Wojciech CHOLEWA, Wojciech MOCZULSKI

Department of Fundamentals of Machine Design
Silesian Technical University, Gliwice, Poland

KNOWLEDGE ACQUISITION IN SHELL EXPERT SYSTEM *MAS*

Summary. *The aim of this paper is to point out the selected problems appearing in processes of knowledge acquisition for expert systems directed towards applications in technical diagnostics. It is reasonable to use as much as possible of multiple knowledge sources, where the mathematical models are (or should be) of great importance. Direct utilisation of such models is difficult, because they point out the causal relationships only, and not the relations between diagnostic symptoms (effects) and the technical state of an object (cause). We identify essence of knowledge acquisition for machine diagnostics and monitoring. We discuss briefly several methods and techniques of knowledge representation. Then we focus on a methodology of an acquisition of inverse models from examples obtained as results of simulations done with known, existing models. Another possibility is to acquire knowledge from experts. Finally, we give a piece of information on the tools made in our Department.*

1. INTRODUCTION

Most of all, expert systems make use of rules. To be effective, expert systems for technical diagnostics must contain a large amount of domain expertise organised appropriately. The designing of large knowledge-bases containing rules is a hard work. The direct acquisition of rules from experts is extremely difficult. Special forms of logs (fault trees, test trees, diagnostic graphs, decision tables), made by experts may support the process of knowledge acquisition. The knowledge engineer (or special program) processes such logs (records) to obtain an appropriate set of rules. For a large set of independent rules we can lose track of them. It is very inconvenient to check their completeness and consistency.

2. ESSENCE OF KNOWLEDGE ACQUISITION FOR MACHINE DIAGNOSTICS AND MONITORING

The process discussed embodies acquisition of knowledge on (Moczulski [21]):

- the given object of monitoring and/or diagnostics and other objects of the same design (structure of the object, history of its exploitation, etc.);
- ways of observation of the object, its conditions of operation during measurement, measured quantities, measuring arrangements;
- diagnostic methods and techniques;
- diagnostic symptoms;
- technical states of the object investigated;
- diagnostic relations (between state features and features of some diagnostic signals previously chosen);
- methods, strategies and/or procedures that an expert applies when he/she wants to solve diagnostic and monitoring problems;
- techniques used to perform a corrective action (repairs or even overhaul), spare parts needed, etc.

It is possible to identify characteristic features of knowledge base of typical diagnostic expert system:

- many levels of representation of data and information that correspond to different degrees of detailed description of the structure of the monitored and/or diagnosed object;
- top-down approach;
- dynamically updated models (the way of proceeding applied in our method consists in updating the parameters of the model of diagnosed object).

Our goal depends on modelling knowledge and expertise of an expert (or a group of experts) that needed to solve problems from *some limited problem domain*. Participants of the process are in particular: an expert (or experts) and a *knowledge engineer*. Recently one believes that it is impossible (or even inexpedient) to eliminate knowledge engineer completely from the discussed process. A result of proceeding is knowledge base put down into a mass storage memory of the computer.

Knowledge acquisition invokes the following operations:

- **designation**: detailed identification of objects (abstract or concrete objects) consisting in finding of objects and giving them proper names;
- **description**: unequivocal attachment of corresponding meanings to these objects (context-sensitive; it is worth to stress that this task is very difficult);
- **organisation**: describes relations that occur between objects; it depends on identification of multiple relations of structural character (i.e. composite structures these objects belong to);

- **identification of relations**: which objects are in relation with others, revealing dependencies between groups of objects;
- **defining or discovering of constraints**: offers additional information on accessible values of features of the investigated objects as well as exceptions.

Traditionally (see Buchanan [3]) knowledge acquisition was understood as a process of *knowledge elicitation* from human expert and then *transferring* knowledge (into computer memory). Thus, *human expert* is asked how to proceed to solve some problem (even, which rules apply), and *knowledge engineer* tries to write these data in the form suitable for computer program. It is possible to apply such *passive approach* successfully to solve only few problems, where:

- all participants (expert, knowledge engineer) must have some knowledge on methods and ways of solving domain problems;
- moreover, all participants have to use terminology that other participant is able to understand.

We think that knowledge engineer should play an important, *active* part that depends on *modelling of expert's knowledge* (see Wielinga et al. [27]). Hence, knowledge engineer:

- *reveals* expert's knowledge;
- *interprets* it using some conceptual framework;
- *tries to formalise* knowledge to make it available in a computer program, e.g. aiding maintenance personnel in solving problems from technical diagnostics and monitoring.

This process is achieved regarding *limiting criteria* that were identified by the knowledge engineer. Having general methodology in mind, one can distinguish *phases*, *operations* and *methods*. It is very important that the *identification phase* and *modelling phase* appear alternatively. Thus, knowledge acquisition becomes *sequential-iterative process*.

3. MAIN PROBLEM

It seems, that complex phenomenological models of the investigated object (or an element of the object) are valuable sources of relationships between technical states and symptoms. By means of such models we are able to point out the influence of different parameters on the activity of the object and it is possible to predict the values V of selected symptoms resulting from the simulated technical state S of the object.

$$S \xrightarrow{v(S):} V$$

The basic question is: *how to invert the mapping $v(S)$ into a mapping $s(V)$* ?

$$V \xrightarrow{s(V):} S$$

This new mapping is of great importance to technical diagnostics and makes it possible to reason about the technical state S of a real object that causes the observed symptoms V. The mapping $s(V)$ forms directly the pattern for a rule in an expert system. The inversion of $v(S)$ into $s(V)$ is not specially difficult if it can be done in the analytic way or if both S and V are real vectors. Of course, the discussed models consist of sets of differential equations with parameters depending on features of the object and with an unknown analytic solution. They result not in a single real vector but in a complex function (e.g. of time) taking its values from the vector space. The values of the solution arise from numerical procedures. As an example we can take into account the equations of motion for the rotating elements of compressors or turbines.

4. SELECTED METHODS OF KNOWLEDGE REPRESENTATION

Two types of knowledge must be represented for the purpose of diagnostics and monitoring, namely *declarative knowledge* and *procedural knowledge*.

We need to represent procedural knowledge if we are going to identify concepts, names of objects and classes of objects, relations between concepts, between concepts and goals and within the group of goals.

Taking diagnostic and monitoring applications into account, it is necessary to represent *procedural knowledge* as well. This includes:

- Algorithms of diagnostic proceeding (also heuristic procedures), algorithms of verification of elements regarding design (e.g. verification of stress and strain in an element given) and exploitation (e.g. calculation of eigenfrequencies of a system compound with a shaft, bearings and foundation of a machine), etc. (see Moczulski [22]).
- Methods of diagnosing (of heuristic nature rather) that may be thought of as general guidelines concerning practical attempts of proceeding.
- Methods of diagnostic reasoning (commonly of heuristic nature as well).

In the following subparagraphs we will discuss briefly the more important methods and techniques of knowledge representation for diagnostic and monitoring needs.

4.1. Fault trees

The fault tree is a well known and widely used tool in the area of reliability analysis and troubleshooting [1], [17], [24]. If one element or unit of a machine does not operate correctly, it causes further abnormal actions of some other elements. Finally, failure propagation may result in a general failure of the machine. The simplest way to find out the source of failure is to trace back the cause-effect pathway, which may be effectively achieved by means of fault trees. To define a fault tree we should identify a list of the main elements and/or functional units (subsystems) constituting the machine. Failures and their conditions are interpreted as events. They may be represented in a graph model as nodes, which are connected by 'cause-effect' links. The links can be clustered by means of AND, OR, XOR gates. From the fault tree we may infer which primal events or combination of primal events cause the root event. To construct the fault tree we should have a thorough knowledge and experience about the design of the machine and its environment.

The fault tree can be extended to a weighted fault tree. The weightings are assigned to the nodes and/or links. They are understood as probabilities or conditional probabilities of occurrence and may be used for reliability calculations.

4.2. Test trees

Fault trees are tools for failure localisation. To locate a basic failure we ought to follow a given set of cause-effect links. Test trees result from fault trees, but they don't represent directly the causal links between the failures. Their nodes contain descriptions of procedures or tests that have to be made. They may be written as weighted graphs. Weights give the measures of cost or time necessary for the diagnose. The sequence of diagnostic or check tasks may be optimised. A test tree can be designed for a selected group of machines. Both kinds of trees don't include explicit information about the relations between the state of the object and the features of diagnostic signals.

4.3. Decision tables

A decision table is a very useful tool for the designing of reasoning systems for technical diagnostics. A basic worksheet for the decision table consists of rows representing conditions (questions, tests), rows representing actions (and/or conclusions), as well as rows representing exits. Each column on the right-hand side of the table contains the definition of a rule. Entries of conditions for these rules consist of expected answers to questions or test results. It has to be assumed [14], that rules are tested from left to right, until all the conditions are fulfilled. It means that not all rules will be tested. For a decision table to be complete each possible combination of entries for the given conditions has to be included once and only once. Missing rules, redundant rules and conflicting rules are not admitted.

4.4. Frames

Frames are an example of object-oriented programming, which makes it possible to generalise, classify and generate abstractions. Frames offer high computational efficiency. They are an interesting tool for the designing of interfaces with users and with external sources of data (e.g. measuring devices). A frame is a description of a real or an abstract object and can be used as a flexible representation of conditions and actions in decision tables. The notion of frames was introduced by [19] as well as [0], [13]. His basic goal was concerned with the designing of a data-base containing encyclopaedic knowledge, needed in common-sense reasoning.

A frame contains slots representing attributes of the object. Slots contain facets connected with values, default values and/or procedures (called demons) by which the values may be obtained. It is important to point out that each facet can contain values or demons. Such an inclusion of demons in frames joins procedural and declarative representations. Some systems distinguish between frames for classes and frames for individuals.

Frames may be arranged in hierarchical structures which make it possible to develop and process the idea about classes without being disturbed by details of any particular object. Such structures are expressed by links, between super-frames (parent frames) and sub-frames (derived frames).

Searching for a slot value for a frame is the basic task in frame systems. Special properties are assigned to the facets: *value*, *if_needed*, *if_added* and *if_removed*. The slot value is assigned to the facet *value*. When we are looking for the value of the slot, the content of the facet *value* is returned. If such a facet is missing, the facet *if_needed* points to the value or to a demon returning the value. Slots that are not present in a frame are inherited from superframes.

Inheritance is the most important feature of frames, which makes it possible to eliminate a redundancy of data and to handle exceptions. It can also be used to generate reasonable default data or assumptions in the case of incomplete information. Special facets, such as *if_added* and *if_needed* may be applied for forward and backward chaining, respectively. Simple hierarchy results in a tree structure of frames.

4.5. Knowledge bases

Most expert systems use a knowledge base. It may be assumed that a knowledge base is a collection of statements describing relationships between entities of the real world as well as abstract concepts. Such statements are variously called sentences, clauses, formulas and most often facts. A common way to represent statements is an *object-attribute-value* triple, e.g. used in the well-known pattern expert system MYCIN [26]. Attributes are general characteristics of properties possessed by objects. The value specifies the particular nature of a property in a given situation. Of course, the sets of such triples are flat (they contain no underlying structure) and the maintenance of great sets is extremely difficult. A dozen of different idea of structuring the knowledge base has been proposed and implemented (e.g. see [15]). Very important is the idea of frames.

4.6. Approximate rules

Diagnostic knowledge results from the experiences of experts. It is not given in a rigorous form and we have often to deal with rules that are true in most (but not in all) cases. It means that the statements and rules in such applications are often uncertain and/or imprecise. Approximate statements can be represented in many different ways, where certain rules and certain statements can be always taken into account as a special case of approximate one. Several ad hoc, empirical and theoretically based approaches to represent approximate statements and rules are known [12], [28], [26], [16].

The simplest approach is the direct application of probability theory and standard Bayesian model. A modification of the probability theory results in the *truth values* T(s) from the range [0,1] (or [-1,1]), assigned to each of the statements. They are interpreted as an extension of two logical values NO=0 and YES=1, onto the ordered set [0,1] of real numbers. Particular implementations differ mainly in the interpretation of the value T(s)=0, which can point statements that are false or which can point only the statements for which we haven't any source of information that they are true. The last case does not mean that there exist some reasons to interpret such statements as false.

4.7. Possibility and necessity

An interesting modification of reasoning patterns was obtained by means of modal logic. The notions of possibility and necessity form conceptual framework for the measures of possibility P(s) and necessity N(s), assigned to statements. Leaving out the rigorous explanation we can interpret the values of these measures as boundaries of a hypothetical range for the unknown truth value. By means of N(s) and P(s) we can distinguish the case of compensated premises pro and contra $N(s)=P(s)=0.5$ from the case with a lack of information $N(s)=0$ and $P(s)=1$.

4.8. Bilateral implication

There exist many examples of dependencies between the state of an object and features of diagnostic signals. They can be interpreted as the following ordered relations:

from the known state of the object *it follows* a special property of the signal

Of course we are not able to use such relation directly for the robust reasoning in the diagnostics because it is highly possible that the same special property of a signal follows from an another state of the object. Moreover we have often no reasons to assume that the discussed relation is a causal relation. To represent correctly all cross-dependencies we ought to write the relations in such a form that both *modus ponens* and *modus tollens* may be applied together. It can be done by means of bilateral implication [3], [2]. The bilateral implication is simply a pair of both underlying implications and may be interpreted as a set of interconnected rules:

from the known state of the object *it follows* a special property of the signal

from the known property of the signal *it follows* a special state of the object

The implication has to be symmetric for *modus ponens* and *tollens* because the result of reasoning ought to be independent from the particular forms of basic statements and questions send to the user.

4.9. Compound decision tables

Decision tables have been proposed as a special tool for writing rules and can be used directly to model knowledge bases for expert systems with forward chaining. After slight modifications of exits, i.e. assuming that each table ought to return a value (e.g. YES or NO) we can speak about such tables as about special cases of conditions. Return values have to be assigned to all columns on the right-hand side of the table. Moreover, a default value ought to be assigned to the whole table in the case of incomplete sets of rules. The table returning a value can be used as a condition in another table. It is very useful to write such a table as a special case of a frame. The entries of the table ought to be written as frames, too. Such modifications allow to simulate some kinds of backward chaining, and allow to write the rules dealing with the knowledge (about the diagnosed objects) and with the meta-knowledge (about the reasoning process) in a similar, uniform way.

The notion of necessity and possibility and the concept of bilateral implication can be used together to extend the idea of sharp decision tables [4]. To apply such an extension of decision tables it is necessary to change some properties of the sharp tables. We should assume, that:

- rules ought to be tested from left to right and all rules have to be tested;
- for a decision table a combination of entries for the given conditions can be repeated many times;
- missing rules, redundant rules and conflicting rules are allowed.

4.10. Diagnostic models

The detection of changes in the internal state of a machine by means of non-destructive tests is one of the main goals in technical diagnostics. Such a detection bases on the features of interaction between the machine and environment. The basic diagnostic model of a machine can be built using a set of features. The methods dealing with the diagnostics of machines are frequently strongly based on experience. Valuable relationships are established intuitively without knowledgeable reasoning or study and corresponding phenomenological models are used for explanation only. It is known, that a direct acquisition of rules from a human expert is difficult and connected with the possibility of erroneous constraints.

More convenient (and easier) for the designer of an expert system, as well as for the end-user, is the acquisition of rules not by direct specification but indirectly (in a hidden form) by means of examples. Examples may be written in a decision table in a similar way as rules. Such examples result from case studies or from simulations and point out the pairs connecting the evaluations of technical states with symptoms. Simulations are particularly interested in applications, where we try to define an appropriate set of rules for the object that exists as a single unit, and where we are not able to collect a representative set of examples from different installations.

4.11. Meta-Knowledge Representation

Knowledge on methods of diagnosing and methods of diagnostic reasoning (in particular) is somewhat kind of *meta-knowledge*. To represent such knowledge one might apply *decision tables*, *frames* and *context*.

Decision tables contain amount of meta-knowledge because they define order in which logical values of (compound) premises of rules are examined and determine the order of firing rules. The order determining examination of statements in the premise is very important, because it is often necessary to undertake some operations (the so-called *side effects*) while evaluating simple premises, that may influence the contents of the working memory, thus yielding other behaviour of the expert system acting upon this knowledge base.

Another possibility of representing meta-knowledge might be a *context* for rules, that is some form of defining a structure of the set of rules and may be applied in the case of simple relationships between objects in the problem domain (where individual rules may be ordered and attached to groups of rules). The context may be also determined by means of *auxiliary statements*. An indirect definition of the context in a set of rules might be an application of a suitable decision table.

5. SUGGESTED WAY TOWARDS SOLUTION OF INVERSE MAPPING PROBLEM (LEARNING BY EXAMPLES)

A frequently discussed way to inverse mapping *v(S)* is connected with the application of neural networks. Neural networks consist of linked processing units called nodes or neurones, where interconnections of nodes may be in general variable. The network propagates the input data through the layers of nodes to the output layer. Each node in the hidden or output layer transforms outputs from nodes in previous layer, to its output. The output depends on the weights assigned to the node and is calculated by means of a selected differentiable, monotone activation function - most often the sigmoid function. The operating characteristics of the net is mainly defined by the topology of the network, and does not depend significantly on the shape of activation function. Weights in all nodes are calculated in the training (supervised learning) phase.

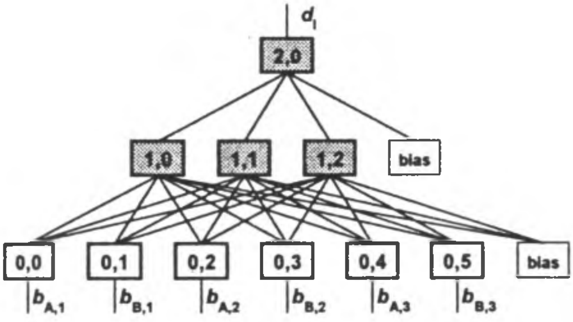

Fig. 1. Neural network (model *N*)

During training the known set of samples in the form of pairs consisting of values of all inputs and expected values of all outputs is given. The outputs are next calculated by the network (i.e. inputs are feed forward), using current values of weights, where initially the values of weights are set randomly. From the comparison of calculated and expected values an error results for each output. The global measure of errors (e.g. sum of their squares) should be minimised. Due to the shape of the activation function it is possible to calculate the degree to which each input to a given node contributes in its output error. Finally it is possible to propagate errors back to previous layers and using appropriate minimising strategy it is possible to update iteratively the weights in all nodes. The calculation of errors and updating of weights is repeated until the network reaches an expected level of quality. A neural network is a highly non-linear system and assuring the stability of such a system is difficult. To obtain the convergence and stability during training and to guarantee jumping over local extremes it is necessary to take into account some special heuristic algorithm for updating of weights (e.g. see [18] or [20]). As the result of training we obtain the set of weights.

We can generate many examples by means of the model *M* representing the mapping *v(S)* and train the network *N* on the examples. The network is taught by the computer program, that produces (with respect to the given model *M*) estimations of the symptoms *V* for randomly selected estimations of the technical state *S*. Estimations of *V* are sent to the input of the net *N* and estimations of the technical state *S* (or the classes of such estimations, only) are expected at the output of the net *N*. Such a way is effective for single real vectors *V*. It is rather difficult and not robust for functions (e.g. for frequency spectra), where the form of the function is most interesting and not the particular

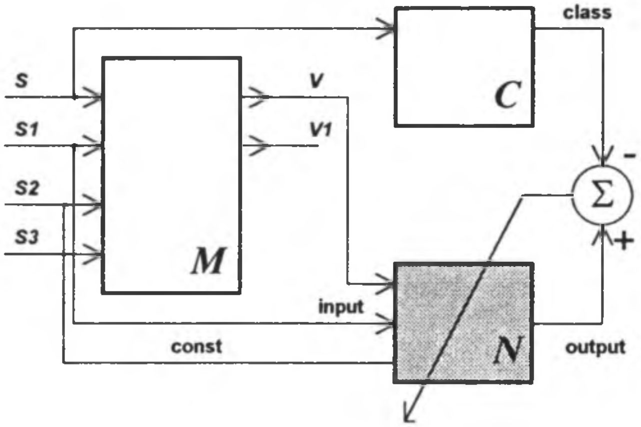

Fig. 2. Inverse model for a given model *M*, as a 'black box' *N* during training:
S - unknown parameters, which should be estimated by means of an inverted model,
S1 - known parameters, *S2* - parameters, that may be considered as constant,
S3 - parameters difficult to measure, that we have to leave out,
V - known parameters, *V1* - unimportant and unknown parameters.

maximum or minimum values. This approach seems to be attractive, since due to existing software the required knowledge of the theory of neural networks and training strategies is minimal.

The model *M* and the mapping *v(S)* allow to calculate exactly one *V* (e.g. the mean trajectory of the shaft in a slave bearing) for the given *S* (e.g. clearances, shape and thermal conditions of the bearing). From the shape of the inverse mapping *s(V)* it follows whether we are able to estimate the state *S* exactly for the given *V* or whether the results are multi-valued, not precise or even uncertain (e.g. we are not able to calculate the clearances from the shape of a mean trajectory of the shaft). For technical diagnostics it is not so important to estimate the absolute values of state parameters. It is enough when we can estimate the relative changes of the values (e.g. we find that the clearances increase) and/or we can point classes of the values only (e.g. small clearances).

It seems to be a paradox, but in a general case we improve the quality of results, leaving some data out and fuzzify the rest. To fuzzify the data we can define transformations for particular parameters converting their real values to a small number of linguistic values - like *small*, *large*. The similar treatment can be done not for single parameters but for the selected subset of parameters. In the space spanned on these parameters we select some regions called classes and define a family of such classes as a family of fuzzy sets, given by their membership functions. It is strongly recommended that the classes are defined directly and that we are able to interpret what it means that the parameters belong to a class. In an opposite case when the classes result from clustering of data such interpretation may be very difficult. The values of membership functions (without the direct knowledge of the parameters used when defining the values of the functions) can be used to recognise the state classes.

6. ACQUISITION OF DIAGNOSTIC KNOWLEDGE FROM EXPERTS ('LEARNING' SUPERVISED BY THE EXPERT)

In the following we assume that knowledge acquisition for the diagnostic and monitoring expert system will be aided by means of computers using a shell expert system.

To describe properties and peculiarities of objects we apply *statements*. It was assumed that the 'world' is restricted that results in a finite set of statements. *Rules of inferring* and *rules of action* are represented by means of *decision tables*. The decision tables (singular or a structure of tables, e.g. network) make it possible to represent some art of *meta-knowledge*. To facilitate development of databases and knowledge bases, several *dictionaries* are introduced, viz. (Cholewa [4]):

- dictionary of names of object classes
- dictionary of objects names
- dictionary of names of attributes
- dictionary of names of values of attributes
- dictionary of statements.

The discussed process should contain the following tasks:

- introductory discover of the need (where the resulting expert system will be applied, which diagnostic tasks should be solved using this expert system, etc.);
- identification of experts;
- drawing up of the description of the needs previously discovered;
- verification of needs and experts;
- elaboration and discussing of the dictionary of concepts (accompanied with synonyms and explanations of meanings);
- description of elementary tasks and sequences of these tasks;
- building model of expert's knowledge concerning domain of interest;
- verification of the model by expert or experts;
- elaboration of test cases by experts;
- testing, elaboration of reports.

Revealing of elementary concepts and tasks is strongly connected with the need to identify structure of knowledge in the field of application. Thus, it is possible to select adequate methods of knowledge representation.

To explain the idea of knowledge acquisition, let us consider the following examples:

A. **Mechanical structure of a particular machine that is an object of diagnostic proceeding.**

Knowledge acquisition may be directed by means of following assumptions:

- Such knowledge is of declarative nature rather and hence it is expedient to apply *frames* that may describe inheritance of properties and/or relations between the elements and/or parts.
- To identify structure of the machine, *top-down approach* is suitable with taking into account a recommended level of details. Progress of knowledge acquisition corresponds to substantial question *'Which elements and/or parts should be distinguished within the element given at the defined (or chosen) level of complexity'*.
- Each basic element has attributes - its inherent features as a geometric form and a system of dimensions (accompanied by declared or default tolerances). At the start of the process of knowledge acquisition it is expedient to input several elementary forms (so-called *primitives*, e.g. corresponding to elementary manufacturing operations) as well as rules used to define tolerances.

B. **Diagnostic and/or exploitation knowledge and skill.**

Apart from declarative knowledge, *procedural knowledge* is often necessary. At the beginning we need 'starting' knowledge concerning general methods of diagnostic proceeding. Knowledge may be introduced into a decision table. This table may then be decomposed into a network of decision tables revealing specific structure of knowledge. Given results of subsequent diagnostic observations (e.g. results of measurements of diagnostic signals) it is likely to generalise or identify diagnostic relations between technical (exploitation) state of the machine and values of symptoms of this state. These relations can also be introduced into decision tables as well.

7. TOOLS

Frames and decision tables may be realised by means of many tools, where object oriented programming is most important. Maintenance Aide Shell MAS [4] (elaborated at the Technical University in Gliwice) is an expert system shell (an extension of VV_SHELL [5]), that can handle frames and uncertain decision tables. It contains the production system, frame interpreter, frame editor, browser/debugger, reference database and an interface for extracting the diagnostically useful statements from the data in the database. The interface isolates the actual knowledge base of MAS from any particular machinery database and can easily be changed to process the data derived from virtually any condition monitoring system provided the database. The reference database contains the information required by the reasoning system and concerning the configuration of the machine train. System MAS contains the neural network generator and signal processing unit, that is capable to co-operate with an analogue-to-digital converter.

MAS runs on the IBM PC family of computers under MS Windows. The frame interpreter of MAS is a processing unit specially designated to handle different types of statement structures, represented by means of frames. It can also handle so called approximate statements and co-operates with other reasoning systems ([8], [11]). LISP-like frame description language allows to take into account different kinds of inheritance of frames. All the elements of frames in MAS are identified by their names. Names need not to be globally unique. We can use the same name for slots in different frames. Such an assumption results in a polymorphism - the names are shared and their meanings depend on the given context. The frame interpreter of MAS enables us to control the degree of encapsulation, achieved by means of demons (making no difference between data and description of data).

8. RECAPITULATION

In the paper some problems concerning organisation of knowledge acquisition process for diagnostic expert systems are dealt with. We decided to solve the problem using programming environment and hardware constituting Local Area Network that operates under control of Network Operating System *Novell™ NetWare 3.11* (see Cholewa and Moczulski [9]). In such an environment we run shell expert system *MAS* being an application of *MS Windows™*. The solution presented in the paper has the following properties:

Synergy: in the programming environment it is possible to run several applications simultaneously, which may aid knowledge acquisition process. Thus, various insights into the developed knowledge base are possible.

Common database: all applications use the same data base. Changes introduced into the data base by an application are immediately accessible to other applications.

Blackboard architecture: we want to apply blackboard model of co-operation between components of the expert system shell, which makes it possible to build multi-agent expert systems.

References

[1] BARTLETT F.C.: Remembering. The University Press, Cambridge 1932.

[2] BROWN J.S., BURTON R.R., BELL A.G.: A step toward creating a reactive learning environment. Int. J. Man-Mach. Stu. 7 (1975), p.675-696.

[3] BUCHANAN, B.G.: New Research on Expert Systems. [In:] HAYES, J.E., MICHIE, D., PAO Y-H (Eds.), Machine Intelligence 10 (1984) , 269-299. Chichester: Ellis Horwood.

[4] CHOLEWA W.: Reciprocal fuzzy implication. First Joint IFSA-EC and EURO-WG Workshop on Progress in Fuzzy Sets. Warszawa 1986, Abstracts p.19.

[5] CHOLEWA W.: Maintenance Aide Shell (in Polish). Internal Report RMT6076. Technical University, Gliwice 1992.

[6] CHOLEWA W.: Frames in diagnostic reasoning. Applied Mathematics and Computer Science, 1993, vol.3, No.3, p.595-612.

[7] CHOLEWA W.: Implementation of Simulation Procedures in Diagnostic Expert Systems. IMEKO XIII World Congress, Torino, Italy (to be published).

[8] CHOLEWA W. et al: VV_SHELL User's Guide and Reference Manual (in Polish). Technical University, Gliwice 1991.

[9] CHOLEWA W., CZOGAŁA E.: Management of statements in frame interpreter of CC_SHELL. BUSEFAL 49 (1991), p.40-49.

[10] CHOLEWA, W., MOCZULSKI, W.: Problems of Knowledge Representation and Acquisition for the Expert System Shell *MAS*. [In:] Practical Aspects of Artificial Intelligence II (1993). Institute of Computer Science, Polish Academy of Sciences. Augustów, Poland.

[11] CHOLEWA W., WHITE M.F.: Inverse modelling in rotordynamics for identification of unbalance distribution. Machine Vibration, vol.??. (accepted for publication).

[12] CZOGAŁA E., CHOLEWA W.: Uncertainty treatment in a fuzzy production system of CC_SHELL. BUSEFAL 48 (1991), p.124-131.

[13] DUBOIS D.,. PRADE H.: Possibility theory - An approach to computerized processing of uncertainty. Plenum Press, New York 1988.

[14] GOFFMAN E.: Frame Analysis. Harper & Row, New York 1974.

[15] HURLEY R.B.: Decision tables in software engineering. Van Nostrand Reinhold Company, New York 1983.

[16] JACKSON P.: Introduction to expert systems. Addison-Wesley, Workingham 1986.

[17] KRUSE R., MEYER K.D. : Statistics with vague data. Reidel Publ.Comp., Braunschweig 1987.

[18] LAPP S.A., POWERS G.J.: Computer-aided synthesis of fault-trees. IEEE Trans. on Reliability 26 (1987), pp. 2-12.

[19] MASTERS T.: Practical Neural Network Recipes in C++. Academic Press, San Diego 1993.

[20] MINSKY M.: A Framework for Representing Knowledge. [in:] Computers and Thought. [ed.:] WINSTON P.H.; McGraw-Hill, New York 1975, p.211-277.

[21] MOCZULSKI, W. (1992): Concept of a Condition Monitoring System. IMEKO TC10 Technical Conference 'Technical Diagnostics', Proc. 672-681, Dresden.

[22] MOCZULSKI, W. (1994): Representation of Features of Elements for Knowledge-Based Computer-Aided Design and Exploitation. [In:] CIM'94 International Conference on Computer Integrated Manufacturing, Zakopane, Poland.

[23] MOCZULSKI, W. (1994): Problems of Knowledge Acquisition for Diagnostic Expert Systems. IMEKO XIII World Congress, Torino, Italy (to be published).

[24] Neural Computing. User's Manual for Neural Works. Neural Ware Inc., Pittsburgh 1991.

[25] PAU L.F.: Failure Diagnosis and Performance Monitoring. Marcel Dekker, New York 1981. (1975 in France).

[26] SHORTLIFFE E.H.: Computer-based medical consultation MYCIN. Elsevier, New York 1976.

[27] WIELINGA, B.J., SCHREIBER, A.Th., BREUKER, J.A. (1992): KADS: A Modelling Approach to Knowledge Engineering. Knowledge Acquisition, 4, 5-53.

[28] ZADEH L.A.: The role of fuzzy logic in the management of uncertainty in expert systems. Elsevier Science Publishers (North-Holland) 1983.

Revised by: Janusz Dietrych