Meeting Re-use Requirements of Real-life Diagnosis Applications

Thomas Guckenbiehl
Fraunhofer-Institut IITB
Fraunhoferstr. 1
76131 Karlsruhe, Germany
guc@iitb.fhg.de
Phone: +49 (0)721 6091-214
Fax: +49 (0)721 6091-413

Heiko Milde
Laboratory for Artificial Intelligence
University of Hamburg
Vogt-Koelln-Str. 30, 22527 Hamburg, Germany
milde@kogs.informatik.uni-hamburg.de
Phone: +49 (0)40 5494-2606
Fax: +49 (0)40 5494-2527

Bernd Neumann
Laboratory for Artifical Intelligence
University of Hamburg
Vogt-Koelln-Str. 30, 22527 Hamburg, Germany
neumann@informatik.uni-hamburg.de
Phone: +49 (0)40 5494-2451
Fax: +49 (0)40 5494-2527

Peter Struss
Technische Universität München
Department of Computer Science
Orleansstr. 34, D-81667 Munich, Germany
struss@in.tum.de
Phone: +49 (0)89 48095-123
Fax: +49 (0)89 48095-203

Abstract

This report describes application-oriented work on diagnosis of three research groups involved in the joint research project INDIA (Intelligent Diagnosis in Industrial Application). Several different tasks related to the diagnosis of real-life systems of diverse domains are addressed: FMEA, workshop diagnosis, generating diagnosis manuals, generating fault trees and operator assistance in post mortem 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. The MAD system for model-based fault-tree generation is presented which exploits several re-usable components for an automatic compilation of fault trees. Finally it is demonstrated, that the techniques for re-using existent information in modeling and diagnosis are not only applicable to models of technical components but also to models of technical processes.

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 maintenance 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, by exploiting modeling techniques 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 uses of model-based techniques beyond the diagnosis procedures which are traditionally associated with model-based diagnosis.

We believe that real-life engineering applications 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. Traditionally, diagnosis matters are a concern only 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. Diagnosis equipment should be developed in close connection with design, and construction data and FMEA data should be exploited.

- 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 take a close look at model-based service and diagnosis support for the automotive industry. 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 a new method for automatically generating fault-tree diagnosis systems from design data. The key idea is to use modeling techniques of model-based diagnosis for an exhaustive computation of faulty behavior. Based on these data, fault trees can be generated automatically and edited preserving correctness and completeness properties.

Section 4 presents an approach to construct temporally structured models of behavior and use them in consistency-based diagnosis. For instance, designers and operators of sequentially controlled production systems seem to prefer this kind of structuring to a purely component-based approach.

2 Re-use of Models and Software in Diagnosis and Fault Analysis of Automotive Subsystems

2.1 Three 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. FMEA is becoming a legal demand and a customer request in a growing number of areas. 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.

2.2 A categorization for re-use of knowledge and software

Beyond this, the tasks, and, hence, computer systems for their support, share more specific elements. Identifying and analyzing them is a starting point for designing the computational tools in a way that allows for 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. We revisit the three tasks discussed above and highlight the knowledge and procedures involved in order to identify re-usable elements.

- **FMEA-Tool**

The basic knowledge that has to be represented is about the functionality of the components, i.e. **component behavior models**, involved in the designed device and their potential **malfunctions**. The second obvious element is the **device structure** (the blueprint) which determines the interaction of the components. Based on this input, the core of the tasks is then to determine potential effects of component failures which we call **behavior prediction**. The following assessment of these effects is based on the probability of such faults and their criticality. The latter requires knowledge about the severity of the violation of **functionality** of the respective subsystem in the vehicle (e.g. whether the result is just noise, increased emissions, or even danger of fire).

- **Workshop diagnosis**

The nature of the task is obviously to **determine components whose failure could possibly cause some observed effects** (“symptoms”). Again, knowledge about the involved parts,

component behavior models, and the particular **structure** of the system is required, as well as **behavior prediction** in order to compare the predicted and the observed effects. This is usually not a one-step procedure, hence **proposal of tests** is another subtask. Because its nature is to **determine actions** that are likely to **lead to observable distinctions between the behaviors** of the diagnostic candidates, it must be based on some form of **behavior prediction**, as well, and on information about such **actions and observations** and, in particular, about their **cost**.

- **Generation of diagnosis manuals**

In contrast to an interactive workshop diagnosis system that responds dynamically to a changing situation (and obtained information), a manual has to cover all possible diagnostic situations starting from some initial symptoms. Hence, the core is **proposal of test plans**, in the form of more or less sophisticated decision trees. Again, **behavior prediction** based on **component models** and **device structure** information is necessary, as is information of possible **actions** and **observations** and their **cost**.

Already this list of elements of knowledge and software gives some hints on their potential re-use. Applying a classification scheme which is part of a general methodology for developing models and model-based systems (see [Struss 99]), we take a closer look at this list.

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

- **physical principles** (sometimes called domain theory), which are fundamental and valid for all systems (devices) in a particular domain and regardless of the special task, 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.

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

Table 2.1: Classifying knowledge and software elements

With respect to the scope of our work in model-based systems, we obtain the classification displayed in Table 2.1. It is the simplest one, in fact it is somewhat oversimplified, as already indicated by the 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. For the software prototypes discussed above, Figures 2.1, 2.2, 2.3 show the respective architectures and indicate the common components of software and knowledge.

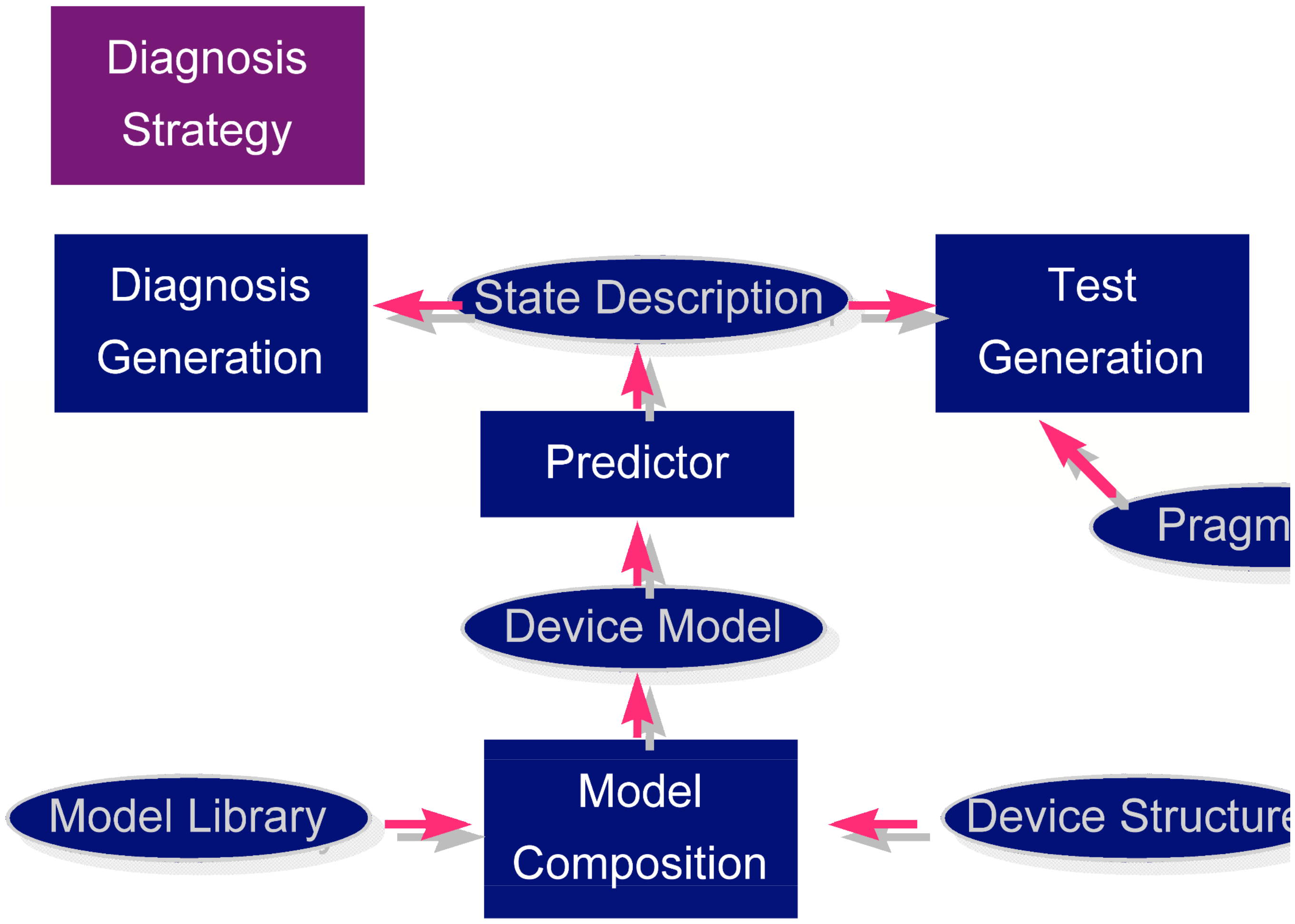

Figure 2.1: Components of the kernel of an off-board diagnosis system

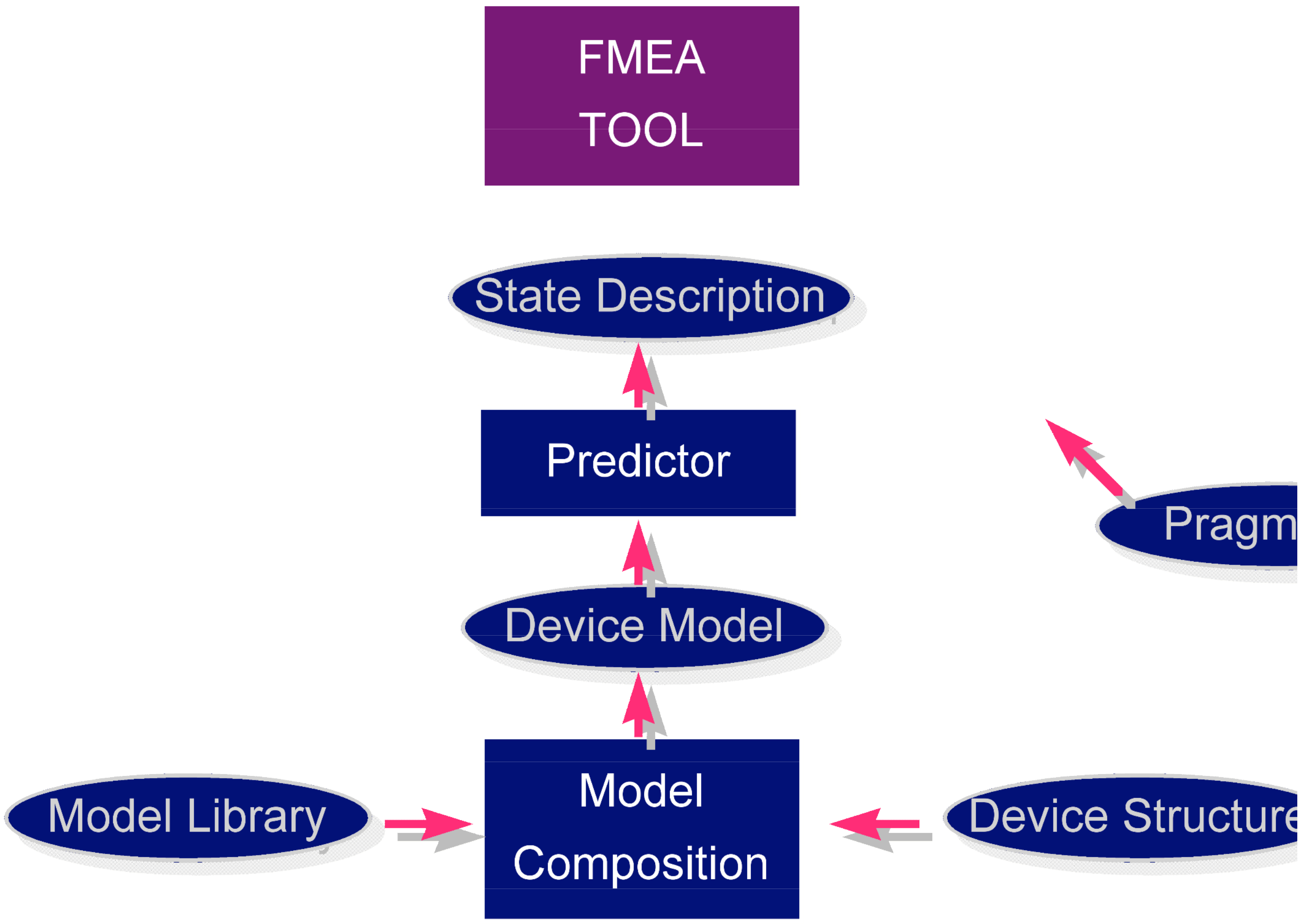

Figure 2.2: Components of an FMEA Tool

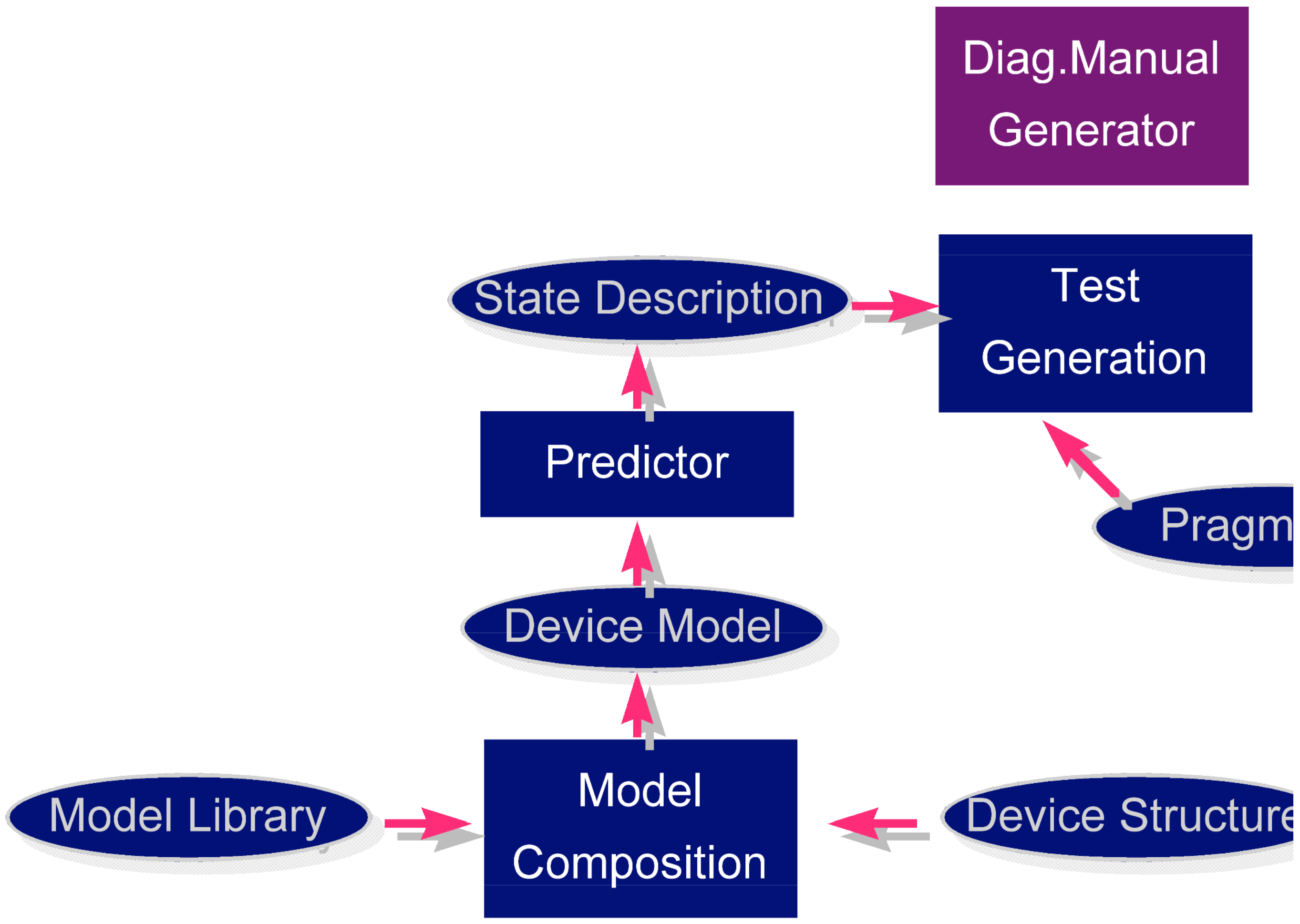

Figure 2.3: Components of the Diagnosis Manual Generator

2.3 Results and 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 tool 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 vertical 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 technology 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.
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, it 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 made explicit 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 may need 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: for the exhaust system, component models, say of pipes and valves, have to include transportation of oxygen, carbon oxide, etc. in the intake and exhaust, whereas for pipes and valves in other parts, 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. [Heller and Struss 96]).

3 Generating Fault Trees

3.1 Application scenario and challenges

More than 100.000 forklifts made by the german company STILL GmbH Hamburg are in daily use all over Europe. In order to reduce forklift downtimes, approximately 1100 STILL service workshop trucks utilize fault-tree-based computer diagnosis systems for off-line diagnosis. 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. The introduction of new diagnosis techniques, however, raises challenges as noted in the introduction.

Model-based fault-tree generation is a promising answer. In particular, grounding diagnosis systems on a model provides a systematic way for modification and re-use of diagnosis equipment. In the STILL application scenario, model-based approaches have to deal with electrical circuits of the transport vehicle domain. These circuits usually consist of components that show a variety of different behavior types, such as analog, digital, static, dynamic, linear, nonlinear and software-controlled behavior. Faults may modify component behavior or may even change circuit structures. Hence, heterogeneous symptoms, such as slight deviations of parameter values or total loss of functionality may occur. In principle, model-based techniques provide a systematic way for predicting the behavior of electrical circuits, including faulty behavior. However, adequate modeling of heterogeneous circuits is still a challenge. The following sections report about the progress which has been achieved.

3.2 Model-based fault-tree generation

In the STILL application scenario, nodes of fault trees represent fault sets. Edges are labeled by the tests (involving measurements, observations, display values and error codes) which must be carried out to verify the corresponding child node. Our approach towards model-based generation of such fault trees is briefly outlined in the following. Due to the application scenario, we focus on the electrical domain, although, in principle, dealing with devices of different technical domains such as hydraulics or mechanics is feasible.

The first step to model-based fault-tree generation is to model a device. This step is supported by a component library and a device model archive (see Figure 3.1). Design data and knowledge from the design process (knowledge concerning intended device behavior, expected faults, available measurements) are integrated into the device modeling process. In a second step, correct and faulty device behavior is predicted automatically by computing the device model. The third step is to build fault trees from behavior predictions. This step is supported by a fault-tree archive and a cost model for the tests which can be performed. Fault-tree generation can be performed automatically or guided by service know-how, i.e. knowledge concerning cost of measurements, preferable fault-tree topologies, fault probabilities. In order to realize these concepts, we implemented the MAD system that is described in the following.
Figure 3.1: Basic concepts of model-based fault-tree generation

3.3 Device modeling and behavior prediction

3.3.1 COMEDI (COmponent Modeling EDItor)

To improve acceptance in industry, the MAD system provides a user interface called COMEDI which is similar to a CAD tool. For device modeling, COMEDI provides two different libraries, a device model archive and a component library. The device model archive allows systematic re-use and modification of device models that were created during former modeling sessions. The component library contains different qualitative models of electrical components. Correct and faulty component behavior is represented. Qualitative modeling is adequate because, usually, faults and symptoms are described qualitatively in this domain.

3.3.2 Standard components

Internally, COMEDI models are mapped to formalized standard components which show well defined and idealized behavior. MAD provides four different standard components: idealized voltage sources, consumers, conductors and barriers. The behavior of idealized voltage sources is well-known from electrical engineering. Consumers are passive and their current/voltage characteristic is monotonic. Idealized conductors do not allow any voltage drop while idealized barriers do not allow any current. Standard components can be connected in combinations of series, parallel, star and delta (triangular) groupings. This simple internal representation of electrical circuits is sufficient for the following reasons.

- In STILL service workshops, only steady-state diagnosis of electrical circuits is performed. Therefore, only steady-state behavior of physical components has to be represented in component models. In particular, an explicit representation of temporal dependencies is not necessary.

- A small number of qualitative standard components suffices, because, often, different physical components show similar electrical behavior, i.e. their current/voltage characteristics differ only slightly. Qualitative versions of these current/voltage characteristics are frequently identical.

- MAD's standard components are deliberately selected so that important behavior classes of the application domain can be adequately represented.

Due to analogies between electrics, mechanics and hydraulics, the internal MAD representation is, in principle, also adequate for other technical domains.

3.3.3 Qualitative values and calculus

Currents and voltages are described by qualitative values consisting of two parts: actual values and deviations from reference values. Similar to the SDSP method [Milde et al. 97], for currents and voltages, there are three qualitative deviation values, i.e. *low (l), normal (n)* and *high (h)* with the obvious meanings. There are five qualitative actual values of currents, i.e. *negative-infinite (ninf), negative (neg), zero, positive (pos)* and *positive-infinite (pinf).* Qualitative values of voltages are similar.

The Tables 3.1 and 3.2 show the qualitative addition of two current values as examples. For the sake of clarity, we omit negative values in the tables. This simple example demonstrates that qualitative deviations cannot be computed independently of qualitative actual values. Hence, qualitative actual values and deviations are associated inseparably and parameters are described by different sets of qualitative values, each valid for one qualitative reference value. In the shaded table cells below, in the left table the result of qualitative addition is *n_pinf*, no matter whether current I2 is characterized by *l_pos* or *n_pos*. In the right table, obviously, the results of qualitative addition are different depending on the qualitative value of I2 (see shaded table cells). Hence, distinct addition tables are defined for each reference value.

I1 | I2	*l_zero*	*l_pos*	*n_pos*	*h_pos*	*h_pinf*
l_zero	*l_zero*	*l_pos*	*l_pos*	*l_pos*	*n_pinf*
l_pos	*l_pos*	*l_pos*	*l_pos*	*l_pos*	*n_pinf*

I1 | I2	*l_zero*	*l_pos*	*n_pos*	*h_pos*	*h_pinf*
l_zero	*l_zero*	*l_pos*	*l_pos*	*l_pos / n_pos / h_pos*	*h_pinf*
l_pos	*l_pos*	*l_pos*	*l_pos*	*l_pos / n_pos / h_pos*	*h_pinf*
n_pos	*l_pos*	*l_pos*	*n_pos*	*h_pos*	*h_pinf*

n_pinf	*n_pinf*	*n_pinf*	*n_pinf*	*n_pinf*	*n_pinf*

h_pos	*l_pos / n_pos / h_pos*	*l_pos / n_pos / h_pos*	*h_pos*	*h_pos*	*h_pinf*
h_pinf	*h_pinf*	*h_pinf*	*h_pinf*	*h_pinf*	*h_pinf*

Table 3.1 and 3.2: Qualitative addition of current values

In order to compute qualitative current and voltage values, local propagation methods are desirable. Local propagation fails, however, except in single cases if carried out in the original network topology. We follow the approach introduced by Mauss [Mauss 98] where networks are transformed into trees which represent the network structure. In particular, series, parallel, star and delta groupings are explicitly represented. Exploiting this tree structure, qualitative behavior can in fact be computed by local propagation.

In order to improve the accuracy of qualitative prediction, MAD offers certain features that prevent spurious solutions. Rather than relying on qualitative versions of basic arithmetics, MAD computes qualitative values for the elements of the structure tree (series and parallel groupings etc.) by a set of qualitative operators which are qualitative versions of complex quantitative equations. In effect, these equations describe the behavior of series, parallel, star and delta groupings. Utilization of complex operators avoids multiple use of simple operators and, thus, avoids spurious predictions. For instance, a voltage divider operator is invoked to compute qualitative voltage values instead of determining current values first and computing voltage values from current values in a second step. In principle, for network analysis, a limited number of operators suffices because of the limited number of standard components and elementary network structures.

Qualitative operators are defined by applying the corresponding quantitative equation to the interval boundaries and to incremental deviations which represent actual values and deviations of qualitative input values. The resulting boundaries and incremental deviations represent the corresponding qualitative output values. The operators are represented by a set of tables comprising more than 30000 entries which had to be generated by computer in order to guarantee reliability. Due to the properties of this qualitative calculus, spurious solutions do not occur at all if the network can be structured into series and parallel groupings of standard components.

3.3.4 Automated behavior prediction

Behavior predictions are performed for all operating modes, faults and fault combinations for which diagnosis support is required. For each operating mode and fault assumption, all symptoms (measurements, observations, error codes, display values) are computed which are in principle available for diagnosis. The output of the prediction step is model-based diagnosis knowledge in form of an extensive table of fault-symptom associations. This table is the basis for fault-tree generation.

3. 4 Fault-tree generation

MAD offers three different possibilities to generate fault trees. First, based on fault-symptom tables, fault trees can be created automatically. Second, fault trees from archives can be re-used. Third, in order to permit manual adaption and modification of fault trees, MAD offers basic editing operations, such as moving a certain fault from one fault set to another and recomputing the corresponding tests. In the following, automated fault-tree generation is presented in more detail. One can choose from the following criteria to guide fault-tree generation.

- **Minimization of average diagnosis cost.** Automated fault-tree generation uses the well-known A*-algorithm to select the tests which minimize the average diagnosis cost according to a cost model (see Figure 3.1).

- **Grouping by observations, error codes, display values.** Fault trees are generated such that subsets of faults correspond to a prespecified symptom. For instance, all faults are grouped together which cause the front lights not to shine correctly.

- **Grouping by aggregate structure.** If the aggregate structure of the device is known, fault trees can be generated such that subsets of faults correspond to the same physical component. For instance, faults occurring on a certain board may be grouped together.

Model-based prediction and automated fault-tree generation guarantee, that fault trees are correct and complete with respect to the underlying device model and the faults and fault combinations considered in the fault-symptom table. All faults considered in the device model occur in the generated fault tree, and tests are selected correctly to discriminate fault sets. This holds even when fault trees are modified manually. Furthermore, average diagnosis cost is minimal within the constraints imposed by a prespecified fault-tree structure.

3.5 Evaluation

We have evaluated the MAD-System in the STILL application scenario and found that using the modeling techniques of MAD with some extensions regarding electronic control units, more than 90% of the faults of the current handcrafted diagnosis system can be handled successfully. The prototypical implementation allows model-based behavior prediction and automatic generation as well as manual modification of fault trees. Furthermore, we successfully integrated these fault trees into existing STILL diagnosis systems.

4. 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 common feature of these systems is that domain experts describe their behavior 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. An important special case are hybrid systems, whose behavior throughout certain steps may be described as continuous while transitions between steps represent discontinuous changes.

To follow the mental models of domain experts by searching for failed process steps in diagnosis of such systems presents several advantages, particularly with respect to re-use. Firstly, temporally structured descriptions of the overall system behavior, such as diagrams of flow of energy, material or information, are much more likely to be found than component-based behavior descriptions. These descriptions can be re-used to guide modeling for diagnosis. With the more widespread use of formal specification and verification languages in engineering there is even the perspective of automating much of this task. Secondly, the proximity between temporally structured diagnosis models and their own mental models helps experts to specify additional information that may be used in diagnosis. Thirdly, there is the chance to re-use diagnosis models for design checking, FMEA or generation of diagnosis manuals, as demonstrated in Section 2. Finally, such diagnosis models mirror the way such technical systems are operated and diagnosed. They should therefore fit more easily into existing traditions and environments. This section presents an approach to diagnosis based on such models, which Fraunhofer IITB developed in the context of INDIA.

4.1 Modeling

Our approach proposes the following guidelines to construct temporally structured models for diagnosis:

1. Identify the steps of the technical process. Try to rely on steps that have already been identified for the considered system (e. g. in existing behavior specifications) and on types of steps that are commonly used in the domain (e. g. the basic physical and chemical procedures of process industry). Steps need not be ordered sequentially, but may be processed in parallel.

2. For every step search in the model library for an unstructured model (also known as Block-Box model) of successful behavior which may be re-used for diagnosis or used as a starting point for modifications or refinements. If no adequate model can be found, construct a new one, which may be added to the library afterwards. In any case, such a model has to contain propositions to infer that the system is processing a particular step, for instance:

   - starting conditions: conditions which necessarily and sufficiently mark the beginning of the step (e. g. a certain event in a particular situation);

   - ending conditions: conditions which necessarily and sufficiently mark the end of the step (e. g. a certain value of a sensor signal);

   - minimal duration constraints: a constant or function of system variables that specifies some lower bound on the duration of a successful step.

   With respect to diagnosis, the model should also contain propositions of some of the following types, which might be called inconsistency detecting constraints:

   - Prerequisites: necessary conditions defined on system variables (e. g. inputs to the system) that have to be met at the beginning of the step if the step should succeed.

   - Required invariants: system properties that have to remain unchanged to complete the step successfully.

   - Guaranteed invariants: system properties that remain unchanged throughout a successful step.

   - Duration constraints: constants or functions of system variables that specify some lower and/or upper bound on the duration of a successful step.

- Success indicators: propositions that necessarily hold at the end of the step if the step completed successfully.

The following criteria should be considered in choosing such propositions:

- The set of propositions has to be consistent.

- Truth of a proposition should distinguish success of the step from (at least one typical way of) failure.

- It must be possible to check a proposition: the negation of the proposition should be inferable from the model and other propositions probably available during diagnosis, like observations or propositions inferred from models of other steps. “Probably available” indicates room for pragmatic interpretation. In order to support re-usability we should make only weak assumptions about the context, in which the model will be used.

- To be useful in diagnosis, prerequisites require starting conditions, duration constraints require either starting or ending conditions, success indicators require ending conditions and invariants require at least propositions of two of the types starting conditions, ending conditions and minimal duration constraints. Otherwise it would not be clear at what time the propositions have to hold.

The model might contain other constraints on system variables that describe the system's behavior during the step, but the information types listed above seem to be particularly useful for diagnosis.

3. For every typical failure of a step look in the model library for a corresponding failure model, which may be re-used for diagnosis or used as a starting point. In analogy to the elements of the success model it should contain propositions that allow to infer that the system processes the step and inconsistency detecting constraints. Obviously, success indicators have to be replaced by failure indicators.

4. Construct a temporally structured model of the overall process which specifies the identified steps and their coupling, e. g. by shared system variables.

5. While identifying the step which failed or the kind of failure may already be sufficient for some preliminary treatment, repair of the system will frequently require to further localize the failure inside a step. In that case we need a structured model of the step. If the step is composed of smaller steps and we want to localize the failure in one of them, we can construct a temporally structured model of the step in the same way as for the overall process. But eventually, we may want to identify a faulty component that caused the failure.

   To this end search the model library for a model of how components interact to implement the step. If no adequate model is available, construct a new component-based model. In any case, such a model should be much simpler than a model that describes how components implement the overall system behavior.

To simplify the presentation, we assumed that structured models specify either steps or components only. However, the approach can be generalized in a straightforward way to models that describe behavior by interacting steps and components.

4.2 Consistency-based diagnosis with temporally structured models

While it is unclear if these types of models may be used for the generation of decision trees, the standard procedure of consistency-based diagnosis may be employed to compare them to observations. To this end we justify every inference 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.

4.3 An example

Our guiding example in INDIA is the chemical distributor (CHD) from THEN GmbH, a system to distribute liquids in a dye house, (see Figure 4.1). 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. Control signals to pumps and valves as well as the sensor signals from a flow meter can be observed by the controller and reported to a logging facility for use in diagnosis. Additionally, an operator can observe the status of valves.

Figure 4.1: Chemical distributor

These steps can be used to construct a temporally structured model of the overall behavior of the CHD. In particular, analysis of the control flow diagram of the control program shows that measuring out of chemicals, transport of the mixture and rinsing may be viewed as instances of a general step type. A temporally structured model for this kind of steps includes as smaller steps: opening the affected valves, starting the pump, waiting until the requested amount has been transported, stopping the pump and closing the valves.

The black-box models of successful steps employ propositions about the values of the valve control signals as starting and stopping conditions. Among the inconsistency detecting constraints duration constraints are particularly useful. For instance, an upper bound on the duration of the measuring and rinsing steps can be derived as a function of the requested amount. Upper bounds on the time needed to open or close a valve are also available. In the past, we have observed a similar benefit from duration constraints in another domain, diagnosis of an assembly station. In general, duration constraints are able to detect failures that cannot be characterized by some unique set of synchronous observations. Typical examples are effects that happen too fast (e.g. because of miscalibrated sensors), happen too slow (e.g. in case of wear or pollution) or do not happen at all (e.g. stuck valves).

Now, assume that some valve of the chemical distributor should open for measuring out some chemical, but is stuck. Since we can observe the control signals, we can detect when steps start and end. Because of the duration constraints, we find that some measuring step does not end in time and hence fails. Now we switch to a component oriented model which describes, how pumps, valves, the flow meter and other components implement the measuring step. From the observed control signals and the models of pumps and valves we infer that flow should be enabled, however we cannot observe any flow. Diagnosis suspects the pump, the flow meter and the valves. Observations of the states of particular valves requested from the operator finally establish the correct diagnosis. Note that analysis of the step models allowed us to temporally focus diagnosis on a particular step. The subsequent analysis on the component level neglected those components that did not take part in the suspected step as well as interactions of the participating components during other steps.

4.4 Discussion

This section presented an approach to consistency-based diagnosis that initially relies on models which structure a technical process into several steps. The example demonstrated the benefits from the approach with respect to knowledge acquisition and complexity of the diagnostic task.

The approach extends the potential for re-use of models from libraries, since step models can be re-used just as component models. This has also been demonstrated in the example.

The approach seems to be applicable whenever experts describe the behavior of a technical system by steps processed sequentially or in parallel. For instance, the behavior of sequentially controlled systems is usually described by event models, such as statecharts. The rapidly growing number of applications of such systems - e. g. industrial plants, air conditioning in buildings, cars and aircrafts, washing machines and video recorders - indicates the relevance of our approach.

An interesting question is how the approach deals with failures of steps that are caused by failures of previous steps. For instance, a manufacturing step that should drill a hole into some product part depends on previous steps to provide the right drill and part. In such a case structured models of the subsequent step would be employed to recognize the component that “transported” the failure, e. g. the part. Then we can choose a model of the overall process that focuses on the history of this component and describes the previously hidden interaction. This seems to be quite similar to the approach hidden interactions are handled in component-based diagnosis. More importantly, it seems to be the way human experts proceed in diagnosis.

5. Conclusions

We have addressed several tasks related to diagnosis 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 identifying re-usable components in three diagnosis-related systems of the automotive domain.

The work on model-based fault-tree generation can be viewed as another example for this general idea. By separating device-specific from task-related knowledge and generic from

pragmatic aspects, and by model-based generation of fault trees, fault-tree diagnosis systems with a
high degree of re-usability can be obtained. 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. Furthermore, an abstraction level is provided where design information or human expert knowledge can be expressed, facilitating re-use between different tasks.

Acknowledgments

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

References

[Heller and Struss 96] 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.

[Mauss, 98] Mauss, J.: Analyse kompositionaler Modelle durch Serien-Parallel-Stern Aggregation, DISKI 183, Dissertationen zur Künstlichen Intelligenz, 1998.

[Milde et al., 97] Milde, H., Hotz, L., Möller, R., Neumann, B.: Resitive 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.

[Struss 99] Struss, P.: Contributions to a Methodology of Model-based Systems for Diagnostic Tasks. In preparation, 1999.