ZOR – Zeitschrift für Operations Research, Volume 32, page 251–269

A Sophisticated Expert System for the Diagnosis of a CNC Machining Center

By K. Althoff, K. Nökel, R. Rehbold, and M. M. Richter[1]

Abstract: As the domains of diagnostic expert systems become larger and more complex, purely associative approaches are no longer adequate solutions. A good example for such a situation is the diagnosis of a CNC machining center where the diagnostic process has a rich inner structure that has to be reflected in the system. In this article we describe three ways of extending the conventional expert system architecture. Firstly, our system uses a structural model of the machining center in addition to production rules. Secondly, we pay special attention to knowledge acquisition which is intimately related to the learning process. Finally, several temporal aspects of the diagnostic situation are addressed explicitly.

Zusammenfassung: Im gleichen Maße, wie diagnostische Expertensysteme in immer umfangreichere und komplexere Aufgabenstellungen vordringen, zeigt sich, daß ausschließlich regelbasierte, assoziative Ansätze keine angemessene Lösung mehr darstellen. Ein typisches Beispiel für eine solche Situation ist die Diagnose von CNC-Bearbeitungszentren, deren komplexe innere Struktur und die spezifische Vorgehensweise des Experten explizit in der Wissensbasis repräsentiert werden müssen. In diesem Artikel beschreiben wir drei Richtungen, in denen die "konventionelle" Expertensystemarchitektur erweitert werden kann, um den gestiegenen Anforderungen zu entsprechen. Unser System stützt sich auf ein Struktur- und Funktionsmodell des Bearbeitungszentrums, das das in Regelform gespeicherte Wissen ergänzt. Die Verwendung des Modells stellt neuartige Anforderungen an die Wissensakquisition, die durch eine enge Verzahnung mit dem Lernmechanismus des Expertensystems unterstützt wird. Schließlich gehen wir auf die zeitlichen Aspekte der Diagnosesituation ein, die eine spezielle Behandlung der dynamischen Maschinenprozesse und der sich darin entwickelnden Fehler erfordern.

Key words: expert systems, diagnosis, deep models, knowledge acquisition, learning, temporal representation, mechanical engineering.

---

[1] Klaus Althoff, Klaus Nökel, Robert Rehbold, and Michael M. Richter, Universität Kaiserslautern, Fachbereich Informatik, Postfach 3049, D-6750 Kaiserslautern, FRG.

0340–9422/88/3/251–269 $2.50

1 Introduction

Most of the existing expert systems deal with diagnostic problems. In principle to establish a diagnosis requires a kind of selection process, namely to isolate the correct diagnosis from other possible hypotheses. This seems to be easier than e.g. to construct a plan, where a completely new object has to be created. Indeed many diagnostic situations are of a structurally simple nature. This is particularly true if the relation between the observed data and the possible reasons for the outcomes of these data can be modelled in the form of production rules. This means that one is investigating a fixed static situation described by facts and rules; in more difficult situations object oriented approaches have been used.

For really demanding diagnostic problems such a static approach is clearly insufficient. Already a superficial view on people who perform the task of finding the reasons for malfunctioning of complex machine aggregates shows that they have to deal with a process of a rich inner structure. An adequate model can no longer consist merely of the data and possible causes for these data. The topic of such a model has to be the whole discourse of finding the diagnosis, i.e. the description of a certain mental process. This process incorporates a number of different activities which are based on and connected with various types of knowledge. Even in the case of the diagnosis of well-defined technical devices this knowledge is to a great extent incomplete, vague and of a heuristic nature. This is a simple consequence of the fact that we are no longer dealing with the machine alone but with the diagnostic process.

In this paper we will concentrate on three different aspects which our model has to reflect. The example which has led us to these investigations is the diagnosis of a CNC machining center. This is the subject of a project in the "Sonderforschungsbereich Künstliche Intelligenz" at the University of Kaiserslautern (West Germany) in cooperation with the WZL at the Technical University of Aachen. It will, however, not be explained here at all because for our present purposes it suffices to answer the questions under consideration in an abstract way.

Firstly our system is model-based. Therefore we have a model (on various degrees of abstractions) of the machine and all questions concerning the machine are solved with respect to that model. The second part is addressed to knowledge acquisition which is strongly connected with the process of problem-solving and the structuring of the knowledge and to some extent cannot be disconnected from learning. Finally we discuss why the diagnostic process is an activity in which time plays an important role. We argue that the structuring of temporal aspects goes far beyond what has been modelled in the various "temporal logics".

2 A Model-Based Approach to Diagnosis

2.1 Why Model-Based Diagnosis?

Up to now most expert systems for diagnosis had little knowledge about the structure and functionality of the object they worked on. Instead, they used several "⟨pattern⟩ → ⟨finding⟩" rules, sometimes enhanced by intermediate diagnoses. These "shallow" rules represented parts of the compiled knowledge of some experts in the field, who themselves gathered it through experience and *their understanding of the way the machine works.*

[Chandrasekaran, Mittal83] point out that knowledge of the underlying structure ("deep" knowledge, also called a model) is not needed, if the compiled knowledge represents all the relevant pieces of that model. The problem in this thesis is of course that the transition from a model to shallow rules is by no means a simple one, since this is an important part of the procedure that makes somebody an expert! As long as there are no satisfying methods to get all relevant information out of the structure (and there is no reason to hope this will change soon) a combined methodology is needed: shallow rules from the expert for speed and deep models, maybe from experts, maybe from plans, for details, unusual cases etc. This approach to diagnosis corresponds to the way human experts attack faults: usually, they quickly find the faulty part using their compiled knowledge; but sometimes things happen that they had not thought of before and they have to use their knowledge about the structure of the machine to continue. Another aspect of human expert reasoning is the ability to select a suitable focus, i.e. the right level of abstraction.

Several approaches to model-based diagnosis have been explored, nearly all of them work on faults in electric digital circuits [Davis85, Genesereth85]. Expert systems for fault diagnosis in mechanical engineering however usually only use shallow knowledge (e.g. rules) without deeper understanding of the machine. There are several reasons why digital circuits are much easier to model than arbitrary machines:

- only one type of connection between components is needed: wires
- like programs in computers, circuits are digital, while machines seem to work more analogically
- in fault diagnosis for circuits there is little or no need for states of components (at least in the examples given in the literature)
- faults in circuits can be found by looking at them statically, while machine faults usually require a dynamic view
- time relations play a much more important role even in simple machines than they do in circuits

– circuits usually follow a global clock signal, while machine components change their states only due to local influence (asynchronous parallelism).

We will now discuss a possible solution to some of the aforementioned difficulties.

2.2 Concepts of the System

Our goal is to create a toolbox system that allows the user to model a machine for diagnosis: not only its structure but also the user's expertise (compiled knowledge), e.g. fault probabilities, simplifying relations. Modelling goes on in three steps:

1. Build up the assembly structure
2. Connect components that functionally belong together
3. Attach additional information (i.e. expertise)

First we need a way to represent the structure: Using a set of given primitive components and several connections we want to build up more complex assembly groups. These assembly groups then can become parts of a still more complex assembly group and so on, until we reach the model of the whole object (machine) to be diagnosed. Subsequently we will only use the word *component* to describe such a machine part, no matter whether it is a complex or a primitive one.

Independently from this structure oriented view we also want to model *systems*, i.e. groups of components that serve the same purpose (e.g. cooling system, power supply etc.) but are not locally connected. This is a way of thinking human experts use, too, when reasoning about faulty devices.

2.2.1 Component Hierarchy

For every concrete instance of a component we model a prototype component, i.e. a class in the sense of object oriented programming. These component classes form a *component hierarchy*, where all components are arranged into a tree depending on the degree of their functional specialization and/or technical realisation. The root of this component hierarchy is the most general component THING. In this tree, related component classes are linked by an *a-kind-of*-edge, not an *is-a*-edge.

Each component class contains information about its interfaces to the outside (called ports), its state and behavior (expressed in constraints between the ports), its composition, i.e. its subparts and internal connections (if it is not a primitive component) and some rules to diagnose typical faults in that part.

2.2.2 *Structure Hierarchy*

While the component hierarchy contains abstract components ordered by specificity, the *structure hierarchy* describes the concrete component and its sub-components. Using the description of the composition from the component hierarchy the internal structure of a component can be built up. Several instances of the same component class can be part of a structure hierarchy, if that subpart is used more than once, e.g. pistons in an engine. The concrete subparts (instances) are connected by *is-part-of*-edges to their concrete superpart. Now it is possible to expand the thus created subparts themselves into their parts, again using the information provided in the component hierarchy. It is easy to see that the structure hierarchy is a tree, too. Its nodes are instances of corresponding component classes, to which they are connected in an *is-a*-relation. Only components that unambiguously belong to the structure of a component are considered as subparts, i.e. the structure hierarchy models the *physical* part-of-relation.

Example:

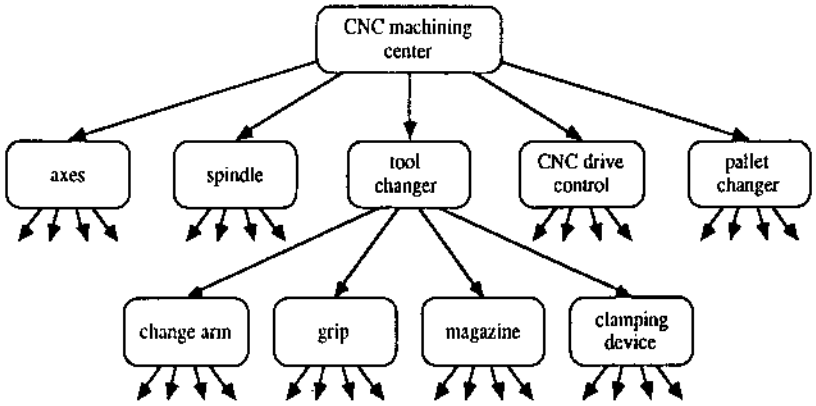

Fig. 1. Part of the structure hierarchy of the CNC machining center

2.2.3 *Functionality*

Beside the dependencies denoted by the is-part-of-relations there are a lot of other relations (connections) between components. These functional dependencies are added to the structure hierarchy as *connected-to*-relations. Information about the connections between subparts can again be found in the component class. By adding *connected-to*-edges to the structure hierarchy its tree character is lost; so we call it the

functional dependency net. Note that *connected-to*-edges can cause cyclic dependencies.

Different from e.g. electrical circuits *connected-to*-edges in mechanical engineering models can have several quite different interpretations (types), e.g. the flow of information, heat, electricity, material (liquid, gas), force etc. Different kinds of *connected-to*-edges of course can also model different intensities of connection.

2.2.4 Component Information

As mentioned above each component class contains several information about its component:

Interfaces (ports)

A component interfaces its neighbors by several types of input and output ports. These ports are its only connection to other components. Each port is connected to another port of the same type via a corresponding *connected-to*-edge.

Behavior and state

Outside of a component its behavior is described by the relations between its ports. This functionality is internally given by constraints between the ports. While these constraints only represent a static view of the component, changes in time also have to be modelled, e.g. using some measures of change. The behavior of a component is closely related to its state. State and behavior are abstracted and efficient versions of the relations that can be achieved using the composition of the component and the behavior of its subparts.

Composition

Each component class contains information about its subparts, the connections between these parts and the connections between some subparts and the ports of the component.

Primitive components need no composition information. A clever selection which parts to model as primitive is important to keep the cost of modelling low and efficiency high. For example it seems to be reasonable to model a component as a primitive if it is always completely replaced if faulty.

Ports of subparts are interlinked by *connected-to*-edges of appropriate types or connected to ports of the component itself via translation relations. *Connected-to*-edges between subparts represent the inner functionality of a part, while translation relations impart that behavior to the outside. Naturally, the behavior described through the functionality of the subparts and their connections should be the behavior directly described for the component.

Translation relations make similar port types of different abstraction levels compatible and can therefore link several ports of subparts to one port of the component.

[Davis85] describes a modelling system for electrical circuits, which allows parameterized components, i.e. meta-components that can be used to build several real components according to some parameter, e.g. an n-bit-adder can be instantiated to any adder for $n \in \mathbb{N}$. Even though this parameterization is not as helpful in mechanical applications as it is in electric ones it can help to keep the cost of modelling low.

Diagnostic module

For every component there is a diagnostic module tailored to the special faults of that part. While there is in principle no restriction on the methods to use for diagnosis we selected the usual production rules to gain profit from the research efforts that went into that direction. Several methods can be used to find faults in a component:

Rule-based examinations and conclusions as in usual diagnostic expert systems (here expertise is incorporated)

- Descent in the structure hierarchy for better localization of some fault, if not already in a primitive component
- Fault simulation by "constraint suspension" [Davis85], i.e. checking out which subparts may have caused the fault by removing selected constraints from the functional dependency net.
- Use of the connections in the system the part belongs to, i.e. only regarding selected neighbors of the same system.

Each of these methods can use its own special focus, i.e. several subparts can be expanded to different levels to provide an optimal examination.

Whether it is sufficient to restrict our model tool to qualitative values is under investigation; sometimes it might be important to reason with quantities. Up to now temporal relations between state intervals and machine parameters are not well covered. The use of (static) qualitative measures of change (e.g. *constant, ascending, descending*) is insufficient to envision the behavior of a machine. Here aspects of simulation play an important role. So modelling with time is an essential field of research.

3 How to Get the Knowledge into the System

In this chapter we give a short description of the main problems we are dealing with while acquiring the necessary knowledge for our diagnostic situation. These problems can be formulated by the following questions which we will try to answer.

1. What kinds of knowledge are relevant for this diagnostic problem?
2. Which knowledge holders own this knowledge and in what kind of way can they be used as knowledge sources?
3. What are the structural consequences for representing this knowledge and modelling the diagnostic problem-solving process?

3.1 Kinds of Knowledge Needed for the Diagnosis of a CNC Machining Center

In our context of diagnostic problem-solving, as in many others, knowledge has to be described on at least three different levels:

1. the cognitive level,
2. the representation level,
3. the implementation level.

The first level serves for identifying the knowledge structures, which are important to find the diagnosis. The second level is used to construct a representation of this knowledge, which focuses on the main aspects of its structures and should be as natural as possible. The third level reflects the pragmatic necessity of efficiently using this knowledge representation.

These levels can be used to differentiate the areas in which knowledge appears, yet there exist other categories for classifying the considered knowledge.

1. Explicit Versus Implicit Knowledge (on the Representation Level)

Knowledge which is directly available without transformation is explicit knowledge. When needed during the problem-solving process implicit knowledge can be generated by the system based on explicit knowledge.

2. Explicit Versus Implicit Knowledge (on the Cognitive Level)

Within the field of Cognitive Science especially the concept of *implicit* knowledge is used in a different sense. It denotes knowledge the respective owner is *not* conscious of and has difficulties in making it transparent to other persons. In this sense knowledge which can be made transparent without problem is called *explicit*. There is much evidence (cf. e.g. [Berry87]) that some kinds of knowledge are directly stored in an implicit form, therefore requiring conscious effort to make it accessible. Implicit knowledge that has originally been represented explicitly is also *compiled* knowledge.

3. Object Knowledge Versus Meta-Knowledge

Object knowledge is knowledge about the machining center (functioning, faults and corresponding causes, machine components, ...). Here the expert is absolutely competent. Meta-knowledge is knowledge about the diagnosis discourse, i.e. knowledge for controlling the object knowledge to govern the diagnostic procedure acquisition and learning processes and to generate explanations. Meta-knowledge (discourse knowledge) is a type of knowledge where the expert may go wrong.

4. Knowledge Consisting of a Partition of the Domain into Classes Versus Knowledge Consisting of Prototypical Examples

A *prototypical example* is a technical term in the field of Cognitive Science denoting a chunk of knowledge. Such an example describes a *class* of applications or things and it must be clear what is typical in that example and what is variable. A partition of the domain into classes is a hierarchical and more abstract way for describing knowledge, but the expert often uses examples to explain his domain.

5. Background Versus Foreground Knowledge

Background knowledge is that part of the domain knowledge which is necessary for a thorough understanding of the nature of the problem. Foreground knowledge is all the knowledge that has a direct influence on the diagnostic procedure.

3.2 Where the Knowledge is Stored and how it Can be Made Usable for the Project

As our diagnostic situation is embedded in the field of Mechanical Engineering, background knowledge mostly consists of the relevant knowledge of the affected subfield. There is plenty of literature available on the subjects of the principal construction and

functioning of machine tools, numerical controls, the production processes used and the necessary testing facilities.

The object knowledge can be extracted from the technical documentation of the machining center, which is not available in public, of course. This includes material about the construction of the machine, the control, the circuitry etc.

Beside the problems of acquiring all this object knowledge, the main problem for big systems is organizing such a high amount of information and identifying the necessary control knowledge. This discourse knowledge is hardly documented and exists mainly in the form of strategies and heuristics in the mind of the service technicians of long standing. Since these service technicians (as other experts, too) are not able to communicate their relevant knowledge completely (because much of it is implicit), here is the point where psychologically motivated knowledge acquisition methods come into play. For a survey of these methods see [Diederich87] or [Olson, Rueter87].

The approach we take is that one member of our knowledge acquisition team (an electrical engineer) takes part in a training in maintaining machining centers at "our" machine tool company. This training provides a kind of knowledge platform upon which a useful processing of all maintenance and repair documents of the last years becomes possible. This results in a measure for evaluating all the symptom-cause-attachments. Such evaluations enable the realization of trouble shooting plans (a kind of generalized flow-charts), which will be used for verification purposes during knowledge acquisition meetings at the machine tool company.

To get all the detailed discourse and object knowledge into the system, one of our aims is developing a knowledge acquisition procedure, which is adequately applicable at these knowledge acquisition meetings. For a lot of good starting points for this see [Prerau87]. Another aim is to develop a knowledge acquisition component, which enables the expert/knowledge engineer to think and act in the respective context using prototypical examples.

3.3 A System Design for Building Big Knowledge Bases

Although most knowledge acquisition systems have chosen a relatively general starting point for automatically classifying and structuring the domain knowledge, we want to be rather close to the respective application. The reason for this is our favoring of the expert's problem-solving behavior, which can be modeled sufficiently only if his way of thinking serves as a *detailed* guideline for the diagnostic procedure. Automatical classification and structuring tends to deviate from this guideline in the end. Since human problem-solving can hardly be separated from learning, this is an important part of our project. The main task here is to enable the system to draw analogous inferences from known examples based on a similarity measure for them.

Our starting point is a small prototype which has explicitly represented domain and control knowledge and in which the needed world of concepts is rather fixed. If the knowledge acquisition component is able to work on the representation of such knowledge a model of the system behavior is accessible for the knowledge acquisition process. This can be compared to the user's specifications (in the form of prototypical examples). Therefore it is possible to build an integrated test environment that allows to look at, execute and/or change many different aspects of the system.

Since examples are a very important knowledge acquisition medium, they will be directly representable within our system. Therefore user specifications can be represented easily. Using the examples the specifications can then be transformed into a more abstract control and knowledge structure. At present they include the following information:

1. Start context
2. Differentiating symptoms
3. Fault causes
4. Strategy how to find the fault causes beginning with the start context by determining the differentiating symptoms
5. Evaluation of the diagnosis strategy
6. Frequency of example occurrence

To give a taste of our explicit knowledge representation we now introduce contexts and meta-contexts as examples of basic knowledge structures:

Contexts

A context is the basic structure for modeling a concrete "part" of the program. Examples are diagnosis contexts (like *data provision, intermediate diagnosis, final diagnosis*, ...), machine component contexts (like *tool changer, magazine, numerical control*, ...), system contexts (like *input, output, browser, debugger, diagnosis, knowledge acquisition, learning, explanation*, ...) or example contexts (sample data). A context includes among other things production rules for diagnosis, production rules for context managing and constraints for modelling purposes.

Meta-Contexts

For every context there exists a corresponding meta-context which represents meta-knowledge of the considered context necessary for processing the context information. A meta-context includes e.g. meta-diagnosis rules or meta-context rules, which

define the strategy of the rule interpreter concerning processing the diagnosis or context rules, respectively.

The fact that both the object knowledge and the control knowledge must be represented explicitly for using a knowledge acquisition component, which bases on the above mentioned ideas, is not really a restriction for their applicability. The reason is that building a big knowledge base requires a very high amount of flexibility and system transparency on its own, which suggests an explicit knowledge representation not only for the object knowledge, but for the discourse knowledge, too.

4 Temporal Aspects of Diagnostic Situations

4.1 A Classification of Temporal Aspects

In the past diagnostic expert system projects have customarily chosen to concentrate on the representation of the static parts of the experts' knowledge, e.g. knowledge about symptoms, tests and faults. The basic assumption underlying this approach is that after a fault has occurred all relevant symptoms are observed simultaneously and the diagnosis is based on a static evaluation of this snapshot of the machine's state. Although this assumption is justified in many cases, critical remarks throughout the literature on diagnostic expert systems suggest that there are aspects of the diagnostic procedure or the faults to be diagnosed which require an explicit representation of their temporal properties. The literature on mechanical engineering (cf. e.g. [Weck85]) mentions temporal data, too, as a source of information for diagnosis. Naturally, the AI and mechanical engineering views of temporal aspects in diagnostic situations do not coincide completely. However, this does not necessarily imply that AI researchers work in disregard of existing knowledge; as seen from the mechanical engineer's point of view some of the aspects that are relevant to AI are instances of common sense reasoning rather than engineering expertise.

Nevertheless there is one important aspect that is common to both perspectives. Diagnosis is not an end in itself, but a means to the end of repairing the machine and restoring it to its full operational capacity. Any improvement of the diagnostic procedure (in particular any speed-up) serves to minimize the duration and, hence, the cost of the standstill. In this sense diagnosis and repair are "normal" components of the production process and are therefore subject to the same economic considerations as all the other parameters in production planning. As a consequence the supervision of the diagnostic process is a special case of a *planning situation* where the diagnosis is pursued only as long as the utility of the additional information to be gained out-

weights the additional effort. The principal decision criteria in this situation are the expected cost and duration of the tests, the penalty costs of the standstill and the urgency of special activities (e.g. due to impending danger). In process monitoring and trouble shooting the characteristics of the planning situation become even more pronounced because the real-time requirements of the process can only be met by judiciously interleaving the diagnostic actions with the primary process.

The second aspect that is repeatedly mentioned in the literature on mechanical engineering concerns *time series of process parameters.* In many cases faults do not manifest themselves in the form of isolated abnormal observations, instead they show up as significant perturbations of statistical aggregates over time (e.g. trends, mean values etc.). Statistical aggregation is a potentially dangerous operation since it eliminates the temporal character of the observations and thus information about the original data is lost. In the case of long time series this may justified because statistical significance is guaranteed by the sheer quantity of measurements; even then it may be potentially misleading to mix aggregates and single observations in e.g. a rule embodying diagnostic knowledge. But in a large number of cases termed *dynamic fault situations* faults are characterized by short sequences of events and machine states rather than long series. Consider for example the following example of a hypothetical fault that results in parameter A's value being below normal, then rising above normal for a period of time and finally dropping below normal again.

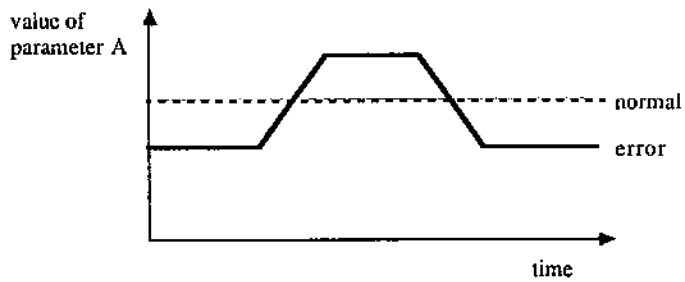

Fig. 2. A fictitious dynamic fault situation

In such a case no single "snapshot" of the machine's state can be an adequate description of the fault situation nor can such a fault be detected by any single measurement. The only way to detect it is to measure parameters at several points of time and to match the values obtained against an explicit description of the evolution of the fault situation over time. We will call such a description a *history* and the set of measurements an *observation sequence.* A very similar problem arises if a parameter is not directly observable but can be deduced from multiple measurements of other parameters.

Two other aspects can be found in the literature on diagnostic systems. In reality the sequence fault → observation → diagnosis → repair is rarely followed in its pure form, in particular diagnosis and repair often cannot be separated in this one-way fashion. In contrast to medical science it is a standard procedure in technical diagnosis to replace individual parts of the machine on a (guided) trial and error basis and thus eventually home in on the cause of the fault. Another situation which calls for an *alternation of diagnostic and repair actions* occurs when the machine has been damaged severely so that it has to be repaired provisionally before the diagnosis can be carried out. Reflecting this procedure in an expert system poses a number of difficult logical (non-monotonicity) and computational (frame problem) problems since some of the earlier observations will have been invalidated by the replacement whereas others will have not (this being the very rationale behind the strategy). A naive solution to the problem would be to restart the expert system each time a part has been replaced and to repeat all observations including those that have not been affected by the replacement. Obviously, this solution cannot be satisfactory since there would be no way for the system to directly compare observations from before *and* after the replacement which might well be crucial for the diagnosis. One way out is to partition time into the intervals between any two consecutive replacements and to index observations with the corresponding interval. As we can see, the only temporal property that remains relevant is the ordering of time whereas durations and more complex temporal relations do not play a role. This simplifies the situation up to a point where the predominant issue is the truth maintenance problem rather than the temporal representation.

Lastly, basing the diagnostic process on a deep model of the machine induces the need for temporal inferences. In the course of the *simulation* or the *envisioning* of the machine's behavior we have to keep track of the state intervals for the various machine parameters and of the temporal relations between them. This aspect has been mentioned before in the section on model-based diagnosis, so we will skip it here.

All in all we have discussed four temporal aspects of diagnostic situations.

Aspect 1: time as a decision criterion in diagnosis planning

Aspect 2: dynamic fault situations/histories

Aspect 3: interleaving diagnosis and repair

Aspect 4: envisioning/simulating the machine's behavior

Each of these aspects requires a specialized knowledge representation formalism and a suitable inference mechanism. In the rest of this section we are going to take a closer look at aspect 2.

4.2 Dynamic Fault Situations: the Representation and Matching of Histories

In paragraph 4.1 we introduced histories as an explicit description of evolving fault situations. If we take a look at conventional rule languages and try to accommodate histories in the terminological framework they provide, we notice that although they all permit complex rule conditions very few of them offer constructs to express the kind of temporal relations between the individual conditions that are needed to represent histories. E.g. in the rule

IF value of parameter A $>$ 30

AND value of parameter B $<$ 80

THEN component XYZ overheated

which is typical of conventional rule formalisms there is no indication whether A and B have to be measured simultaneously or merely in the course of the same consultation with any amount of time in between. Usually the implicit assumption is made that all observations have indeed been made simultaneously or – somewhat weaker – that the observation conditions have been kept sufficiently constant during the whole consultation so that variations of the parameter values can be neglected. While this is a useful abstraction in many cases, it definitely does not hold in the case of histories where parameters are measured at different points of time *in order to* detect changes. The weaker form of the assumption does not hold either: frequently the observer intentionally changes the observation conditions thereby inducing variations in the parameter values. Consequently the expressive power of traditional rule formalisms (more specifically: of traditional rule conditions) has to be extended.

This extension can be carried out in different ways. At the moment the choice of an optimal representation for histories is still an open problem. At least three of them which all have their individual merits and are currently under investigation have to be mentioned here.

4.2.1 Time Series Analysis

If we start from the procedure that is usually employed in mechanical engineering, the mathematical techniques of time series analysis (cf. e.g. [Anderson75]) seem to be the natural choice. They are suited especially well to the compact description of long time series. In the examples studied so far, however, the series are too short so that the advantages of a compact description do not compensate the conceptual and analytical overhead. Whether the break-even-point can be reached at all in diagnostic situations

cannot be decided at this point of time. Furthermore it has to be investigated for which applications the assumptions of the stochastic process models are satisfied. Although we suspect that the methods of time series analysis would have to undergo substantial changes to become useful for knowledge representation they cannot be ruled out as a candidate for the representation of histories on the basis of present evidence alone.

4.2.2 Temporal Logic

Another approach makes use of the various logics that have been proposed for the treatment of temporal information in AI. Historically, these temporal logics have many ancestors in mathematical logic. In AI they were first applied in two subfields which are unrelated to diagnosis: planning and natural language understanding. Many early problem solving programs ran into serious trouble when they assumed a static surrounding. Frequently, examples of "intelligent" planning behavior rely on the exploitation of parallelism or side effects which can be formalized only poorly using traditional situation calculi which were state of the art then. At the same time, but independently, linguists became interested in temporal logic because a better understanding of discourses seemed to require the knowledge of the speaker's intentions; the description of these intentions which include plans and goals raised similar questions as the plans in problem solving.

Among the various proposals for time logics especially those of James Allen [Allen83] and Drew McDermott [McDermott82] have gained wide-spread recognition. Both provide a basic vocabulary for time references and relations between temporal referents. In James Allen's time logic the basic referents are time intervals which can stand in any of the thirteen possible interval relations (before, during, overlaps etc.) to each other. Using intervals and relations we can describe simple histories by specifying sequences of consecutive state intervals for selected machine parameters. Consider the following (fictitious) example:

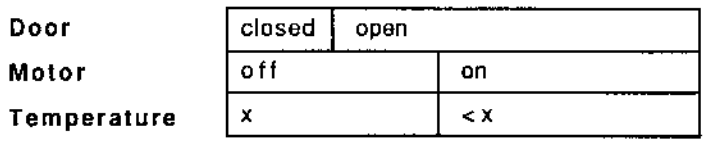

Fig. 3. State interval representation of a history

In the diagram time flows horizontally from left to right, the three bars correspond to the selected parameters and each state interval is marked off as a bar segment. The lengths of the segments are not significant; all that matters is the relative position of the intervals with respect to each other. [Allen83] discusses extensions of his basic logic (e.g. absolute time references, reference intervals, multiple time lines) which are needed to capture more complex diagnostic plans.

Let us suppose that the description of a history (as the one above) forms the condition part of a diagnostic rule that suggests a certain fault. Given such a rule the rule interpreter has to match the incoming observations against its condition. The result of a successful match is a mapping from the observations in the observation sequence to compatible state intervals in the history. At any time during the diagnostic process an initial part of the observations has already been made while the results of the other measurements are still unknown. In contrast to conventional pattern matching where the pattern and the data are specified completely we have to deal with a complete pattern (the description of the history) and incomplete data which accumulate over time. In many cases non-monotonicity is an inescapable consequence of such a situation. Interestingly, though, matching histories against observations is monotonous since a measurement that does not fit a history cannot be compensated by another observation later in the sequence (allowing for observation errors might complicate matters, of course). To take advantage of the accumulating data as soon as possible the matching operation can be carried out incrementally assigning a modal status to each history which is updated after each observation. This status also includes the information about the partial mapping from observations to intervals to which the pattern matcher has committed itself so far.

4.2.3 Discourse Representation Theory

The third approach is motivated by the knowledge engineer's view. Histories – or more completely, whole diagnostic plans consisting of observations and actions – have to be extracted from the expert's account of his knowledge. This knowledge is usually presented in the form of a natural language discourse. All proposals that we have seen so far have in common that the transition from the expert knowledge to the internal representation requires an interpretative intermediate step. As an example in Allen's time logic the characteristic machine parameters have to be singled out before the interval structure can be constructed. In general this may not be easy because the meaning contents of a discourse is to be represented in a language which was designed for a totally different purpose. If we accept this as the key problem the logical next step would be to use the formalisms developed by linguists explicitly for the representation of discourses. Even if their expressive power did not exceed that of the other

approaches, at least the existence of algorithms for the extraction of discourse representation structures from texts would make them highly attractive for knowledge acquisition.

A good candidate for a representation language is Hans Kamp's Discourse Representation Theory (DRT) (cf. e.g. [Kamp81], [Kolb83]) which has been used before for knowledge representation purposes. One of the objectives of the LEX project that was carried out jointly at the IBM research center at Heidelberg and at the University of Tübingen was the representation of legal texts in an advisory expert system. Although legal texts contain fewer temporal references than histories the LEX approach addresses a number of key problems that occur in the treatment of histories as well. Assessing the feasibility of an extension of Discourse Representation Theory for the representation of more complex temporal references is one of our current research interests.

5 Conclusion

We have pointed out that a comprehensive treatment of the diagnostic procedures in technical diagnosis has to include more than just factual knowledge in the form of production rules. The combination of rules with a structural model of the machining center improves the system's performance when confronted with unforeseen faults. Hypotheses can be verified by modifying the model in correspondence with the suspected fault and envisioning the results. Furthermore the diagnosis can be explained within the same conceptual framework that the expert uses. In contrast to other diagnostic applications which can be treated statically the temporal properties of the faults and tests have to be taken into consideration on different levels of abstraction. After a summary description of four important temporal aspects we have given a more detailed account of dynamic fault situations, their representation in an expert system and the pragmatics of their use in a rule-based setting. One should note that although for didactic reasons we have referred to the rule paradigm several times throughout this section, all of the ideas presented can be applied equally well in a model-based expert system. The acquisition and structuring of the different kinds of knowledge needed in our system poses a number of problems that need special attention and blend with the simulation of the expert's learning behavior.

References

[Allen83] Allen JF (1983) Maintaining knowledge about temporal intervals. Comm ACM 26/11:832–843

[Anderson75] Anderson OD (1975) Time series analysis and forecasting – the Box-Jenkins approach. Butterworths

[Berry87] Berry DC (1987) The problem of implicit knowledge. Expert Systems 4/3:144–151

[Chandrasekaran, Mittal83] Chandrasekaran B, Mittal S (1983) Deep versus compiled knowledge approaches to diagnostic problem-solving. International Journal on Man-Machine Studies 19:425–436

[Davis85] Davis R (1985) Diagnostic reasoning based on structure and behavior. In: Bobrow DG (ed) Qualitative reasoning about physical systems. MIT-Press, Cambridge, pp 347–410

[Diederich87] Diederich J (1987) Wissensakquisition. GMD-Arbeitspapier Nr. 245, St.Augustin

[Genesereth85] Genesereth MR (1985) The use of design descriptions in automated diagnosis. In: Bobrow DG (ed) Qualitative reasoning about physical systems. MIT-Press, Cambridge, pp 411–436

[Kamp81] Kamp H (1981) A theory of truth and semantic representation. In: Groenendijk JA et al. (eds) Formal methods in the study of natural language I. Amsterdam, pp 277–322

[Kolb83] Kolb H-P (1983) Aspekte der Implementation der Diskursrepräsentationstheorie. Universität Tübingen, Forschungsstelle für nat.-sprachl. Systeme

[McDermott82] McDermott D (1982) A temporal logic for reasoning about process and plans. Cognitive Science 6:101–155

[Olson, Rueter87] Olson JR, Rueter HH (1987) Extracting expertise from experts: methods for knowledge acquisition. Expert Systems 4/3:152–168

[Prerau87] Prerau DA (1987) Knowledge acquisition in the development of a large expert system. AI Magazin Summer 1987, pp 43–51

[Weck85] Weck M (1985) Werkzeugmaschinen – Bd 3 Automatisierung und Steuerungstechnik. VDI-Verlag, Düsseldorf

Date/Location	Conference	Information
30.05. - 01.06.88 Pisa (Italy)	International Workshop on Generalized, Concavity, Fractional Programming and Economic Applications	Dipartimento di Statistica e Matematica Applicata all'Economia Prof. Alberto Cambini Via Ridolfi 10, I-56100 Pisa Tel.: 050/598201
14. - 16.06.88 Oulu (Finland)	3rd IFAC/IFIP/ IFORS/IEA Conference on Analysis, Design and Evaluation of Man-Machine Systems	Prof. Raimo P. Hämäläinen Inst. of Mathematics Univ. of Technology 02150 Espoo, Finland
27. - 30.06.88 Tokyo (Japan)	FTCS 18 The Eighteenth International Symposium on Fault-Tolerant Computing	International Congress Service Inc., Kasho Building 2-14-9 Nihonbashi Chuo-ku, Tokyo 103 Japan, telex: 0222-3585 ICS
19.06. - 01.07.88 Sicily (Italy)	International Summer School on "Nonsmooth Optimization and related topics"	Prof. Frances Giannessi Dipartimento Di Matematica Universita di Pisa Via Buonarotti, 2 I-56100 Pisa
06. - 08.07.88 Paris (France)	EURO IX TIMS XXVIII Operations Research Management Sciences and New Technologies	AFCET EURO-TIMS Congress 156, Boulevard Pereire 75017 Paris, France
11. - 13.07.88 Lissabon (Portugal)	International Workshop on Project Management and Scheduling	Prof. J. Pinto Paixao APDIO-CESUR-IST AV. Revisco Pais 1000 Lissabon, Portugal
18. - 22.07.88 Paris (France)	12th World Congress on Scientific Computation	The Secretary 12th IMACS World Congress IDN, BP 48 F-59651 Villeneuve d'AscQ Cedex
24. - 29.07.88 Lausanne (Switzerland)	ECCE 88 European Conference on Computers in Education	ECCE 1988 Prof. Bernard Levrat Centre Universitaire d'Informatique 12, rue du Lac CH-1207 Genève