototypen, Tiefenmodellierung und kooperative Wissensentwicklung

22 Seiten

D-94-03*Franz Schmalhofer*

Maschinelles Lernen: Eine kognitionswissenschaftliche Betrachtung

54 Seiten

D-94-02*Markus Steffens*

Wissenserhebung und Analyse zum Entwicklungsprozeß eines Druckbehälters aus Faserverbundstoff

90 pages

D-94-01*Josua Boon (Ed.)*DFKI-Publications: The First Four Years
1990 - 1993

75 pages

D-93-27*Rolf Backofen, Hans-Ulrich Krieger, Stephen P. Spackman, Hans Uszkorcit (Eds.)*

Report of the EAGLES Workshop on Implemented Formalisms at DFKI, Saarbrücken

110 pages

D-93-26*Frank Peters*

Unterstützung des Experten bei der Formalisierung von Textwissen INFOCOM - Eine interaktive Formalisierungskomponente

58 Seiten

D-93-25*Hans-Jürgen Bürckert, Werner Nutt (Eds.)*

Modeling Epistemic Propositions

118 pages

Note: This document is available for a nominal charge of 25 DM (or 15 US-\$).

D-93-24*Brigitte Krenn, Martin Volk*

DiTo-Datenbank: Datendokumentation zu Funktionsverbgefügen und Relativsätzen

66 Seiten

D-93-22*Andreas Abecker*

Implementierung graphischer Benutzungsoberflächen mit Tcl/Tk und Common Lisp

44 Seiten

Note: This document is no longer available in printed form.

D-93-21

Dennis Drollinger

Intelligentes Backtracking in Inferenzsystemen am Beispiel Terminologischer Logiken
53 Seiten

D-93-20

Bernhard Herbig

Eine homogene Implementierungsebene für einen hybriden Wissensrepräsentationsformalismus
97 Seiten

D-93-16

Bernd Bachmann, Ansgar Bernardi, Christoph Klauck, Gabriele Schmidt

Design & KI

74 Seiten

D-93-15

Robert Laux

Untersuchung maschineller Lernverfahren und heuristischer Methoden im Hinblick auf deren Kombination zur Unterstützung eines Chart-Parsers
86 Seiten

D-93-14

Manfred Meyer (Ed.)

Constraint Processing – Proceedings of the International Workshop at CSAM'93, St.Petersburg, July 20-21, 1993
264 pages

Note: This document is available for a nominal charge of 25 DM (or 15 US-\$).

D-93-12

Harold Boley, Klaus Elsbernd, Michael Herfert, Michael Sintek, Werner Stein

RELFUN Guide: Programming with Relations and Functions Made Easy
86 pages

D-93-11

Knut Hinkelmann, Armin Laux (Eds.)

DFKI Workshop on Knowledge Representation Techniques — Proceedings
88 pages

Note: This document is no longer available in printed form.

D-93-10

Elizabeth Hinkelman, Markus Vonderden, Christoph Jung

Natural Language Software Registry (Second Edition)
174 pages

D-93-09

Hans-Ulrich Krieger, Ulrich Schäfer

TDLEExtraLight User's Guide

35 pages

D-93-08

Thomas Kieninger, Rainer Hoch

Ein Generator mit Anfragesystem für strukturierte Wörterbücher zur Unterstützung von Texterkennung und Textanalyse
125 Seiten

D-93-07

Klaus-Peter Gores, Rainer Bleisinger

Ein erwartungsgesteuerter Koordinator zur partiellen Textanalyse
53 Seiten

D-93-06

Jürgen Müller (Hrsg.)

Beiträge zum Gründungsworkshop der Fachgruppe Verteilte Künstliche Intelligenz, Saarbrücken, 29. - 30. April 1993
235 Seiten

Note: This document is available for a nominal charge of 25 DM (or 15 US-\$).

D-93-05

Elisabeth André, Winfried Graf, Jochen Heinsohn, Bernhard Nebel, Hans-Jürgen Profitlich, Thomas Rist, Wolfgang Wahlster

PPP: Personalized Plan-Based Presenter

70 pages

D-93-04

DFKI Wissenschaftlich-Technischer Jahresbericht 1992

194 Seiten

D-93-03

Stephan Busemann, Karin Harbusch (Eds.)

DFKI Workshop on Natural Language Systems: Reusability and Modularity - Proceedings
74 pages

D-93-02

Gabriele Schmidt, Frank Peters, Gernod Laufkötter
User Manual of COKAM+

23 pages

D-93-01

Philipp Hanschke, Thom Frühwirth

Terminological Reasoning with Constraint Handling Rules
12 pages

PuK - 95

Beiträge zum 9. Workshop "Planen und Konfigurieren"

Susanne Biundo, Wolfgang Tank (Hrsg.)

D-95-01
Document