Integrating Model-Based Monitoring and Diagnosis of Complex Dynamic Systems

Franz Lackinger
Wolfgang Nejdl
Christian Doppler Laboratory for Expert Systems
Technical University of Vienna,Austria
e-mail: lackinger@vexpert.dbai.tuwien.ac.at

Abstract

In this paper we present a new approach to model-based monitoring and diagnosis of dynamic systems. We extend related research work which concentrates either on monitoring (i.e. fault detection) or on diagnosis (i.e. fault localization) and therefore misses important aspects of effective and efficient troubleshooting.

The presented DIAMON algorithm[1] uses hierarchical models to monitor and diagnose dynamic systems. DIAMON is based on the integration of teleological parameter-based monitoring models and repair-oriented device-based diagnosis models. It combines consistency-based diagnosis with model-based monitoring and uses an extension of the QSIM-language for the representation of qualitative system models. Furthermore, DIAMON is able to detect and localize a broad range of non-permanent faults and thus extends traditional diagnosis which exclusively deals with permanent faulty behavior. The operation of DIAMON will be demonstrated on a real-world example in a multiple-faults scenario.

[1]DIAMON means **DIA**gnosis and **MON**itoring Algorithm

1 Introduction

Knowledge-based monitoring and diagnosis of dynamic systems has tremendously gained in significance over the last years. Early approaches in this field mainly used heuristically derived causal relations between faults and their causes to determine a technical system's faulty behavior. Although many implemented expert systems using these concepts offered a high runtime performance, they all suffered from well-known disadvantages like incompleteness and inflexibility.

On the other hand, research in the field of model-based diagnosis has typically concentrated on static technical systems (e.g. [Dav84, Gen84, Rei87, dKW89, FLN90]). Although these systems have been successfully applied to diagnosis in static problem domains (e.g. digital circuits), they are incapable of dealing with dynamic mechanisms which exhibit time-varying behavior.

Only a few model-based approaches trying to monitor and/or diagnose dynamic systems have been published in the past (e.g. [DK89, Ng90]). Most of them resort to qualitative simulation as an inference engine during the troubleshooting process to predict possible behavior patterns. Additionally, the architecture of the used qualitative system models is either non-hierarchical or contains discrete layers of abstraction.

Unfortunately, all approaches concentrate either on monitoring (i.e. fault detection) or on diagnosis (i.e. fault localization) and therefore miss important aspects of effective and efficient troubleshooting. We present an integrated algorithm which performs both tasks and which can therefore be used as a widely applicable general framework for troubleshooting dynamic systems.

We start in Section 2 with a discussion of the various problems that occur in the course of troubleshooting dynamic systems. In Section 3 we present our approach how to solve these problems and describe the important properties of DIAMON. The operation of DIAMON is shown on a real-world control problem - troubleshooting a central heating system. Section 4 discusses related research work and compares it with our approach, closing with some comments on future work.

2 Troubleshooting Dynamic Systems

In order to build an integrated control system doing both monitoring and diagnosis in dynamic environments, we have to be aware of the issues which are involved in such a task. This section will therefore be devoted to clarifying the relationship between dynamic and static diagnosis and monitoring.

2.1 Dynamic vs. Static Diagnosis/Monitoring

While diagnosis of a faulty component in a static system is a comparatively straightforward sequential process of fault detection, measurement selection and fault localization, this process grows more complex in a dynamic system.

Example 1 [Central Heating System:] Consider the central heating system depicted in figure 1. The figure contains the device components (e.g. ROOM, BOILER, RADIATOR, etc.) and parameters (e.g. $R_{Control}, TR_{Water}, T_{Room}, etc.$). The function of the central heating system is to guarantee that the actual room temperature T_{Room} equals the intended room temperature T_{wanted}. The function and the structure of this device will be explained in more detail in Section 3.

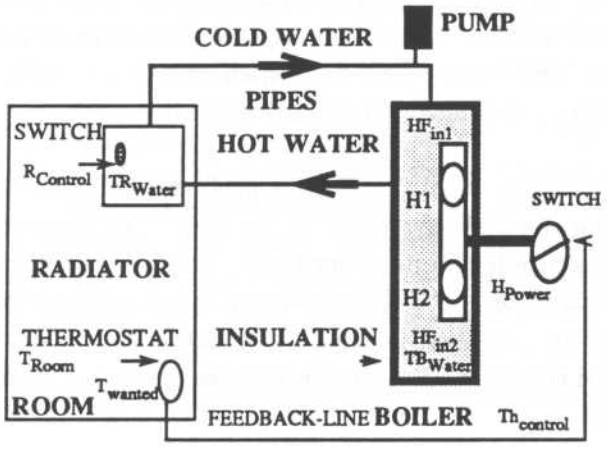

Figure 1: Central Heating System

In the course of monitoring and diagnosing this dynamic system, we have to solve the following problems:

Multiple Layers of Detail: Due to the use of several layers of models (monitoring models, hierarchical diagnosis models), we must take a changing system description and component set into account. In order to improve efficiency, real-time monitoring tends to minimize the amount of parameters under observation, and constraints are usually not component-oriented. In our example, checking the thermostat and the room temperature is enough for most monitoring purposes. Hierarchical diagnosis models use mostly component-oriented constraints at different abstraction levels. In our example, the most detailed level includes all the components depicted in figure 1. Hierarchical models can of course also be used in static diagnosis, but dynamic systems usually need a greater variety of model layers.

Temporal Observations: Observations consist of sets of measurements at different time points, determining the value of a specified set of parameters at different times. As a consequence, taking measurements sequentially is usually not feasible in a dynamic system as all parameters in a measurement set have to refer to the same system state (which changes over time). Additionally, the observation rate (i.e. the time period between measurements) is usually chosen such that all system states can be observed.

Time Dependent Behavior: We have to take various modes of correct behavior and various system states into account. For example, after turning the thermostat to a higher temperature, the temperature difference between room and thermostat temperature is perfectly normal. However, if they are not equal after a sufficient time period has elapsed, this difference leads to a conflict set.

Intermittency of Faults: Static diagnosis as in [Gen84] or [Rei87] assumes that the mode (*ok* or *ab*) of a component does not change during the diagnosis process (*non-intermittency of component modes*). However, this may or may not be recognized in the different situations which are characterized by the parameter sets. A dynamic diagnosis system has

to handle this *intermittency of faults*. For example, if one of the heating elements is not working, we detect this fault only after a sufficient time period has passed and the room is still not warm enough. Additionally, the monitoring system should be able to detect even *transient* faults, which may be caused by external events not modeled in the system.

Intra State Consistency: Some diagnosis engines (like [Dav84]) use multiple test vectors (parameter sets) which are assumed to be independent and describe only one system state. In dynamic systems we have to describe the system in several system states. Therefore not only conflicts of a parameter set with respect to the correct system state are relevant, but also conflicts describing faulty transitions between these system states have to be included into diagnosis. In our example, assume two system states *heat* and *cool*. The transition from *heat* to *cool* takes place, if $T_{room} > T_{high}$, while the system changes from *cool* to *heat*, if $T_{room} < T_{low}$. If such a transition takes place too early (i.e., if the heater turns off too quickly), we get a conflict with the specified behavior.

The above discussion shows that the current state-of-the-art of model-based diagnosis which is limited to static systems and permanently faulty behavior is unable to solve the discussed problems. It is the goal of our approach to extend model-based diagnosis with respect to dynamic systems and to solve most of the above mentioned problems.

2.1.1 Monitoring - a Necessity in Dynamic Systems

Having designed a structural and behavioral model of a static device we can unambiguously derive predictions about its correct behavior. Therefore, we do not have to monitor a static system as its behavior is precisely determined.

However, making predictions about the correct behavior of dynamic systems is a more complicated task.

If we resort to qualitative modeling techniques and use qualitative simulation for behavior prediction, we can usually predict more than one possible behavior pattern. Consequently this requires a monitoring phase in the troubleshooting process to detect inconsistencies between sets of observations and possible correct behavior patterns.

On the other hand, we claim that it is not necessary to observe all possible correct behavior patterns. It is sufficient to determine whether the device fulfills its function or not. For example, we will use a car long after some small deviations from its original behavior specification have occurred. In general this function of a device can be modeled according to its purpose TEL (which, for example, is expressed in first order sentences) by a teleological monitoring model (denoted with TEL_M) following the condition

$$TEL_M \wedge \bigwedge_{c \in COMP_{TEL_M}} ok(c) \models TEL.$$

Note that in the above definition $COMP_{TEL_M}$ will be the dynamic system itself (i.e. we can use the abbreviation $ok(device)$). However, we can easily extend this general concept to allow subpurposes of subparts of the system.

In contrast to [Fra89] who proposes the derivation of teleological models from an envisionment, we derive such models heuristically. We claim that a teleologic model contains meta-knowledge about the purpose of the device which can only be determined by a human expert and which includes such knowledge as which faults can still be tolerated.

While monitoring, we have to check only the accordance of the actual behavior with this teleological model under the assumption that the device works correctly.

Definition 1 (Monitoring) *Monitoring* is the process of testing whether

$$TEL_M \wedge ok(device) \wedge OBS_M$$

is consistent.

Usually fault detection alone is not sufficient and we need an additional diagnosis phase to provide sufficient information about fault localization for repair.

Additionally, we have to clarify how the choice of a qualitative model influences the monitoring and diagnosis process.

2.2 Viewing Qualitative Models from Different Perspectives

In order to clarify the differences between simulation, diagnosis and monitoring models, we have to analyze the task that uses these models.

2.2.1 Simulation Models

An important issue concerns the structure of qualitative models. In most cases simulation does not use hierarchical models. Although [FD89] introduce component-connection models which allow hierarchical modeling at an abstract level, the translated QSIM-constraints again use only one level and are therefore *flat*.

Doing simulation we are mainly interested in the behavior of the dynamic system over time. This behavior is optimally deduced from a structural model which uses time-varying parameters as its model primitives and which does not necessarily have to contain component-oriented knowledge.

Doing simulation we want to derive *all* behavior patterns with respect to the qualitative model no matter whether they describe faulty or correct behavior. Note that the term *all* refers to the chosen level of qualitative abstraction in the simulation model - of course we can not guarantee the derivation of the complete set of behavior patterns including all possible levels of abstraction.

The partial inclusion of models which describe physically impossible behavior is caused by the qualitative nature of the representation language and the problem-solving strategy of the simulation algorithm ([Kui85]).

2.2.2 Monitoring Models

We claim that the main task of monitoring is to check in real time if the purpose of the system is still fulfilled. A monitoring system observes under real-time constraints a small set of parameters and determines their correctness with respect to the system description. Consequently, efficiency is one of the main goals.

Therefore, a monitoring model usually contains only a few parameters and constraints necessary to describe the teleological purpose of the device. Additionally, such a monitoring model will not contain hierarchies in most cases.

Informally, we restrict the set of simulation models by adding teleology in accordance with the previous discussion about teleologic monitoring models.

Note that a monitoring model can be viewed as an instantiation of the system's teleology. In general, however, system teleology may cover a *broader range of purpose* than the monitoring model actually expresses.

2.2.3 Diagnosis Models

Simulation and monitoring models are both built using parameters and constraints. This parameter-oriented view, however, is not sufficient for diagnosis. Parameters can represent fault symptoms (e.g. the exceeding of thresholds), but faults are usually related to physical components. We assume here that the main aspect of diagnosis is *to provide information for repair*, and only mechanical devices (and not parameters) can be repaired [2]. We therefore have to relate purely parameter-oriented constraints (e.g. QSIM-constraints) to the device components that we want to diagnose and to repair.

Diagnosis models include component-connection information and are typically hierarchically structured. It is obvious that the lowest-level model has to contain only components which can be repaired, also called *smallest replaceable units* or *SRUs*. The choice of the SRUs depends on the availability of a repair-technician as well as on cost considerations.

An additional problem stems from the ambiguity of qualitative models. It is well-known that qualitative simulation produces behavior patterns that are physically impossible. These behavior patterns do not cause problems for diagnosis as they are physically impossible and can never occur in reality.

However, there exists a third class of derivable behavior patterns (in addition to correct and physically impossible), faulty behaviors, which can lead to diagnostic problems. Simulation of a qualitative model of correct behavior can result in behavior patterns which could stem from a fault model as well, even if correct and faulty behavior do not overlap in reality. This is caused by the inexactness of the qualitative model describing dynamic behavior, whose abstractions can sometimes remove the difference between different states.

We have already stated that simulation and monitoring models both are usually flat and do not contain a structured hierarchy.

Diagnosis, on the other hand, greatly benefits from hierarchical models. Complex real-world technical systems include intrinsic hierarchies which are related to the modularity or the purpose of the mechanism. Note that not only a device-oriented structural model can be hierarchical; we use hierarchical, parameter-oriented structural models as well to represent decomposable

[2]The adjustment of parameters is a struggle against fault symptoms, not against fault causes.

physical quantities (e.g. fluid flow).

We have of course described extreme positions for these models. Sometimes it might be useful to express sub-purposes of sub-mechanisms. On the other hand, diagnosis coupled with repair is also often oriented towards reconstruction of a teleology which may be different from the one used for monitoring. This topic is discussed in more detail in [FGN90b].

The above discussion shows that diagnosis models are more restricted than simulation models [3] but cover a broader range of expertise than monitoring models do.

In the above section we have clarified the problems that we have to solve in the course of monitoring and diagnosing dynamic systems. In the following part of the paper, we present our approach which solves most of these problems.

3 The DIAMON System

The following section introduces the **DIAMON** system, a qualitative reasoner for model-based diagnosis and monitoring, which incorporates the principles discussed in the previous section. After a short presentation of an application example which will demonstrate the capabilities of DIAMON we start with the discussion of some modeling aspects and continue with the presentation of the algorithm.

3.1 Example: Central Heating System

Figure 2 shows the constraint-network of the central heating system: the parameter model and the device model are connected by constraints. The device-model contains three hierarchies consisting of 1 component (Central Heating), 3 components (Room, Boiler, Pipe-System) and 8 components (Thermostat, Radiator, Heater1, Heater2, Switch, Insulation, Pipes, Pump).

We use a flat monitoring model and a hierarchical diagnosis model which consists of two levels. The monitoring level is modeled with 2 parameters and 1 constraint. The first diagnosis level (describing the 3 device-components)

[3] with respect to their qualitative expressiveness

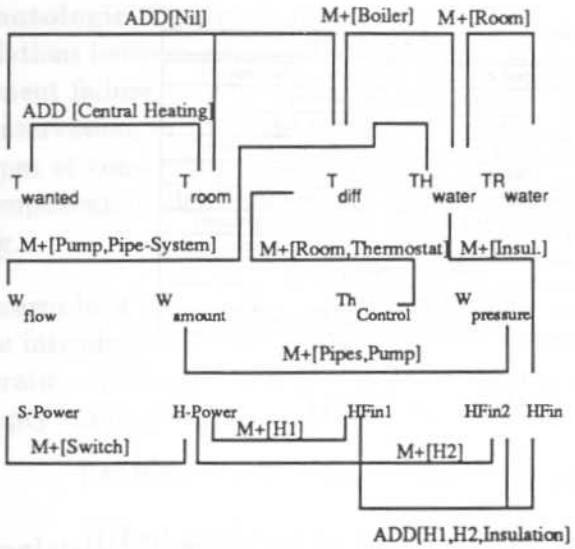

Figure 2: Constraint Network for the Central Heating

uses 7 parameters and 5 constraints, the SRU-level (describing the 8 device-components) which is used for repair consists of 16 parameters and 13 constraints.

Table 1 explains those parameters of our model that are used for monitoring and diagnosis.

3.2 Qualitative Modeling using Device Constraints

We extend the QSIM-language of [Kui86] by adding device-oriented knowledge to the constraints.

Definition 2 (Device-Constraint) The syntax of a *device-constraint* is grammatically defined as follows

$$Dev\_Constr := ((constr)(l\_of\_comp)(corresp\_values)^{+})$$

where $(constr)$ and $(corresp\_values)$ refer to the traditional QSIM definitions of $(constraint)$ and $(corresponding - values)$, and (l_of_comp) denotes the list of device components which are associated by the constraint.

PARAMETER:	EXPLANATION:
T_{wanted}	intended room-temperature (thermostat)
T_{Room}	actual room-temperature
TB_{Water}	actual water-temperature in the boiler
TR_{Water}	actual water-temperature in the radiator
W_{flow}	water-flow in the pipe-system
$Th_{Control}$	control signal from the thermostat (feedback)
$R_{Control}$	signal for the radiator-switch (exogenous)
$Switch$	signal for the water-heater switch
H_{Power}	Heating Power (total)
HF_{in1}	Heating Power from Heater 1
HF_{in2}	Heating Power from Heater 2

Table 1: Model Parameter

Example 2 The device-constraint

$$((M^{+}(PA\ PB))(COMP1\ COMP2)((lm1\ lm2)(lm3\ lm4)))$$

is interpreted as follows:

- parameter PB is a monotonic function of parameter PA
- the constrained parameters are associated to the device-components COMP1 and COMP2
- if the qualitative value of PA is landmark value lm1 (or lm3), then PB simultaneously has the qualitative value lm2 (or lm4).

According to the introduced device-oriented extension, we distinguish between four classes of device-constraints in our modeling language.

Teleologic Device-Constraints: A teleologic device-constraint contains parameters that are used during the monitoring cycle to detect a first fault symptom. It is obvious that the device itself (at its highest level of abstraction) is the device-oriented part in such a device-constraint.

Example 3 In our example, the following teleologic device-constraint states the desired relationship between the intended and the observed room temperature:

$$((EQUAL(T_{wanted}, T_{Room}))(Central\_Heating)((cold\ cold)(t_{room}\ t_{room})(hot\ hot)(inf\ inf)))$$

Tautologic Device-Constraints: These constraints exclusively denote relations between parameters that cannot cause inconsistencies due to a component failure. We use them to denote laws of nature, for example energy-conservation, or to relate computation parameters. In contrast to the other types of constraints a tautologic device-constraint is not associated to any component as it can not provide any diagnostic information for troubleshooting.

Example 4 In our example, we use a tautologic device-constraint to relate the intended room temperature, the actual room temperature and the temperature difference (which is a computation parameter). NIL denotes the empty component set.

$$((ADD(T_{diff}, T_{Room}, T_{wanted}))(NIL)((0\ cold\ cold)(0\ t_{room}\ t_{room})(0\ hot\ hot))))$$

Singleton Device-Constraints: Some device-constraints are explicitly associated to only one component. They express restrictions on internal operations of the component which do not affect its surroundings. An inconsistency between such a device-constraint and the observed values is therefore explained by the assumption that the component is behaving abnormally.

Example 5 A device constraint is associated to the radiator, that expresses a qualitative state of the radiator-signal:

$$((EQUAL(R_{Control}, ON))(RADIATOR)(on\ on))$$

Set Device-Constraints: This class of device-constraints associates at least two components. Although the included components are usually situated at the same level of abstraction, some set device-constraints might associate components at different levels of abstractions as well. This is a useful strategy for sophisticated focusing techniques during diagnosis. Clearly, an inconsistency between constrained values in a set device-constraint and observed values can only be explained by the fact that not all concerned components can be assumed to work correctly.

Example 6 The monotonic functional relation (expressed by an M^+ – *constraint* (see [Kui86])) between the amount of water in the pipe-system and the pressure of water associates the pipes and the pump:

$((M^{+}(W_{Pressure}, W_{Amount}))(PIPES, PUMP)((0\ 0)(p_{normal}\ full)))$

In this case, we use a set device-constraint to avoid the introduction of additional parameters which are not observable. Additionally, set device-constraints represent the behavior of connected groups of SRUs (we cannot replace an *abstract* component like PIPE-SYSTEM).

In contrast to [FD89], we do not view device-oriented models to be at a higher level of abstraction than parameter-oriented models are. Instead, we use different levels of abstraction for the integrated modeling language which concerns device- and parameter-oriented aspects simultaneously.

3.2.1 The Hierarchical, Qualitative Modeling Architecture

Monitoring and diagnosis of dynamic systems frequently requires hierarchically structured models. This is due to the complexity of dynamic systems as well as to the different tasks for which the models are used.

The modeling architecture of DIAMON consists of three layers:

Monitoring Layer: The monitoring layer of DIAMON contains a teleologic device-constraint model which is used to check for faults in the system. All parameters in the monitoring layer are continuously observed to guarantee fast fault detection. Clearly the choice of an optimized monitoring model (containing as few parameters as possible) is extremely important for the efficiency of monitoring.

Note that if the monitoring layer contains enough parameters to pinpoint the failure of a specific part of the device, then this focusing is done on the first diagnosis level. In this case the first diagnosis level could actually consist of the same parameters as the monitoring level, but additionally include device components.

Diagnosis Layer: The diagnosis layer of DIAMON contains hierarchically structured device constraints which are used for the dynamic refinement of fault localization. It is obvious that the complexity of a device is represented in the hierarchical architecture of the diagnosis layer.

Repair Layer: The repair layer of DIAMON contains device constraints which exclusively relate SRUs. If the diagnosis process reaches the repair layer (i.e. all actual diagnoses only contain SRUs), it switches control to the repair process.

3.3 Algorithm

The following section introduces the algorithmic principles of DIAMON. Using the central heating as an example we demonstrate how DIAMON detects and localizes multiple faults.

3.3.1 The Extended HS-DAG-algorithm

Previous approaches to monitoring have usually concentrated on the control of predefined thresholds. Conversely we view both monitoring and diagnosis from the consistency-preserving point of view which allows us to cover a broader range of detectable faults. The limits of detectability are due to the expressiveness of the underlying constraint language, not to the monitoring/diagnosis algorithm itself.

DIAMON is built on top of the *HS-DAG*-algorithm [GSW90] which is an improved version of the Reiter-algorithm [Rei87] for model-based diagnosis. The device-constraints are checked for consistency by a constraint-propagator. If an inconsistency in a constraint is detected, DIAMON adds the associated components to the conflict set.

We have extended the basic *HS-DAG*-algorithm to deal with dynamic systems similarly to the algorithm presented in [Ng90].

3.3.2 Dynamic Model Zooming

We have developed a continuous strategy of dynamic model zooming for the diagnosis layer. *Zooming* denotes the process of focusing the diagnosis process to the relevant parts of the model. In our current implementation we use a *breadth-first* zooming strategy which recursively zooms in the hierarchically deeper level for *all* components which are part of a diagnosis. This strategy guarantees a diagnostic process which is optimal w.r.t. the amount of detectable faults.

Best-first-zooming, on the other hand, allows the integration of heuristic information and takes only small sets of diagnoses into account at a given point of time. However, in comparison with breadth-first-zooming, a lot of faults are not detected which can question many safety-critical applications.

The careful integration of both strategies will be a topic for future research. We will allow the integration of heuristic information for model-zooming whenever possible and if no safety-criteria are violated.

3.4 The Monitoring/Diagnosis Cycle

Continuing the above discussion we present a formal definition of the DIAMON-algorithm.

Let M be the system description consisting of a sequence of qualitative models $m_0, m_1, ..., m_n$, Σ denotes a set of diagnoses Δ (following [Rei87]).

According to our hierarchical modeling concepts, m_0 is the monitoring layer, m_n denotes the repair layer and m_k *with* $0 < k < n$ are the models of the diagnosis layer. $m_{actual} \in M$ denotes the current working model used for troubleshooting.

Let P_{m_i} denote the set of parameters of model m_i and let C_{m_i} denote the set of components of model m_i for $0 \leq i \leq n$, $m_i = P_{m_i} \cup C_{m_i}$.

Let further *succ* be the binary relation $M \times (C_M \times C_M) \rightarrow M$ with $succ(m_i, \Sigma) = m_{i+1}$ for i = 0 to n-1. In our current implementation, *succ* is realized by a *breadth-first* zooming strategy which refines all device-components of a dynamic diagnosis.

Let further D_s denote a sequence of sets of measurements $d_0, d_1, ..., d_s$ with $cardinality(d_i) \leq cardinality(d_j)$ and $d_i \subseteq d_j$ for $j \geq i$. $d_{actual} \in D_s$ is the current working set of measurements.

Finally let $follow$ be the unary relation $D_s \rightarrow D_s$ with $follow(d_i) = d_{i+1}$ for i = 0 to n-1. DIAMON's implemented $follow$-relation extends $P_{m_{actual}}$ by $P_{succ(m_{actual})} = P_{m_{actual}} \cup follow(d_{actual})$.

In the following description, $CONSISTENT(m, d)$ denotes the call of the theorem prover which checks whether model m and measurement set d are consistent. The result of $CONSISTENT(m, d)$ is either the empty set or the set of conflicts *con*. $HSDAG(con)$ denotes the call of the HS-DAG algorithm which returns the set of diagnoses (i.e. the minimal hitting sets for *con*) and $READ(d)$ describes the observation process of parameter set d.

Accordingly the DIAMON-algorithm consists of four steps:

1. **INITIALIZATION:**
   $m_{actual} := m_0$;
   $d_{actual} := d_0$;
   $\Sigma := \emptyset$;

2. **MONITORING:**
   WHILE $((con := CONSISTENT(m_{actual}, d_{actual})) = \emptyset)$ *DO*
   $READ(d_{actual})$;

3. **DIAGNOSIS:**
   REPEAT
   $\Sigma := HSDAG(con)$;
   $m_{actual} := succ(m_{actual}, \Sigma)$;
   $d_{actual} := follow(d_{actual})$;
   $READ(d_{actual})$;
   $con := CONSISTENT(m_{actual}, d_{actual})$;
   UNTIL $(\Sigma \subset m_n)$

4. **(REPAIR:**
   REANIMATE; GOTO 1)

In this paper we concentrate on monitoring and diagnosis and do not discuss repair strategies. We are currently investigating and evaluating various strategies how to restore teleology in a dynamic system doing repair (see also [FGN90b]).

3.5 Example: A Fault Scenario in the Central Heating System

Applying DIAMON to the on-line control of the central heating system yields the following results for the detection and localization of a double fault.

Assume that the radiator is switched off and one heating element (H2) is permanently faulty. DIAMON correctly detects that T_{Room} differs from T_{wanted} and switches to the first diagnosis level. Additional parameters and constraints are zoomed in (state $s2$), i.e. the parameters are measured and the constraints are evaluated. This is done again in state $s3$ to reach the SRU-level. The diagnosis process finally localizes the double fault $[RADIATOR, H2]$.

Table 2 summarizes the control process (s0 - s3 denote the qualitative system states (see [Kui86])).

	$s0$	$s1$	$s2$	$s3$
T_{wanted}	T_{Room} std	T_{Room} std	T_{Room} std	T_{Room} std
T_{Room}	T_{Room} std	($Cold$ T_{Room}) dec	$Cold$ std	$Cold$ std
TB_{Water}			($Cold$ HOT) dec	($Cold$ HOT) dec
TR_{Water}			$Cold$ std	$Cold$ std
$Th_{Control}$			ON std	ON std
$R_{Control}$				0 std
$Switch$				$CLOSED$ std
H_{Power}				ON nil
HF_{in1}				ON std
HF_{in2}				0 std
Layer:	Monitoring	Monitoring	Diagnosis	Repair
Actual Diagnoses	nil	[Central-Heating]	[ROOM], [BOILER]	[RADIATOR,H2]

Table 2: Double-Fault-Diagnosis

4 Conclusion and Related Work

We have presented a new approach to model-based troubleshooting dynamic systems which uses an extended constraint language for hierarchical system models and which is based on consistency-preserving algorithmic concepts for monitoring and diagnosis.

Our work is closely related to that of [DK89] and [Ng90] and it can be seen as a continuation of their results. [DK89] present *MIMIC*, a sophisticated program for model-based monitoring based on qualitative simulation. In [Ng90], the algorithm of [Rei87] for model-based diagnosis is extended w.r.t. to the incremental diagnosis of dynamic devices. The DIAMON system differs from these approaches in some important ways.

First, we use a combined model-based algorithm which integrates both monitoring and diagnosis and which additionally uses hierarchical models and a dynamic zooming strategy.

Second, DIAMON's fault coverage is more complete than those of [DK89] and [Ng90]. In [DK89] faults can be missed due to the use of pre-simulated fault models which do not allow the detection and localization of unanticipated faults. The algorithm of [Ng90] can miss faults if the heuristically chosen incomplete sets of measurable parameters (as proposed in the example of [Ng90]) do not represent the actual fault scenario.

In contrast to [Ng90], we allow many-to-one relations between components and constraints and thus achieve more expressiveness in our qualitative modeling language. For example, we can relate components at different levels of abstraction.

The use of a hierarchical modeling architecture for troubleshooting is closely related to to the work of [Ham91] who distinguishes between a functional and a physical (repair) model for diagnosis of digital circuits.

We differ from [DK89] in that we do not use fault models and inductively derived fault hypotheses for monitoring. Rather, we rely on the behavior discrepancies to the correct behavior model, which allows us to handle unanticipated faults. Additionally, we do not have to simulate a sometimes large set of fault hypotheses, which leads to better complexity of our system (see also [FGN90a]). However, an integration of fault models or physical impossibility axioms along the lines of [FGN90a] is possible, if this proves necessary.

An interesting topic which we are currently investigating is the question of ontologies for reasoning about non-permanent faults in dynamic systems (pseudo-transient and transient).

We are also currently integrating this work with work done by the *Fault-Tolerance* community on reasoning about time in the context of mean time to failure (MTTF) and mean time to repair (MTTR).

Acknowledgements

This work was done in the context of the Christian Doppler Laboratory for Expert Systems, which is supported by the Austrian Industries. We especially thank Benjamin Kuipers who provided the Common LISP-implementation of QSIM (Q2) and several new research reports of his research group. Additionally, the discussion with our colleagues has been helpful in shaping our goals and solutions. Brian Falkenhainer and anonymous referees provided valuable comments on an earlier draft of this paper. Alois Haselböck implemented most of the DIAMON system as part of his M.S. Thesis [LH91].

References

[Dav84] Randall Davis. Diagnostic Reasoning Based on Structure and Behaviour. *Artificial Intelligence*, 24:347–410, 1984.

[DK89] Daniel Dvorak and Benjamin Kuipers. Model-Based Monitoring of Dynamic Systems. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1238–1243, Detroit, August 1989. Morgan Kaufmann Publishers, Inc.

[dKW89] Johan de Kleer and Brian C. Williams. Diagnosis with Behavioral Modes. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1324–1330, Detroit, August 1989. Morgan Kaufmann Publishers, Inc.

[FD89] David W. Franke and Daniel L. Dvorak. Component Connection Models. In *Proceedings of the Workshop on Model Based Reasoning*, pages 97–101, Detroit, 1989.

[FGN90a] Gerhard Friedrich, Georg Gottlob, and Wolfgang Nejdl. Physical Impossibility Instead of Fault Models. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 331–336, Boston, August 1990.

[FGN90b] Gerhard Friedrich, Georg Gottlob, and Wolfgang Nejdl. Towards a Theory of Repair. Technical report, Technical University of Vienna, November 1990. Submitted for publication.

[FLN90] Gerhard Friedrich, Franz Lackinger, and Wolfgang Nejdl. Redefining the Candidate Space in Model-Based Diagnosis. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pages 277–282, Stockholm, August 1990.

[Fra89] David W. Franke. Representing and Acquiring Teleological Descriptions. In *Proceedings of the Workshop on Model Based Reasoning*, pages 62–67, Detroit, 1989.

[Gen84] M. R. Genesereth. The Use of Design Descriptions in Automated Diagnosis. *Artificial Intelligence*, 24:411–436, 1984.

[GSW90] Russell Greiner, Barbara A. Smith, and Ralph W. Wilkerson. A Correction to the Algorithm in Reiter's Theory of Diagnosis. *Artificial Intelligence*, 41(1):79–88, 1989/90.

[Ham91] Walter Hamscher. Modeling Digital Circuits for Troubleshooting. *Artificial Intelligence*, 1991. to appear.

[Kui85] Benjamin Kuipers. The Limits of Qualitative Simulation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 129–136, 1985.

[Kui86] Benjamin Kuipers. Qualitative Simulation. *Artificial Intelligence*, 29:289–388, 1986.

[LH91] Franz Lackinger and Alois Haselböck. Qualitative Models for Simulation and Control of Dynamic Systems. In *AISB91*, Leeds, U.K., April 1991. Springer-Verlag. to be published.

[Ng90] Hwee Tou Ng. Model-Based, Multiple Fault Diagnosis of Time-Varying, Continuous Physical Systems. In *Proceedings of the IEEE Conference on Artificial Intelligence Applications (CAIA)*, pages 9–15. IEEE Computer Society Press, 1990.

[Rei87] Raymond Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32:57–95, 1987.