Meeting Re-use Requirements of Real-life Diagnosis Applications

Thomas Guckenbiehl[1], Heiko Milde[2], Bernd Neumann[2], and Peter Struss[3]

[1] Fraunhofer-Institut IITB, Fraunhoferstr. 1, 76131 Karlsruhe, Germany
guc@iitb.fhg.de
[2] Laboratory for Artificial Intelligence, University of Hamburg,
Vogt-Koelln-Str. 30, 22527 Hamburg, Germany
milde@kogs.informatik.uni-hamburg.de
neumann@informatik.uni-hamburg.de
[3] Technische Universität München, Department of Computer Science,
Orleansstr. 34, 81667 Munich, Germany
struss@in.tum.de

Abstract. This report addresses re-use issues in computer-based diagnosis. It is shown that in order to obtain re-usable components it is useful to categorize the knowledge and software for a diagnosis system along two dimensions, generality and genericity. Several new contributions to diagnosis technology are presented which illustrate different re-use categories and show the benefits of improved re-usability. The contributions pertain to different tasks related to the diagnosis of real-life systems of diverse domains: FMEA, workshop diagnosis, generating diagnosis manuals, generating fault trees and operator assistance in post mortem diagnosis. The work has been performed by three research groups involved in the joint research project INDIA (Intelligent Diagnosis in Industrial Application).

1 Introduction

Efficient development of diagnosis equipment, use of available resources, and re-usability of software components are the main advantages which industry expects from innovative diagnosis technology. This has been the experience of the authors in several diagnosis projects with industrial partners, in particular in the joint research project INDIA (Intelligent Diagnosis in Industrial Applications). In this project, three teams, each consisting of a research institute, a software supplier, and an industrial production company, have joined to apply model-based diagnosis technology to real-life diagnosis problems and pave the way for successful applications elsewhere. The particular diagnosis problems provided by the industrial partners represented an interesting subset of industrial diagnosis applications. One application area is on-line diagnosis of automotive equipment, another one is off-line diagnosis support for transport vehicles, the third one deals with operator assistance in post mortem diagnosis of machinery.

It is well-known that model-based methods promise applications with attractive problem-solving capabilities and significant economical advantages. Reviewing the attractive features of model-based diagnosis, the main benefits are connected with the

compositionality and transparency of the model, from which diagnosis knowledge can be generated. Compositionality bears the potential for re-using components, building component libraries and inheritance hierarchies, alleviating version control and easing modifications. The transparency of component-based behavior descriptions may add further benefits, including complexity management, exploitation of information from the design phase, and a large degree of compatibility with other life-cycle product data including documentation. Hence, important benefits can be gained from model-based diagnosis technology.

All this can be stated without reference to a particular diagnosis procedure. In fact, one of the insights which this report wants to convey is about possible (re-)uses of model-based techniques beyond the diagnosis procedures which are traditionally associated with model-based diagnosis.

We believe that real-life engineering applications and re-use of model-based techniques may become possible at a large scale provided the characteristics of today's diagnosis practise are taken into consideration:

- First of all, many producers of technical systems provide only limited diagnosis support of their products to begin with. There are only few large market segments where producers develop sophisticated diagnosis support (e.g. the automotive and aircraft industry). This is changing, however, as the cost of maintenance personnel becomes more important and improved service is required to remain competitive. But in many cases, the initial demand for diagnosis support will be quite modest.
- A second point to observe is the industrial tradition of employing decision trees or fault trees. These techniques have been developed from a maintenance rather than from a design perspective. Since traditionally, diagnosis matters are a concern mainly to the service division of a company, not to the design division. However, as technical systems become larger and more complicated, the design of decision trees becomes more demanding and problems arise. Furthermore, frequent product changes cause excessive costs for the maintenance of such diagnosis equipment. Hence there is growing awareness of the need for re-usability of diagnostic information.
- A third problem to cope with is the natural desire of industry to perform changes in small steps. The introduction of model-based reasoning for complete automation of diagnosis is often perceived as too different from the traditional ways of doing diagnosis. Existing know-how would become worthless and new know-how would have to be acquired. As noted above, the organizational structure would be affected, with diagnosis tasks shifting from the service to the design division.

This led us to focus our diagnosis research on ways to exploit the advantages of model-based diagnosis techniques compatible (to some degree) with existing industrial traditions and requirements.

In Section 2 we identify several different kinds of re-use and propose a classification scheme which can be generally applied to re-use phenomena in complex industrial environments. Section 3 presents new diagnosis methods for three application domains, illustrating different ways to improve re-usability. It is shown that computer-based behavior models can be used for various real-life diagnosis tasks and thus play a key part for improving re-usability of diagnosis components.

2 A Categorization for Re-use of Knowledge and Software

2.1 Different Tasks - Shared Knowledge and Skills

During the life cycle of a product, a broad range of different tasks are performed, from conceptual design via production planning to maintenance. This includes various forms of fault analysis and diagnosis, such as failure-modes-and-effects analysis (FMEA), design for diagnosability, test generation and testing, creation of on-board diagnostics, production of diagnosis guidelines, workshop diagnosis, repair and reconfiguration. Despite their different goals and conditions, many of these tasks are based on similar knowledge, information and skills which are (or ought to be) exchanged between the people and departments performing these tasks. Identifying and analyzing these shared elements is a starting point for designing computational tools in a manner that exploits re-use of these elements in an optimal way. In a knowledge-based approach, such elements are both collections of represented knowledge fragments and software components.

Obviously, information and knowledge about the subject, the technical system itself, is central to all tasks, and model-based systems reflect this explicitly. It comprises the *device structure* (the blue print) and knowledge about the *behavior of its components*, both under *normal operation* and in the presence of some *malfunction.* Furthermore, there is knowledge about the *function*, i.e. the *purpose* of a system or a particular *role* to be fulfilled by a component or subsystem. For planning and performing tests and repairs, information about the *assembly* of parts, *accessibility* of probing points, *equipment* required and the *cost* associated with the actions are important.

A fundamental reasoning skill required to exploit this knowledge (and, hence, a candidate for software procedures) is the capability to *infer global system behavior* from the structure of the device and the behavior of its components. The various diagnostic tasks require procedures for *relating observations* (“symptoms”) *to hypotheses of faults*, whilst testing is based on *determining actions* that are likely to *lead to observable distinctions between different behaviors.*

Already this incomplete list of elements of knowledge and software gives some hints on their potential re-use, but it is still too coarse-grained. Applying a classification scheme which is part of a methodology for developing models and model-based systems (see [7]), we take a closer look at this list.

2.2 Classification

The first distinction is made according to the generality of the respective entities of knowledge, information, or methods:

- *physical principles*, which are fundamental and valid for all systems (devices) in a particular domain and regardless of the special task (sometimes called domain theory) e.g. Ohm's Law,
- *device-specific* (or device-type-specific) entities, i.e. knowledge and information characterizing a particular system, for instance the value of the parameter resistance of a part of the device,
- *task-specific* entities: knowledge, information, and algorithms for solving a certain type of problem, for example an algorithm for computing a decision tree based on a

set of behavior models and possible measurement points.

Obviously, reflecting this distinction in the design and implementation of systems helps to re-use the physical principles for several devices and in different tasks. Of course, exploiting device information in different tasks and applying a problem solver to various devices is also a desire. This requires a systematic separation of “How” and “What” (the problem solver and its subject, that is), a principle which is central to knowledge-based systems and to model-based systems, in particular.

Our experience shows that a structuring of knowledge and information and design of software components cannot be optimal w.r.t. re-use unless at least one more, orthogonal, distinction is made. We need to separate

- *generic* knowledge and methods from
- a representation and treatment of *pragmatic* elements.

In our context, the former are related to representation of and reasoning about the abstract behavior of a system according to the “laws” of physics, while the latter reflect the particular physical implementation of a system and the preconditions imposed on a special task by the real environment. To illustrate this in the context of our work, the circuits in a car subsystem in two vehicles may have the same structure in the sense of the blueprint and include components of the same type, nevertheless differ fundamentally in how they are laid out and installed in the vehicles. This means, their behavior models are the same and will, for instance, lead to the same diagnostic candidates. However, the respective test sequences to be generated have to be different, because the accessibility of probing points varies significantly. While this is determined by the device, the actual costs of a test sequence may, furthermore, depend on the pragmatic context the task has to be performed in, for instance the equipment available in a certain type of workshop.

With respect to the scope of our work in model-based systems, we obtain the classification displayed in Table 1. It is the simplest one, in fact it is somewhat oversimplified, as already indicated by overlap between device- and task-specific pragmatic elements: the actual actions that have to be performed to carry out a test may be determined by both the physical device and the contextual constraints on a task. More caveats are discussed below. Nevertheless, an analysis along these lines already provides useful criteria for the design of knowledge representation schemes, data base structures, and modularization of software under the aspect of re-use.

2.3 Discussion

Although the distinctions made in the above analysis are fairly obvious, they are not necessarily present in current practice of product documentation, actual work processes, and traditional software systems. In fact, the scheme provides a way for analyzing the inherent limitations of different approaches to software tools for the kind of tasks considered here. Many tools for engineering tasks do not provide a clear distinction along the horizontal axis at all turning them into unique solutions, and even if they are model-based, they do not provide a method for composing the device model by re-using physical principles. Fault-tree-based diagnosis offers a general inference mechanism for processing the trees, but the trees inseparably merge component knowledge, device structure, and task knowledge including the pragmatic aspects. This is why this technol-

ogy is obsolete, unless a way is provided to generate the trees from first principles as described in Section 3. But limitations become evident also for several AI techniques. Case-based systems, for instance, do not represent the principled layer and do not separate device-specific from task-specific knowledge, which also holds for neural-network-based solutions.

Table 1. Classifying knowledge and software elements

<table>
<tr><th></th><th>generic</th><th>pragmatics</th></tr>
<tr><td>physical principles</td><td>behavior model fragments
(component library)</td><td></td></tr>
<tr><td rowspan="2">devices</td><td rowspan="2">structure
parameters
behavior (intended and faulty)
state</td><td>function
criticality
tolerance
assembly / replaceable units
measurement points</td></tr>
<tr><td rowspan="2">actions
• measurements
• disassembly
equipment
cost</td></tr>
<tr><td>tasks</td><td>model-based inference mechanisms
• model composition
• behavior prediction
• model-based diagnosis
• model-based test generation</td><td>inference mechanism
reflecting pragmatics
• cost-oriented test proposal
/ test plan generation</td></tr>
</table>

We would like to emphasize that the discussion clearly shows that the problem of re-use cannot be solved by means of general software engineering techniques, but requires an analysis of the problem domain at the knowledge level and then, of course, appropriate software architectures and knowledge representation facilities. This is why knowledge-based systems, Artificial Intelligence, and, in our application domain, model-based reasoning techniques promise significant progress and superiority to traditional approaches.

As a side remark should be stated that much of what we discussed does not only apply to re-use of knowledge through computer systems, but also to re-use of knowledge by humans. If knowledge is not structured and indexed appropriately, for example by mixing device and task knowledge, it is likely to be useless to a human expert who is working in a different context. This is why the solutions advocated here are also a contribution to the area of knowledge management which receives more and more attention.

Despite the fact that we will not easily find the necessary distinctions present in current practice, the analysis is not an academic one. Rather, it has a significant, if not decisive, impact on the competence and flexibility of software systems and on the cost of producing and maintaining them. Taking the classification into consideration when designing knowledge bases and software solutions will already have a tremendous effect. In our case studies, not only sharing of the component library and the device model

across the three tasks could be demonstrated, also software components, such as model composition and behavior prediction, are re-used for the prototypes.

However, we have to mention that the schema presented above needs extensions, refinements, and even further research. In particular, although we may be able to identify the general principles and generate a device model from them, the result may not be appropriate in a particular context:

- The *task* may influence the granularity of the *behavior* model. While a qualitative model may do for the early phase of design and also for diagnosis, the final design needs a numerical model.
- *Pragmatic aspects*, for instance the replaceable units, can have an impact on the appropriate granularity of the *structural and behavioral model*. Another example is the interaction between the function (purpose) of a device and its behavior model: component models, say of pipes and valves, have to include transportation of oxygen, carbon oxide, etc. in the intake and exhaust in order to analyze problems with emissions in vehicles, whereas for other pneumatic or hydraulic subsystems (for instance, those controlling valves), only pressure matters.

As an answer to such issues, the transformation of generic, compositional models under task-dependent and pragmatic criteria is part of our work (see e.g. [1]).

3 Towards Re-use in Practical Applications

In this section we describe work in three application areas which illustrates re-use of diagnostic components. Note that this work has been driven by real industrial requirements and not by academical interests to demonstrate re-use. Hence also other aspects besides re-use will be touched.

3.1 Three Automotive Applications

In collaboration with Robert Bosch GmbH as a major supplier of mechatronic car subsystems, the Model-based Systems and Qualitative Modeling Group (MQM) at the Technical University of Munich works on three prototypes that support different tasks related to fault analysis during the life cycle of a product:

Failure-Mode-and-Effects Analysis (FMEA). This task is performed during the design phase of a device. Its goal is to analyze the effects of component failures in a system implemented according to the respective design. The focus is on assessment of the criticality of such effects, i.e. how severe or dangerous the resulting disturbance of the functionality is in objective or subjective terms (e.g. inconvenience for the driver, environmental impact, risk of hazards). In addition, the probability of the fault and its detectability is considered. Based on this analysis, revisions of the design may be suggested. Because of the safety and environmental aspects, it has to be as "complete" as possible, not only w.r.t. the faults considered, but also in the sense that all possible effects under all relevant circumstances (driving conditions of a vehicle, states of interacting subsystems, etc.) have to be detected.

Workshop Diagnosis. The diagnostic task in the repair workshop starts from a set of initial symptoms which are either customer complaints or trouble codes generated and stored on-board by one of the Electronic Control Units (ECU) responsible for various

subsystems of the vehicle. Except for the obvious cases, some investigations, tests, and measurements have to be carried out in order to localize and remove the fault, usually by replacement of components.

Generation of Diagnosis Manuals. The mechanic in the repair workshop is educated or guided by diagnosis manuals produced and distributed by the corporate service department (on paper, CD-ROM, or via a network). Here, engineers compile various kinds of information (tables, figures, text) which is required or useful for carrying out the diagnosis. Such documents have to be produced for each variant of the various subsystems and specific to a particular make, type, and special equipment of a vehicle, a broad set of issues for a supplier. The core of each document is a set of test plans that guide fault localization starting from classified customer complaints or trouble codes of the ECU.

In practice, these tasks are usually not extremely difficult to perform by an expert. However, they can be time-consuming since they have to be carried out for each specific instance of a general device type which includes collection of all the information specific to this instance. This situation of routine work applied to a large set of variants justifies computer support to be developed. And because knowledge about physical devices is the key to solving the tasks, model-based systems offer a perspective.

With the background discussed in Section 2, a major goal in our work was to demonstrate the possibility of re-use of knowledge and software modules and, particularly, of grounding the tools on the same model libraries. As depicted in Figure 1, the software basis of the three tools can share several software components (exploiting the modeling and diagnosis environment RAZ'R [6]):

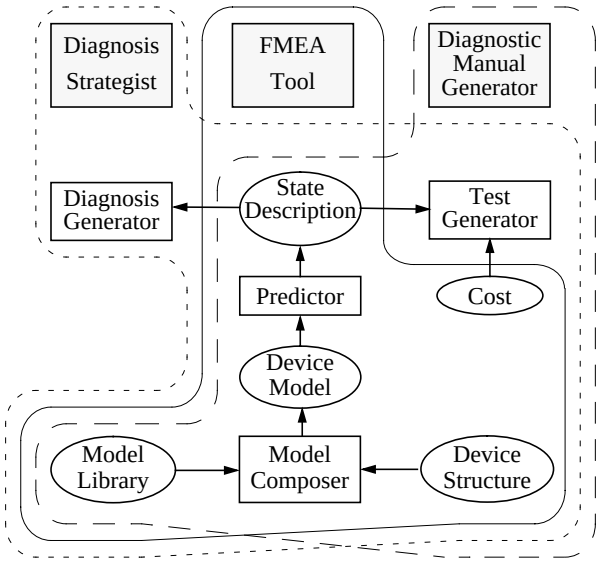

Fig. 1. Tools for workshop diagnosis, FMEA and diagnosis manual generation

- The *FMEA-Tool* is based on model composition and behavior prediction from which the task specific part has to extract the information that is critical to the function (see [8]).
- *Workshop diagnosis* additionally needs the generation of diagnostic candidates and could exploit a test generation (or measurement proposal) module. The latter is not included in our prototype which is intended to only demonstrate the possibility of interactive diagnosis based on customer complaints and trouble codes (see [9]).
- The tool for the *generation of diagnosis manuals* adds an explicit representation of car subsystems and automatic test generation to the classical functionality of an authoring system.

3.2 Model-based Fault Tree Generation

Computer-based systems for off-line diagnosis constitute the main diagnostic equipment for more than 100.000 forklifts made by the german company STILL GmbH. The diagnostic knowledge provided by these systems is contained in fault trees consisting of fault sets as nodes and tests along the edges. To support the diagnosis task of service engineers, the system guides through the fault tree by asking for tests until the possible faults have been narrowed down to a single one.

Due to the complexity of the electrical circuits employed in forklifts, fault trees may consist of more than 5000 nodes. When forklift model ranges are modified or new model ranges are released, fault trees are manually generated or adapted by service engineers who apply detailed expert knowledge concerning faults and their effects. Obviously, this practice is costly and quality management is difficult. Hence, there is a need for computer methods to systematically support modifications and re-use of components of diagnosis systems.

Model-based fault-tree generation has been developed to improve this situation. The idea is to generate diagnostic knowledge by model-based computer simulation and then use this knowledge to automatically generate the fault-tree structure for the diagnosis system. Figure 2 shows the main steps.

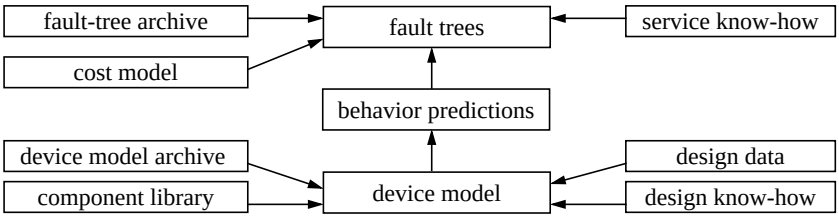

Fig. 2. Basic concepts of model-based fault-tree generation

The first step is to model a device using design data and expert knowledge concerning intended device behavior, expected faults, available measurements etc. Modeling is supported by a component library and a device model archive. As a second step, correct and faulty behavior is computed based on this model, resulting in a potentially large table of behavior predictions. The third step is to build a fault tree by grouping faults into

fault sets and selecting tests. This step is supported by a fault-tree archive and a cost model for the tests.

Behavior prediction and fault-tree formation involve several innovative procedures which can only be briefly sketched in the following. The procedures have been implemented in the prototypical system MAD (Modeling, Analyzing and Diagnosing).

For behavior prediction - which is restricted to steady-state behavior - aggregates are modeled by idealized voltage sources, consumers, conductors and barriers, or their equivalents for domains other than the electrical. Also integrated circuits and software-in-the-loop may be included based on steady-state models. A qualitative calculus is employed - similar in scope to the CIRQ system of [3] - where currents and voltages are described by qualitative values and deviations from reference values. To improve the accuracy of qualitative prediction, a propagation technique has been developed which prevents spurious solutions.

Fault trees may be generated automatically from fault-symptom tables and a cost model, using the A* algorithm to optimize average diagnosis cost. In order to permit re-use and manual adaptation, MAD offers editing operations, such as moving a certain fault from one fault set to another and recomputing the corresponding tests. Due to model-based prediction, fault trees are correct and complete with respect to the underlying device model and the faults (or fault combinations) considered in the fault-symptom table.

We have evaluated model-based fault-tree generation in the STILL application scenario and found that using the modeling techniques of MAD, more than 90% of the faults of the current handcrafted diagnosis system can be handled successfully.

3.3 Searching for Failing Steps of a Technical Process

The approaches to model-based diagnosis discussed in this paper, so far, are adopting the view of “searching for faults of components”. However, for many technical systems an alternative approach seems preferable, which searches for failures in steps or phases of a technical process rather than components, at least in the initial phases of diagnosis. The work of Fraunhofer IITB in INDIA has demonstrated that the standard algorithms for consistency based diagnosis may be used for this search. The approach relies on models, which structure a process into a directed graph of steps or phases as well as on models of successful completion and failure of such steps.

Our approach is preferable if domain experts describe the overall behavior of a technical system as interconnected steps and use such mental models in focussing diagnosis (“Which step failed?”). The duration of a step in a certain system behavior is not necessarily fixed but may depend on certain events, like variables reaching certain values, time-outs or operator commands. Such systems are particularly common in the process and manufacturing industries.

Re-use of temporally structured models is crucial in process industries, since the essential know-how of many companies is condensed into a library of procedures. New procedures are usually developed by modifying successful procedures from the library. Re-use of step models in process technology is supported by a strong tendency to describe and classify steps on an abstract level (cf. [2]).

Our approach employs standard consistency-based diagnosis by justifying every in-

ference from a structured model with correctness of the structure, every inference from a model of a successful step with success of the step and every inference from a model of a failed step with the proposition that the step failed in the corresponding way. A diagnosis then states that one or more steps have failed. Such diagnoses may be refined by analyzing structured models for the suspected steps. Standard correctness and completeness results of consistency-based diagnosis are still valid for temporally structured models, and the quality of diagnosis still depends on the precision of the models.

Our guiding example in INDIA is the chemical distributor (CHD) from THEN GmbH, a system to distribute liquids in a dye house. Domain experts describe a typical task processed by the CHD as measuring out certain amounts of certain chemicals, transporting the mixture to the requesting dying machine and finally rinsing the pipes involved with water. These steps can be used to construct a temporally structured model of the overall behavior of the CHD (Figure 3).

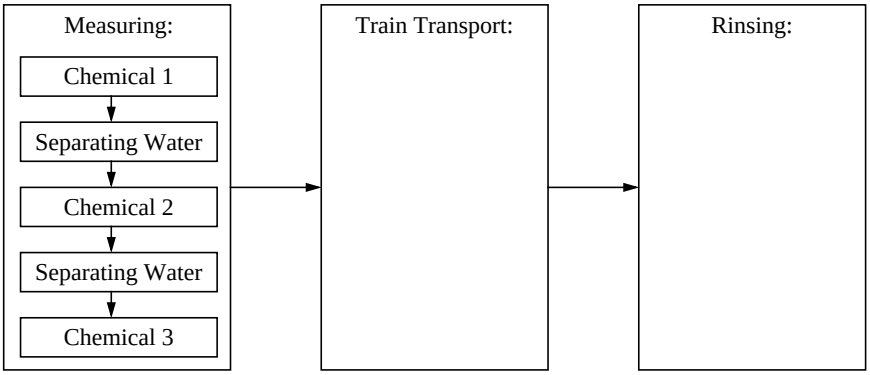

Fig. 3. Composing a train of chemicals and sending it to a certain dyeing machine

In developing models for our approach one can employ the same techniques for information re-use which were described in Section 2 with respect to component models. For instance, analysis of the control flow diagram of the control program of the CHD shows that measuring out some chemical or the separating water, transport of the mixture and rinsing may be viewed as instances of a general step type depicted in Figure 4, and hence the same set of models may be used to describe these steps.

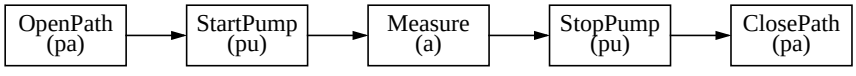

Fig. 4. Transporting amount a with pump pu along path pa

4. Conclusions

We have addressed re-usability in connection with several diagnosis tasks in an industrial environment. From our experience, re-usability is one of the prominent features model-based techniques have to offer in this domain.We have shown that in order to

improve re-usability it is useful to categorize the knowledge and software for a diagnosis system along two dimensions, generality and genericity.

This has been illustrated by presenting innovative diagnosis approaches for different tasks in diverse domains. In the automotive domain, several components could be identified which are re-usable across different tasks. The work on model-based fault-tree generation demonstrates improved re-usability by separating device-specific from task-related knowledge and generic from pragmatic aspects. It is interesting that this approach combines advantages of model-based diagnosis systems with the familiarity of traditional fault-tree systems.

Finally, we have demonstrated that the re-usable building blocks for modeling the behavior of devices need not represent components. Consistency-based diagnosis techniques remain applicable if a technical process is described as a series of steps constrained by certain propositions for correct or faulty behavior.

Acknowledgments

We would like to thank our colleagues at IITB, LKI and TUM as well as our partners in INDIA for their valuable contributions. This work has been supported by the Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie (BMBF) under the grant 01 IN 509 D 0, INDIA.

References

1. Heller, U., Struss, P.: Transformation of Qualitative Dynamic Models - Application in Hydro-Ecology, in: 10th International Workshop on Qualitative Reasoning (QR-96), Fallen Leaf Lake, Ca (AAAI Press), 1996.
2. Hemming, W.: Verfahrenstechnik. Würzburg. 7. Auflage 1993. (in german)
3. Lee, M. H., Ormsby, A. R. T.: Qualitative Modeling of the Effects of Electrical Circuit Faults, Artificial Intelligence in Engineering, vol. 8, pp. 293-300, 1993.
4. Mauss, J.: Analyse kompositionaler Modelle durch Serien-Parallel-Stern Aggregation, DISKI 183, Dissertationen zur Künstlichen Intelligenz, 1998. (in german)
5. Milde, H., Hotz, L., Möller, R., Neumann, B.: Resistive Networks Revisited: Exploitation of Network Structures and Qualitative Reasoning about Deviations is the Key, in: Proc. DX-97, 8th International Workshop on Principles of Diagnosis, 1997.
6. http://www.occm.de/
7. Struss, P.: Contributions to a Methodology of Model-based Systems for Diagnostic Tasks. In preparation, 1999.
8. Struss, P., Malik, A., Sachenbacher, M.: Qualitative Modeling is the Key to Automated Diagnosis. 13th World Congress of IFAC, San Francisco, CA, USA, Pergamon, 1996.
9. Struss, P., Sachenbacher, M., Dummert, F.: Diagnosing a Dynamic System with (almost) no Observations. Workshop Notes of the 11th International Workshop on Qualitative Reasoning, (QR-97) Cortona, Italy, June 3-6, pp. 193-201, 1997.