KNOWLEDGE BASED SUPPORT WITHIN CONFIGURATION AND DESIGN TASKS

Hans Kleine Büning
FB 17 – Praktische Informatik
Universität-GH- Paderborn
Warburger Str. 100, 33095 Paderborn
Germany

Benno Stein
FB 17 – Praktische Informatik
Universität-GH- Paderborn
Warburger Str. 100, 33095 Paderborn
Germany

ABSTRACT

Manufacturers of complex technical systems are faced with the problem to configure their products according to the customers' demands. Due to the complexity of the products and the partly sophisticated technical background, the process of configuration can turn out to be a difficult problem.

When solving a configuration problem, at least a *technical* problem has to be solved: The configuration program must produce a correctly configured system from its inputs specified by a user. However, we found that realizing an *adequate formulation* of configuration knowledge and configuration problem instances is just as important as solving the configuration problem from its technical standpoint. As a consequence, when developing a configuration system, as much domain specific knowledge as possible has to be considered, i.e. "compiled" into the system.

This paper deals with knowledge based concepts that support configuration and the routine design process respectively, where a central idea is the following: Generate configuration constraints from the visual information specified by a user who describes the problem. In order to illustrate our ideas, the configuration system AKON and *adeco*[1] will be presented.

[1]The system *adeco* has been developed within a cooperative project together with the institute "Meß-, Steuer- & Regelungstechnik", Universität Duisburg, Prof. H. Schwarz. This project is supported by the DFG (Project-No. Kl 529/3-1 "WITA").

INTRODUCTION

We define configuration as a process of creating complex systems out of a finite set of predefined objects. This process may enclose the selection, the assemblage, some adaptation, and perhaps the arrangement of these objects and must lead to a correct configured system. I.e., the system has to meet all given technical constraints and customers' demands.

Rather than checking the function of a configured system through simulation, the configuration process is controlled by heuristics, associative constraints, or simple functional dependencies. Nevertheless, depending on the domain and its particular constraints, each of the above mentioned tasks can turn out to be very complex.

Another class of problems – lets call them "routine design problems" – also deal with a finite set of objects to be configured. But, in contrast to the above, checking the function of such a composed system means to check its *behavior*. An example for this kind of problem is the design of hydraulic systems: Objects like pumps, valves, pipes, are connected together, where each change of a component's parameter or a circuit's structure will lead to a new global behavior of the system.

Until now exist a lot configuration systems, in fact, particular configuration platforms/ shells like PLAKON and COSMOS have been developed[2]. Such configuration shells pro-

[2]An introduction to configuration and associated problems is given in [Brown & Chandrasekaran 89], [Marcus et al. 85] and [McDermott 82]. Detailed information about the configuration platforms PLAKON and COS-

vide certain mechanisms to formulate both knowledge of control and configuration constraints. Using such a platform, a certain configuration problem is tackled by adapting the given mechanisms. Beside these technical aspects of configuration it is an essential job to realize the *adequate formulation* of configuration knowledge and configuration problem instances.

This paper deals with knowledge based concepts that support the configuration and the routine design process respectively. A central idea is the following: Generate as much configuration constraints as possible from the visual information specified by a user who describes the problem. In order to introduce both relevant configuration tasks and our approaches, two systems will be presented:

- The system AKON has been developed to perform the configuration of telecommunication systems and is supposed to support the following tasks: (i) Sales consultation on customers' site and (ii) checking of proposed telecommunication systems. AKON allows a graphical specification of functional dependencies. The emergent dependency network is processed by means of a balance algorithm which is controlled by rules.
- To support the checking and the setting into operation of hydraulic plants we have been developing the system *ardeco*. With this, hydraulic circuits can be manipulated easily in order to examine them with regard to logical faults, stationary behavior and connection constraints. While a problem is specified, i.e., a circuit is drawn, *ardeco* automatically generates the corresponding knowledge bases and all necessary topological information.

CONFIGURING TELECOMMUNICATION SYSTEMS WITH AKON

Configuring telecommunication systems can turn out to be a complex task: Switching boards that interconnect hundreds of extensions, telephones, and other terminals have to be equipped with plug-in-cards. Additionally, it has to be checked whether all customers' demands can be fulfilled. For this, technical components like power supplies, boxes, control units, etc. have to be adapted.

Since technical progress goes on, new components and systems are introduced from time to time. Therefore, the configuration of such systems became a complex task. In order to help the sales personnel to concentrate themselves upon sales consultation, the configuration system AKON has been developed.

MOS can be found in [Cunis et al. 91] and [Heinrich 91].

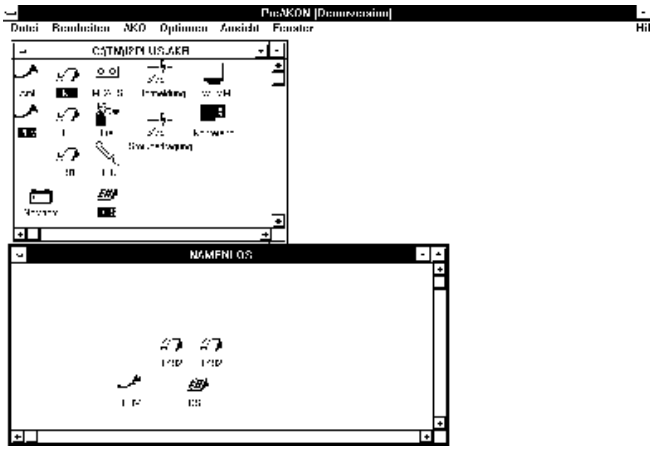


FIGURE 1: AKON'S APPLICATION MODE

The next subsections will introduce essential concepts of this systems.

Working with AKON

When developing AKON, important goals have been

- reduction of the settling-in period to a minimum
- flexibility with regard to maintenance and future extensions
- graphical representation of systems to be configured.

A typical working scenario is the following: While being on a customer's site, a salesperson models the given desired demands by dragging icons from a catalog into the application window (cf. Figure 1).

If desired (and known if at all) connection lines are drawn that indicate relations between extensions and telephones.

At any time, this manual configuration process can be stopped and the automatic configuration process can be invoked. Depending on the modus preset, all necessary components will be added automatically with or without asking the user for confirmation. Figure 2 depicts the result: a ready configured system.

Knowledge Representation and Processing

Basic Data Structures. Within AKON, a technical system is described *model-based* using the concept of functionalities that are provided or required by the components. Given this, configuring a system means selecting components (their type and number) in such a way that all demands can be satisfied. The following simple example may illus-

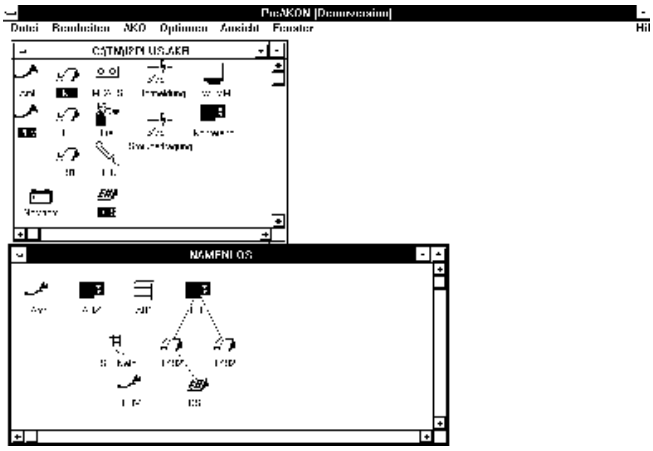


FIGURE 2: A COMPLETED CONFIGURATION

trate the idea. Let's consider we had the following components:

Interface Card A
Supplies:
- 1×V24-interface

Demands:
- 1×plug-in slot
- power (20mA)

Box B
Supplies:
- 4×plug-in slot
- 1×power supply socket

Demands:
- power (150mA)

Power Supply C
Supplies:
- power (500mA)

Demands:
- 1×power supply socket

If we decided to configure a system that provides a V24–interface, new demands come up when selecting component A in order to provide the desired function. These demands can be satisfied by the components B and C which in turn also have to be selected. The next two paragraphs outline, how this configuration process is realized.

The Balance Algorithm. The balance algorithm, supplemented by rules, is an important inference component of AKON (cf. [Kleine Büning et al. 93]). Given the example above, the balance algorithm would principally work as follows:

At first, all initial demands will be entered into a balance (V24-interface in our case). Now, for each open demand, components that help to satisfy the demands will be selected from the knowledge base while the balance is updated with all functionalities associated with the selected components.

To work more efficiently and comfortably, this basic configuration algorithm is refined by the following concepts:

- Local balances. With the aid of local balances one is able to formulate exactly which components have to be connected among themselves.
- Selection knowledge. Often exist several alternative components that satisfy an open demand. In this case, the selection of a component to be the new update-candidate is controlled by heuristics – example: "Select the component which will bring the least number of new demands in its wake."
- Rules. The dependency network of all components' supplies and demands control the configuration process. I.e., the major part of the control knowledge is described *locally*. However, in some cases it is practicable to specify *global* constraints. Within AKON, such portions of knowledge can be formulated using rules.
- Succession knowledge: The quality and efficiency of the configuration- (better: inference-) process is mainly influenced by the determination which open demand should be satisfied next. This kind of global control knowledge is generated automatically: Interpreting the local component descriptions, a dependency network (a directed graph) is built up. The topological sorting of the graph's nodes yields a necessary criterion of succession. This information isn't compelling in all cases and can be refined using additional heuristics, if desired.

Spatial Constraints. The balance algorithm provides the type and the number of the components needed for an entire configured system. Besides this selection problem also a connection problem and an arrangement problem have to be solved:

- Connection Knowledge. Even when the selection problem is solved, it remains difficult to process all connections' information in order to check if all desired connections can be provided by the system. Within AKON, a Least-Commitment-strategy is applied during the process of checking/assigning connections between components. The idea of this strategy is the following: "Put off a decision until it becomes inevitable or the solution space is decreased decisively."
- Arrangement Knowledge: In contrast to the above, the arrangement knowledge is not part of the technical configuration problem. Rather it is intended to improve the user interaction: Since a configured system can be

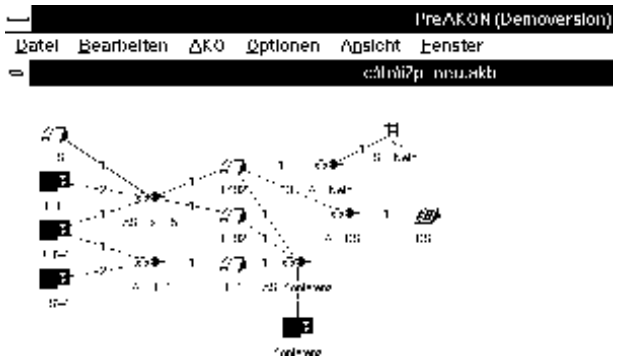


FIGURE 3: KNOWLEDGE REPRESENTATION IN AKON

composed of more than hundred components, a clearly arranged graphical representation (e.g. on the laptop's display) is needed. AKON can generate such an arrangement automatically, using heuristics that describe criterions of clarity. The application of these criterions is controlled by a graph search algorithm of type A* which operationalizes the search for a satisfying arrangement.

Knowledge Acquisition

At the beginning of this section we gave an idea, how to use AKON in its application mode. Consequently, the acquisition of new knowledge is based upon graphical concepts too. Components and functionalities are represented by icons. A functional dependency between a component and a functionality is set up every time an icon is dragged on another.

After such an operation a line appears between the related objects, where the line's dashing style indicates the type of the relationship: A solid line stands for the relation "component-provides-functionality", a dashed one indicates a demand. Figure 3 depicts a part of the graphical representation of a knowledge base in AKON.

Compared to the conventional formulation of rules, this technique of specifying and representing knowledge makes dependencies between components explicitly visible and is much easier to maintain.

SUPPORTING THE CONFIGURATION OF HYDRAULIC CIRCUITS

A hydraulic circuit consists of mechanical, hydraulic, and – increasingly – electronic components. Configuring a hydraulic system is a complex process. Within this process that starts with first design concepts and ends with setting the readily installed system into operation, there exist a lot of tasks that can be supported by a computer (cf. [Eversheim 90]).

In order to assist the checking of a circuit's function we have been developing the prototypic system *ardeco*. A central idea of *ardeco* is the following: The support of tasks within the configuration process should be oriented by the typical working documents of the domain experts. That means here, it must be possible to specify a checking problem graphically. I.e., when manipulating a graphic object, the associated physical and topological knowledge has to be processed also.

To achieve such a deep connection between the graphical representation and the technical specification, a graphical as well as a technical description language is provided by *ardeco*. These two languages can be used to define complex objects which are characterized by a picture structure and a technical behavior description. Furthermore, it is possible (and necessary) to specify dependencies between the syntactical information (the picture structures) and the semantics (the technical description) of an object.

Working with *ardeco*

Within *ardeco*'s application mode, a user selects objects from a catalog and arranges them in order to build a new hydraulic system. When drawing lines between the components' gates, the appropriate types of connection lines will be selected and instantiated. Within this connection, it will be checked if both gates incident to a connection line are of the same type. All necessary information concerning the topology is generated as well. The Figure 4 shows *ardeco*'s application screen.

Within a hydraulic system, all parameters belonging to components (e.g. hydraulic resistances, valve positions) and connections (pressure values, forces etc.) can be predefined or changed. For arbitrary parts of a network, constraint propagation can be invoked any time. The derived values and possible faults can be displayed within the circuit (cf. Figure 5).

While checking the circuit illustrated above, an inconsistency has been detected by the local propagation mechanism: The position of one valve contradicts some demands specified at a cylinder.

Knowledge Representation and Processing

Modeling Topological and Stationary Knowledge.

ardeco distinguishes between elements, gates, connections and sources (or sinks resp.). These elements will be used to describe a systems topology, causal dependencies, and the

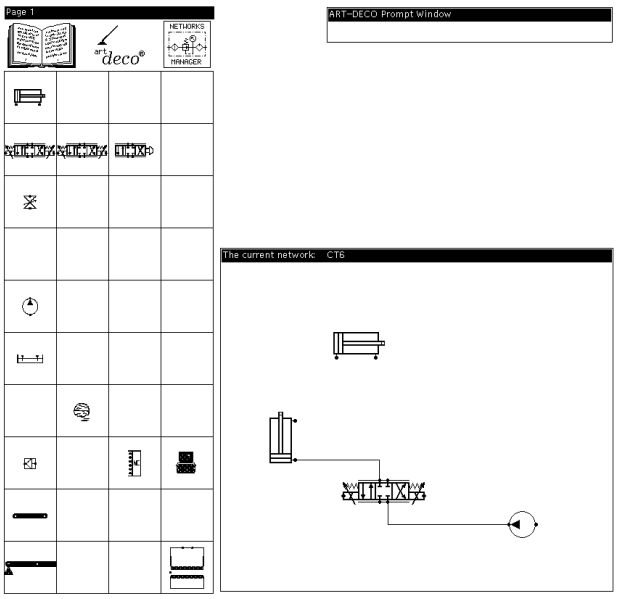


FIGURE 4: *art deco*'s APPLICATION MODE

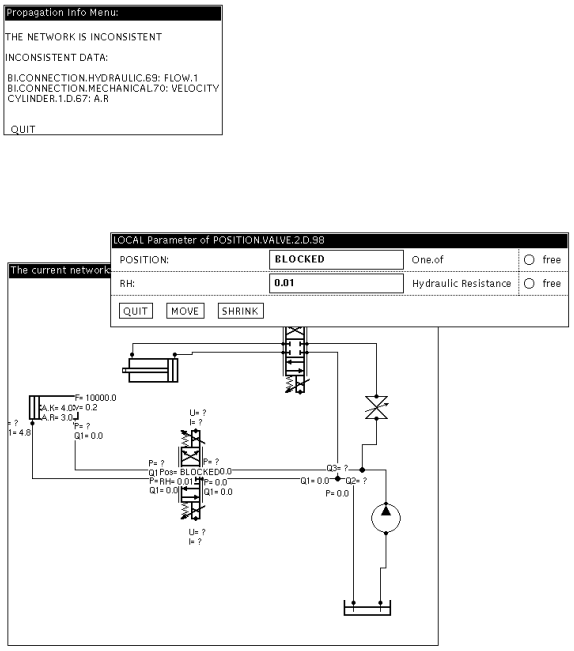
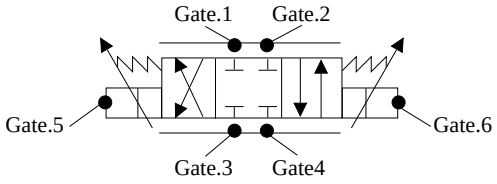


FIGURE 5: CHECKING A GIVEN CIRCUIT

flow of information within a technical system (cf. [Kleine Büning & Stein 93]).

The medium to be transported (e.g. electric current, fluids) is specified by sources/sinks of the appropriate type. A component of a real system is described by an element e that defines constraints between all *accessible* sources/sinks of the medium (cf. Figure 6).

To describe the technical behavior of components, i.e. their constraints, a formal behavior language has been developed. This language provides both a qualitative level of description and a quantitative one. The example below depicts a valve and a part of its behavior description:




```
flow[GATE.1] =
 (CASE
  WHEN position[SELF] IS CROSSED: - flow[GATE.4]
  WHEN position[SELF] IS PARALLEL: - flow[GATE.3]
  OTHERWISE: 0)

pressure[GATE.4] = pressure[GATE.1]
 - SIGNUM(flow[GATE.1]) · flow[GATE.1]^2 · rh[SELF]
```

Processing Stationary Knowledge. Behavior descriptions are worked off using the method of constraint propagation (cf. [Gosling 83]). We realized a constraint propagation approach to keep the processing of behavior flexible and causal: Dependencies can be traced back and explanation that base upon local derivations can be generated. If local propagation comes to an end, other mechanisms are invoked in order to derive further information:

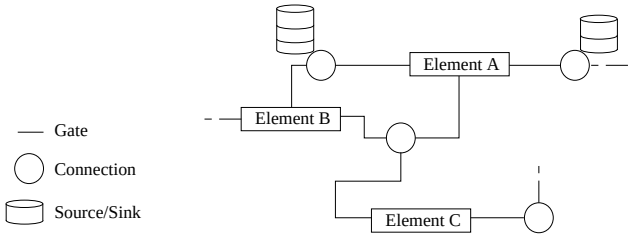


FIGURE 6: DATA STRUCTURE ELEMENTS IN *art deco*

- Reducing structural complexity. This module investigates a circuit and introduces for series/parallel structures additional constraints. This results in a new network that can be worked off using local propagation.
- Numerical methods. Linear equations can be extracted from constraints and processed using algebraic transformation. The solution of non-linear equation systems is part of the current work.
- User inquiry. The user can be requested to specify further information. This module is part of the current work and will employ knowledge-based techniques in order to implement heuristics for parameter selection.

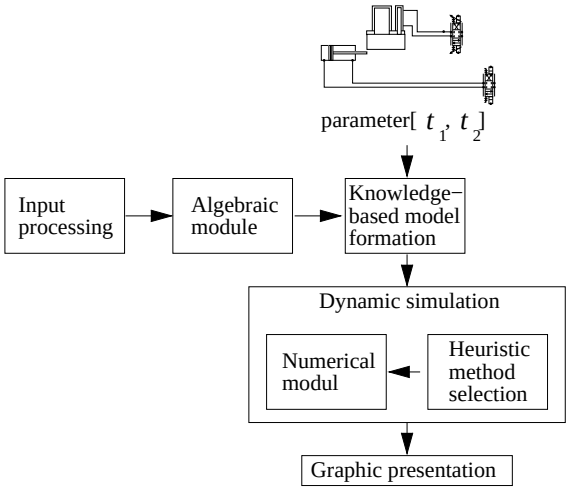


FIGURE 7: THE DYNAMIC CHECKING MODULE

Examinating Knowledge of Dynamic Function. In *ardeco* the static examination of a mechatronic system is followed by a dynamic examination of function. The description of the dynamic behavior of a system involves quantities known from statics, in particular the information responsible for the dynamic behavior. The Figure 7 shows the structure of the dynamic module that is currently being developed.

In the following the function and the interplay of the submodules is described briefly:

Processing the inputs takes for every component not yet known to the system a description of the dynamic behavior in form of order n differential equations which are to be provided by the user. The algebraic submodule reduces for each new dynamically specified component the order n differential equations to a system of differential equations which is of order 1. The submodule of model formation has as additional input-parameters the user selected subnetwork and the time interval needed for the dynamic simulation. Along with the known, different levels of model depths of the dynamic component description, a knowledge driven graph algorithm yields a subcircuit of the system. During the dynamic simulation the dynamic function examination for that part of the system which was determined by the model formation process is carried out. The selection of the appropriate numerical method is controlled by the module of method selection. The respective choice takes place in a context sensitive manner, i.e. for each phase of integration the method that suits best according to heuristic criteria is determined and applied.

Diagnosing Fault Behavior. Diagnosing faults in hydraulic circuits is part of the current research. Roughly spoken, diagnosing a hydraulic system means to explain observations of faulty behavior (cf. [deKleer & Williams 89]).

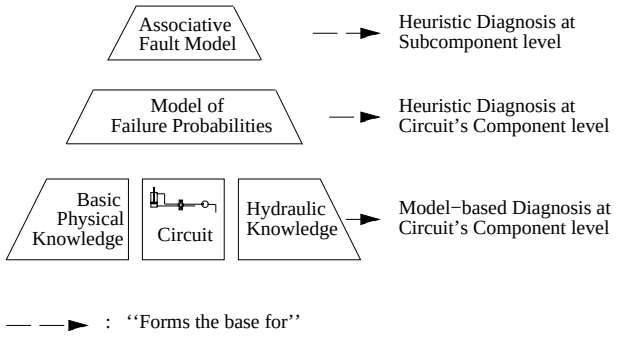


FIGURE 8: DIFFERENT LEVELS OF DIAGNOSIS

E.g., if an expert tries to explain the observation that the cylinder's piston drives out too slowly, he starts his diagnosis ignoring the exact relationships between the piston plane, the piston velocity and the flow value. Rather, he follows up the piping's structure in order to determine a set of components that contains the misbehaving one. Since he knows about the components' failure probability, he is able to make a well-founded assumption as to which component out of the determined set could be defect. At this point, the expert will refine the diagnosis using his knowledge about the malfunctioning *inside* of components.

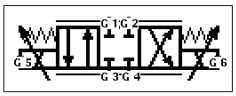
When analyzing this scenario, we identify three kinds of diagnosis knowledge where each portion of knowledge contains a diagnosis problem that needs to be solved in its own manner. Our idea is, rather than looking for an overall strategy, to identify different diagnosis problems that can be solved independently. The Figure 8 depicts this concept graphically.

Knowledge Acquisition

A general idea of *ardeco* is to provide equivalent concepts for "normal" user interaction as well as for acquiring additional knowledge. When the acquisition mode is activated, new classes of components can be defined where existing classes might serve as templates.

ardeco provides a module that enables (i) the generation of structured pictures and (ii) the specification of dependencies between picture hotspots and a component's behavior description. Interpreting this information, *ardeco* automatically generates the according data structures and constraint information for both the knowledge processing and the user interaction. The Figure 9 shows a part of the behavior descriptions that are defined for a valve.

Introducing new properties within such a behavior description will cause *ardeco* to update the according component definition automatically.



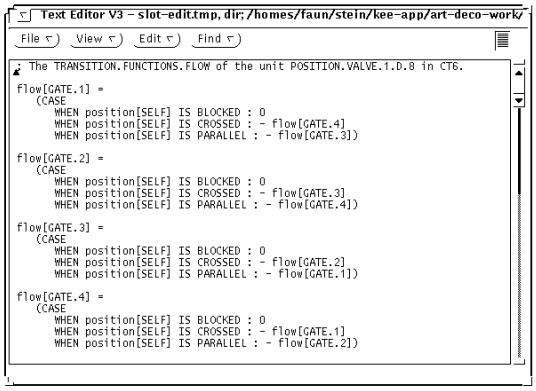


FIGURE 9: SPECIFYING COMPONENT BEHAVIOR IN *artdeco*

SUMMARY

Configuration systems can support the configuration and the routine design process successfully. Both, a configuration system and a routine design system were presented:

The system AKON is intended to support the sales personnel on the customer's site. Moreover, AKON realizes a concept to extend and maintain the knowledge base graphically. To perform the configuration process efficiently, knowledge based technics like a balance algorithm, the Least-Commitment-strategy and graph search algorithms are employed.

When configuring a hydraulic system, an engineer is faced with a lot of problems. The most important ones are: checking of the stationary behavior, dynamic examination of function, and detecting faults. Each of these problems needs a deep model to become operationalized on a computer. Within this connection, we have developed concepts for the representation and the processing of hydraulic knowledge. These ideas have been realized within the prototypic system *artdeco*.

The systems AKON and *artdeco* have the following key ideas in common: (i) *local description* of a component's functionalities or its behavior, and (ii) *graphical specification* of a configuration problem instance.

Generally spoken, our philosophy is to support the processing of "standard" tasks in an adequate way. This can be achieved by modeling the typical working documents of the domain experts on a computer.

REFERENCES

Brown, J. S., Chandrasekaran, B., 1989, "Design Problem Solving," Pitman.

Cunis, R., Günter, A., Syska, I., 1991, "Das PLAKON Buch – Ein Expertensystemkern für Planungs- und Konfigurierungsaufgaben in technischen Domänen," Springer Verlag, Berlin.

Eversheim, W., 1990, "Inbetriebnahme komplexer Maschinen und Anlagen," VDI-Verlag, Düsseldorf.

Gosling, J., 1983, "Algebraic Constraints," Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA. 15213.

Heinrich, M., 1991, "Ressourcenorientierte Modellierung als Basis modularer technischer Systeme," *Proceedings, 5th Workshop "Planning and Configuring"*, LKI-M-1/91, Hamburg.

Kleine Büning, H., Curatolo, D., Stein, B., 1993, "AKON – Reference Manual," Universität-GH Paderborn.

Kleine Büning, H., Stein, B., 1993, "Ein wissensbasiertes System zur Inbetriebnahmeunterstützung," *Internal Report*, Universität Paderborn, FB17 – Praktische Informatik.

deKleer, J., Williams, B.C., 1989, "Diagnoses with Behavioral Models," *Proceedings, IJCAI 89*, p.1324.

Marcus, S., McDermott, J., Wang, T., 1985, "Knowledge Acquisition for Constructive Systems," *Proceedings, 9th IJCAI*, Los Angeles.

McDermott, J., 1982, "R1: A Rule-Based Configurer of Computer Systems," *Artificial Intelligence 19*, pp.39-88.