Computer Assisted Mechanics and Engineering Sciences, 9: 21–40, 2002.

Real-time diagnostic expert systems

Wojciech Cholewa
Faculty of Mechanical Engineering, Silesian University of Technology
ul. Konarskiego 18, 44-100 Gliwice, Poland

(Received October 8, 2001)

The aim of the paper is to point out the most important factors that should be taken into account during the designing of expert systems for technical diagnostics and advanced condition monitoring. The first such factor is the proper design of unified databases. It was assumed that discussed system consists of a network of coexisting and related nodes containing active statements looking for an equilibrium state. Such network represents a diagnostic model. Diagnostic models describe the relations between observed symptoms and their causes, i.e. the technical states of the object. Direct specification of such models is difficult due to the complex nature of state-symptom relations. An interesting idea is connected with example based inverse diagnostic models. Suggested solutions simplify the development and reduce maintenance costs for the whole system. A very important benefit for industrial application is the opportunity to arrange an incremental development of the final diagnostic expert system.

Keywords: expert systems, blackboard, reasoning strategy, inverse models

1. Introduction

The strategy for reliable plant operation can be defined as the condition monitoring of normal and faulty states of a technical processes on the basis of measured features of interaction between the process and environment, to aid or yield an automatic decision on the process performance. One of the advancing ways of state recognition is monitoring of the vibrations immanently generated by running machinery. Recent measuring techniques enable to observe the great amount of features of diagnostic signals. Successful vibration measurement and analysis require an intimate familiarity with types of measurement, transducer characteristics and application, as well as the capabilities and limitations of the diagnostic instrumentation.

Resulting features are often compared with warning, pre-alert and alert levels to detect whether observed quantities overcome pre-set limit values. Other, more sophisticated methods for machine monitoring and failure diagnostics are required to ascertain safe and economic operation of complex machinery as well as the quality control of products.

Among a great number of other, more sophisticated methods of failure detection and condition monitoring applied to technical diagnostics, the development of expert systems, which aims at emulating the way utilised by human experts for solving problems, plays a fundamental role. They can significantly improve the symptom interpretation and the system monitoring in case of malfunctions or algorithmically difficult to describe systems and processes.

Reasoning in expert systems in the area of technical diagnostics is concerned with incomplete, imprecise and even erroneous input data sets and domain knowledge. In such conditions it is recommended to use special kinds of data and the knowledge representation as well as appropriate problem-solving strategies. The blackboard model, introduced in speech recognition systems, seems to be the most suitable. Currently there have been a lot of papers dealing with blackboard systems and containing general suggestions as well as remarks on details. An interesting overview of the evolution of blackboard systems was presented in [2].

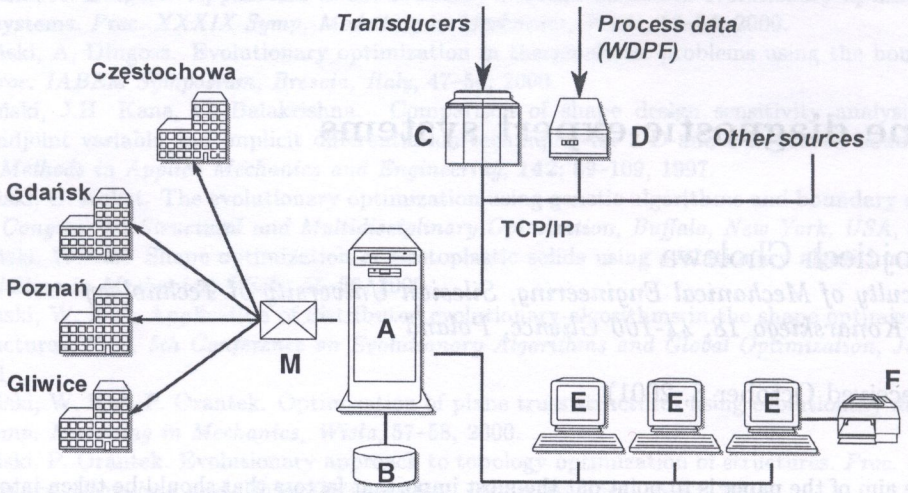

Fig. 1. Main components of the diagnostic system DT200-1

A few years ago the co-operating teams from Institute of Fluid-Flow Machinery Polish Academy of Sciences in Gdańsk, Silesian University of Technology in Gliwice, University of Technology in Częstochowa, University of Technology in Poznań, Institute of Energy in Warszawa as well as hardware firms ControlSoft and EnergoControl in Krakow started a project dealing with the development of monitoring and diagnostic system DT200-1 [11] (Fig. 1) for the 200 MW steam turbine in power plant Kozienice. The number of features registered by the system exceeds 6000. Currying out the project we have found that the designing of databases and real time reasoning engines for such systems is a very hard task. It was supposed that the system should contain an expert system, which should be able to work together with external simulation units and to utilise expert knowledge given in a form of rules. The project was carried out under one assumption that the best way is to apply the blackboard model with the knowledge sources using frames [3, 7]. It was observed that blackboard systems are used, among others, for interpretation problems (such as signal, speech and image understanding), which are characterised by huge search spaces. Large search spaces are typical in expert systems for medical diagnostics (great number of registered and distinguished diseases). Fortunately this is not true in expert systems for technical diagnostics, where the number of incorrect states, faults and damages is far lower.

This paper results from tests on a prototype dynamic expert system DT200-1 and presents an opinion that it is reasonable to introduce a new model for real-time expert systems in the area of technical diagnostics, which differs slightly from the classical blackboard systems and which can be used in all applications where the logical values of hypotheses are dynamic, i.e. varies with time. Blackboard systems deal with complex problems, and there are many features of the system, that should be optimally selected in order to guarantee successful solutions, where the effective control is a critical feature. The main feature of the proposed simplified blackboard is the application of *'active statements'*. The idea of an active statement results from the idea of statements (propositions) and from the idea of frames. Active statements form a net of coexisting and related nodes looking for an equilibrium state. Moreover, they can be modified directly by external tasks (e.g. data acquisition processes or interfaces to measuring units) as well as they can be read by external observers (e.g. users of the system or user interfaces).

A small (not too large) number of considered hypotheses is the main assumption of the presented system. That means that the reasoning may be analysed as a process carried out in a 'closed world', where all possible hypotheses are known and can be enumerated.

The second assumption is related to the issue of monotonicity of the system. Traditional deductive logic is always monotonic. For a diagnostic expert system there are no reasons to assume that the

reasoning will be monotonic, because due to our incomplete domain knowledge previous conclusions may be incorrect when new or changing evidence is obtained.

To follow the next assumption deals with temporariness of statements. All statements are marked with time-stamps and all reasoning steps are done for a given time (GMT). Moreover, all statements are recorded together with logs of the time-stamps and values of their changes (with their history). Such assumption is necessary for a real-time system, where it is impossible to carry out the reasoning process without any time delay between premises and conclusions.

To conclude, the last assumption deals with knowledge representation. Reasoning in the system is carried out with respect to the domain knowledge represented by rules written down in the form of inequalities of statements values [4]. The most attractive feature of the model results in the capability of implementing the whole reasoning engine in a standard database management system, which is used for data recording connected with activities of machinery under investigation. The suggested solution simplifies the development, reduces maintenance costs for the whole system, and allows incremental knowledge acquisition [17] as well as incremental development of an expert system to be performed.

2. STATEMENTS

Projects related to applications of expert systems which ought to contain a great amount of domain expertise that is not given in a rigorous form (e.g. technical diagnostics), result in conclusions that we have to deal with rules which are true in most, but not in all cases. It means that the statements and rules in such systems are often uncertain, with the different reasons of uncertainty such as randomness, lack of precision, approximate values, etc. Uncertain statements and rules can be represented in a lot of different ways, where certain rules and certain statements can always be taken into account as a special case of the uncertain ones.

To reason in an expert system some information about the real world is needed. Such information can be presented in the form of statements, called sentences and most often facts. Of course, from the direct use of the notion *fact* a lot of misinterpretations may result. Statements describing the real world express only the particular point of view or a belief that something has happened (will happen) or has been (will be) done. Statements depend on current (or default) contexts. Statements are applied to describe features of real and abstract objects. They are not independent, real existing facts.

It is important to make the following distinction:

a fact – something that exists in the real world, irrespective of our perception,

a statement – a record of our opinion about selected facts.

We can record our opinion about the given fact by means of different statements. The measure expressing our belief that the particular statement is true will be called *the value of the statement.* The value of the statement should be distinguished from *the content of the statement.*

2.1. Dynamic statements

As it was mentioned in the introduction all statements should be marked with time-stamps, which are necessary for a real-time system, to carry reasoning in a time-varying environment.

Statement x may be written down as the following tuple,

$$x = \langle o, a, v, w, t, b \rangle, \tag{1}$$

or

$$x = \langle o(x), a(x), v(x), w(x), t(x), b(x) \rangle\,, \tag{2}$$

where

'o, a, v' – content of the statement x, i.e. proposition (sentence) stating that the defined attribute a (e.g. mean oil temperature) of the given value v (e.g. 55°C) belongs to the designated object o (e.g. oil in the journal bearing #2),

t – time (moment or period),

b – value of the statement x, i.e. degree of belief that the proposition 'o, a, v' is true at the time t.

w – weight of the statement x, i.e. degree of its importance or significance.

The object o may be defined as an element belonging to the family of all subsets of the so-called *primary objects* (e.g. parts of machine elements or even fragments of these parts). Such assumption allows regarding interactions between elements of a machine of interest in a uniform way as a special kind of (abstract) objects.

2.2. Values of statements

Value of the statement x measures the acceptance of the appropriate proposition. The simplest approach is to consider the values $v(x)$ of statements x as logical values

$$v(x) \in \{ \text{YES, NO} \}. \tag{3}$$

Diagnostic knowledge is mostly acquired from experts [17]. We are often forced to apply rules that are regarded to be true in most cases (but not always). Hence, rules and statements implying conclusions may be uncertain and/or imprecise. Several empirically and theoretically based as well as ad hoc approaches representing and recording approximate statements are known (see e.g. [7, 12, 20]). The simplest approach is the direct application of the theory of probability and the standard Bayesian model. A modification of the theory of probability results in the *truth values* $T(x)$ from the range [0, 1], assigned to each statement x,

$$v(x) = T(x) \in [0, 1]. \tag{4}$$

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 $T(x) = 0$, which can point at statements that are false or only those statements for which we have not got any sources of information confirming that they are true indeed. The latter do not mean that any reasons to consider such statements as false exist.

An interesting extension of the truth values results from modal logic. The notions of possibility and necessity form a concept for the measures of *possibility* $P(x)$ and *necessity* $N(x)$ assigned to the statements x. Apart from a rigorous explanation, we may consider the values for these measures

$$[N(x), P(x)] \subset [0, 1]. \tag{5}$$

as boundaries of a hypothetical range of the unknown truth value

$$0 \leq N(x) \leq T(x) \leq P(x) \leq 1. \tag{6}$$

It is useful to assume that any statement x, such that $N(x) \neq 0$ and $P(x) \neq 1$, does not exist. From the assumption it follows that $N(x)$ and $P(x)$ may be substituted by a single value $NP(x)$,

$$NP(x) = (N(x) + P(x))/2, \tag{7}$$

and

$$\begin{aligned} N(x) &= \text{ if } (NP(x) \leq 0.5) \text{ then } 0 \text{ else } (2 * NP(x) - 1), \\ P(x) &= \text{ if } (NP(x) \leq 0.5) \text{ then } (2 * NP(x)) \text{ else } 1. \end{aligned} \tag{8}$$

2.3. A dictionary of statements

To simplify the reasoning process we ought to assume that objects, attributes and even values form elements of corresponding (finite) dictionaries

$$o \in DictionaryOfObjects \tag{9}$$

$$a \in DictionaryOfAttributesOfObjects \tag{10}$$

$$v \in DictionaryOfValuesOfAttributes \tag{11}$$

where *DictionaryOfAttributesOfObjects* and *DictionaryOfValuesOfAttributes* are defined for classes of objects and not for their instances. The assumption that reasoning engines of diagnostic expert systems run in a closed world, justify usefulness of the dictionary of statements

$$\langle o, a, v \rangle \in DictionaryOfStatements \tag{12}$$

Dictionaries (9)–(12) may be organised efficiently as tables of relational databases.

3. DATABASES

During last years it was a lot of suggestions on the designing of databases for purposes of technical diagnostics. Many vendors and users have implemented various relational databases and modelled machinery data in various ways. The main disadvantages of such databases are incompatible data formats, restricting the possibilities for exchange of data and common development of metadata.

3.1. Data warehouse

The terms *data warehouse* (and *data mart*) were introduced in the domain of information processing for business decision-making.

The data warehouse is an integrated store of information collected from a single or multiple sources, transformed into meaningful units. Information is stored with time and history information to allow for effective decision support. It forms the foundation for decision support and data analysis, because collected data is consolidated, consistent and subject oriented. Data warehouse provides multiple views of information to different users. Unification of design tools reduces the cost of building data warehouses, but the complexity of the data structure and complicated managing results in high requirements for data administrators.

The common feature of all decision-support database applications is that they do not change data. Database may be queried in a read-only manner. Moreover it is not necessary to run such applications in real-time. It means that the contents of the database may be updated periodically only, without the continuous connections with external sources. Due to the low update requirements and expected high performances of queries the discussed database should use many indexes, where it is clear that heavy indexing of production databases is impossible. Information may be organised in a very denormalised manner to satisfy common query requirements and improve query response time.

Over the past few years, a distinction has been articulated between data warehouses and data marts. A data mart is defined as a subset of the contents of a data warehouse stored separately in its own database. A data mart contains a detail and summary data focused on specific area. It is loaded down from data warehouse or taken directly from operational sources. The advantages of data marts are simpler implementations with lower costs, protection of valuable information stored in data warehouses, lower volumes of data and faster response times. The data mart projects are more manageable than complex enterprise data warehouse systems.

3.2. Integration of computer environment

The needs referring to integration of computer environment which use software aiding technical diagnostic are widely known and do not need justification. They can be considered from different points of view. Initially, the tasks dealing with assurance of data exchange by different modules of software cooperating in one operational memory have been solved. The suggested unification of multiaccess modules meeting requirements of COM (Component Object Model) resulted in high effectiveness of software. CORBA (Common Object Request Broker Architecture) specification is an extension of these requirements, which allows creation of systems with Remote Procedure Call (RPC). Application of proper libraries API (Application Programming Interface).is a must in using COM and CORB specifications. However, there are slight differences limiting code transfer between these environments.

One of effective ways of the above mentioned integration is application of unified databases, which enable simple data exchange between different programs. Nevertheless, it is not sufficient in all situations. It seems that the usage of XML language is a satisfactory supplementation of such bases and in chosen applications even a mean substituting the need of unification.

3.3. Multilayer software

The desire to unify ways of communication of end user with software and the search for optimal strategy of software development have caused the increase of importance of mulitlayer structures of software.

General model of software structure including a dialogue with user has been presented on Fig. 2, where data access layer, data processing layer, reasoning layer and result presentation layer have been distinguished. Traditional structure (Fig. 2a) can be characterised by the fact, that all layers appear in presentation layer software, designed for end user. In two-layer structure (Fig. 2b) access

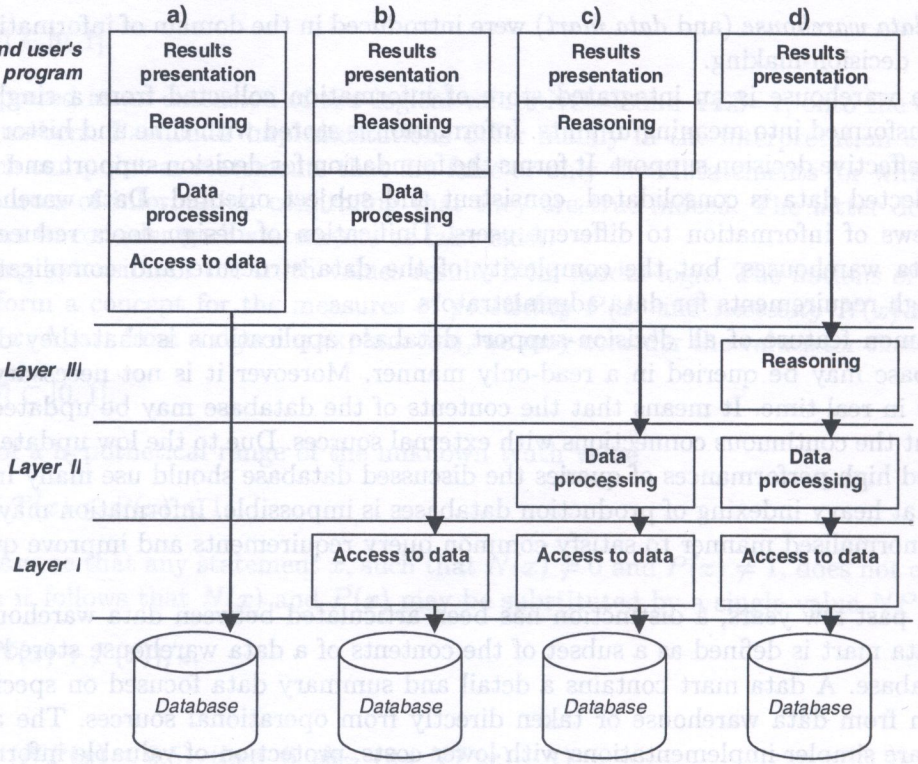

Fig. 2. Software structure model: a) "traditional", b) two-layer, c) three-layer, d) multilayer

data procedures (reading, recording, deleting, blocking, collision elimination etc.) are distinguished from utility program and constitute an external layer (mutual for many programs). In three- layer structure (Fig. 2c) data processing is distinguished as a successive external layer. Determination of further external layers depends on the application of the software. Such actions aim at possibly great simplification of presentation software. It can be assumed that Internet browser should be optimal presentation software appearing in multilayer structure (Fig. 2d). Such solution allows to use typical, reliable solutions concerning cooperation of different software modules in distributed environment (in successive layers).

3.4. An example: Mimosa

An interesting solution was presented as MIMOSA (Machinery Information Management Open Systems Alliance) [15] relational database schema by means of which developers of machinery information software can set their machinery database specification. The integration of many sources of machinery data was also a key ingredient to the expected long-term success of the MIMOSA conventions. The basic framework of storing site, site database, plant service segments, assets, model/part information, measurement locations, data measurement sources, transducers, ordered lists, and alarms are specified in the conceptual schema. In addition, the method of storing single-valued numeric data, Fast-Fourier Transform data, and time waveforms is also specified.

MIMOSA is the excellent solution for off line systems. Unfortunately the great number of related tables results in time consuming transactions carried out on the database and it is rather difficult to implement this idea to large, on line systems.

3.5. Databases in diagnostic system DT200-1

The idea of data warehouses and data marts was implemented in the DT200-1 system (Fig. 3), making use of Microsoft SQL Server. The system consists of two main databases DT4D111 and DT4D150. Information about the process (flows, temperatures, power, etc.) and features of diagnostic signals (vibrations, displacements etc.) is acquired by means of two services DT3D219 and DT3D220.

All databases are updated asynchronously. Long term database DT4D111 stores all historical data. It is periodically cleared. Short term database DT4D150 consists of the following parts (Fig. 3):

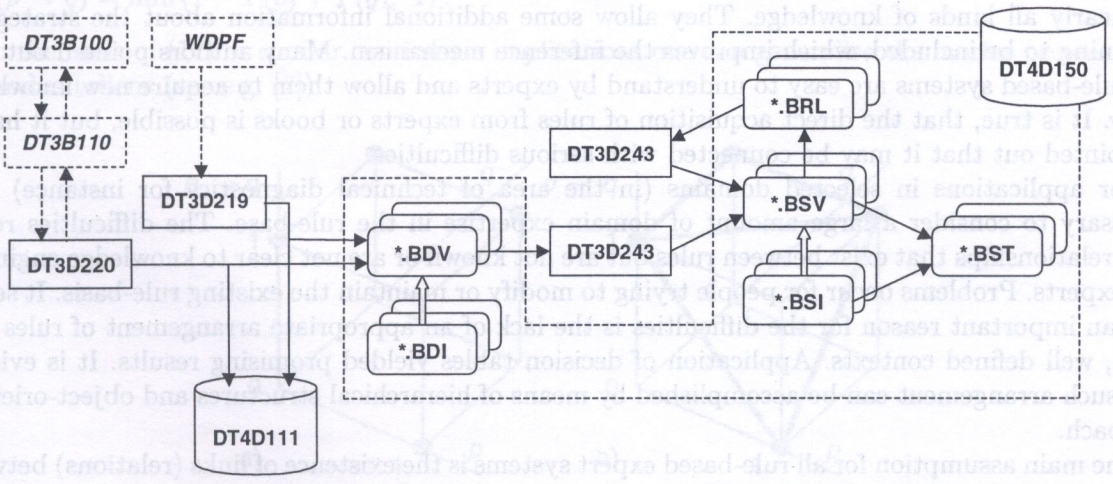

Fig. 3. Selected parts of databases in DT200-1 system

- the table BDV containing transformed (Fig. 4) values of selected features, where the definitions and descriptions of features are stored in the table BDI,
- the table BSV containing values of statements, where the definitions and descriptions of statements are stored in the table BSI.

The table BSV acts as the blackboard, which is updated by the service DT3D243 with the respect to rules stored in the table BRL. The blackboard is presented to the user by means of an internet explorer. Scripts stored in the table BST set the scope of presentation.

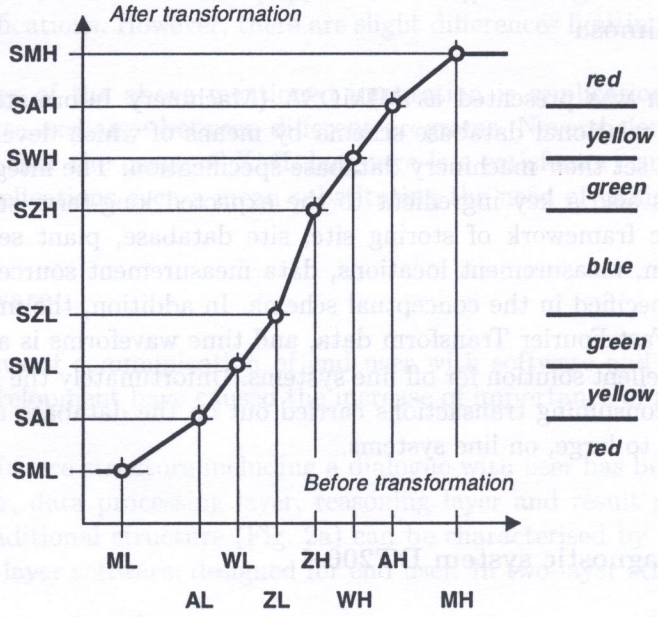

Fig. 4. Transformation of feature values

4. KNOWLEDGE REPRESENTATION

A critical step in the development of an expert system is the acquisition of the expert knowledge [17]. Most of all expert systems make use of rules (and structures basing on rules) capable of representing nearly all kinds of knowledge. They allow some additional information about the strategy of reasoning to be included, which improves the inference mechanism. Many authors pointed out that the rule-based systems are easy to understand by experts and allow them to acquire new knowledge easily. It is true, that the direct acquisition of rules from experts or books is possible, but it has to be pointed out that it may be connected with serious difficulties.

For applications in selected domains (in the area of technical diagnostics for instance) it is necessary to consider a large amount of domain expertise in the rule-base. The difficulties result from relationships that exist between rules but are not known or are not clear to knowledge engineers and experts. Problems occur for people trying to modify or maintain the existing rule-basis. It seems that an important reason for the difficulties is the lack of an appropriate arrangement of rules into small, well defined contexts. Application of decision tables yielded promising results. It is evident that such arrangement can be accomplished by means of hierarchical structures and object-oriented approach.

The main assumption for all rule-based expert systems is the existence of links (relations) between the statements. Links between the statements represent the domain knowledge. How to represent the links? By means of such links represented as rules conclusions about new values of statements can be

drawn, which result from the known values of the other statements. Many authors express an opinion that the rules in a rule-base represent causal links between the statements. This is often not true. It is clear that the statements should be connected by links representing their cross-dependencies, but such dependencies need not be causal. For example, the statements p and q in (13) as well as r and s in (14) form pairs of linked statements (but the dependencies are not causal):

$$\underline{\text{if}}\ x\ \underline{\text{then}}\ p; \qquad \underline{\text{if}}\ x\ \underline{\text{then}}\ q; \tag{13}$$

$$\underline{\text{if}}\ r\ \underline{\text{then}}\ y; \qquad \underline{\text{if}}\ s\ \underline{\text{then}}\ y; \tag{14}$$

4.1. Continuous implication

Mathematical logic introduces the notion of implication. If p and q are given statements, then the implication $p \rightarrow q$ is true when q is true or p is false. Classical logic uses two basic patterns for the deduction in propositional calculus, called *modus ponens* (15) and *modus tollens* (16):

if the rule (implication)	$p \rightarrow q$	is true	
and the premise	p	is true	(15)
then the conclusion	q	is true	

if the rule (implication)	$p \rightarrow q$	is true	
and the premise	$\neg q$	is true (i.e. q is false)	(16)
then the conclusion	$\neg p$	is true (i.e. p is false)	

It is important to point out, that:

- if one knows only that p is false (or respectively, that q is true) one is not able to arrange a reliable inference about the logical value of q (or respectively p),
- one is able to draw reasonable conclusions only when the implication $p \rightarrow q$ is true, because from the assumption that the implication is false it follows only that p has to be true and q has to be false.

For the purpose of reasoning by means of truth values one needs an extension of the concept of implication. A two-valued implication is shown in Fig. 5a. To extend the definition into a definition of continuous implications which are needed to deal with the truth values $T(s) \in [0,1]$ we have to define a surface (a function) spanned on the points shown in Fig. 5a. An interesting example of continuous implication is the Łukasiewicz's implication (Fig. 5b):

$$T(p \rightarrow q) = \min\left(1 - T(p) + T(q),\ 1\right). \tag{17}$$

On the basis of (17) or similar continuous implications one can generalise the modus ponens and the modus tollens (see e.g. [7]).

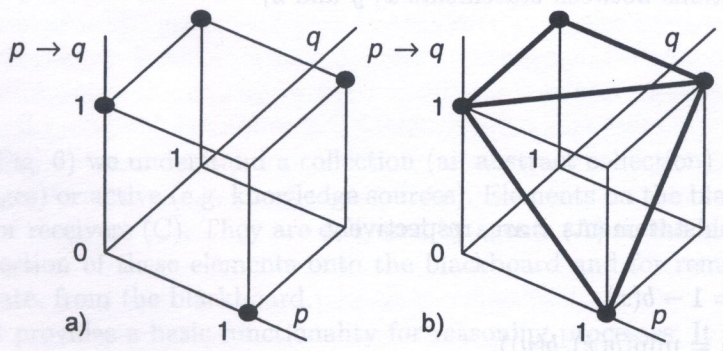

Fig. 5. Implication: a) two-valued, b) continuous (17)

4.2. Bilateral implication

It has been mentioned that we often have no reasons to assume that the relation represented by a rule in a rule-base is a causal relation. How to model the causality and how to model the lack of causality? One way is to write the relations down in such a form that both the modus ponens and the modus tollens may be applied together. This can be done by means of bilateral implication. The bilateral implication $p \leftrightarrow q$ is simply a pair of both underlying implications $p \rightarrow q$ and $q \rightarrow p$. The bilateral implication has to be symmetric for modus ponens and modus tollens to make the result of reasoning independent of the particular forms of statements (*object X has the property A* and *object X has the property* $\neg A$) and questions sent back to the user.

The notion of necessity and possibility and the concept of bilateral implication can be used together. They result in the following joint (generalised) modus ponens and modus tollens:

$$\begin{array}{lll} T(p \rightarrow q) & & \\ T(q \rightarrow p) & & \\ \hline N(p),\ P(p) & \text{such that} & N(p) \leq T(p) \leq P(p) \\ N(q),\ P(q) & \text{such that} & N(q) \leq T(q) \leq P(q) \end{array} \tag{18}$$

where for known values $T(p \rightarrow q)$, $T(q \rightarrow p)$, $N(p)$, $P(p)$ and for the continuous implication (17) we have

$$\begin{aligned} N(q) &= \max\left(0,\ N(p) + T(p \rightarrow q) - 1\right) \leq N(p), \\ P(q) &= \min\left(1,\ P(p) - T(q \rightarrow p) + 1\right) \geq P(p). \end{aligned} \tag{19}$$

The bilateral implication allows writing rules containing both the necessary and sufficient conditions, in a uniform way.

4.3. Uncertain statement calculus

Another form of the knowledge representation is concerned with uncertain statement calculus. One may consider the following statements x, y and z,

$$x = \langle o(x),\ a(x),\ v(x),\ w(x),\ t(x),\ b(x)\rangle\,, \tag{20}$$

$$y = \langle o(y),\ a(y),\ v(y),\ w(y),\ t(y),\ b(y)\rangle\,, \tag{21}$$

$$z = \langle o(z),\ a(z),\ v(z),\ w(z),\ t(z),\ b(z)\rangle\,, \tag{22}$$

and the following relations between statements x, y and z,

$$z = \text{NOT } x, \tag{23}$$

$$z = x \text{ AND } y, \tag{24}$$

$$z = x \text{ OR } y, \tag{25}$$

where the values of the statements z are, respectively,

$$b(z) = b(\text{NOT } x) = 1 - b(x), \tag{26}$$

$$b(z) = b(x \text{ AND } y) = \min(b(x), b(y)), \tag{27}$$

$$b(z) = b(x \text{ OR } y) = \max(b(x), b(y)). \tag{28}$$

4.4. Necessary and sufficient uncertain conditions

We know a lot of different ways to represent the uncertain domain knowledge (e.g. the uncertain implication). They have been presented in a good deal of papers, and they work with simple examples. Of course the acquisition of knowledge for such representations is not an easy task. More adequate (in the author's opinion) is a treatment related to two clear notions: the so-called *necessary condition* and *sufficient condition.*

It is interesting to see that given the values of two statements x and y with values

$$b(x) \in [0,1] \qquad \text{and} \qquad b(y) \in [0,1], \tag{29}$$

we can point out that x is a sufficient condition for y as follows,

$$b(y) \geq b(x) - \delta, \tag{30}$$

In a similar manner we can point out that y is necessary condition for x,

$$b(x) \leq b(y) + \delta, \tag{31}$$

where δ is a constant value,

$$\delta \in [0,1], \tag{32}$$

setting an uncertainty degree for the sufficient and necessary conditions, where for certain (i.e. not uncertain) conditions we have

$$\delta = 0. \tag{33}$$

By means of the necessary and sufficient conditions we can represent the conjunction (24) and (27) of statements (20) and (21) as a set of inequalities

$$b(z) \leq b(x) + \delta, \qquad b(z) \leq b(y) + \delta, \tag{34}$$

as well as alternative (25) and (28) of statements (20) and (21),

$$b(z) \geq b(x) - \delta, \qquad b(z) \geq b(y) - \delta. \tag{35}$$

5. Reasoning system

Main assumptions for the discussed system (a small number of considered hypotheses, non-monotonicity, temporariness of statements) were mentioned in the introduction. The system DT200-1 consists of the following main parts: a simplified blackboard, an interface to measuring units and a user interface.

5.1. Blackboard

By a blackboard (Fig. 6) we understand a collection (an abstract collection) of elements, which are passive (e.g. messages) or active (e.g. knowledge sources). Elements on the blackboard are accessible to their observers or receivers (C). They are delivered by agents (A) to the blackboard manager (B) responsible for insertion of these elements onto the blackboard and for removal of these elements which are out of date, from the blackboard.

The blackboard provides a basic functionality for reasoning processes. It forms a view attached to a database and acts as an intermediary between the database and the user or tasks participating in the reasoning process.

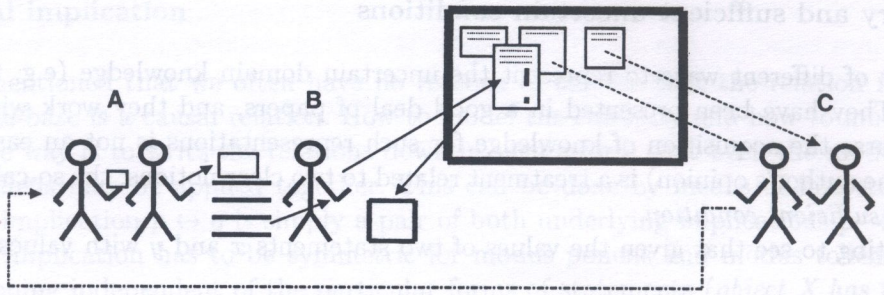

Fig. 6. A blackboard-like diagnostic system

5.2. Limited resources

A great number of papers and existing systems using the idea of blackboards do not take into account the needs connected with applications implemented for real-time tasks. Suggested algorithms assume that agents on the blackboard have sufficient knowledge, enough advance time and all the necessary resources to participate optimally in the reasoning. Such conditions do not represent real situations in technical diagnostics.

An interesting solution for the issue of time is the idea of an any-time algorithm, which can be interrupted at any time and which always returns a reasonable result. The quality of the result improves over time.

Looking for an optimal reasoning process one can assign a set of selection criteria as well as a set of heuristics to each agent and it is possible to identify an interesting search problem. Of course to solve such an optimisation task all agents have to lose the significant portion of their execution time. For real-time systems it is necessary to take into account, that computational resources used by the agent should be devoted exclusively to the analysis of the results of tests (measurements), without making any decision concerned with the order of performance of the task [1].

It seems, that the most promising generic way is to design two co-operating systems:

- a real-time system ('fast' system) running a quasi-fixed plan (program) of tests and/or tasks,
- a non time-critical system ('slow' system) producing optimal fixed plans (programs) of tests (tasks) for the first system.

Both systems run loops of actions lasting short periods (e.g. few seconds) for the first case and long periods (e.g. a few hours) for the second one.

5.3. Simplified blackboard

The main difference between the idea of a blackboard introduced in Hearsay [2] and the proposed simplified blackboard is the design and contents of the elements on the blackboard. To underline the differences the elements will be called *'active statements'* (instead of *'knowledge sources'*). The idea of the active statements results from the concept of statements (propositions) and from the concept of frames. Active statements (elements of the simplified blackboard) represent opinions about the object of interest. They form a net of coexisting and related nodes (Fig. 7), looking for an equilibrium state. Moreover, they can be modified directly by external tasks (e.g. data acquisition processes or interfaces to measuring agents) as well as external observers (e.g. users of the system or user interfaces) can read them.

An active statement contains demons, i.e. a special kind of procedures taking effect when one would like to look up or modify a value of the statement. One advantage of the demons is that they are acting automatically. Each demon is specific to one or more of the statement operations: if_needed, if_updated, if_inserted, ...

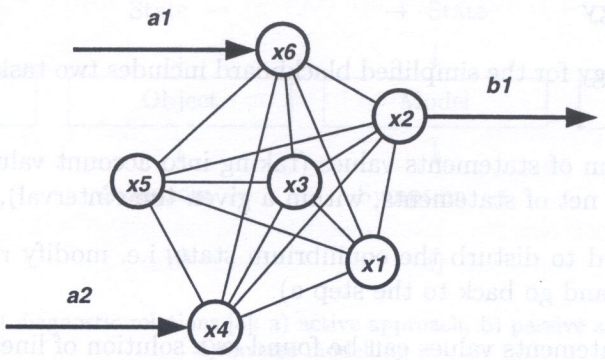

Fig. 7. A simplified blackboard; a – from external tasks, b – to external observers, x – active statements

Statements allocated on the simplified blackboard belong to two classes:

- direct statements, whose values do not depend on other statements and have to be set directly,
- indirect statements, whose values depend on values of other statements.

An indirect statement may be interpreted as a special case of a conclusion (the right-hand side part of a rule). The dependence on premises may be expressed as:

- conjunction (and connective) of necessary premises — e.g. (34),
- alternative (or connective) of sufficient premises — e.g. (35),
- aggregation (agg connective) of sufficient independent conditions [7].

Due to the previous assumptions all statements may change their values with time. Statements on the blackboard collect their history. An interesting question is how to take into account differences occurring in reasoning with indirect statements supported permanently by positive premises and with indirect statements supported only occasionally by positive premises. It can be done by means of accumulation (acc connective) of the values of the statements. Accumulation of the values may be interpreted as a special case of an abstract low pass filter for statement values (varying with time) where the time constant of the filter models the importance of particular premises.

5.4. Non-monotonic reasoning

Traditional deductive reasoning is always monotonic. Adding new premises never invalidates old conclusions. Of course continuous observation of an object may result in varying features of diagnostic signals, i.e. it may result in varying premises for diagnostic system. Such variations are intractable in deductive logic. The only way to deal with such premises is to run the reasoning engine repeatedly (every few seconds) from scratch and present conclusions on a plot. The proposed solution may be unstable, and may depend strongly on measurements noise and deviation.

An alternative technique is to use non-monotonic logic. The non-monotonic reasoning has been studied in the context of human common sense reasoning capable of dealing with incomplete information. It should be clear that previous conclusions may be incorrect when a new evidence is obtained. The influence of environment and the varying evidence on conclusions may be modelled as continuous or discrete process, where the results of reasoning are represented by continuous values assigned to uncertain conclusions.

5.5. Reasoning strategy

The general macro-strategy for the simplified blackboard includes two tasks, forming a continuously executed loop:

a) look for an equilibrium of statements values (taking into account values of nodes and relations between nodes in the net of statements, within a given time interval),

b) 'shake' the blackboard to disturb the equilibrium state, i.e. modify randomly and slightly the values of statements and go back to the step a).

The equilibrium of statements values can be found as a solution of linear programming problem. Well known procedures (e.g. [10, 14, 19]) enable to calculate unknown vector $\underline{x}$ (containing N elements) minimising the value z

$$z = \underline{c}^T \underline{x} \tag{36}$$

where $\underline{c}$ is a constant vector, and where it is assumed that for the solution $\underline{x}$

$$\underline{x} \geq \underline{xmin}, \qquad \underline{x} \leq \underline{xmax}, \tag{37}$$

$$\underline{Ax} \leq \underline{b}, \tag{38}$$

for

$$\underline{b} \geq \underline{0} \tag{39}$$

where $\underline{A}$, $\underline{b}$, $\underline{xmin}$, $\underline{xmax}$ are constant. The matrix $\underline{A}$ and the vector $\underline{b}$ represent the domain knowledge given as a set of necessary and sufficient conditions (34) and (35).

Such strategy (consisting of recurrent tasks) may be applied to objects with continuous (but slow) changes of their state over time. The strategy is appropriate for diagnostic systems (but it may be not appropriate for protecting systems).

6. Inverse diagnostic models

The main component of each expert system, and of each diagnostic expert system in particular, is a knowledge base consisting for example of rules indicating the potential sources for observed symptoms. The knowledge acquisition process may be an expert-opinion-based one [17]. The goal of such process is to collect either complete diagnostic rules or only diagnostic relations. However this sort of knowledge is not, and has newer been, precisely defined. It seems that more objective sets of diagnostic relations may be acquired not directly from experts but from experiments carried on engineering objects (Fig. 8a,b) or on exhaustively validated models of such objects. To a large extent the last one may be considered as a special kind of simulation diagnostics (Fig. 8c).

In an active approach to identification of state-symptom relations (Fig. 8a) defects are deliberately introduced into the investigated object, to reveal corresponding symptoms. The passive approach is based on the analysis of the observed object dynamic behaviour, to identify symptoms corresponding to the changing state of the object. The latter, however, requires a large number of objects to be observed and series of experiments planned over a long period of time, while the active experiments exclude any large-scale objects due to high potential risk.

Simulation approaches, in which models are used to map states into symptoms, have recently partially replaced the destructive experiments with high running costs. Their usefulness is obvious, although they require a thorough experience on the parameterisation of potential defects and correct interpretation of results obtained through the simulation.

It is possible to introduce an alternative approach to identification of symptom-state relations, distinguished by the use of inverse diagnostic models (Fig. 8d) instead of model based modules. It

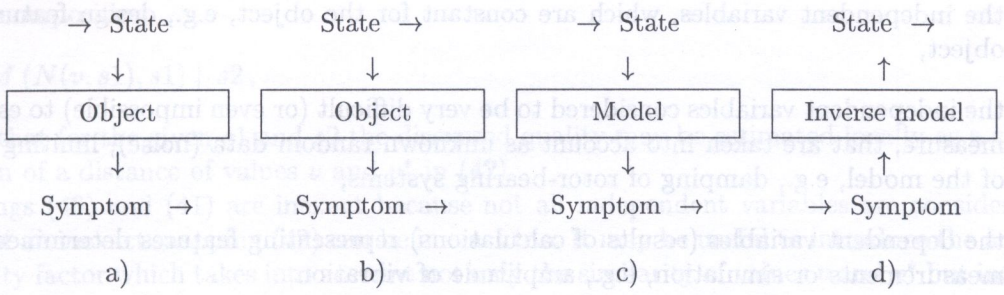

Fig. 8. Identification of diagnostic relations [8]: a) active approach, b) passive approach, c) simulations, d) inverse modelling

is assumed that a validated model of the considered object is given. The method is based on the idea of numerical inversion of dynamic models of engineering objects [8, 9]. Of course, it should be evident that in certain circumstances an inverse model could not be clearly defined. Correctly designed inverse diagnostic models may be included in the knowledge base of an expert system as special representation of diagnostic rules. It seems to be the promising solution for the time critical dynamic expert system.

6.1. Models *M* and *N*

Let M be an existing mathematical model of an object O as shown in Fig. 9. The model M maps the multidimensional space of input data values $\{s, s1, s2, s3\}$ expressing causes into space of values of the output data values $\{v, v1\}$ expressing effects. The model can be represented as a black box in the form of an algorithm or computer programme converting data $\{s, s1, s2, s3\}$ into data $\{v, v1\}$. In most cases it should be assumed, based on the current status of applied modelling methods and computing algorithms (e.g. FEM), that the analytical form of the model M is not available.

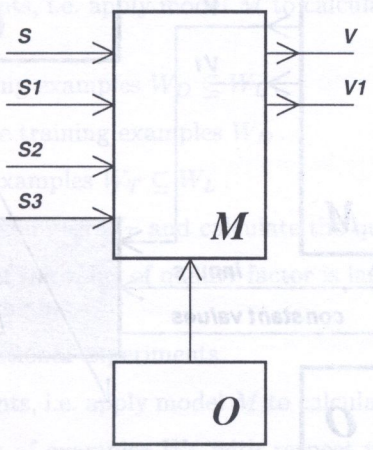

Fig. 9. Inputs and outputs of the model M of an object O

Inputs $\{s, s1, s2, s3\}$ and outputs $\{v, v1\}$ introduced in the model M can be interpreted as follows:

s – the independent variables representing the expected solution of the diagnostic process – the most common defect parameters, e.g., unbalance distribution of rotating elements,

$s1$ – the known independent variables determining working conditions of the object, e.g., rotational speed,

$s2$ – the independent variables, which are constant for the object, e.g., design features of the object,

$s3$ – the independent variables considered to be very difficult (or even impossible) to estimate or measure, that are taken into account as unknown random data (noise), limiting accuracy of the model, e.g., damping of rotor-bearing systems,

v – the dependent variables (results of calculations) representing features determined through measurements or simulation, e.g., amplitude of vibration,

$v1$ – the unknown or skipped dependent variables.

The model M describes case-effect relations between inputs and outputs. Parameters of the model result from identification, which can be carried out on real-life objects. The main question discussed in this chapter is how to apply the model M to obtain an inverse model N (Fig. 10) transforming the known observed output $\{v\}$ and known observed input $\{s1\}$ into unknown input $\{s\}$ with a constant input $\{s2\}$. One does not assume that the input $\{s2\}$ is known — one assumes that it is constant only.

There are reasons to suppose that an exact and unique inverse model does not exist as the considered inputs and outputs do not take into account all features. Moreover, from the above observation about the model M, it does not follow that an inverse model exists at all. What can be done in this situation? One possibility is to simplify the task. There is no other choice, but to accept that the considered relations are subject to inaccuracies and errors, as well as to abandon the requirement for very high precision of the calculated features. Such conscious step of cutting down the accuracy can be realised for example by the classifier C (Fig. 10) transforming exact expected valued of features $\{s\}$ into indexes $\{\text{class}(s)\}$ of their classes.

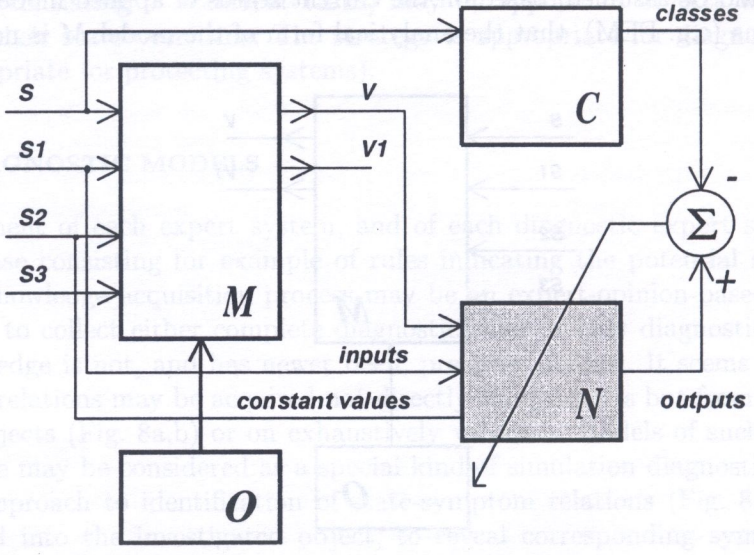

Fig. 10. Inverse model N for the given model M of the object O and classifier C

Let us assume that one has found an inverse model N for the given model M (it is explained e.g. in [5], how it can be done). An interesting question that should be considered now, is how to assess the quality of the inverse model N.

From the following pair of inexact mappings ,

$$v = M(s, s1) \mid s2, \tag{40}$$

$$s = N(v, s1) \mid s2, \tag{41}$$

and their composition

$$v^* = M\,(N(v, s1), s1) \mid s2\,, \tag{42}$$

it follows that for the given $s1$ and $s2$ the discussed quality may be estimated locally as a similarity or function of a distance of values v and v^* in (42).

Mappings (40) and (41) are inexact because not all independent variables are considered. The result v^* of an inexact mapping (42) can be inexact too. It may be useful to introduce the additional local quality factor which takes into account not only the similarity of values v and v^* but inaccuracy of v^* too.

6.2. Global inverse models

From the presented assumptions, it follows that the analytical inversion cannot be applied to this problem. Thus, numerical methods has to be used. Very important feature of the discussed models is their range of validity

- global models valid over the whole learning data set,
- local models valid in a cylindrical neighbourhood of a selected element $\underline{x}$ in the learning data set.

In particular cases the structure of the model N may be designed by experts, converting the problem into identification of parameters. Such situation is a special one and may be considered in a framework of a more general problem. The general method to look for the mapping N (see Fig. 10) is to consider the black box, which is trainable (Fig. 11).

a) Design random experiments.
b) Carry out the experiments, i.e. apply model M to calculate the initial set of learning examples W_L.
c) Select a subset of training examples $W_D \subseteq W_L$.
d) Identify model N on the training examples W_D.
e) Select a subset of test examples $W_T \subseteq W_L$.
f) Test model N on test examples W_T and calculate the quality factor.
g) Finish the calculations if the value of quality factor is large enough or the number of steps carried out is too high.
h) Design appropriate additional experiments.
i) Carry out the experiments, i.e. apply model M to calculate additional examples W_A.
j) Modify the learning set of examples W_L with respect to additional examples W_A, e.g., append W_A to W_L.
k) Go to the step c).

Fig. 11. Simplified basic algorithm for identification of the model N

An essential feature of the algorithm (Fig. 11) is the loop with the permanent modifications of the set of examples. It is important to stress that the above algorithm is not restricted to modelling of given data produced by the model M. Steps connected with the calculation of examples by means of the model M may be substituted by or mixed with experiments on real technical objects.

6.3. Local inverse models

Interpolation schemes model data locally at an arbitrary point $\underline{x}$ given a value of the function $\underline{y}(\underline{x})$ and a set W_P of pattern data consisting of known exemplary values of the function. A group of interpolation algorithms may be presented in the form of weighted averages of known values of the function [18], where the weights are defined as functions decreasing monotonically with the distance between $\underline{x}$ and pattern point $\underline{x}_i$. For weights it is possible to take into account the different accuracy from one point to another, where the more accurate point has greater influence on the model than the others.

6.4. Planning of simulation experiments

Basic kinds of models as well as calculation of the model parameters and testing of the models were discussed e.g. in [7] and [5]. Having based on the presented remarks it is possible to select and identify the particular model for the given set of examples. In order to carry the whole task the general method for selecting examples is still needed.

Easy to use, ad hoc method for designing of experiments consists in planning of experiments with regularly distributed examples in the space of values of inputs $\{s, s1, s2\}$ to the model M (Fig. 9). Of course, it should be clear that for all non-linear models such distribution does not guarantee a regular distribution of values of inputs $\{v, s1, s2\}$ to the inverse model N (Fig. 9). To correct the distribution of examples it is possible to identify such region in space of values of inputs and outputs where the density of the distribution is the smallest. In the middle of such region the next example should be generated. The presented method for identification of the best new example can be interpreted as a generalisation of a special search for the middle of the largest hole in a given piece of hard Swiss cheese.

An interesting alternative method for the designing of simulation experiments is the application of genetic algorithms [13, 16], belonging to the class of stochastic search methods operating on populations of solutions and not on single solutions only. This ability to operate directly on population of solutions is very useful for identification of sets of pattern data.

More details on types of considered models, appropriate global and local algorithms as well as genetic approaches to planning of simulation experiments may be found in [5] and [6].

7. Diagnostic system DT200-1

The DT200-1 system [11] has a hierarchical structure and consists of autonomic modules associated with each other. The signal acquisition subsystem (Fig. 1C), destined to preprocessing of signals measured by means of absolute and relative vibrations transducers, was separated from the main unit (Fig. 1A) of DT200-1 system. Asynchronous operation of this module does not allow direct prompting access to server resources and is a common reason for introducing a buffer for data store in this module. Signal analysis and data processing are realized e.g. according to the following engagement: frequencies range 1.5 Hz – 20 kHz, harmonic range – $1/3f_0$, $1/2f_0$, $2/3f_0$, $1f_0$, $\cdots$ $8f_0$, with phases measurement for 1, 2, 3, 4 harmonics, synchronic analysis resolution – 20 r.p.m., knowledge of rms value, peaks values, instantaneous value histograms and averaging all known estimates.

Co-operating modules (Fig. 1) can be run on several computers. Asynchronous transactions allow for co-operating programs to work under different operating systems. It enables rational usage of simulating software called remotely by DT200-1 system.

Diagnosers' terminals (Fig. 1E) were engaged to be equipped in computers with Microsoft Windows NT operating system. The multiple document interface (MDI) was used to prepare users interface, which allows opening several documents at the same time, making simultaneous presentation of current and historical data possible. The software enables moving data and pictures to other

Windows environment applications. The explanation and annotation texts are available to simplify access to system resources and to allow interpretation system functioning results. The DT200-1 system user can access an appropriate software module by the menu shown on the screen as task and resources tree.

The supervision modules enable controlling all simultaneously presented basic information (process parameters, vibration, trends, and stress) for the whole turbine. User interface enables several ways of presentation of measured signal values: pictograms, values diagram on the alarm/warning levels background, numerical values. The following convention of colors is used to mark values: blue – very good value, green – proper value, yellow – value exceeds warning level, red – value exceeds alarm level, white – measurement path damage.

Diagnostic expert subsystem DT3D100 is a dynamic system destined to non-monotonic reasoning under variable environment conditions. The system works in short time cycles. Structure of the DT3D100 databases, knowledge bases and inference engine was developed on the statement concept. The statement pronounces on observed facts or represents defined opinion. Original solution introduced in the DT3D100 system deals with using active statements representing an opinion about investigated object by means of agents taking place on an open blackboard. The active statements may form a net of co-operating units, which are modified constantly to move the whole network to equilibrium. Besides, each node of the network may be directly modified by external tasks (measurement events) and the nodes are available for external observers (users). All statements can change their values with time.

An original algorithm was used in the DT3D100 system, which enables accurate as well as uncertain reasoning. Results of reasoning are presented by means of particular interface, consisting of hierarchical presentation of temporal statement values depending on time. The values are shown on this view by colors symbolizing danger levels.

The DT3D100 expert system is a shell system with empty knowledge and databases. In order to use this system to aid monitoring of turbine technical state, its knowledge base must be filled with proper portion of knowledge enabling diagnostic reasoning at dynamically changing database contents representing results of measurements. This task belongs to knowledge gathering process. Experiments were undertaken to gather knowledge, among others by means of inverse diagnostic models based on collections of examples (results of measurements and simulation) describing dynamic features of the turbine observed thorough relative shaft vibrations as well as absolute vibrations of bearing suspensions.

8. Conclusions

The presented model of a dynamic diagnostic expert system seems to be an interesting modification of the well-known blackboard systems. It can be used in all applications where the number of hypotheses is not too large.

At present, a limited applicability of expert systems in monitoring tasks results from the enormous costs of acquisition and validation of knowledge bases. A special attention has been drawn to the application of known cause-effect dynamic models of investigated objects. Such models transform features of technical states into diagnostic symptoms. Their numerical inversion makes it possible to identify the global as well as local diagnostic models, with the latter ones being more appropriate. It was mentioned that the inverse diagnostic model forms a good background for the numerically based acquisition of diagnostic rules. In general, an inverse model (in the form of mapping of symptoms into states) can be developed on the basis of correctly validated theoretical model represented in a numerical form. The model should be considered for dynamic conditions. To obtain appropriate results the case-based distribution of examples should be carefully selected. Effective optimisation of sets of examples may be carried out as a genetic process.

The need to use suitable databases in monitoring and diagnostic systems of technical objects is commonly recognised by majority of users. Different forms of such databases have been discussed,

paying attention to the necessity of unification of structures and access methods. The need of application of numerous, diverse software, individually chosen for a user of a given asset as well as the lack of versatile experts on technical diagnostic justify purposefulness of works aiming at setting proper standards. The existence of such standards and their observation by the software authors should allow simultaneous application of program modules originating from different experts (competent in a given group of assets), thus, limiting purchase costs of software and facilitating their collection. Currently big projects are being carried, which aim at elaboration of such standards and MIMOSA can be an example of such a big, international project. It has been indicated that database unification can be insufficient for proper integration of environment where diagnostic reasoning is carried. For example a necessity may arise to introduce documents describing an object in XML language, using appropriate schemas. It will be especially needed while using multilayer software.

The most attractive feature of the discussed reasoning system is its capability of implementing the whole reasoning system in a standard database management system. It allows incremental development of the knowledge base, among others by means of inverse diagnostic models.

ACKNOWLEDGMENTS

The support from the State Committee of Scientific Research (KBN) in Warsaw is gratefully acknowledged.

REFERENCES

[1] D. Ash. *Diagnosis Using Action-Based Hierarchies for Optimal Real-Time Performance.* Ph.D. dissertation, Computer Science Department, Stanford University, 1994.

[2] N. Carver, V. Lesser. *The Evolution of Blackboard Control Architecture.* CMPSCI Technical Report 92-71 [available on the Internet], October 1992.

[3] W. Cholewa. Frames in diagnostic reasoning. *Journal of Applied Mathematics and Computer Sciences*, **3**(3): 595–612, 1993.

[4] W. Cholewa. Blackboards in diagnostic expert systems (in Polish). *Pomiary, Automatyka, Kontrola*, **4**, 123–128, 1998.

[5] W. Cholewa. Genetic approach to inverse diagnostic modelling. *Proceedings PPAM'99, Kazimierz Dolny*, 510–524, 1999.

[6] W. Cholewa. Example-based inverse diagnostic models. *Bulletin of the Polish Academy of Sciences*, **49**(2): 359–377, 2001.

[7] W. Cholewa, J. Kaźmierczak. *Data Processing and Reasoning in Technical Diagnostics.* WNT, Warszawa 1995.

[8] W. Cholewa, J. Kiciński, eds. *Inverse Diagnostic Models* (in Polish). The Silesian University of Technology, Gliwice, 1997.

[9] W. Cholewa, M.F. White. Inverse modelling in rotordynamics for identification of unbalance distribution. *Machine Vibration*, **2**: 157–167, 1993.

[10] G.B. Dantzig. *Linear Programming and Extensions.* Princeton University Press, Princeton, 1963.

[11] W. Cholewa, J. Kiciński, eds. *DT200-1* (in Polish). Wydawnictwo IMP PAN, Gdańsk, 1998.

[12] D. Dubois, H. Prade. *Possibility Theory. An Approach to Computerised Processing of Uncertainty.* Plenum Press, New York, 1988.

[13] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison Wesley, Reading, 1989.

[14] A. Grace. *Optimization Toolbox for Use with Matlab. User's Guide.* The MathWorks Inc., Natick, 1995.

[15] *Machinery Information Management Open Systems Alliance MIMOSA* – `http://www.mimosa.org`

[16] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, Berlin, 1996.

[17] W. Moczulski. Methods of knowledge acquisition for the needs of machinery diagnostics (in Polish). *Zeszyty Naukowe Politechniki Śląskiej*, No. 1382. Wydawnictwo Politechniki Śląskiej, Gliwice, 1997.

[18] M.J.D. Powell. A review of methods for multivariable interpolation at scattered data points. In: I.S. Duff, G.A. Watson, eds., *The State of the Art in Numerical Analysis*, 283–309. Clarendon Press, Oxford, 1997.

[19] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery. *Numerical Recipes in C.* Cambridge University Press, 1988.

[20] L.A. Zadeh. *The Role of Fuzzy Logic in the Management of Uncertainty in Expert Systems.* Elsevier Science Publishers, North-Holland 1983.