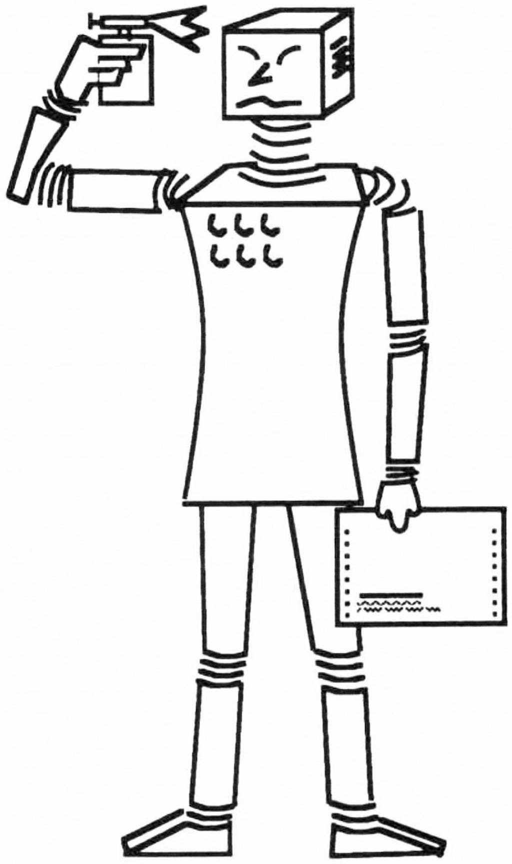
Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
D-6750 Kaiserslautern

SEKI - REPORT



Knowledge Acquisition in the Domain of CNC Machining Centers: the Moltke Approach

K.-D. Althoff et al

SEKI Report SR-89-09

Knowledge Acquisition in the Domain of CNC Machining Centers: The Moltke Approach

K.-D. Althoff et al
Fachbereich Informatik, Universität Kaiserslautern
Postfach 3049, D-6750 Kaiserslautern, W.-Germany

This research was supported by the Deutsche Forschungsgemeinschaft, SFB 314 (D2)

Knowledge Acquisition in the Domain of CNC[1] Machining Centers: the MOLTKE Approach[2]

K.-D. Althoff, S. Kockskämper,
R. Traphöner, W. Wernicke
University of Kaiserslautern
Dept. of Computer Science
P.O. Box 3049, D-6750 Kaiserslautern
West Germany

B. Faupel
Technical University of Aachen
WZL
Chair for Metrology and Quality Assurance in Production
Steinbachstr. 53B, D-5100 Aachen
West Germany

Abstract

MOLTKE is a research project dealing with a complex technical application. After describing the domain of CNC machining centers and the applied KA methods, we summarize the concrete KA problems which we have to handle. Then we describe a KA mechanism which supports an engineer in developing a diagnosis system. In chapter 6 we introduce learning techniques operating on diagnostic cases and domain knowledge for improving the diagnostic procedure of MOLTKE. In the last section of this chapter we outline some essential aspects of organizational knowledge which is heavily applied by engineers for analysing such technical systems (*Qualitative Engineering*). Finally we give a short overview of the actual state of realization and our future plans.

1. Introduction

The aim of the MOLTKE[3] -project is to show that a complex technical problem can be put to an economical solution using expert system technology. The domain of CNC machining centers is used as an exemplary domain. The solution encompasses the diagnosis of the CNC machine as well as the adaptation of such a diagnosis system to the further development of the machining center by the manufacturer.
The complexity of this problem requires rather an integration of different methods than straight forward knowledge acquisition (KA) methods. Firstly, an engineer has to interpret the construction plans of the machine and to interview the domain experts of the machine manufacturer. Secondly, a software tool has to be applied that enables the engineer to implement the diagnosis system using his own concepts and without having to know too much about the programming system. Thirdly, a natural knowledge representation is needed which can make the acquisition of the involved domain heuristics as easy as possible.

2. The Domain of CNC Machining Centers

Today, the development of expert systems for diagnosis problems has a high bearing on mechanical engineering. The motivation for the diagnosis of a CNC machining center is based on the requirement of increasing the reliability and applicability of complex machines. These depend on the expenditure for repair and service. The expectation of short duration for detecting and removing actual faults of machining centers can only be satisfied by using qualified service personnel with special education. The real situation in industry shows that the potential of appropriate experts is limited. It is expected to have instruments supporting diagnosis. The difficulty of fault diagnosis arises from the complex technical realization of these machines; the behaviour and function, of course, rests on the coordination of different machine components (electric, electronic, mechanical, hydraulic and pneumatic). Experts like service technicians from the machine manufacturer arrange this knowledge. Therefore, fault diagnosis can only be accomplished if it is possible to evaluate information about the

1) CNC = Computerized Numeric Control
2) The work presented herein was partially supported by the Deutsche Forschungsgemeinschaft, Sonderforschungsbereich 314, project X6 MOLTKE
3) Models, Learning and Temporal Knowledge in an Expert System for Technical Diagnosis

function of the machining center and its components. The task of building a fault diagnosis system requires a systematic analysis of the following knowledge classes:

1. Special technical knowledge (deep knowledge) and
2. Special heuristic knowledge (surface knowledge).

The characteristic components of machining centers are axes, spindle, tool and pallet changer and the control unit (CNC). The function of all components is realized with electric and mechanical elements. The specific knowledge about the machine components and different elements deals with:

- local features that show how these components work and
- the connections between the particular systems, which describe the global function of specific machine working cycles. The I/O-signals of sensors and components are important sources for the status of the connection ports.

Other kinds of technical knowledge are fault probabilities. One field of quality assurance is the analysis of weak points and the life-span of specific machine elements. One can draw conclusions from these results about those elements having a higher probability in diverse fault situations.
The geometrical accuracy of workpieces is very important in the manufacturing process. The cause for product deviations originates from the factors implied in process. The starting-point for the diagnosis of production faults are the data from special measuring helps (coordinate measuring machine, gauge, etc.). Such technical knowledge requires a deep understanding of the different involved subjects: control engineering, (mechanical) machine elements etc.
The significance of heuristic knowledge is shown by way of human experts solving the given fault problems, e.g. which strategies and abilities they use to recognize the relationship between different information. It is the quick and sure apprehension of apparently independent data which is a characteristic of experts. In (nearly) all fault situations they have a lot of similar fault examples (this leads us to the introduction of diagnostic cases in the next chapter).
The quality of a diagnosis system is essentially influenced by the appropriated information sources (electric and mechanical design plans, fault probabilities, machine and system manuals, motivated and talkative experts) and - that is decisive - the ability of structuring, connecting and evaluating all acquired details being used for building a knowledge base.

3. Applied Indirect Knowledge Acquisition Methods

The machining center we use for diagnosis (Maho MC600) is available in the mechanical engineering institute at the WZL Aachen. Therefore a significant amount of know-how concerning the functional behavior of the machine can be acquired from our cooperating engineer. Additionally, all the following information was evaluated for the realization of the knowledge base of MOLTKE:

Specific technical knowledge
- electric design plans
- mechanical design plans
- hydraulic design plans
- pneumatic design plans
- statistic quality assurence reports
- fault probabilities
- machining manuals
- measurement manuals
- error messages

Human experience knowledge
- observations of the service technician
- communication and discussions with other service personnel

Our engineer had the opportunity to take part in a special training program of the manufacturer during which he could interview expert sevice personnel. Service reports and plans of similar machines were obtained. Many other companies have been visited to get information both about the causes of deviation of the geometry of workpieces and the structure of the control unit. KA methods which have been applied are the result of studying all the mentioned source material and additional unstructured interviews with expert service personnel. For the acquisition of the heuristic knowledge, *diagnostic cases* have been introduced to "mirror" concrete diagnostic situations. They consist of a machine fault and all the symptoms which have been acquired by the respective service technician.

Such a natural knowledge representation is a minimal requirement, because of the KA problems described in the next chapter.

4. Knowledge Acquisition Problems

A main point for the evaluation of the specific technical knowledge are the already existing possibilities of the CNC for finding machine failures. In most cases the control unit provides an error message, if a malfunction during the machine cycles or an emergency stop can be noticed. The CNC generates warnings without stopping the manufacturing process, if no severe disturbances occur. One problem is the varying information content of the error messages. In some cases the fault causes can directly be read from the error statement, in other situations the message is only the starting point for the diagnosis process. The error messages are divided into two main fault classes [Pfeifer, Held, Faupel88]:

CNC fault message	PLC[1] fault message
system errors	I/O-errors
programm errors	axes- and spindle errors
data errors	
working errors	

The condition for creating PLC error messages depends on the observation of the temporal alternation of the I/O-signals in comparison with the correct function of the machine cycles. It is not quite trivial to select a diagnosis strategy based on an occurence of an error message only. First of all the temporal sequences of cycles of the machine parts have to be known. In general the service personnel looks for the machine status at the point of time when the error was determined. The machine part cycles can be described simply by the following sequence:

1. input signal
2. controlling different elements (switches, relays, valves, ...)
3. feedback impulse to the CNC

The exact correlation between the design and the realization of the machine cycles has to be obtained from several plans and diagrams. Thus the control information for individual elements can be acquired from the electric, hydraulic and mechanical design plans. The sequence of the part cycles cannot yet be derived from such plans. The access to this very important information about the function and behavior of the machine may be possible by testing the function of a real machining center or by extracting the details from complex PLC-programs. In general it is problematic to get this secret information belonging to the protected know-how of software companies.
Another problem of KA is the optimization of problem solving steps and strategies as a function of different fault situations. Appropriate strategies and steps of the diagnostic procedure are determined by the different goals that serve as a guideline for the respective service technician, like the orientation according to fault probabilities, the experience, the difficulty of testing and measuring, and temporal estimations of the expenditure of testing and measuring. All relevant facts have to be considered for finding optimized diagnosis steps. Often the fault probability suggests errors of electric switch elements, but in the case of hidden switches the necessary tests are too expensive. Usually less probable fault causes are verified in such cases. The known fault probability reflects the analysis of all types of machining centers. Characteristic weak points of particular machines cannot be derived from global statistic informations. An important problem is the classification of fault symptoms by degree of complexity:

difficult	easy
noise and vibration derivation	error message
smell derivation	state of electric and mechanical
temperature derivation	elements

The correlation of "difficult" symptoms to possible fault patterns is influenced by subjective decisions, i.e. a noted deviation in the normal noise level can only be used for diagnosis, if a

[1]) PCL = Programmable Logic Control

connection to the actual fault situation exists and - that's important - the noise impression can be combined with machine components.
The expenditure for acquiring the expert knowledge differs with the different subjects. The analysis of all information about electric components is extremely relevant for the diagnosis of our domain. The KA of mechanical sequences is essentially difficult and also unfit for diagnosis. The exchange of mechanical elements like the spindle bearing lasts more time than any exchange of electrical components. A further problem is the description of the necessary qualification and test methods for analysing mechanical faults. The minimal requirement the domain experts have to meet is to know all about the function and behaviour of CNC machining centers, as this is the central part of their education by the manufacturer. The real experience of these experts is developed (only) during their practical work as service technicians. The main problem of the KA is the identification of such heuristic knowledge.
The intention of the consultation of the expert is at first focused on the registration of the global methods that the expert uses in most fault situations. The generalization of the global methods leads to a systematic procedure of the diagnosis process. The next step is the analysis of the diagnostic procedure used in special cases. The exclusive experience, i.e. the intuition of unconventional handling of diagnosis steps, is only obvious in difficult diagnosis problems. By consulting several experts the own observation impresses different methods and strategies for diagnosis. Similar to other domains, different opinions of experts are usual. The possibility for an outsider knowledge engineer to decide between several expert opinions is still an unsolved problem. Finally, the KA with the "use" of a knowledge engineer is an iterative process for optimizing and completing the knowledge base.

5. Towards a more Direct Knowledge Acquisition Mechanism

To understand the ideas and the mechanisms applied in the KA tool we will describe, it is important to examine the underlying structures of the knowledge. Therefore we first give a closer look to the knowledge representation in MOLTKE.

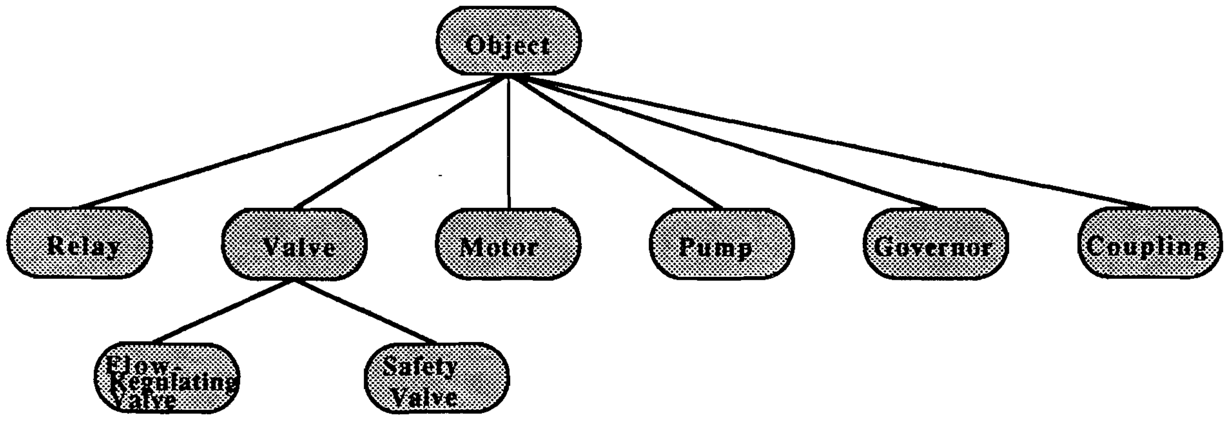
5.1. Knowledge Representation

In the domain of CNC machining centers, we have to deal with large knowledge bases. In order to organize and manage them, it seems necessary to split up the knowledge vertically as well as horizontally.

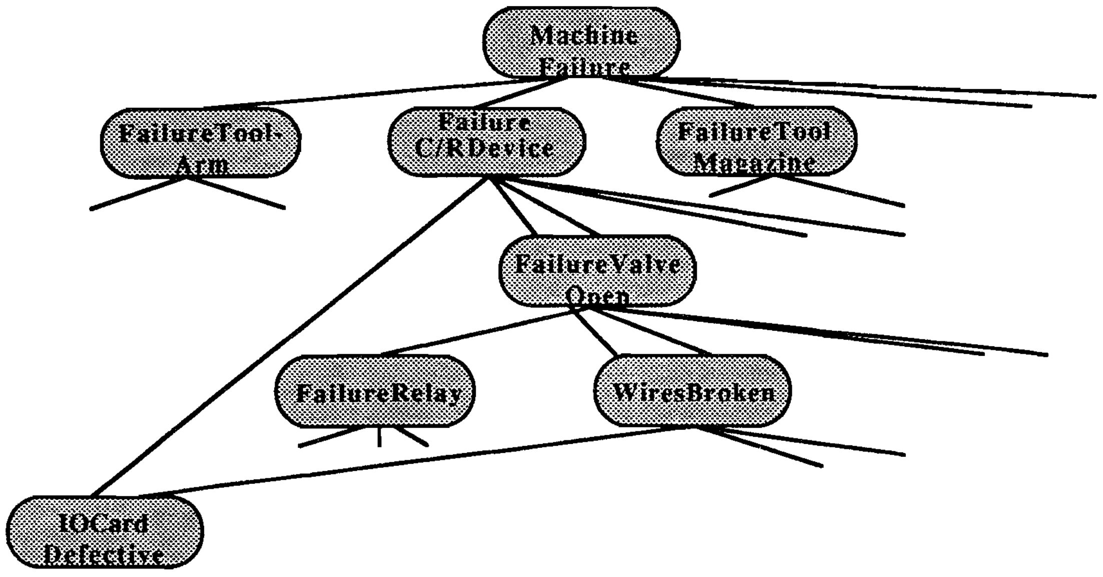
5.1.1 Vertical Modularization

In MOLTKE there are two kinds of taxonomy:

Component Hierarchy - Every component of the CNC-machine is represented as a frame. The components are structured hierarchically by is-a relations. Every component frame describes an abstract component that has the typical features of some part of the machine. Depending on the level of specificity, this hierarchy can extend over several layers. Concrete components are instances of component frames at the lowest level of this hierarchy.



Context Heterarchy - Besides this component hierarchy there exists a diagnosis heterarchy, in MOLTKE called context heterarchy. The knowledge in MOLTKE is organized on different levels of abstraction. This corresponds to the vertical modularization. The resulting modules are called contexts. Each context represents a diagnosis, which leads to more specific diagnoses represented by its subcontexts. Contexts are connected with their subcontexts by refinement links.



5.1.2 Horizontal Modularization

The knowledge represented in each context is subdivided in the following categories, which are explicitly and seperately represented.

Control Knowledge - Control knowledge is represented, as already mentioned, seperately from domain-specific knowledge. This is of decisive advantage for the development of large knowledge bases, because an application-oriented system often requires problem-specific control strategies. For this reason there is a need for a formalism that allows a simple description and modification of control strategies. In MOLTKE we use control rules to describe the proceeding of the system.

Knowledge about fault situations - In MOLTKE facts about a fault are represented as preconditions of contexts. These preconditions are declarative descriptions of the symptomatic pattern of a fault. In a given situation it is easy to check in the subcontexts, whether there is a fault or a reference to a fault. This is useful if there is data available that is not directly requested by the system, but sporadically supplied by sensors. Furthermore, the explicit representation of faults makes it possible to verify hypotheses of the user and allows an easier integration of a case-based mechanism, because diagnostic cases have a similar description.

Knowledge about Order - This kind of knowledge is necessary to answer the question: "Which symptom must be asked next ?". It contains a description of how to prove a fault, i.e. how a situation can be reached as quickly as possible, in which a fault can be verified. Knowledge about order is not necessary to choose a special test, but to choose which symptom is to be asked next. Possibly there are several tests for a symptom which differ in cost and time. The symptom itself determines which test is favourable at a given moment.

Determinations - Determinations represent correlations between symptom values. They can be either of functional or of empirical nature (for a formal definition see also 6.1). Functional correlations (total determinations) between symptom values can be taken from construction plans of the machine or can be generated automatically out of a model of the machine [Rehbold89]. Total determinations correspond to constraints between symptom values. They need to be used in one way (like rules), because in the diagnosis a defect in the machine is assumed. Partial determinations can be generated from diagnostic cases on the basis of empirical data.

An essential part of the knowledge base of MOLTKE is represented in the form of rules and formulas. For an efficient treatment of that knowledge a RETE-similar network is built. This network is used to evaluate the preconditions and rules of the contexts in an event-driven manner. The basis for evaluation is a 3-valued logic, that simplifies, among other things, the treating of incomplete knowledge.

5.1.3. Knowledge Acquisition

KA in MOLTKE is simplified by separating the different kinds of knowledge. The facts about the symptomatic of a fault is relatively easy to obtain from the expert. It is much more difficult to acquire the heuristic knowledge: "Which action must be done next to reach a diagnosis ?".
The following principles must be taken under consideration when ascertaining symptoms:

- minimization of cost and time
- concentration on the most relevant symptoms
- avoidance of a confusing order when ascertaining symptoms
- priority treatment of directives from the user

Often a diagnosis can be reached in different ways. The way which leads to a diagnosis can differ in cost and time of the tests which must be carried out. The decision of which way to choose depends on many criteria:

- educational background of the operating and service personnel
- experience of the operating and service personnel
- concrete diagnosis situation
- cost, time and accuracy of the tests
- frequency of symptoms/faults
- urgency of a diagnosis

There are six possible strategies resulting from these factors:

- importance-oriented strategy for choosing symptoms (i.e. symptoms which are relevant to detect a failure of a part of the machine)
- frequency-oriented strategy (statistical data about failures of a part of the machine)
- cost/time-oriented strategy (i.e. cost/time of the test which must be carried out)
- difficulty-oriented strategy (i.e. difficulty of the test which must be carried out)
- derivation-oriented strategy (avoiding useless questions to the user, because this strategy prefers symptoms which can determine other symptom values)
- individual strategy (depending on the experience of the user)

As the table shows, the proceeding during the diagnosis process is highly dependent on the qualification of the diagnostic staff, e.g. many tests are not allowed to the machine operator. It is important for the acceptance of the system, that it is flexible with regard to its proceeding, i.e. it must not pursue only one of these strategies, but depending on the kind and qualification of the user, it must put different strategies at the user´s disposal and allow him, at any time, to check the symptoms he wants to.

Requirements:

- realization of different strategies
- consideration of user´s directives
- possibility to combine different strategies

Realization:

a. According to the object-oriented paradigm a strategy object is defined for each of these strategies (strategy objects define a partial order on symptoms/tests). Every context receives a strategy object which is equipped with a default strategy before an exact strategy is specified. This default strategy can be set by the user at the beginning of the session or can be fixed by the system. Furthermore, the strategy object contains information about the number of symptoms that appear in a context, as well as details about the number of symptoms that fulfill a certain criterion (these slots are set up automatically by if-added demons).
b. There are ordering rules to establish exceptions from standard strategies (but also for individual strategies, e.g. proceeding in accordance to machine cycles).
c. There is a menu, available at any time and containing all unknown symptoms, that allows the user to determine the symptom he wants to.
d. The combination of different strategies is realized by the introduction of connection operators.
e. The combination of strategies and rules is also realized by connection operators.

tests / qualification level	1	2	3
opt. testing easy to recognize	D5 T5	D2 T1-2	D1 T1
opt. testing parts of the machine: relay, valves, motor,IO-state	D- T-	D4 T2	D1 T1
electr. testing resistance, stress, current	D- T-	D4 T1	D2 T1
hydr. testing components: pumps, valves, manometer, pipings, seals, flow-regulating valves	D- T-	D4 T3	D1 T2
pneum. testing tool holding fixture, worktable, access door, governors	D- T-	D4 T3	D1 T2
noise identification, interpretation	D10 T-	D4 T-	D5 T-
smell identification, interpretation	D10 T-	D7 T-	D5 T-
feeling identification, interpretation	D10 T-	D7 T-	D5 T-
control technology (metrology) governors, drive system, motor, measuring system,sensors	D- T-	D8 T5	D3 T4
control technology (CNC-technique) program technique, testprograms	D- T-	D8 T5	D3 T5
mech. testing gearing, guidances, bearings, couplings shafts,studs, shutters	D10 T5	D8 T5	D3 T5
methodology used instruments, methods	D- T-	D8 T5	D5 T3

Explanation:
qualification 1 --> machine operator
qualification 2 --> maintenance personnel
qualification 3 --> service technician
D = difficulty valuation: 1-10 - : not given
1 = very easy, 2-3 = easy, 4-6 = less difficult, 6-8 = difficult, 9-10 = very difficult
T = time valuation: 1- 5 - : not given
1 = very short, 2 = short, 3 = less long, 4 = long, 5 = very long

The determination of order when ascertaining symptoms can be done principally in three ways:

a. Establishing the order on the basis of syntactical criteria
b. Establishing it based on semantical criteria
c. Combination of syntactical and semantical criteria

Often there is much information missing, which is important for a useful proceeding in finding a diagnosis. That is the main problem, which arises by the incremental building of the knowledge base.

To examine this difficulty, we make a subdivision in four phases as a basis for building the knowledge base. Each phase corresponds to a certain level of knowledge of the knowledge engineer. According to this, the demands on the system are as follows:

- Possibility of incremental input of faults and symptoms
  There are two tools at the user´s disposal: FrameBrowser and RuleBrowser. The RuleBrowser is a tool for the input of rules and formulas as well as for their management and organization. The FrameBrowser serves for the treatment of the definition of symptoms, tests and faults. Both tools were created on the model of the SystemBrowser of Smalltalk-80, i.e. their use corresponds to the usual Smalltalk philosophy.
- Mechanism to determine the proceeding in finding a fault (i.e. the system is able to run, even if the syntactic and semantic criteria are incomplete)

We give now a simple example from the domain of CNC-machines as an illustration of the individual phases. Our example shows only some details of the complex frame-structure.

5.2 A Demonstration of the Mechanism

5.2.1. Declarative Phase

Which objects do exist ? - In this phase a description of faults (contexts) is carried out on a relative high level of abstraction (in the sense of simple diagnosis) and with it a description of the symptoms and tests.

context: MachineFailure	
precondition:	true
symptoms:	(ErrorCode, IN34)
refinements: (FailureClamping/ReleaseDevice, FailureToolArm)	
diagnosis:	MachineFailure
strategy:	default

context: FailureClamping/ReleaseDevice	
precondition:	(ErrorCode = I59)
symptoms:	(IN36,OUT7,Valve5Y1, OUT24,Valve5Y2,Wires, IN32)
diagnosis: FailureClamping/ReleaseDevice	
strategy:	default

Definition of the symptoms, appeared in this context:

symptom: ErrorCode	
type:	(I47, I67, I59)
importance:	very important
possible tests:	ErrorCode-check

symptom: IN34	
type:	(logical 0, logical1)
importance:	important
possible tests:	IOCard-check

symptom: IN36	
type:	(logical 0, logical 1)
possible tests:	IOCard-check
importance:	important
symptomGroup:	(OUT7, OUT24, IN32)

symptom: Valve5Y1	
type:	(switch-position 0, switch-position 1)
possible tests:	Valve-Check
symptomGroup:	Valve5Y2

Definition of the tests:

test: ErrorCode-check	
precondition:	qualification 1
output:	Please check the error code !

test: IOCard-check	
precondition:	qualification 2
output:	Please check the IO-state < > !

test: IOCard-Check	
precondition:	qualification 2
output:	Please check the IO-state < > !
difficulty:	qualification 2 -> easy to acquire qualification 3 -> very easy to acquire

test: Valve-Check	
precondition:	qualification 2
output:	Please check the switch state of the valve <>
difficulty:	qualification 2 -> less difficult to acquire qualification 3 -> very easy to acquire

symptom: Wires	
type:	(ok, broken)
possible tests:	Wire-Resistance-Check

test: Wire-Resistance-Check	
precondition:	qualification 2
output:	Please check the
difficulty:	qualification 2 -> less difficult to acquire qualification 3 -> very easy to acquire

All other symptoms are defined in the same way.

A diagnosis run in this phase would lead to the following order when ascertaining symptoms (under the assumption that the importance-oriented strategy is the default strategy):

Output:	Please check the error code !
Input:	I59

After the input I59 the context FailureClamping/ReleaseDevice is reached. The default strategy does not lead us anywhere, because there is no information about the importance of symptoms. Because of the existing details about difficulties of tests, it is switched to the difficulty-oriented strategy and the user will be informed of this change. The new strategy supplies the following partial order of symptoms:

(IN36, OUT24, IN32, OUT24, Valve5Y1, Valve5Y2, Wires)

The values of this symptoms will be asked to the user with regard to the above order (symptoms that are collected in a symptom group, will be asked together). In this case the only diagnosis can be FailureClamping/ReleaseDevice, because there are no further refinements of the context specified.

5.2.2. Refinement Phase

Which more precise relations do exist ? - In this phase an introduction of more detailed diagnoses in the form of intermediate and final diagnoses is carried out.

context: FailureValveOpen	
precondition:	(Valve5Y1 = close)
symptoms:	(Relay5K1, Wires, Diode, Stress)
refinements:	(FailureRelay,FailureWires, FailureDiode,FailureVoltage)
strategy:	default

context: FailureClamping/ReleaseDevice	
refinements:	FailureValveOpen
strategy:	default

symptom: Relay5K1	
type:	(open, close)
possible tests:	Relay-Check

test: Relay-Check	
precondition:	qualification 2
output:	Please check the relay < > !
difficulty:	qualification 2 -> easy to acquire qualification 3 -> very easy to acquire

symptom: Wires	
type:	(ok, broken)
possible tests:	Wire-Resistance-Check
importance:	important

test: Wire-Resistance-Check	
precondition:	qualification 2
output:	Please check the wires < > !
difficulty:	qualification 2 -> less difficult to acquire qualification 3 -> very easy to acquire

symptom: Diode	
type:	(short-circuited, not short-circuited)
possible tests:	Resistance-Check
importance:	very important

test: Resistance-Check	
precondition:	qualification 2
output:	Please check the diode < > !
difficulty:	qualification 2 -> less difficult to acquire qualification 3 -> easy to acquire

symptom: stress	
type:	(existing, not existing)
possible tests:	Voltage-Check

test: Voltage-Check	
precondition:	qualification 2
output:	Please check the voltage < > !
difficulty:	qualification 2 -> less difficult to acquire qualification 3 ->easy to acquire

context: WireBroken	
precondition:	(wires = broken)
refinements:	./.
correction:	Wire-Repair

A system run in this phase may possibly lead to a final diagnosis (and to a proposal for a correction) on a lower level of abstraction, because there are more contexts defined in the slot 'refinement' of the context FailureValveOpen.

5.2.3. Procedural Phase

How to use the objects defined in the previous phases? - The procedural phase introduces strategies and ordering rules for establishing a symptom order.

context: FailureClamping/ReleaseDevice		
Establishing the syptom order with ordering rules:		
(IN38 = logical 1)	-->	(check OUT7)
(OUT7 = logical 0)	-->	(check Valve5Y1)
(Valve5Y1 = close)	-->	(check Valve5Y2)
(Valve5Y2 = open)	-->	(check Wires)
(Wires = ok)	-->	(check IN32)
strategy:	importance-oriented before difficulty-oriented	

MachineFailure strategy <-- importance-oriented, FailureClambing/ReleaseDevice strategy <-- rules
In this phase the system holds more information with regard to the strategies. The uppermost context will now be started with the importance-oriented strategy. The context 'FailureClambing-/ReleaseDevice' will be provided with ordering rules, because the user has decided to go on in accordance to the machine cycles. After each ascertainment of symptoms, the system checks whether the precondition of a context is fulfilled. If this is the case, a context change takes place.
The context 'FailureValveOpen' contains as strategy a combination of the importance and the difficulty-oriented strategy. The connection operator *before* has the following meaning: First determine an order of the symptoms that has details about importance, then apply the difficulty-oriented strategy to the remaining symptoms. It is conceivable that there are more connection operators, but they are not explained here.

5.2.4. Phase of Realization

Which correlations between symptom values do exist? - In this phase the input of constraints for symptom values follows.

context: FailureClamping/ReleaseDevice		
Input of determination rules:		
(Valve5Y1 = closed)	-->	(OUT7 = logical 0)
		(IN38 = logical 1)
(OUT7 = logical 0)	-->	(IN 38 = logical 1)
strategy:		derivation-oriented before rules

This context gives now priority to the derivation-oriented strategy. This strategy prefers those symptoms which enable the system to derive other symptom values. At first, according to the connection, Valve5Y1 is asked, because of the state of this valve the values of OUT7 and IN38 can be derived. Otherwise, in the next step, the value of OUT7 will be asked. After this the ordering rules will be evaluated.

6. Towards more Automatical Knowledge Acquisition Methods

This chapter is related to an algorithm for generating determinations [Russell86] described in [Althoff, Kockskämper, Maurer, Stadler, Weß89]. We begin by defining the notion of determinations in the sense of MOLTKE. After giving a extensively modified version of the original algorithm we introduce a knowledge-based extension which enables the method to improve the capabilities of the underlying expert system. Some ideas for the type of the used knowledge are discussed subsequently. Finally, we describe our approach to extend the method to a more powerful one.

6.1. The Definition of Determinations

In MOLTKE determinations represent the relevance of a set of symptoms S1 to a set of symptoms S2, i.e. S1 determines S2 (S1 $\succ$ S2), if the fulfillment of S1 allows the derivation of the values of S2. The simple formal definition given by [Russel86] is:

$$P \succ Q :\Leftrightarrow \forall z\, [P(z) \wedge Q(z) \rightarrow \forall u\, [P(u) \rightarrow Q(u)]]$$

where, in our sense, the predicate P denotes all those symptoms which are common for some cases and Q some which are not. In the example above P(x) holds if the case x contains symptom S1 and Q(x) holds if case x contains symptom S2. We distinguish two kinds of determinations. A determination is called total if the values of S2 can be inferred from S1 with the probability 1, it is called partial if the probability of the inference is less than 1. The probability is computed by a determination factor δ [Russell86] :

$$\delta = \frac{|\{\, x \mid P(x) \wedge Q(x) \,\}|}{|\{\, x \mid P(x) \,\}|}$$

where P and Q are defined as denoted above. A partition of the interval [0,1] renders the attachment of each determination to one of the following probability categories: total, of-high-probability, probable, of-tolerable-probability, of-low-probability and unprobable, respectively.

6.2. The Syntactic Approach

The approach described in [Althoff, Kockskämper, Maurer, Stadler, Weß89] works on the similarity of concrete empirical cases, i.e. comparing cases with cases. This fact implies a number of problems, especially the main purpose of learning determinations is not taken into account.
Learning determinations will help to improve the actually known best way to do a diagnosis. In this sense the best way is the one which ascertains as few symptoms as possible. Instead, the original method generates as many as possible determinations, hoping that some of them are useful. We now give an improved method which compares cases with diagnoses.
First, we introduce some helpful terms:

(1) A case c is given by a set s of ascertained symptoms and an ascertained

hypothesis h written as $c_h(sm)$[1].

(2) A diagnosis d is given by a set s of symptoms to ascertain and a hypothesis h written as $d_h(sm)$.

(3) Let $d_h(sm_1) \dots d_h(sm_n)$, $n \geq 1$ be the already known diagnoses of hypothesis h. Then we define the set S_h of all known symptoms for h as $S_h := sm_1 \cup \dots \cup sm_n$.

(4) The set $\mathbf{D}_h$ of possible determinations for hypothesis h basing on S_h is defined as

$$\mathbf{D}_h := \{(\{s_1, \dots, s_n\} \succ s) \mid (s, s_1, \dots, s_n \in S_h) \wedge (s \notin \{s_1, \dots, s_n\} \wedge (\forall i,j \in [1..n]: (i \neq j) \Rightarrow (s_i \neq s_j)) \}$$

(5) Then, the set D_h of all known or, so to say, learned Determinations for hypothesis h is explained by $D_h \subseteq \mathbf{D}_h$

According to our aim of comparing cases with diagnoses, we have to define an order, which works on cases as well as on diagnoses. The subset relation induces a partial order on the set of all cases and diagnoses belonging to one hypothesis.

Definition: $a(sm_1) <_c b(sm_2) :\Leftrightarrow sm_1 \subset sm_2$ with sm_1, sm_2 defined as above and a,b $\in \{c_h, d_h\}$

Now we need to define in which case a diagnosis is minimal according to a given set of symptoms and known determinations basing on the defined order $<_c$:

Definition: A minimal diagnosis $[d_h(sm)]^m$ is defined as follows: Assume S_h, D_h to be defined as above. Let $d_h(sm)$ be some diagnosis. Then $d_h(sm)$ is called minimal if the following condition holds: $sm = S_h - \{\sigma \mid \exists (\{s_1, \dots, s_n\} \succ \sigma) \in D_h \}$

According to the partial order $<_c$, the diagnosis $[d_h(sm)]^m$ is a minimal element.

Now, we give the modified algorithm, which compares the empirical cases to the minimal diagnoses:

<u>Input:</u> A set of cases c_{all} with some hypothesis h_1 to h_n

<u>Output:</u> A Modification of the sets D_{h_i} for all $i \in [1 .. n]$

<u>Algorithm:</u>

Create a partition such that $c_{all} = c_1 \cup \dots \cup c_n$ with $c_i = \{ c_{h_i}(sm_{i_k}) \mid$ case with hypothesis $h_i \}$.

```
For all $c_i \in c_{all}$ do
    For all $c_{h_i}(sm_{i_k}) \in c_i$ do
        If $(c_{h_i}(sm_{i_k}) <_c [d_{h_i}(sm)]^m)$ then $D_{h_i} := D_{h_i} \cup (sm_{i_k} \succ (sm - sm_{i_k}))$ endif
    endfor
endfor
```

The determination factor δ is computed as follows

$$\delta = \frac{|\{\chi_{h_i}(\sigma) \mid ([d_{h_i}(sm)]^m <=_c \chi_{h_i}(\sigma)) \wedge (\chi_{h_i}(\sigma) \in c_i)\}| + 1}{|\{\chi_{h_i}(\sigma) \mid (c_{h_i}(sm_{i_k}) <=_c \chi_{h_i}(\sigma)) \wedge (\chi_{h_i}(\sigma) \in c_{all})\}| + 1}$$

It might seem to be a problem to get the initial minimal diagnosis $[d_{h_i}(sm)]^m$, but all the information needed is represented in the described context heterarchy (see chapter 5.).

6.3. Cases and Some Knowledge-based Modifications

To improve the method still further, we will have a closer look to the empirical cases and their types. According to the purpose of the method we can distinguish the following types of cases:

(1) $c_h(sm_1)$, $[d_h(sm_2)]^m$: $sm_1 = sm_2$

[1]) We denote a single symptom by 's' and a set of symptoms by 'sm'

(2) $c_h(sm_1), [d_h(sm_2)]^m$: $sm_1 \subset sm_2$

(3) $c_h(sm_1), [d_h(sm_2)]^m$: $sm_1 = (sm_2 \cup \Delta)$ where $\Delta \neq \emptyset$ is a set of additional symptoms; $\Delta \cap sm_2 = \emptyset$

(4) $c_h(sm_1), [d_h(sm_2)]^m$: $sm_1 \subset (sm_2 \cup \Delta)$ where Δ as in (3) and $sm_1 \cap \Delta \neq \emptyset$ holds

Types (1) and (3) can be neglected because they do not represent any improvement of $[d_h(sm_2)]^m$ and therefore there is no possibility to generate any determination. The already described algorithm of 6.2 handles type (2) in a satisfactory manner. The fourth type was not handled yet at all because the syntactical methods are not powerful enough. An examination shows that there are two interpretations to take into account:

(a) The symptoms of $(sm_2 - sm_1)$ are meaningfully replaced by the symptoms of Δ. To motivate this thought imagine, e.g. a service technician who uses a totally different strategy than the expert system. He could try to ascertain symptoms the expert system never would. But nevertheless the technician does a better diagnosis. In this case it is desirable to generate a determination which expresses this capability.

(b) There are only some additional redundant symptoms which have no meaning to the resulting diagnosis.

It is only possible to solve this situation if the system is equipped with some additional knowledge. To handle the appearance of such additional symptoms we follow the approach of introducing the notion of Qualitative Engineering. That means knowledge, in our terminology, on the level of the machine's principal technical organization. For example an information like "a relay is switched by an I/O-state".
Thereby we are able to handle interpretation (a) as follows: If it is possible to find an explanation for the additional symptoms according to the qualitative technical knowledge, then it is justifiable to assume that the actual case has interpretation (a). So the determination $sm_1 \succ (sm_2 - sm_1)$ will be generated where the condition $\Delta \subset sm_1$ holds. In principle the symptoms of Δ could be totally new. So it is possible to cover some modifications on the machine's construction, except those which result in actually unknown faults.

On the other hand, if we cannot find an explanation for a symptom of Δ it is not possible to decide whether we are able to handle the actual case as if it has interpretation (b) or not. Because we are not interested in generating an overhead of useless determinations, the operation we apply at the momentary state of our work is to do nothing.

6.4. Extending the Method

6.4.1. Motivation

So far, as described above, the method is strongly restricted. The similarity between two cases is only based on the syntactical equality of symptoms and the use of some explanations. Therefore it is not possible to consider such properties of cases like a more common symmetry. But symmetry can be a very interesting kind of similarity, which leads towards an intensified use of the notions of Qualitative Engineering and determination. This property of symmetry is often found in the currently examined domain of technical diagnosis. A reasonable fact, if the high complexity of this domain is taken into account. Imagine a CNC-machine which has to change the tools automatically. This mechanism has to fulfill two functions among others:

1. clamp a tool
2. release a tool

These two processes are reverse to each other. A tool is released by a hydraulic pressure onto a piston which opens the chuck. Then reducing the pressure clambs the tool. This contrary behavior continues throughout the whole mechanism. But in both situations the same error may occur, for example a piping of the hydraulics breaks or there is a defect in the control system:

Example: A similar case-context pair with reverse symptoms

a context	case no. 59
error-code(i59)	error-code(i34)
switch-position(valve_5Y1,1)	switch-position(valve_5Y1,0)
switch-position(I/O-state_IN36,1)	switch-position(I/O-state_IN36,0)
switch-position(I/O-state_OUT7,1)	switch-position(I/O-state_OUT7,0)
switch-position(valve_5Y2,0)	switch-position(valve_5Y2,1)
state(piping-system,ok)	state(piping-system,ok)
state(chuck,clean)	state(chuck,clean)
switch-position(I/O-state_IN32,1)	switch-position(I/O-state_IN32,0)
defect(I/O-card)	defect(I/O-card)

Now imagine a situation where, e.g. only the last three symptoms and the error-code of case no. 59 are given. Using the algorithm of the preceeding chapters, the comparison of such a case with the given context would not come up with a usable result.

6.4.2. Extension

The unsatisfactory result in the example is caused by the fact that the more complex similarity between the context and the case is not covered. Now, we introduce an extension which does so. The introduced algorithm for syntactical similarity-oriented learning of determinations uses the first definition given below. The similarity is determined by equality only. Our extension introduces an additional general transformation which replaces the simple equality.

Definition (old): Simple Similarity of symptoms. Let **S** be the set of all possible symptoms. The Simple Similarity $=_s$ of two symptoms $s_1,s_2 \in$ **S** is defined as follows:

$$s_1 =_s s_2 :\Leftrightarrow s_1 = s_2$$

Definition (new): Similarity of symptoms. Let **S** be the set of all possible symptoms. The Similarity $=_{si}$ of two symptoms $s_1,s_2 \in$ **S** is defined as follows:

$s_1 =_{si} s_2 :\Leftrightarrow \exists t,t' \in \mathbf{T} : [t(s_1) = t'(s_2)]$ where **T** is the set of all introduced transformations with $t \in \mathbf{T} \Rightarrow t : \mathbf{S} \rightarrow \mathbf{S}$

It is easily seen, that the old definition is a specialization of the new one if the identity is chosen as the transformation. Additionally, the definition of the subset relation has to be updated by the substitution of simple equality with our new definition. Now situations like the one mentioned above are covered:

Example: Generating determinations by using transformations

context	case no. 59*
error-code(i59)	error-code(i34)
switch-position(valve_5Y1,1)	switch-position(valve_5Y1,0)
switch-position(I/O-state_IN36,1)	
switch-position(I/O-state_OUT7,1)	
switch-position(valve_5Y2,0)	switch-position(valve_5Y2,1)
state(piping-system,ok)	state(piping-system,ok)
state(chuck,clean)	state(chuck,clean)
switch-position(I/O-state_IN32,1)	switch-position(I/O-state_IN32,0)
defect(I/O-card)	defect(I/O-card)

$$t := \begin{cases} \text{not} & s = \text{switch-position}(a,b) \\ & \text{with a,b out of the domain of definition of switch-position} \\ & \text{and not(switch-position(a,b))} := \text{switch-position}(a,\neg b) \\ \text{id} & \text{else} \end{cases}$$

Using the defined transformation t the algorithm of 6.2. is able to detect the similarity between the context and the case. Then the following symptoms are determined:

t(switch-position(I/O-state_IN36,1)) = switch-position(I/O-state_IN36,0)
t(switch-position(I/O-state_OUT7,1)) = switch-position(I/O-state_OUT7,0)

6.4.3. A Closer Look to Transformations and Qualitative Engineering

The question which now arises is what transformations really are? To give an answer on this we first describe where the transformations come from.
Transformations are gained out of the knowledge we denote as Qualitative Engineering. Remember the example above. There the used "not" can be derived from a fundamental rule which describes a common principle of machine organization: "If the task of a functional part of a machine is to undo an action, then this is normally done by applying the according steps backwards". Adding some basic information about the relations of valves, relais, I/O-state and so on as introduced in 6.3. we are able to generate the not-transformation. (Technically, the necessary knowledge aquisition is done by analyzing the concrete machine cycles using some methods of inductive learning). Thus our notion of transformation represents an operationalization of Qualitative Engineering.
In our sense Qualitative Engineering means to have knowledge about the correlations of basic machine components or, more precise, knowledge about the basic organization of the machine. E.g. valves can be controlled by relays. To use such a kind of knowledge it is not important to know how this control is physically done. Without organizational knowledge the use of transformations would not be practicable.
Qualitative Engineering could also mean to have knowledge about basic machine components and their fundamental function. E.g. a valve can be represented as a component whose function is to regulate material flows. To use such a kind of knowledge it is not important to consider properties of a special instance of a valve. At the moment we have not integrated this interpretation in our methods.

6.4.4. Usability

To show that this extension is useful, let us mention some of its advantages and their reasons:
Size - Less cases are needed to get a "good" set of determinations. In the example from 6.4.2 using the old, simpler definition of similarity causes the need of an additional case which fits better to the context than the given one. A comparison to case 59 would not supply the correct symptoms. The consequence is, that applying the extended definition causes a smaller case base to do good determination learning.
Competence - In addition to the preceding part (6.3), which described the use of the notion of Qualitative Engineering for the generation of explanations as an extension of the underlying syntactical method, the method´s competence increases by introducing transformations. As shown in the example, the use of the new definition implies applying additional knowledge not only to avoid making errors but also to increase the method´s power.
Necessity - It is possible to take advantage of similarities by introducing new meta-rules into the expert system which infer the rules of diagnosis for contexts like the given one, from the rules of a similar context by applying transformations. A smaller knowledge base will be the result. Keeping this in mind, it is necessary to have the ability of recognizing cases which are similar to a context in the sense of the introduced meta-knowledge. Otherwise the knowledge would get filled up with redundant and useless determinations.
Efficiency - The efficiency of the method increases because operationalising the additional knowledge implies not to have to generate an often used explanation for each symptom or case which is affected.

7. State of Realization

The KA mechanism has been realized to meet the main requirements our engineer had stated during the development of the knowledge base [Kockskämper89]. The component for the generation of determinations is currently under development [Traphöner89], [Wernicke89]. We will start soon with the implementation of the extension of this method and the integration of Qualitative Engineering into this component. There already exists a case-based reasoning system in MOLTKE [Althoff, Kockskämper, Maurer, Stadler, Weß89], [Stadler, Weß89] which is used in addition to the basic diagnosis system partially introduced in chapter 5. MOLTKE is implemented in Smalltalk-80 and runs on all workstations and personal computers on which the corresponding virtual machine is available [Althoff, Faupel, Nökel, Rehbold89].

8. Conclusion

The KA tool we presented in chapter 5 is intended to serve as a supporting mechanism for further works. Based on a successful integration of the methods for processing experience, introduced in chapter 6 and the underlying diagnosis system, we are actually working on, we hope to get towards a more automated knowledge acquisition. This includes at first, the automatical adaptation of the diagnosis system to a modified CNC machining center, and, at second, the vague possibility to generate a technical diagnosis system automatically by using a detailed machine model in connection with our case-based learning approach enabled by Qualitative Engineering. (See also [Rehbold89])

9. Acknowledgements

We would like to thank Michael M. Richter for introducing the idea of using Qualitative Engineering for our purposes in the above mentioned manner. Finally we would like to thank Klaus Nökel and Robert Rehbold for many constructive discussions within the MOLTKE project, which were essential for the initiation of some new approaches.

10. References

[Althoff, Faupel, Nökel, Rehbold89]
Althoff, K.-D., Faupel, B., Nökel, K., Rehbold, R.: MOLTKE.- to appear in: Extended abstracts of the workshop on knowledge-based diagnosis systems, GMD, Bonn, April 1989

[Althoff, Kockskämper, Maurer, Stadler, Weß89]
Althoff, K.-D., Kockskämper, S., Maurer, F., Stadler, M., Weß, S.: Ein System zur fallbasierten Wissensverarbeitung in technischen Diagnosesituationen.- Proceedings ÖGAI-Jahrestagung, 1989

[Kockskämper89]
Kockskämper,S.: Diskussion möglicher Wissensrepräsentations- und Inferenzmechanismen zur Fehlerdiagnose eines CNC-Bearbeitungszentrums und deren Implementierung als Expertensystem in Smalltalk-80.- diploma thesis, University of Kaiserslautern , 1989 (forthcoming)

[Pfeifer, Held, Faupel88]
Pfeifer, T., Held, H.-J., Faupel, B.: Aufbau einer Wissensbasis für Fehlerdiagnosesysteme von Bearbeitungszentren. VDI-Z VDI-Verlag 10/88.

[Rehbold89] Rehbold, R.: Deriving Causal Rules from Structure Descriptions in a Technical Diagnosis Domain, SEKI-Report 89-1 University of Kaiserslautern, 1989

[Russel86] Russel, S.J.: Analogical and Inductive Reasoning.- Ph.D.-Thesis, Stanford University, Dez.1986

[Stadler, Weß89]
Stadler, M., Weß, S.: Konzept und Implementierung eines fallbasierten, analogieorientierten Inferenzmechanismus und dessen Integration in ein regelbasiertes Expertensystem zur Diagnose eines CNC-Bearbeitungszentrums.- project thesis, University of Kaiserslautern, 1989

[Traphöner89] Traphöner, R.: Ein Konzept für die Verarbeitung von Erfahrungswissen in MOLTKE.- project thesis, University of Kaiserslautern, 1989 (forthcoming)

[Wernicke89] Wernicke, W.: Ein System zur Verarbeitung von Erfahrungswissen in MOLTKE.- diploma thesis, University of Kaiserslautern, 1989 (forthcoming)