ELSEVIER

Future Generation Computer Systems 9 (1993) 329–337

FGCS
FUTURE GENERATION COMPUTER SYSTEMS

Issues in practical model-based diagnosis †

R.R. Bakker *, P.C.A. van den Bempt, N.J.I. Mars, D.-J. Out, D.C. van Soest

University of Twente, Department of Computer Science, P.O. Box 217, 7500 AE Enschede, Netherlands

Abstract

The model-based diagnosis project at the University of Twente has been directed at improving the practical usefulness of model-based diagnosis. In cooperation with industrial partners, the research addressed the modeling problem and the efficiency problem in model-based reasoning.

Main results of this research are that (1) re-use of electronically available descriptions of systems for diagnostic purposes is possible, and (2) efficient reasoning can be realized using PDE, hierarchic models, and a simple diagnostic strategy. We have built a prototype diagnostic system which shows the technical feasibility of model-based diagnosis in a computer system.

The project was concluded in July 1993 by the development of the 'Diagnostic Toolbox'. The Diagnostic Toolbox supports the modeling of technical systems for diagnostic purposes, and it contains several model-based reasoning methods.

Key words: Model-based diagnosis; Modeling; Computational efficiency; Tools

1. Introduction

Modern design and production methods have resulted in complex technical systems which, considering their complexity, seldom fail. In case of a problem, a service engineer will have to troubleshoot the system effectively and efficiently. The low frequency of failures, the high complexity of technical systems, and the existence of many variants of a system may complicate the troubleshooting process.

Due to the lack of experience in troubleshooting, expert systems that are based on this type of knowledge are difficult to construct. Engineers use, in addition to their experience, descriptions of a system and technical principles. The use of technical knowledge and descriptions of systems for diagnostic purposes is the focus of research in *model-based diagnosis*, a subfield of Artificial Intelligence. Introductions in model-based diagnosis are [6] and [16]. A collection of frequently cited articles in model-based diagnosis is [8]. Most research is based on de Kleer and Williams [9] and Reiter [13].

Model-based diagnosis uses a model of a technical system, observations, and specialized reasoning methods to obtain diagnostic hypotheses. With additional information, like probe and test results, we can discriminate between these hy-

* *Corresponding author*. Email: bakker@cs.utwente.nl

† This research has been sponsored by SKBS. SKBS is a Dutch foundation that stimulates research cooperation between universities and industry in the domain of knowledge-based systems.

SSDI 0167-739X(93)E0033-V

potheses in order to identify the actual defects. A diagnostic model of a system is often a simulation model; the behavior of a correctly operating system can be predicted with such a model. A diagnostic model consists of components, behavior of components, and connections between components. The diagnostic hypotheses that can be generated using such a model consist of sets of possibly defective components.

Central research issues in model-based diagnosis have been *modeling* and *effective and efficient reasoning*. The problem of modeling is to acquire a technical description of a system that can be used for diagnostic purposes. A prerequisite for any modeling approach is a good evaluation criterion for a diagnostic model. The reasoning problem is to develop methods that identify the faults in a system at low costs. The General Diagnostic Engine (GDE) of De Kleer [9] is an effective reasoning method for a limited class of systems. GDE's computational problems limit its application to small-sized systems (fewer than 10 components).

1.1. The diagnosis project at the University of Twente

Since 1988, our research has been directed at these central issues in model-based diagnosis. We have addressed the research questions in cooperation with several industrial partners in the framework of project A1 of SKBS. The goal of this research project has been to narrow the gap between model-based diagnostic theory and its practical application.

In this paper we present an overview of the results of this research project. The central research issues have been addressed in a pragmatic way. Re-use of already available technical information of a system (like CAD-descriptions of a system) has been our focus of research in modeling. We will discuss the transformation of CAD-descriptions of digital systems into diagnostic models in Section 2.

We have addressed the reasoning problems in various ways. Makarovič [10] has described a method of symbolic reasoning that solves part of the effectiveness problem and he has developed a preprocess that identifies all potential conflicts [1] in advance. However, serious computational problems remain. We have, therefore, developed an approximate reasoning method, called PDE (pragmatic diagnostic engine), that generates only a small number of conflicts to give probe advice [1]. As these conflicts are computationally easy to identify, preprocessing is only important in models with complicated behavioral descriptions of components.

Out [11] describes strategies for action selection in the more general setting of troubleshooting. A computationally simple strategy called *weighted elimination* turned out to select near-optimal probes or repairs in a number of experiments. We discuss in Section 3 the notion of *diagnostic indistinguishability* of components. In case components are indistinguishable, they can be grouped together leading to an hierarchic model. This may result in a large efficiency gain. Recognizing indistinguishable components is computationally simple. Indistinguishability of components can, therefore, also serve as criterion for model evaluation in a modeling process.

In addition to the scientific problems, two *organisational problems* exist that complicate the development of practical model-based diagnostic applications. The first one is mainly a start-up problem. Very few diagnostic applications have been realized and are in practical use. Therefore, little experience exists in estimating development costs and return on investment. In our research we have developed two prototype diagnostic systems to obtain some ideas on this problem. The first prototype we developed is a program for diagnosing Philips P9000 computer systems; we will describe this prototype briefly in Section 4.1. The other prototype is a program for diagnosing an industrial chemical process installation.

[1] Important concepts in model-based diagnosis are a conflict, a diagnosis and a dependency record. Loosely formulated: a conflict is a set of assumptions about the state of components which together with the observations leads to an inconsistency. A diagnosis is a set of assumptions about the state of components that explains the observations. A dependency record contains a value for a signal together with the assumptions used to obtain that value.

The other organisational problem is the lack of tools for developing model-based diagnostic applications. In Section 5, we will briefly discuss the diagnostic toolbox that has been developed in our project.

2. Modeling

Van Soest [15] has studied the re-use of existing models of a system for troubleshooting. Any modeling method requires an evaluation of the model obtained. The resulting model should be checked on its usefulness for diagnostic reasoning. Van Soest uses the following criterion for an adequate model:

A model is adequate if the relations between the observation points differ for each state that has to be distinguished.

This corresponds to the concept of identifiability in system theory. Scarl [14] also pays attention to diagnostic distinguishability, with the aim of minimizing the requirements on sensors. He does not describe how diagnosability is checked.

Distinguishability can be checked by constructing the equations that describe the relation between probing points, for each state-assignment that should be distinguishable. These equations are then pairwise compared. When equations are equal, the corresponding sets of state assignments are indistinguishable. There is one problem with this: the problem of comparing two arbitrary relations is undecidable. In Section 3.3 we present a different approach to check the adequateness of a model. This approach identifies indistinguishable components based on the structure of the model and characteristics of ports of components that can be locally determined by the behavior of components.

2.1. Re-use of models

Modeling from scratch can be very difficult. For many technical systems descriptions and models already exist. In Van Soest [15] the re-use of design descriptions of electronic systems has been described. Common formats for describing designs of electronic systems are EDIF and VHDL.

The models that are derived from design data, either in EDIF or in VHDL format, describe only part of the behavior of a system. Only those parts of the system are described that contribute to the correct behavior of a system, and only that part of the correct behavior is described that is assumed to be relevant by the designer. The influences between parts that are described are also only those that contribute to the correct system behavior. Also, behavioral descriptions might not be suitable to identification of the potential faults. In other words, it may be that:

- not all components in the model are fallible components;
- there are more connections in the model than are necessary to distinguish between potential diagnoses;
- the components have ports that are not necessary for distinguishing between diagnoses;
- there are too many component states;
- the behavioral descriptions are too detailed;
- not all fallible components are contained in the model;
- not all connections necessary to distinguish between potential diagnoses are present in the model;
- components miss ports that are necessary for making the distinction between potential diagnoses;
- there are component states missing;
- the behavioral descriptions are not detailed enough.

A model derived from design data might or might not be an adequate model. The model will have to be analyzed to make sure that all potential diagnoses are distinguishable. Also, it might be possible to simplify the model, by removing components, ports or connections, or by simplifying behavior.

Despite these shortcomings of models derived from design information, design data might still be useful. It provides the modeler with a partitioning of the system into parts, and gives at least a portion of the behavior of these parts. An interesting project in that respect is the design knowledge capture project for Space Station

Freedom [17]. In that project, not only the designs are electronically stored, but also the design decisions. The researchers envisage a greater support for (re)design, manufacturing and maintenance than with traditional techniques. The benefits for maintenance are mainly thought to lie in better availability of information about the system. No explicit support is provided for modeling for MBD, but this would have been a natural extension to the project.

3. Reasoning

3.1. Effectiveness

The effectiveness problem in model-based diagnosis consists of checking if a particular diagnostic hypothesis is consistent with all observations obtained. In general, consistency checking is undecidable, see Fig. 1.

Most diagnostic algorithms (e.g. GDE [9] and HT [5]) use *value propagation* to identify inconsistencies. Makarovič [10] shows that constraint propagation and symbolic processing can be more effective than value propagation. As a result in the example above the following equations are obtained:

$$C = A^5 + A^4 + A^3 + A^2 + A$$

based on ok(C_1) and ok(C_2)

$$C = B^5 + B^4 + B^3 + B^2 + B$$

based on ok(C_2).

In the example, neither value propagation nor symbolic reasoning are capable to determine the (in)consistency of diagnostic hypothesis [C_1]. As the problem itself is undecidable, the effectiveness problem in model-based diagnosis cannot always be solved.

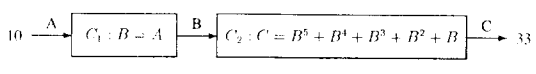


Fig. 1. The values of the variables in the model are integers. checking the consistency of diagnostic hypothesis [C_1] ([C_1] denotes component C_1 is defective) is undecidable.

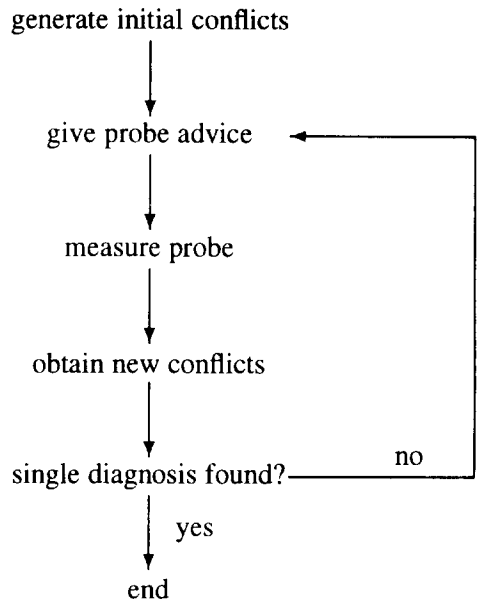


Fig. 2. A graphical description of PDE.

3.2. Pragmatic diagnostic reasoning

The Pragmatic Diagnostic Engine (PDE) is an adaptation of GDE [9] that is directed at improving run-time performance. For the purpose of this article only some of the aspects of PDE will be summarized. A more detailed description of PDE can be found in [1].

The aim of PDE is to provide an efficient diagnostic engine. In PDE the expensive (NP-hard) transformation of conflicts into diagnoses is omitted, and probe advice is given based on a limited number of minimal conflicts. The general concept of model-based diagnosis based on conflicts is shown in Fig. 2.

PDE uses probe advice based on a limited set of minimal conflicts. The probability mass of state assignments excluded by conflicts plays an important role in the modification of PDE to meet resource limitations; we will discuss this in some depth. First, the relevant theory of probe advice based on diagnoses, as described in [9], will be summarized.

We assume that prior failure probabilities of components are known, and that components fail independently. For each probing point x_i possi-

ble values v_{ik} can be measured (the most likely values v_{ik} for probing point x_i are contained in the dependency records computed for x_i). The expected benefit ΔH from measuring probing point x_i is computed by:

$$\Delta H = -\sum_k p(x_i = v_{ik}) \log\, p(x_i = v_{ik}) + \zeta.$$

In this expression $p(x_i = v_{ik})$ is the conditional probability that the value of probing point x_i equals v_{ik} given the current set of diagnoses; ζ is a rest term that can be ignored in the context of this paper. The probing point with maximum ΔH discriminates best between likely diagnoses.

The probabilities $p(x_i = v_{ik})$ can also be computed using the current set of conflicts. The general idea is to consider which conflicts result from measuring $x_i = v_{ik}$, and to compute the effect of these resulting conflicts on the probability mass of state assignments (normal/abnormal) to components that are still possible.

In a similar way as the set of remaining diagnoses, the set of conflicts that remains after measuring $x_i = v_{ik}$ can be computed using the dependency records.

The initial set of conflicts in PDE consists of obvious and semi-obvious conflicts. Obvious conflicts are conflicts that can be computed by simulating the model (forward propagation). Semi-obvious conflicts can be found by back-propagation of measured values in the model. In a series of experiments, it turned out that these conflicts are easily computed (in $O(n^2)$) and probe advice based on these conflicts is almost equal to GDE's probe advice.

3.3. Indistinguishability

Another source of inefficiency for diagnostic reasoning is shown in the following example. Consider the model depicted in Fig. 3. Suppose that the observations are that the input is '0', the output of c_6 is '1' and that of c_{11} is '0'. We can use generate and test to determine the minimal diagnoses, giving us $\{[c_2], [c_3], [c_4], [c_5], [c_6], [c_1, c_7], [c_1, c_8], [c_1, c_9], [c_1, c_{10}], [c_1, c_{11}]\}$. A succinct way of describing this set is: either one component from $\{c_2, \ldots, c_6\}$ is abnormal, or c_1 is abnormal *and* one component from $\{c_7, \ldots, c_{11}\}$ is abnormal.

Now suppose that we observe the output of c_6 to be '0' and that of c_{11} to be '1'. The resulting minimal diagnoses can now be described as: either one component from $\{c_7, \ldots, c_{11}\}$ is abnormal, or c_1 is abnormal *and* one component from $\{c_2, \ldots, c_6\}$ is abnormal.

A third case is when both outputs are '1'. In this case we can succinctly describe the set of minimal diagnoses as: either c_1 is abnormal, or one component from $\{c_2, \ldots, c_6\}$ is abnormal *and* one Component from $\{c_7, \ldots, c_{11}\}$ is abnormal.

This example shows that components $c_7, \ldots, c_{11}$ are indistinguishable from a diagnostic perspective. The same hold for components $c_2, \ldots, c_6$. The model in Fig. 3 could be simplified by clustering all indistinguishable components. We now define:

Two components c_1 and c_2 are indistinguishable if for every possible combination of observations, every minimal conflict contains either: both c_1 and c_2, or neither c_1 nor c_2.

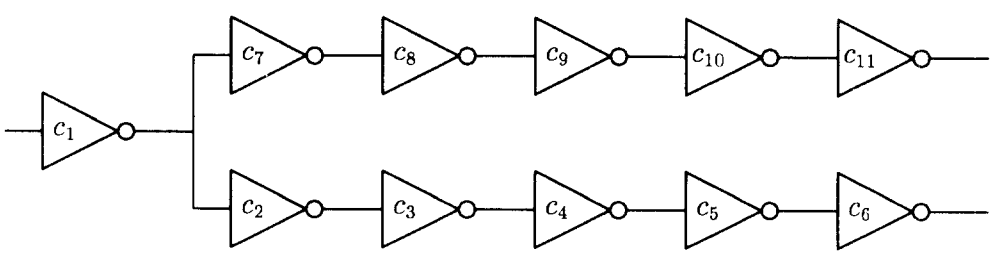


Fig. 3. A model consisting of 11 inverters.

Generating maximal clusters of indistinguishable components is very complex (in general, the problem is undecidable [11]). The following algorithm identifies a set of indistinguishable clusters in polynomial time ($O(n^2)$). Very often, each cluster in this set will be maximal, however, it may happen that some clusters are not maximal, for further details see [11].

In the algorithm the notion of a non-computable input is used. This is an input of a component which is not relevant in some computations of the output. For example, both inputs of an AND-gate are non-computable as a single input of 0 determines a 0 output of the AND-gate. Furthermore, the set $Infl(c)$ consists of all components that influence directly or indirectly an input of component c.

Input. A model, the set of components is denoted by C.

Output. The set ALLC of clusters of indistinguishable components.

Step 1. ALLC $:= \{\{c\} | c \in C \wedge c$ is an output component$\}$

Step 2. Take a component v, not in any cluster, such that all of the Components in $Infl(v)$ are in a cluster. If this is not possible, terminate.

Step 3. If v directly influences components in different clusters then $ALLC := ALLC \cup \{\{v\}\}$ and go to Step 2.

Step 4. If all components directly influenced by v are influenced through only non-computable inputs then $ALLC := ALLC \cup \{\{v\}\}$ and go to Step 2.

Step 5. Let C be the cluster that contains the components directly influenced by v. $C := C \cup \{v\}$. Go to Step 2.

The proof of correctness of this algorithm is rather complicated and can be found in [11]. In the example presented above the result of the algorithm is $ALLC = \{\{c_1\}, \{c_2, \ldots, c_6\}, \{c_7, \ldots, c_{11}\}\}$. These clusters can be used as components in a model at a higher level of abstraction. If the probing costs of connections can be grouped in classes, a multi-level hierarchical model arises, see [7].

3.4. Troubleshooting strategies

Troubleshooting a system consists of three different activities:

- using the model to reason about hypotheses about the malfunctioning of the system (we also call this diagnostic reasoning);
- observing the system;
- repairing the system.

As nothing in life is free, we can associate a cost with each of these activities: diagnostic reasoning takes computing time, observing the system takes time and effort, repairing the system takes time, effort and possibly spare parts. These costs can usually be differentiated within the activities: some observations are more difficult to make than others, some hypotheses are computationally harder to check than others and some repairs are easier and cheaper than others.

To manage these costs, we need a strategy: a procedure that schedules the different activities that we can undertake.

As our prime motivation for introducing a strategy for model-based troubleshooting is cost, a good strategy is a strategy that minimizes cost. That is, a strategy that minimizes the sum of:

- all the costs involved in diagnostic reasoning;
- all the costs involved in observing the system;
- all the costs involved in repairing the system;
- all the costs involved with the strategy itself.

In [11], an overview is given of troubleshooting strategies based on an estimate of total expected costs. The problem in all of these approaches is that no good cost estimation function has yet been found in case that probe cost, test cost, or repair cost differ.

Out [11] shows in a series of experiments that an *opportunistic* strategy based on direct cost and yield outperforms the strategies based on total cost estimates. Out defines the following:

Yield for a probe: $1 - \sum p_i(1 - p_i)$, where

$$p_i = P(\text{probe} = \text{value}_i)$$

Yield for a repair: $(1 - P(\text{system fixed}))^2$.

The ranking of the troubleshooting action is in order of increasing yield × cost.

The additional advantage of Out's opportunistic strategy is that identifying the best troubleshooting action is computationally simple (in $O(n^2)$ where n is the number of components in the system).

4. Realizing model-based applications

From a management perspective, it currently is hard to assess the value of model-based diagnostic applications. Mainly due to a lack of experience in building model-based diagnostic applications, the development costs are hard to estimate. The lack of commercial model-based reasoning tools further increases the development cost of applications.

The uncertainty in the costs of developing applications may also be present in the yield of an application. A diagnostic application reduces the service time; it is not obvious, however, that such a reduction is always profitable. In case service time reduction prevents a trip in a chemical plant, its usefulness is obvious. When only the time is reduced that an operator uses to identify a fault, no cost reduction has been obtained.

In our research project, we have developed a prototype diagnostic system for Philips P9000 computer systems. The department Customer Service of Philips Apeldoorn considered this domain as relevant both from a technical and a management point of view. The project has been stopped before a complete prototype was realized because of a reorganisation of the Philips company.

4.1. Diagnosing Philips P9000 computer systems

The Philips P9000 is a series of minicomputer systems. They can be configured in many ways, ranging from a one-processor unit with a terminal to many multi-processor units in a network, together with numerous peripherals like terminals, workstations, dot-matrix printers and laser printers, modems and the like. The processor units themselves can also be configured: processors, disk units, tape units, network controllers, and memory can all be present in various amounts. The number of components in a P9000 configuration can range from 10 to 1000. For each piece of hardware that is added, some software has to be present in the system, or some parameters have to be adjusted. The computer system itself must have information about its configuration. This information is recorded in a number of files, collectively called the *configuration database*. In the configuration database, all hardware components and software components are described, and many aspects of their relations.

The number of different configurations that exist of these computer systems (different for each customer) makes diagnosis difficult. Not only must the troubleshooter come to know the hardware configuration in all its details, but also the parameters that are adjusted and the software that is installed. For this reason, attention turned to automation of the troubleshooting process. In addition, we have studied how to acquire diagnostic models of P9000 computer systems by reusing the configuration information.

The prototype diagnostic system we have built identifies hardware faults at subsystem level using simple observations, like successful or failing UNIX commands. The prototype gives advise on the commands that are most useful in a particular troubleshooting session. The model of a particular P9000 system is obtained automatically by transforming the configuration information. A full description of this prototype system is given in (Bakker et al. [2]).

5. Diagnostic toolbox

In our experience, realizing diagnostic applications involves a selection of reasoning methods and reasoning strategies. As a tool was lacking that supports prototype development, we have implemented a Diagnostic Toolbox that offers good experimental possibilities. The Diagnostic Toolbox is based on global specifications [12] which are detailed using the Yourdon method in combination with the CASE-tool SDW and the formal specification language Z [3,4]. The Diag-

nostic Toolbox has been implemented in C on a Sun Sparc Station 1 under X-Windows. Its functionality consists of:

Model editor, a graphical model editor for creating and modifying a diagnostic model of a system;

Diagnostic reasoner, a series of programs that execute diagnostic reasoning steps like conflict generation, diagnosis generation, probe advice and test advice;

Diagnostic user interfaces, a graphical interface for presentation of the diagnostic results and user input for additional steps;

Strategy composer, programs that allows a user to compose diagnostic strategies, check these strategies for consistency and execute these strategies on the actual diagnostic context and data;

Log and statistics, programs that log the diagnostic session and produce statistics.

6. Conclusion

From a technical point of view, modeling a system for diagnostic applications is still difficult. This problem is decreased by using existing models of a system, like design descriptions or configuration information. Model-based reasoning can be made tractable by different approaches.

Most technical problems in model-based diagnosis are solved, but organisational problems remain. As long as only a few applications have been realized it is hard to estimate the development cost. The yield of an application highly depends on the current process of troubleshooting and the possibility of reducing break-down costs.

References

[1] R.R. Bakker and M. Bourseau, Pragmatic model-based diagnosis, in *Proc. 10th European Conf. on Artificial Intelligence* (Aug. 3–7, 1992) Vienna, Austria, B. Neumann, ed. (Wiley, Chichester, New York, 1992) 734–738.

[2] R.R. Bakker, P.A. Hogenhuis, N.J.I. Mars and D.C. van Soest, Re-using configuration information: High-level diagnosis of Philips P9000 computer systems, in *Industrial applications of knowledge-based diagnosis*, G. Guida and A. Stefanini, eds. (Elsevier, Amsterdam, 1992) 3–24.

[3] P.C.A. van den Bempt, Yourdon specifications of the Diagnostic Toolbox, University of Twente, Memorandum UT-KBS-92-28, SKBS/A1/92-14, Enschede, the Netherlands, 1992.

[4] P.C.A. van den Bempt and R.R. Bakker, Z specifications of the Diagnostic Toolbox: the diagnostic module, University of Twente, Memorandum UT-KBS-92-30, SKBS/A1-92-18, Enschede, the Netherlands, 1992.

[5] R. Davis, Diagnostic reasoning based on structure and behavior, MIT Artificial Intelligence Laboratory, A.I. Memo 739, 1984.

[6] R. Davis and W. Hamscher, Model-based reasoning: Troubleshooting, in *Exploring Artificial Intelligence*, H.E. Shrobe, ed. (Morgan Kaufmann, San Mateo, CA, 1988) 297–346.

[7] E. Hammink, D.J. Out and R.R. Bakker, Controlled reasoning and hierarchy in model-based diagnosis, in *Proc. 11th Internat. Conf. on Expert Systems and Their Applications*, Avignon (1991) 217–230, SKBS-A1/91-01.

[8] W. Hamscher, L. Console and J. de Kleer, *Readings in model-based diagnosis* (Morgan Kaufman, San Mateo, (4, 1992).

[9] J. de Kleer and B.C. Williams, Diagnosing multiple faults, *Artificial Intell.* 32 (1987) 97–130.

[10] A. Makarovič, Parsimony in model-based reasoning, University of Twente, Ph.D.-Thesis, Enschede, 1991.

[11] D.-J. Out, Strategies for efficient model-based troubleshooting University of Twente, Ph.D.-Thesis, Enschede, 1993.

[12] F. van Raalte and D.C. van Soest, High-level specifications of "The Diagnostic Toolbox", University of Twente, SKBS/a1/92-07 Memorandum UT-KBS-92-15, Enschede, 1992.

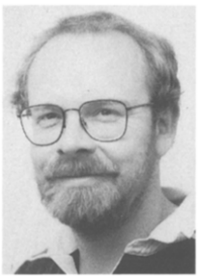
[13] R. Reiter, A theory of diagnosis from first principles, *Artificial Intell.* 32 (1987) 57–95.

[14] E. Scarl, Analysis of diagnosability, in Working Notes *Second Internat. Workshop on Principles of Diagnosis*, Milan, Italy (Oct. 14–16, 1991) 191–200.

[15] D.C. van Soest, Modeling for model-based diagnosis, University of Twente, Ph.D.-Thesis, Enschede, 1993.

[16] P. Struss, Model-based diagnosis-progress and problems, in *Proc. 3rd Internat. GI Congress "Knowledge-Based Systems"*, Munich (1989) 320–331.

[17] D.B. Wechsler and K.R. Crouse, An approach to design knowledge capture for the Space Station, in *Space Station Automation II, Proc. SPIE, Vol. 729*, Wun C. Chiou, ed. (1986)106–113.



René R. Bakker is currently associate professor in the Knowledge-Based systems Group at the University of Twente. His research interests include model-based troubleshooting and resource-limited reasoning. He received a B.Sc. and M.Sc. in applied mathematics from the University of Twente. His Ph.D.-study at the University of Twente was directed at representing and structuring scientific knowledge in semantic networks.

Paul C.A. van den Bempt graduated in Computer Science at the University of Twente. He has worked on the domain of model-based troubleshooting for three years. His main interest is application-oriented research on this domain. He realized a diagnostic toolbox and worked on the development of a diagnostic system for an industrial chemical process. He intends to start a Ph.D. research project on a sub-domain of model-based troubleshooting in the near future.

Nicolaas J.I. Mars obtained his B.Sc., M.Sc. and Ph.D. in electrical engineering from the University of Twente in 1972, 1974 and 1982, respectively. He leads the Knowledge-based Systems Group in the Department of Computer Science at the University of Twente, where he is professor of Computer Science. His research interests include large-scale knowledge bases and provable expert systems.

Dirk-Jan Out studied Computer Science at the Technical University Delft from 1983 to 1989. From 1989 to 1993 he was employed by the Knowledge-Based Systems Group of the University of Twente. In 1993 he received his Ph.D. for his thesis 'Strategies for efficient model-based troubleshooting'. He is currently employed as researcher at the Telematica Research Center Enschede.

Dick C. van Soest studied Electrical Engineering at the University of Twente from 1979 to 1988. In his studies, he put a great emphasis on Computer Science; he did his Master's thesis in the department of Computer Science, in the Knowledge-Based Systems group. After his studies, he went to work in the Knowledge-Based Systems group, in the project on model-based diagnosis. After two years of working as a researcher, he decided to start work for a Ph.D. In 1993 he finished his thesis, entitled 'Modeling for model-based diagnosis', and received his Ph.D.