19957 20969

N95-27390

S15-19

P. 13

Model-based monitoring and diagnosis of a satellite-based instrument.

André Bos,* Jörg Callies, and Alain Lefebvre.†

Abstract

For about a decade model-based reasoning has been propounded by a number of researchers. Maybe one of most convincing arguments in favor of this kind of reasoning has been given by Davis in his paper on diagnosis from first principles (Davis 1984). Following their guidelines we have developed a system to verify the behavior of a satellite-based instrument GOME (which will be measuring Ozone concentrations in the near future (1995)). We start by giving a description of model-based monitoring. Besides recognizing that something is wrong, we also like to find the cause for misbehaving automatically. Therefore, we show how the monitoring technique can be extended to model-based diagnosis.

1 Introduction

1.1 Testing complex systems

Before space systems, like satellite-based instruments, go into orbit, it is important to validate the system's functioning thoroughly. However, as systems become more and more complex, the effort needed to verify these systems becomes enormous. Traditional testing methods validate system behavior by applying test inputs and comparing observed to expected output behavior. Care must be taken that all possible interactions between subsystems are covered. Unfortunately, experience shows that it is nearly impossible to do complete testing, and most systems possess some unknown –and unwanted– behavior. In these cases it is very important to know if the system (*e.g.*, when it is in orbit) behaves correctly. For example, a faulted component of an Ozone measuring instrument may influence the measurements negatively. So, it is important to recognize malfunctioning as soon as possible. However, for a human controller it is just impossible

*TNO-TPD-TU, Instrumentation Department, P.O.Box 155, 2600 AD Delft, The Netherlands. bos-a@tpd.tno.nl.

†ESTEC-ESA, ERS Project Division, Earth Observation Department, P.O.Box 299, 2200 AG Noordwijk, The Netherlands. {jcallies,alefebvr}@estec.esa.nl.

PRECEDING PAGE BLANK NOT FILMED

PAGE 194 INTENTIONALLY BLANK

to monitor system's performance in every detail. An automatic system is needed to keep track of the system.

In this paper we describe a technique to validate systems: *model-based monitoring and diagnosis*. Note that there is no intent to replace existing test techniques; it is an *additional* method that is used to detect the errors that remain after traditional testing and when the system is in operation. The test method here presented is model-based. That is to say, a behavior description is used to predict how the system should behave, and the predictions are compared to the actual observations. If an inconsistency arises, then it is assumed that something is wrong and an error is signaled.

1.2 GOME

We have applied the test method to verify the GOME instrument (ESA 1993). GOME, short for Global Ozone Monitoring Experiment, is an instrument that will be mounted on ESA's ERS-2 satellite. Its purpose is to measure Ozone concentrations in the earth's atmosphere. This is done by comparing the sun's spectrum measured directly to the spectrum of sun light that has been reflected and travelled twice through the earth's atmosphere.

Apart from a diode array for measuring the spectra, the instrument has a number of supporting subsystems. Such as a command interpreter for interpreting and executing of commands send by ground control; a data acquisition unit for sending the measured spectrum and house keeping data to ground control; a mirror unit for scanning the earth's atmosphere; a heating unit for temperature control; etc.. All in all, GOME is a rather complicated system and its behavior is hard to verify.

1.3 Overview of the paper

In Section 2, we start by describing a monitoring system that is used to verify GOME's behavior. A monitoring system checks if a system is functioning correctly, however, the cause of a malfunctioning is not reported. This is part of the functionality of a diagnostic system. In Section 3, we extend the description to a diagnostic system that is currently being implemented for GOME. Finally, in Section 4, some conclusions are drawn and future work is described.

2 Monitoring

As already described we have implemented a *model-based* form of monitoring. We assume that something is wrong whenever the model's predictions are contradicting the observations of the system's behavior. That is, a description of normative behavior is used to verify the system.

In this section we will (1) formalize the method of model-based monitoring in a way so that it is easily extended to model-based diagnosis, see Section 3; (2) describe the implementation of it; and (3) discuss some of the results of applying a monitor program to the GOME instrument.

2.1 Characterizing model-based monitoring

In this section we will establish a conceptual framework for defining model-based monitoring. Central in this framework, and also in that for diagnosis, is that we view a description of system behavior (a model) as a formal system. That is, the behavior description is a set of sentences taken from some kind of language with a logic attached. It is important to note that we do not restrict ourselves to predicate or first-order logics. To the contrary, we view formal systems in which algebraic or differential equations can be expressed as important candidates for logics in which the behavior of a system can be expressed. Viewing the behavior description as a formal system eases the definition and implementation of monitoring and diagnostic system, but may also introduce notations that may seem awkward in the context of system theory. For example, a numerical integration step is –in the logical context– considered as a derivation rule, *e.g.* Euler's can be stated as:

$$\frac{\begin{array}{ll} x(t) = C_1, & \text{with } C_1 \in \mathbf{R}^n, \\ x'(t) = Ax(t) & \end{array}}{x(t+1) = AC_1 + C_1,}$$

with $x(t) \in \mathbf{R}^n$ and $A \in \mathbf{R}^n \times \mathbf{R}^n$. The derivation of σ from a set Σ is denoted as:

$$\Sigma \vdash \sigma.$$

Consider for example the case of dynamical simulation. Let *SIMMOD* denote a dynamical simulation model and *INIT* its initial conditions both expressed in some formal system with Euler's integration step as derivation rule. Then the set

$$PRED = \{p : SIMMOD \cup INIT \vdash p\}$$

contains all the predictions that can be obtained by applying the derivation rules of the formal system.

Using the logical terminology, we define a system to be monitored as follows:

Definition 2.1 *A system to be* monitored *is a triple* $(OBS, MODULES, SD_m)$, *where*

- *OBS, the observations, is a finite set of observations each of the form*

$$v = \langle value \rangle,$$

where v *is a variable, and* $\langle value \rangle$ *a value of appropriate type.*

- *MODULES, the modules, is a finite set of so-called* modules. *Modules are introduces to denote subsystems that are supposed to be functioning independently. For each module a separate set of behavior relations is defined, as will be explained next.*

- *SD_m, the system description (for monitoring), is a finite consistent set of behavior relations for each module. The general form of a modular behavior specification is as follows:*

  $$M \supset \langle \textit{behavior relations for } M \rangle,$$

  where $M \in MODULES$ and $\supset$ denotes material implication *(if ..., then ...).*

  Due to a possible incomplete knowledge of the system, e.g. the current state is not known, we allow alternative behavioral relations per module; however, exactly one of these behavioral relations must be true. That is, each instance of "$\langle$behavior relations for $M\rangle$" is of the form:

  $$rel_1 \oplus \cdots \oplus rel_n,$$

  where rel_i is e.g. an algebraic or differential equation, and $\sigma_1 \oplus \sigma_2$ denotes the fact that either σ_1 or σ_2 is true, but not both[1]*. $\oplus$ is also called a* choice *operator.*

When monitoring a system, an error message must be generated whenever the predications made by the system description are contradicting the observations. A contradiction occurs whenever a prediction assigns a value to a variable that is incompatible to the observations[2]. Deciding whether two values are incompatible is problem and type dependent. For example, for real-valued variables normally a range on the values is defined; for variables with a discrete domain the values have to match exactly. Furthermore, because the different modules are assumed to be working independently, we can give an indication where something is going wrong by stating the module responsible for generating the contradiction. This leads to the following.

Definition 2.2 *Let $(OBS, MODULES, SD_m)$ be a system to be monitored. An error message for a module $M \in MODULES$ is generated whenever*

$$SD_m \cup OBS \cup \{M\}$$

is inconsistent[3]*.*

[1] $\sigma_1 \oplus \sigma_2$ is an abbreviation for $\sigma_1 \vee \sigma_2$, and $\neg\sigma_1 \vee \neg\sigma_2$.

[2] Because SD_m is assumed to be consistent, contradictions may only occur due to a mismatch between prediction and observations.

[3] Note that presence of M in the formula enables the use of its behavior relations.

It is important to note that we assume that only one module –and no combination of modules– is responsible for a contradiction. In other words, multiple faults (de Kleer and Williams 1987) are not captured by this definition (this will be handled in the section on diagnosis, see Section 3). Note, however, that during a monitoring session more than one module may generate an error message. If we recall the initial purpose of the monitoring system, *viz.* the verification if a system functions correctly, the restriction to single faults is not that serious. We assume that the modules are chosen so that one module captures the behavior of a subset of the system constituents. On the occurrence of an inconsistency, we know that the culprit is to be found within that subset.

2.2 Implementation

We have implemented a monitoring system to verify GOME's behavior. In Figure 1 the overall layout of the program is given. To simplify the implementation,

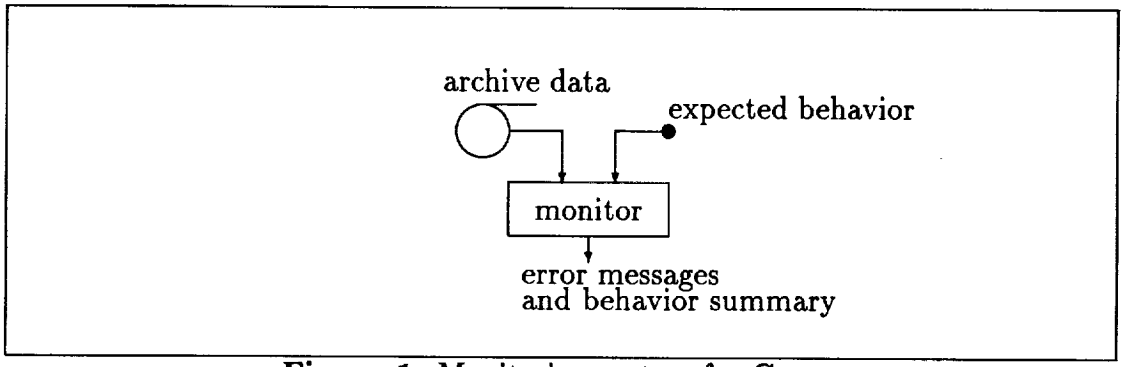

Figure 1: Monitoring system for GOME

the observations (OBS) of GOME's behavior are first stored on (Bernoulli) disks before GOME's operation is analyzed. A snapshot (the values of all GOME's variables) is taken each 1.5 secs. and is stored in what is called *archive data.* The contents of a single snapshot is called a *packet. Packet numbers* are used to address packets.

The *expected behavior* comprises the system description per module[4]. SD_m can be considered as a kind of simulation model of the system where the behavior relations are centered around the modules. Note that SD_m is not truly a simulation model because the choice operator introduces alternative behaviors per module. So, no conclusive predictions can be made using SD_m; it can only be used to do a consistency check.

[4]In the current implementation the program and the system description is coded in C (Kernighan and Ritchie 1978). The behavioral relations are decoded as procedures; a more elegant –at least viewed from a logical and a maintenance perspective– implementation would use a declarative description of both the behavior and the derivation (*e.g.* Euler's rule) relations.

The *monitor* program reads each packet from the archive data (OBS), 'simulates' each module M using the expected behavior (SD_m) and checks if

$$SD_m \cup \text{OBS} \cup \{M\}$$

is inconsistent. If so, an error message with some additional information is printed.

The following example gives some feeling for the implementation of the monitoring program.

Example 2.1: Consider the operation of the setting of the mirror's mode. Informally, SD_m contains relations that describe the following behavior:

> If a command that sets the mirror in swath mode is in the current packet, then after N packets[5] ($= N \times 1.5$ secs.) the mirror position is changing according to a linear relation defined on the packet number.

Now, if a mirror-setting command is found in the current packet, the monitor program checks after a delay of N packets the mirror position. □

As an extra, the monitor program prints for each packet –what we call– a *behavior summary* with the most important status information of GOME's operation. For example, the behavior summary contains the last submitted command, the mirror's and coolers' mode, and so on. This extra information comes in handy when the cause for malfunctioning is searched (either manually or automatically with a diagnostic system).

2.3 Results

The monitoring system as described above has been applied to the GOME instrument. It should be clear that a monitoring instrument does not perform a full functional test. Types of behavior that are not enabled during the verification process will not be tested for correct functioning. As we have already mentioned, it is an additional method of testing. Although GOME was tested fairly intensively, the monitoring program did expose a number of faults. To give some feeling for the type of faults, we name a few: (1) The integration time (for measuring sun light) was set incorrectly on a number of occasions; (2) synchronization faults of timers on receipt of a command; (3) a too slow operating timer; (4) inaccurate scan mirror positioning during swath mode; (5) documentation faults (other process variables are measured than documented); (6) etc..

3 Diagnosing

The monitoring program has been proved to be useful for validating the correct functioning of GOME. However, when an error message is generated, the cause

[5] Actually, this number N depends on the current packet number modulo 4.

for misbehaving has to be searched manually. It is interesting to have a system that not only recognizes that something is wrong, but also is able to find the cause of misbehaving. This functionality is part of *diagnostic* systems.

3.1 Characterization

Similar to model-based monitoring, the characterization of model-based diagnosis uses a logical terminology, see *e.g.* (Reiter 1987; de Kleer and Williams 1987; de Kleer and Williams 1989).

Definition 3.1 *A system to be* diagnosed *is -again- a triple* $(OBS, COMP, SD)$, *where*

- *OBS, a finite set of observations, defined as in the case of monitoring.*
- *COMP, a finite set of components. Components are akin to modules, however, behavior is assigned to individual components. In this way, it is possible to extract responsible components for a discrepancy in observed and expected behavior.*
- *SD, the system description (for diagnosis), similar to the model-based monitoring case, except that the behavior relations are defined per component.*

  In the diagnostic case we assume that a component working in a mode. *A mode represents a physical 'condition' (so to speak) of a component. For example, we have:*

  - *A* normal *mode, i.e. the component is working as intended.*
  - *One or more* fault *modes, i.e. the -faulted- component is working according to a known behavioral relation.*
  - *An* abnormal *mode, i.e. the component is not working as intended but we have not anticipated its fault behavior as in the previous case.*

  Now, the general form of a behavior relation is:

$$Mode(c) \supset \langle governing\ eq. \rangle,$$

  where $c \in COMP$, *and "*$\langle governing\ eq. \rangle$*" describes how the component's variables are governed when* c *is working in mode* $Mode(c)$. *If the* $Mode(c)$ *is the abnormal mode, then the equation is such that no predictions can be made.*

  To each mode of a component a prior *probability is assigned. This prior probability is used during the computation of diagnoses as will be explained shortly.*

SD can be considered as a component-centered simulation model. That is, behavior relations are given per component, so components responsible for a discrepancy can be isolated. In GOME's case we have the following.

Example 3.1: We consider two components: c_m and c_c representing the mirror unit and command interpreter, respectively. A normal functioning mirror unit (c_m) can scan the earth's atmosphere either at a fixed position or rotating:

- At a fixed position, indicated by the predicate $\mathit{fixed}(c_m, t)$ being true for all time instances t the mirror is fixed. The position of the mirror at time instance t has a constant value: $pos(t) = m_k$[6].
- With a scan angle, indicated by the predicate $\mathit{swath}(c_m, t)$ being true for all time instances t the mirror is rotating. The position of the mirror at time instance t has a value that is linearly dependent on t described by the function $f(t)$[7].

In Figure 2 the behavior of the mirror is given.

$$Normal(c_m) \supset \left[\begin{array}{lcl} \mathit{fixed}(c_m, t) & \supset & pos(t) = m_k \\ \vee & & \\ \mathit{swath}(c_m, t) & \supset & pos(t) = f(t) \\ \vdots & & \end{array}\right].$$

Figure 2: Mirror unit behavior

The command interpreter c_c sets, among other things, the predicates $\mathit{fixed}(c_m, t)$ and $\mathit{swath}(c_m, t)$ if a corresponding command has been received[8], see Figure 3. □

Now a *diagnosis* is an assignment of modes such that no predictions can be made that are contradictory to the observations. We use the following definitions.

Definition 3.2 *A* mode assignment *is a conjunction of mode predicates for all* $c \in COMP$*:*

$$\bigwedge_{c \in COMP} Mode_c(c).$$

[6]This is simplified, actually the position can be controlled.

[7]Again this is simplified; it is possible to control the maximum angle of rotation.

[8]It is assumed that once the predicate $\mathit{fixed}(c_m, t)$ or $\mathit{swath}(c_m, t)$ is believed, it stays true until is explicitly asserted false.

$$Normal(c_c) \supset \left[\begin{array}{lll} cmnd = fixed_{mirror} & \supset & fixed(c_m, t+N) \wedge \\ & & \neg swath(c_m, t+N) \wedge \\ & & \cdots \\ \vee & & \\ cmnd = swath_{mirror} & \supset & swath(c_m, t+N) \wedge \\ & & \neg fixed(c_m, t+N) \wedge \\ & & \cdots \\ \vdots & & \end{array}\right]$$

Note that N denotes the delay after which a scan mirror command is issued.

Figure 3: Command interpreter behavior

Definition 3.3 *A* diagnosis *for a tripel* $(OBS, COMP, SD)$ *is a mode assignment* Γ *such that:*

$$SD \cup OBS \cup \{\Gamma\}$$

is consistent.

Recall that assuming a mode assignment (Γ) results in a set of of governing equations describing expected behavior. If this set of equations predicts a value that is inconsistent with the observations, the assumption represented by Γ must be wrong. That is, the mode assignment Γ is not a diagnosis.

In general there are multiple diagnoses and computing all diagnoses can be very time consuming. However most of the times we are only interested in the *most probable* (de Kleer and Williams 1989). Using the prior probabilities of the modes we first test the most likely mode assignments for consistency. If the consistency test succeeds, the posterior probability can be computed by incorporating the number of observations that are explained by the mode assignment[9] as is described in (de Kleer and Williams 1989).

If a highly probable diagnosis Γ contains one or more fault (or abnormal) modes, it is likely that the corresponding components are the culprit.

Example 3.2: Consider the example of the mirror unit and the command interpreter again. Assume that we observe that the mirror is *not* moving after a swath command has been given. Using only these observations, we can only assume that either (or both) the mirror unit or the command interpretor is malfunctioning. However, the command interpreter controls other components

[9]Note that the mode assignment which assigns the abnormal mode to all components yields always a consistent theory, but does not explain any observation.

as well. So, if we see that, *e.g.*, the integration time is set correctly, then it is less likely that the command interpreter is malfunctioning and only the diagnosis stating that the mirror unit is the culprit remains. □

Implementation The implementation of the diagnostic system that is currently being under development is based on the work of de Kleer, Williams and Forbus, see (de Kleer and Forbus 1993). We make use of a best-first search algorithm in order to compute the most probable diagnoses, see (de Kleer and Williams 1989) for a detailed description.

3.2 Multiple models

The problem with contemporary diagnostic systems is twofold: (1) It is hard to construct a behavior model (SD); and (2) the computation of diagnoses is very hard.

Concerning the construction of a behavior model, one has to realize that in order to obtain non trivial diagnoses more than one aspect of system behavior must be described. For example, as Davis (Davis 1984) points out, for the detection of a solder-bridge between two pins of an IC, not only a electrical but also a geometrical model is needed. That is, one needs different *views* on system behavior. In case of space systems a lot of aspects, like electrical, mechanical, thermal, etc., play an essential role in the behavior of a system.

Concerning the computational hardness. In general, the computation of a set of most probable diagnoses is exponential in the number of components/relations in the behavior description. This means that there is no guarantee that a set of most probable diagnoses can be computed in acceptable time.

As solution for both problems *approximations* of behavior descriptions are propounded, see *e.g.* (Struss 1992; Bos 1994; Nayak 1994). There are two special types of approximations: *weak* and *strong* abstractions.

We start with weak abstractions.

Definition 3.4 *A system description SD_1 is* weaker *than SD_0 (the more accurate description), if everything that can be derived from SD_1 can also be derived from SD_0.*

Weak abstractions can be used to construct views, *i.e.*, models describing a single (or restricted set of) aspect of behavior. Other examples of weak abstractions include qualitative reasoning schemes (de Kleer and Brown 1984; Forbus 1984) for continuous systems, and temporal abstractions (Hamscher 1991) for digital systems. Weak abstractions can be used to speed-up reasoning using the following property:

Property 3.1 *If a combination of mode assignments (a conflict set) yields an inconsistent set (see Definition 3.3) for the weaker description, then that combination will also yield an inconsistent set for the more accurate description (Bos 1994).*

In general, reasoning over an abstraction is less costly than over the more accurate description. So, we may start reasoning over the abstractions to get (relatively) fast but coarse[10] diagnoses. If we like to refine the answers, we know that the conflict set found sofar need not be considered again. In this way we can prune the search space induced by the more accurate description.

Strong abstractions are defined as[11]:

Definition 3.5 *A system description SD_1 is* stronger *than SD_0 (the more accurate description), if everything that can be derived from SD_1 can also be derived from SD_0.*

Strong abstractions can be applied where the original description allows for a choice between two of more outcomes. For example, if the original model describes that either in this time instance or in the next a certain event occurs, the strong abstraction states one of the possibilities. Strong abstractions can also be used to speed-up reasoning by using the following.

Property 3.2 *If a combination of mode assignments yields an consistent set, i.e. a diagnosis (see Definition 3.3), for the stronger description, then that combination will also be a diagnosis for the more accurate description (Bos 1994).*

So, if one chooses one of the outcomes by selecting a strong abstraction and no contradictions are found, then in the more accurate description contradictions will also not be found.

In (Struss 1992; Nayak 1994; Bos 1994) *heterogenous* frameworks for multiple models are propounded. In these frameworks it is possible to have multiple abstractions of a given models and these abstractions can be stated in different languages. For example, both a qualitative model (de Kleer and Brown 1984; Forbus 1984) and a hierarchical abstraction (Hamscher 1991) can be used as an approximation of, say, a differential model. So, a modeler can select the formalism best suited for describing (an approximation of) system behavior. The result is a partial order on system descriptions, see Figure 4 for an example. In this figure, $SD_i \rightarrow SD_j$ denotes the fact that SD_i is an (either a weak or strong) abstraction of SD_j.

[10]Because the abstractions are weaker than the accurate descriptions we may, for example, oversee a diagnosis.

[11]It is important to note that stronger is not equivalent to more accurate.

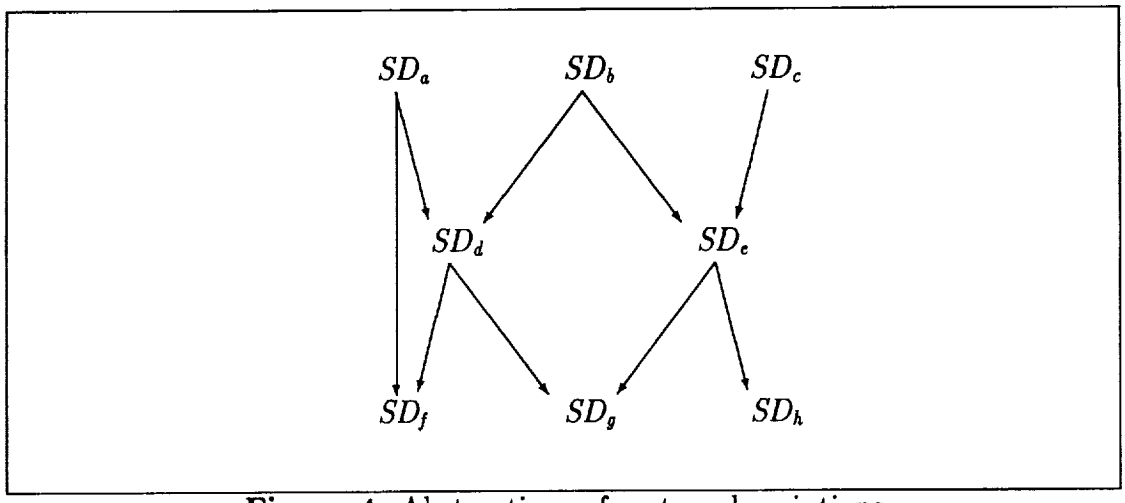

Figure 4: Abstractions of system descriptions

4 Conclusions and future work

Conclusions We have developed a monitoring system using a model-based technique. Such a system can be of great help for verifying system behavior. We have applied the monitoring system to the GOME instrument and revealed a number of discrepancies in expected and observed behavior. However, a monitoring system does not pinpoint the cause of malfunctioning; therefore, a diagnostic system should be used. A diagnostic system can be defined in a way similar to monitoring systems.

Future work We are currently developing a diagnostic system for GOME. The system will make use of abstractions in order to speed-up reasoning and to describe different aspects of system behavior.

Acknowledgements

Discussions with Joost Rosier (TNO-TPD) and Cees Witteveen (Delft University of Technology) improved previous drafts considerably.

References

Bos, A. (1994, October). Can abstractions be used to speed up diagnostic reasoning? In G. Provan (Ed.), *DX-94, The Fifth International Workshop on the Principles of Diagnosis*, New Paltz, NY, pp. 21–25.

Davis, R. (1984). Diagnostic reasoning based on structure and behavior. *Artificial Intelligence 24*(1), 347–410.

de Kleer, J. and J. S. Brown (1984). A qualitative physics based on confluences. *Artificial Intelligence 24*, 7–83.

de Kleer, J. and K. Forbus (1993). *Building Problem Solvers.* MIT Press.

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

de Kleer, J. and B. C. Williams (1989). Diagnosis with behavioral modes. In *Proceedings IJCAI-89*, pp. 1324–1330.

ESA (1993). GOME, global ozone monitoring experiment, interim science report. Technical Report SP-1151, ESA, Noordwijk, The Netherlands.

Forbus, K. D. (1984). Qualitative process theory. *Artificial Intelligence 24*, 85–168.

Hamscher, W. C. (1991). Modeling digital circuits for troubleshooting. *Artificial Intelligence 24*, 223–271.

Kernighan, B. W. and D. M. Ritchie (1978). *The C programming language.* Prentice Hall.

Nayak, P. P. (1994, October). Diagnosis with multiple theories. In G. Provan (Ed.), *DX-94, The Fifth International Workshop on the Principles of Diagnosis*, New Paltz, NY, pp. 217–226.

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

Struss, P. (1992). What is in SD? In W. Hamscher, J. de Kleer, and L. Console (Eds.), *Readings on Model-based Diagnosis.* Morgan Kaufmann Publishers.