8. Soft Computing Models for Fault Diagnosis of Conductive Flow Systems

Viorel Ariton

This chapter focuses on the fault diagnosis of artefacts often met in industry, but not only, that execute various functions involving conductive flows of matter and energy, i.e., multifunctional conductive flow systems (MCFSs). The proposed MCFS abstraction is close to the human diagnostician way of conceiving entities and relations on physical, functional and behavioural structures. Diagnosis reasoning, performed by human diagnosticians, is intrinsically abductive reasoning. This chapter presents the abduction by plausibility and relevance in a connectionist approach. The case study on a hydraulic installation of a rolling mill plant gives examples on the knowledge elicitation process and on the diagnostic expert system building and running.

8.1. Introduction

Fault diagnosis of complex systems is often a difficult task, due to the incomplete, imprecise and uncertain knowledge on behaviours and interactions encountered in the real-life context. Diagnostic reasoning is abductive reasoning, thus it is different from the common (deductive) reasoning. The latter starts from causes and leads to effects, hence the "explanation" is based on a definite space of causes to a definite set of effects, while the first starts from effects to reveal causes. Hence, the "explanation" is based on a presumed space of causes with many-to-many links to a (reduced) space of effects. In real life, the diagnosis itself proceeds differently for similar target systems running in different contexts. On top of those difficulties, one may notice that computer applications for fault diagnosis face the modelling and the parameter identification burdens, both after a challenging knowledge elicitation effort on the target area.

Consequently, fault diagnosis of complex systems often relies on human diagnosticians, who usually perform knowledge acquisition on faulty behaviours, later used to "recognize" faults from (some) instance effects. In a simple view, they use a mapping of faults to effects, for searching causes possibly linked to the instance effects, and sequentially refining the diagnostic based on knowledge in the area and from practice.

The artificial intelligence community concerned with diagnosis obtains the mapping either by methodical experiments – exhausting the faults' space and collecting the effects – or by means of some knowledge of human experts from practice. However, the computational models for fault diagnosis also require methods to reduce the many-to-many relations of the reverse mapping from effects to faults, which commonly are known as human diagnostician's deep knowledge.

The diagnostic is then obtained: (1) using a matching procedure from actual effects to possible faults – as in the case-based diagnosis or in the neural/causal network-based diagnosis, (2) using a transformed effects space regarding the difference from the expected and the actual behaviour labelled with faults – as in the model-based diagnosis, or (3) using an "intelligent" look up procedure performed through a combined effects space, according to human diagnostician knowledge on phenomena specific to the target system in normal and faulty running – as in knowledge-based fault diagnosis.

Computational models of the above approaches have shortcomings at both phases above, most of them revealed when the target system runs in a real context:

a) For the faults-to-effects mapping phase: cases (1) and (2) above involve experiments which are barely possible for (all) faults, hence no complete mapping is possible, while in case (3), the mapping involves additional structures on causal relations between faults and effects, coming from some explanations of phenomena taking place.
b) For the diagnostic decision phase: in cases (1) and (2), the computational models are simpler but the diagnostic not entirely reliable, while in case (3) the backward chaining from effects to faults is applied in specific ways to the various running contexts.

Knowledge handled in cases (1) and (2) is often identified as “shallow knowledge,” while that in case (3) is considered “deep knowledge.” In usual cases, target systems involve flow conduction; hence the effects propagate throughout the (entire) system and thus make the diagnosis much more difficult. In that case, the combinatorial growth of the faults-effects mapping – cases (1) or (2), and because the deep knowledge refers to the model of the entire system – case (3). However, for systems in real life, neither the complex mapping nor the (many) complex models are possible, and that’s why the human diagnostician’s role is crucial. It is worth noting that running contexts of real systems are of greatest importance, while identical systems may behave differently – due to age, environment, maintenance.

The present chapter first states some considerations on the diagnosis as an abduction problem solving which exhibits an intrinsic connectionist nature: the many-to-many relations of the effects to causes may get forward (excitatory) links meant for activation of plausible causes, then relevant causes result from competition between the plausible ones. The artificial neural network (ANN) implementation of the connectionist model is enriched with specific architectural features (structures of neural sites) meant to solve all types of abduction problems met in the literature.

The nodes of the connectionist model are manifestations, symptoms and faults. Human diagnosticians handle such concepts in a discrete and qualitative way. In order to obtain a sound representation of the concepts and their qualitative relations, the chapter develops the analysis on modelling means that lead to discrete knowledge pieces and their relations, as human diagnosticians handle, regarding normal and faulty behaviour of a target system.

The chapter focuses on the class of conductive flow systems that perform more functions at a time; such systems are most encountered in technical and economical domains, and due to their multifunctional and flow conduction natures they are termed *multifunctional conductive flow systems* (MCFS). Sections 8.4 and

8.5 develop appropriate knowledge elicitation schemes, in a multi-modelling approach, to discriminate concepts and relations, as knowledge pieces involved in fault diagnosis.

All concepts and relations take part in appropriate Computational Intelligence models which combine human diagnosticians' deep and shallow knowledge on the target system behaviour, based on *fuzzy logic and possibilistic modelling* of incomplete and imprecise deep knowledge on manifestations, and based on *neural network* blocks for abductive problem solving of both fault diagnosis and next best test policy in refining the diagnostic.

The neural networks embed the shallow knowledge as data sets from practice and experiments for the plausibility links between faults and manifestations. Deep knowledge helps finding the relevant causes (from the plausible ones), and it is embedded in the neural sites of the specific abduction problems on manifestations and faults in the target system. Also, it is embedded in the links between faults and their specific symptoms corresponding to the four "orthogonal transport anomalies" (first introduced in (Ariton, 2003)). Additionally, the deep knowledge on the physical structure of the target system is embedded as the projection structure of neural blocks, each corresponding to a Bond Graph junction of the flow conduction system (Ariton, 2001).

Whilst deep and shallow knowledge are combined and embedded in the neural network, the training does not require exhaustive experiments on faults in the complex target system (which are barely possible in real life), and the diagnosis exploits the common view on the whole system as an interconnection of modules; to each module a neural network block is attached, thus easier to handle and train. The architectural features that embed the deep knowledge allow a better and comprehensive diagnostic, and also offer the opportunity to generate dedicated diagnosis applications for each concrete complex target system and its real-life running context. That opportunity is of most importance for the diagnosis task while two identical target systems may behave differently. While for the control task of a system it is natural to provide all homeostatic conditions to obtain the intended aim, the diagnosis task deals with the system as it is, in its real context and local conditions.

8.2. Diagnostic Problem Solving by Abduction

Abductive reasoning is a challenge for philosophy, science and practice. Abduction is sometimes creative while it puts effects before causes (Bylander *et al.*, 1991; Schurz, 2002). Computer applications require effective computational models, commonly focusing on the connectionist nature of the abduction problems (Peng and Reggia, 1990; Ayeb *et al.*, 1998).

8.2.1. Abduction Problems in Diagnosis

In the real world, fault diagnosis involves *open spaces* of manifestations and faults, while both are not completely known in real contexts. Unlike deductive reasoning ,

which focuses a definite aim and may consider the targeted part isolated from the whole, abductive reasoning (e.g., in diagnosis) may not ignore causes (and effects) without corrupting the result (i.e., the diagnostic). For example, if the excessive heat of the air around a hydraulic installation is neglected, one may assert that abnormal running is due to a faulty component – which may be false; a similar case arises when ignoring the quality of the mineral oil flow.

In fault diagnosis, the *cause* may represent one or more faults occurring at a moment, and the *effects* are subsequent deviations from the normal running that appear. A huge number of causes come from combinations of various faults and various external events, so the set of all possible causes is never taken into consideration (it is not realistic). On the other hand, some effects are observed and become *manifestations*, and some are not "visible" – due to the lack of information (e.g., no sensors).

Both aspects presented in the previous paragraph are facts of the intrinsic knowledge incompleteness of the diagnosis, actually of abductive reasoning in general. So, diagnosis always deals with open spaces of causes and effects; moreover, it deals with imprecise and uncertain knowledge of human experts on the real behaviour of the target system. However, for feasibility reasons, both the space of causes and the space of effects should be closed spaces. In this respect, special classes of causes and effects should be introduced – e.g., the "normal" situation or "unknown" causes.

Studies of Bylander *et al.* (1991) on abductive reasoning reveal four categories of abduction problems:

a) *independent* abduction problems – no interaction exists between causes;
b) *monotonic* abduction problems – an effect appears at cumulative causes;
c) *incompatibility* abduction problems – pair of causes are mutually exclusive;
d) *cancellation* abduction problems – pair of causes cancel some effects, otherwise explained separately by one of them.

Ayeb *et al.* (1998) have a sound approach in this respect. They introduce a fifth category:

e) *open* abduction problems – when observations consist of three sets: present, absent and unknown observations.

The discrimination of the abduction problem type is specific to the particular behaviour of the target system and it is a matter of deep knowledge of the human diagnostician on causes and effects in the local context. For each type of abduction problem, Section 8.2.4.2 presents a suitable architectural feature, which may enter the neural network implementation for the abductive problem solving.

8.2.2. Abductive Reasoning through Plausibility and Relevance

Direct causal links between effects and causes may represent *plausibility criteria* (Bylander *et al.*, 1991). From the set of all plausible causes, only a subset represent actual causes, usually obtained through a parsimonious principle. Konolige (1992)

considers the minimum cardinality as a *relevance criterion*, and applies it to the set of plausible faults to obtain the diagnostic subset.

In the presented approach the concept of relevance gets a specific representation, namely, it assumes some grouping of plausible causes – following specific points of view, then selecting the most relevant causes from a group – following competition or sorting/choosing procedures (Ariton and Ariton, 2000). In the connectionist implementations, plausibility links get direct representations as forward links between specific effects to specific causes As a concept, the "relevance" is not often discussed in the literature, so below a special attention is given to the subject.

A *relevance group* is a set of causes that are hardly likely to occur the same time – e.g., the set of faults for a particular component in the target system; in other words, a faulty component may exhibit only a small number of faults at a time (usually, only one). The point of view from which causes may enter a relevance group is the *relevance scope*, and it reflects the human diagnostician's deep knowledge on the faulty behaviour of the target system. The *relevance criterion* is the method used in selecting relevant cause(s). In order to perform selection, a quantitative quotient (e.g., "certainty" or "activation") is provided to rank causes. Following the relevance criterion (usually "minimum cardinality"), the selection of "most relevant" causes proceeds, e.g., by competition inside the group – for the connectionist implementation, or by choosing the cause with greatest activation. Other relevance criteria may state specific order of causes or specific quantitative relations between activations.

In the case of fault diagnosis, the minimum cardinality is usually applied as a relevance scope for the single fault diagnosis, disregarding it refers to a component or to the whole target system. However, the concept of relevance may be extended to the selected aspects met in real-life situations, i.e., to other "relevance features." For example, in conductive flow systems, a group of faults may indicate "leakage" symptom, so they all form a relevance group; if some of such faults in the group are plausible, the most relevant will be the one exhibiting the maximum relevance feature (in that case "leakage").

The abduction problem solving proceeds by applying plausibility and relevance criteria to the sets of all effects and causes, as further described; the input is the set of instance effects and the output is the set of plausible and relevant causes – which form the diagnostic. In Sections 8.2.4 and 8.2.5, the plausibility and the relevance get connectionist models adequate to computational implementation.

8.2.3. Connectionist Approach to Abduction

Many-to-many causal relations between faults and manifestations get reversed when reasoning by abduction. However, no inverse exists for the complex relations when real problems are under concern – e.g. fault diagnosis of a real complex installation. In such a case, one fault evokes many manifestations and the same manifestation is evoked by many faults. Moreover, manifestations may enter conjunction grouping to one fault, whereas disjunction grouping for others.

8.2.3.1. Qualitative Plausibility and Quantitative Relevance

It is worth noting two interesting characteristics of the above concepts: *plausibility is qualitative* and *relevance is quantitative*. So, in order to find:

- plausible causes, one should use some qualitative processing to select all causes complying with the observed current situation, e.g., asserting the faults related to the instance manifestations that appeared;
- relevant causes, one should use some quantitative processing to select only causes exhibiting a certain degree (e.g., greater than a given threshold value) from the set of plausible ones.

The practical conclusions on issuing a connectionist model for abductive reasoning by plausibility and relevance are:

- the activation mechanisms involved in plausibility criteria should allow a "logical overload" of numbers toward the qualitative processing on causes;
- the competition mechanisms for relevance criteria should assess (numerical) *degrees* which enter the quantitative processing on relevance of causes.

The logical overload is meant for affecting "quantities" (e.g., numbers) in order to become "qualities" (i.e., meanings) thus suited for plausibility criteria; the meaning is attached to each range of values, corresponding to the significance of that range taken from the deep knowledge of domain experts. The simplest logical overload attaches two complementary meanings for the two ranges of numerical values obtained after splitting the whole domain based on a border value (i.e., a threshold) with certain significance for the variable.

That simplest logical overload is actually used in the neural network implementation of the plausibility: if the link strength to a fault-neuron, coming from a manifestation-neuron, is greater than 0.5 (the doubt threshold), then the link is "important" and gets that meaning. Therefore, it has to pass the gates into the fault-neuron, i.e., enter the input function (the stimuli sum). Otherwise, it is "not important" and hence the gate to the fault-neuron is blocked, i.e., the input stimulus does not enter the input function (actually, the input value is set to 0). Practical examples on how to use the logical overload in specific abduction problems in neural network implementation are presented in the next subsections.

8.2.3.2. Parallel Plausibility and Sequential Relevance

Relations between causes and effects (in this direction) correspond to the deductive explanations and indicate which causes determine which effects. The many-to-many relations between effects and causes (in the reverse direction) show which effects may evoke which causes, but instance effects do no indicate instance causes (that really occurred), while no inverse of the direct relations exists. Therefore, in the general case, complex relations between effects and causes naturally lead to a *connectionist model* which, in an artificial neural network (ANN) implementation, will present excitatory links for the plausibility and competition links for relevance.

In a general approach, abduction problem solving proceeds by multiple applications of the following functions (Ariton and Ariton, 2000):

- *plausibility*(*P_CRITERIA, EFFECTS*) – which originates the plausibility of each element from the set of *CAUSES*, based on the set of instance *EFFECTS*, and according to plausibility criteria *P_CRITERIA*.
- *relevance*(*R_CRITERIA*, *CAUSES*) – which yields the subset of *CAUSES* selected from the set of plausible ones, observing *R_CRITERIA* specific to each relevance grouping resulted from the relevance scopes.

Note that entities in *CAUSES* and *EFFECTS* sets exhibit values in [0,1] interval. The above functions apply to each entire set of entities: first, all instance *EFFECTS* contribute to activation of plausible causes (so they attain nonzero values), then the entire set of *CAUSES* enters the relevance competition (repeatedly) while the less plausible causes already have near-zero values, thus eliminated. That assures a "classical" connectionist implementation in the ANN approach.

P_CRITERIA refer to deep knowledge of human experts (related to known causal relations between effects and causes) or they refer to shallow knowledge after the ANN train, following data collected from experiments on causes and effects. Plausibility may operate in parallel on EFFECTS to activate the related.

R_CRITERIA refer solely to deep knowledge of human experts on the various cases where causes show specific relations between them, specific links to running contexts or particular behaviours. Relevance processing is repeatedly (sequentially) applied, until a final definite set of causes (i.e., the diagnostic) achieve the highest stationary activation. In single fault diagnosis, the cardinality accepted for the diagnostic set is 1, in multiple fault diagnosis cardinality is greater than 1. How sequential diagnosis proceeds is presented in Section 8.7.3.

8.2.4. Neural Models of Plausibility for the Abduction Problems

In the neural network model, plausibility refers to forward (excitatory) links between effects and causes. A cause (e.g., fault) becomes the output neuron F_i and an effect (e.g., manifestation) becomes an input neuron M_j. The activation of a cause is the result of cumulative action effects associated to it, and it may be expressed by the well-known neural activation function applied to inputs M_j:

$$F_i = f\left(\sum_{j=1}^{|M|} w_{ij}\, M_j + \theta_i\right) \tag{1}$$

i.e., each manifestation from the set $\boldsymbol{M}$ (with $|\boldsymbol{M}|$ the cardinality) evokes, in a specific measure (i.e., weight) w_{ji}, the fault F_i, if the sum becomes greater than the threshold θ_i.

However, human diagnosticians often take into account a manifestation linked to a fault in a simple, "logical manner" (Ariton and Palade, 2004): manifestation M_j is "valid" (as a witness) for a fault F_i only if its activation is greater than a threshold, specific to the given manifestation-to-fault link. In the simplest way, if any two neurons M and F have activations in [0, 1] and the weight on their link is w, the maximum contribution of M to F is w (when M=1) and it is

still "valid" when M>0.5 (when M is above the doubt value) – i.e., its contribution is greater than w/2.

The logical overload consists in attaching certain linguistic attributes to the generic input I of a cause neuron, e.g., exceeding the doubt level, w/2:

if $I > w/2$ *then* I = "valid" *else* I = "not valid" (2)

This way, each link's strength is logically overloaded, and it makes possible the logical aggregation of effects to (evoked) causes, as required by each type of abduction problem.

8.2.4.1. Neural Sites and Specific Logical Aggregation

The ANN computational model of abduction for plausibility of the logical aggregation of input-effects to cause-neurons is performed by means of dedicated "neural sites," as specific architectural features that may embed deep knowledge in the connectionist model, beside the native shallow knowledge – which is embedded by training. The logical aggregations envisaged are (Ariton and Palade, 2004):

i) *disjunctive aggregation*, performed by the "disjunctive site" through the default cumulative processing (that is already the input function of the "classical" neuron), i.e., all m inputs cumulate their activation I_j:

$$O = \sum_{j=1}^{m} I_j \quad (3)$$

ii) *conjunctive aggregation*, performed by the "conjunction site," whose output O obeys the rule given by Eq. 4. After the logical overload, the inputs I_1, I_2 are aggregated according to the truth table from Figure 8.1f:

if $I_1 > w_1/2$ *AND* $I_2 > w_2/2$ *then* $O = I_1 + I_2$ *else* $O = 0$ (4)

iii) *negation*, performed by the "negation site". The output O is obtained from input I according to Eq. 5 and the truth table in Figure 8.1g:

$$O = w - I \quad (5)$$

Note that the logical aggregation upon links' strengths modifies only the input value of the cause-neuron; it does not affect the usual processing inside the neurons in the original neural network (i.e., input or activation neuron functions). So, the training and the recall procedures do not change (e.g. for perceptron or counterpropagation neural networks).

8.2.4.2. Structures of Sites and Neurons for Different Abduction Problems

Each type of abduction problem in Section 8.2.1is solved through a specific structure of neural sites, involving forward links from effects to causes as follows:

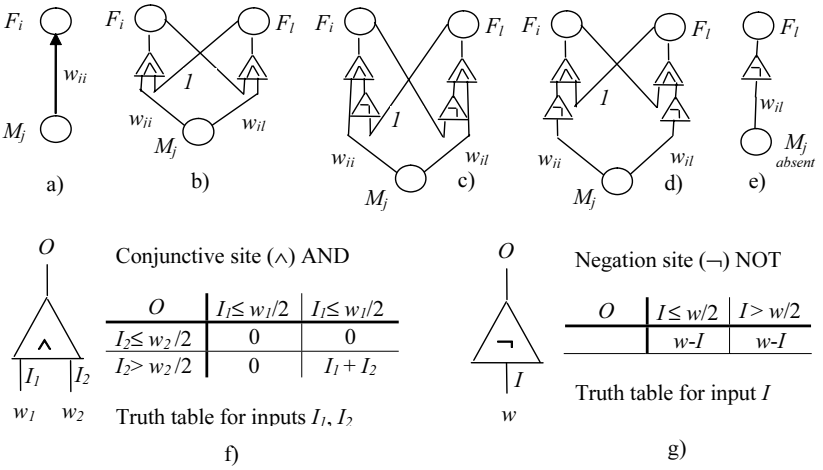

Figure 8.1. Various abduction problems solved by neural network features using logical overload of the links between neurons.

a) For *independent* abduction problems – excitatory links apply directly from the effect M_j to the corresponding cause F_i (see Figure 8.1a). If there also exists a conjunction grouping of the effects to the cause, conjunction site(s) are used at the input of the cause-neuron. Note that, by default, the neuron implements a disjunctive grouping of inputs through its input (sum) function (Eqs. 1 and 3).

b) For *monotonic* abduction problems – the causes F_i and F_l both evoke the same effect M_j, hence they suffer conjunction with one another and with the common effect through conjunction sites, as shown in Figure 8.1b, and expressed by the rule:

$$F_i \leftarrow F_l \text{ AND } M_j, F_l \leftarrow F \text{ AND } M_j \quad (6)$$

c) For *incompatibility* abduction problems – the pair F_i and F_l of causes are mutually exclusive (i.e., they are not both active at the same time), both evoking the same effect M_j. Each of them suffers conjunction with the negation of the other cause and with the common effect, as shown in Figure 8.1c, and expressed by the rule:

$$F_i \leftarrow NOT\ F_l \text{ AND } M_j, F_l \leftarrow NOT\ F_i \text{ AND } M_j \quad (7)$$

d) For *cancellation* abduction problems – the pair of causes F_i and F_l reduce the effect M_j when both occurred, although each of them evokes it separately. They suffer conjunctions as in Figure 8.1d, according to the following rule:

$$F_i \leftarrow F_l \text{ AND NOT } M_j, \quad F_l \leftarrow F_i \text{ AND NOT } M_j \quad (8)$$

e) For *open* abduction problems – the difficulty is dealing with absent effects, so the cause F_i is activated if no effect M_j exists (Figure 8.1e), according to the rule:

$$F_i \leftarrow NOT\ M_j \quad (9)$$

Links between cause-neurons in abduction problems of types b, c, d, have all weights between cause-neurons equal to 1 if they are symmetric (one to another), else they are set according to deep knowledge of the human expert.

Plausibility criteria are now embedded in:

- weights of the forward links between effects and causes – shallow knowledge;
- neural sites structures attached to cause-neurons (according to respective abduction problem) – deep knowledge;
- thresholds set for the site's inputs – deep knowledge.

The training procedure embeds the shallow knowledge by strengthening links between effects and causes as from the training patterns. At the recall phase, the sites trigger the inputs of the neurons just to obtain plausible causes; so, they only avoid activating less plausible causes, but do not modify the values of activations of the plausible ones – according to instance values of the (input) effects appearing. Even the structure of the neural network looks different, the original training procedure of the (two-layer) neural network does not change (no matter the type of the neural network used – e.g., perceptron, counterpropagation).

8.2.5. Neural Models of Relevance and Layered Modularization

The neural model of the relevance is competition. Relevance assumes a numerical value attached to causes, and the relevant cause(s) have the highest values that also exceed a given threshold. The cardinality of the relevant set of causes is 1 if "winner takes all" competition applies, or greater (if a relaxed competition applies). So, the relevant causes observe the minimum cardinality condition.

Relevance is a sequential processing: each relevance criterion is applied one after another in a given order, each criterion assuming the following steps:

i) Consider plausible causes in the current relevance group whose values exceed the given threshold.

ii) Start *competition* between causes inside the relevance group.

iii) *Select* relevant cause(s) observing the given cardinality (1 for single fault diagnosis).

Both pieces of information, the order of the relevance criteria applied and the causes belonging to each relevance group, are a matter of the human diagnostician's deep knowledge on refining the diagnostic. The numerical values involved in competition and the selection of causes come from the plausibility processing of causes based on instance effects.

Due to the fact that plausibility activates in various degrees the causes, competition always proceeds on the whole relevance group of cause (not only on the plausible ones); less plausible have lower (or zero) values and are easily eliminated, so the computational procedure is applied identically.

8.2.5.1. Relevance Scope

Any cause should enter a relevance group, i.e. no cause is relevant by itself while it is either already known or permanent. A relevance group usually consists of causes that share the same characteristics (Ariton and Ariton, 2000). For example, faults occurring at a given component form a relevance group, faults exhibiting "leakage" symptom at a given module form a relevance group, etc. Note that one cause (e.g. fault) may take part in more relevance groups, due to its properties.

The groups of causes are actually obtained by performing some modularisation on the entire set of causes observing relevance criteria that fall into one of the following categories:

- Scope on *physical structure* – concerning the physical units as locations for causes: all the faults at the module level form a relevance group, and all the faults at the component level form a relevance group;
- Scope on *functional structure* – concerning the specific running contexts (i.e., activities or process phases) in which causes are "visible": all the faults whose effects appear only when the piston of a hydraulic cylinder is moving form a relevance group;
- Scope on *generic effects* – usually concerning the same symptom: all faults evoking "leakage" symptom form a relevance group, while those evoking "clogged" symptom form another relevance group.

The relevance criterion is usually the minimum cardinality on plausible causes, meaning that causes are *unlikely to appear simultaneously*. It is applied at the various unit levels (physical or functional). Other relevance criteria are: *faults more likely* to occur (due to component's age or state – as from human diagnostician's experience), *faults requiring further observations* (by means of human operator tests), etc. In such cases, to each cause is attached a numerical value necessary in the processing presented above.

8.2.5.2. Layered Modularisation of Causes

A cause may enter various relevance groups of the same set of causes, in a *layered modularisation*. Each layer refers to a scope – regarding the modularisation of the set of causes, for each relevance scope obtaining two (or more) "relevance groups." For example, some layers refer to the physical structure: one layer contains groups of causes associated to modules and another one to components; other layers refer to generic symptoms associated to faults: those producing "leakage" and those producing "obstruction." For each layer a specific modularisation occurs, corresponding to the scope it represents.

Suppose that the layered modularisation of causes is performed according to n relevance scopes, so n-times partitioning of the same set of causes is obtained. Each layer L of relevance induces a specific modularisation of causes and has a specific weight W^L in the economy of the diagnosis. A layer (and its scope) may be more relevant than another, provided weights are normalized, i.e.:

$$\sum_{L=1}^{n} W^L = 1 \tag{10}$$

The relevance criteria, scopes and layers, groups and weights of layers all come from the deep knowledge of human diagnosticians, and they are indicated during knowledge elicitation time. The competition that takes place over causes in a relevance group, is independent of the forward plausibility processing in the neural network structure, no matter what ANN implementation is chosen. So, the relevance may be added without altering the original neural network functioning to an appropriate feedforward ANN architecture.

8.2.5.3. Relevance of the Faulty Situation Against the Normal Situation

A component is the final location in fault isolation, corresponding to the set of all faults as possible causes of some faulty behaviour of that component. However, the space of faults should be completed with the "normal" situation. The neural network output layer will contain F_0, F_1,…, F_{n-1} neurons indicating faults, and the F_n neuron indicating the normal situation.

The F_n cause (and neuron) is of capital importance, while the *NORMAL* situation enters the relevance competition along with the *FAULTY* situation. So, before fault isolation proceeds, the fault detection attests the *FAULTY* situation against the *NORMAL* one. The relevance group is the set of F_0, F_1,… F_n causes, and the relevance criterion (Eq. 11) asserts the *FAULTY* situation:

$$\textit{if } \exists Fi > 0.5 \ (\mathrm{i} = 0,1,\ldots,\mathrm{n}-1) \wedge \sum_{i=0}^{n-1} Fi > n \cdot Fn \ \textit{ then FAULTY} \qquad (11)$$

In other words, if any of the activated faults has a truth value greater than the "doubt value," and the relative level of the *NORMAL* situation is greater than all current (activated) faults, then the *FAULTY* situation is credited.

In conclusion, the connectionist model for abduction problem solving, using plausibility and relevance presented in this paper, is fully functional for all categories of abduction problems, as well as for disjunctive and conjunctive groupings of effects to a cause.

The proposed neural network model for abduction is a two-layer feed-forward neural structure, similar to perceptron or counterpropagation, that is completed with neural site structures for plausibility and relevance grouping / competition for relevance. The presented approach is more natural and simpler than the unified connectionist model for abduction presented by Ayeb *et al.* (1998). It also allows various "classic" ANN implementations, if appropriate feedforward and competition links are provided.

8.3. Aspects of Human Knowledge Usage in Fault Diagnosis

Fault diagnosis deals with concepts as fault, fault mode, manifestation, symptom or anomaly. The diagnostic problem solving is commonly conceived in two stages: Fault Detection, then Isolation of the actual faults (Palade *et al.*, 2002; Uppal *et al.*, 2002; Bocaniala *et al.*, 2004; 2005). The literature in the field defines the above concepts slightly different from one researcher to another, depending on the

approach or the actual implementation or method proposed. Diagnosis (DX) approaches deal with Artificial Intelligence (AI) and Cognitive Sciences concepts (Cordier *et al.*, 2000) and are closer to the human diagnostician way of acting.

In real life, fault diagnosis faces three types of inconveniences with respect to the faulty behaviour of a target complex system (Davis, 1993):

- *Incomplete knowledge* – the set of all (single or multiple) faults, effects and relations between them is not completely known. Diagnosis relies on a small set of causal relations (deductive) and empirical associations between faults and causes, and on a vague idea on how to proceed in FDI. Some manifestations are not known, while the human operator may supply information from test points, if required. When propagated effects exist, they increase the uncertainty on the faulty behaviour (e.g., in conductive flow systems (Ariton, 2001)).
- *Imprecise knowledge* – there is perpetually a drift in any measured value of a variable, the human expert having only a clue on abnormal ranges of values for each variable.
- *Uncertain knowledge* – when they have occurred, manifestations may not be entirely "abnormal"; that is, faults and manifestations occur "with some degree," they have truth values attached.

Aiming the computational modelling, the present approach is pragmatic: it considers definite meanings for the concepts above, allowing the representation of knowledge incompleteness, imprecision and uncertainty, assuming it comes from human diagnosticians' deep and shallow knowledge on faulty behaviour of a target real-world system.

8.3.1. Knowledge Pieces Involved in Diagnosis

Human diagnosticians' deep knowledge refers to the structure of the system under diagnosis and to the expected normal behaviour, while shallow knowledge refers to faulty behaviour at module and/or component levels. The structure of the target system consists of modules and components, as units conceived by designers, and accepted by diagnosticians to master the system's complexity. Modules and components are usually conceived as functional units. In the literature, the module is a structure of components, but the component does not have a clear meaning. It may suffer further decompositions (see Section 8.6.2.1), but nevertheless a component is conceived as the final location for faults or manifestations.

In the following definitions, we make use of the term *piece of knowledge*, stressing that the concept defined is obtained through an appropriate processing on the physical reality to extract (discrete) objects and logical meanings. A cognitive neutral numerical value X_k gets meanings (depending on the value range or particular situations) that are expressed by truth values $X_k \in [0,1]$, where $X_k = 1$ means that the concept is certain or complete. The concept may be a state (expressed by a noun) or a grade (expressed by an adjective or an adverb).

8.3.1.1. Component

A component is "a piece of equipment accepted by the human diagnostician as being sufficient for fault isolation" (Ariton and Ariton, 2000). Of course, it is a convention how much "detailed" a component is, while the human diagnostician decides what unit exhibits "pointed" causes for abnormal behaviours. After all, it is a matter of troubleshooting: deciding the location of the cause is the first step in removing the faulty unit (for further removing the disorder). How "small" (or how "low") the components are is a decision of the elicitation made upon the system under the diagnosis, when the fault isolation granularity is established.

8.3.1.2. Disorder

A disorder refers to nonconformities in the actual behaviour of the target system, against the expected one – which is designed and considered "normal." In order to obtain a feasible diagnosis system, the space of causes has to be a closed space, so it includes: disorders taking place at *components* (e.g., damages or ill tuning), *flow* (e.g., bad quality), *environment* (e.g., abnormal surrounding conditions) and *human operation* (technological discipline). Note that environment includes all neighbour systems: technical systems ambient atmosphere, etc., which may affect the target system's running.

8.3.1.3. Fault

A fault is a simple *piece* of knowledge regarding a physical nonconformity located at a component. Fault is a human concept with intrinsic discrete and logical natures: it has a name, usually expressed as a proposition about the disorder, and a degree of uncertainty – usually expressed in terms of a truth value $F_l \in [0,1]$. If $F_l \geq 0.5$, then it is above doubt that fault F_l occurred. From the human diagnostician point of view, the truth value is a *measure of plausibility* of a fault. The set $\boldsymbol{F}$ of all "known" faults should be decided at the elicitation time, each for a specific disorder or for a class of disorders, and reflecting the open space of effects induced by the incomplete knowledge. Open spaces should be closed by completing with generic "disorders" of the kind "not known" or "undecided," also with locations of the kind "out of target system limit." The *fault mode* refers to a specific disorder induced by a certain fault in a given process phase.

8.3.1.4. Manifestation

A manifestation is a simple piece of knowledge attesting to an abnormal value of an observed variable, during a certain running context of the target system. In the entire set $\boldsymbol{M}$ of manifestations, some may reach the diagnosis system by sensors (from continuous or binary variables), and others by human operator tests on observed variables in the process (from human senses – as adjectives, or from test points – as numbers). The manifestation's truth value $M_r \in [0,1]$ indicates how certain is the state or a grade exhibited, and it reflects our knowledge imprecision and uncertainty.

8.3.1.5. Symptom

A symptom is a complex piece of knowledge that refers to a certain behaviour coming from the deep knowledge on the target system and the domain. Symptoms

evoke classes of faults and induce some partition S on the entire set of faults $\boldsymbol{F}$. Some symptoms provoke disjunctive partitions (e.g., faults in the "leakage" class/symptom do not belong to the "clogged" class/symptom), others provoke non-disjunctive partitions. A fault that evokes more than one nondisjunctive symptom cumulates its plausibility (it is more relevant). The primary and secondary effects, witnessed in conduction flow systems, are symptoms: primary effect is the one located at the faulty component, secondary effect is the one located at the nonfaulty component due to propagated deviations of variables values (deviations from the expected "normal" values).

8.3.1.6. Process

Process phase is a complex piece of knowledge that refers to a certain state of the process, with certain duration in the functioning of the target system. From the human diagnostician point of view, a process phase characterizes the context in which the diagnosis takes place. While in the real system's running the process phase is "expected" to happen, its truth value P asserts the degree to which the context is really known, during the current slice of time in the process evolution. Process phases induce partitions on the set $\boldsymbol{M}$ of all manifestations and on the set $\boldsymbol{S}$ of all symptoms.

All the "evaluations" made by the (automated) diagnosis system to obtain truth values for manifestations, symptoms, process phases, and faults evoke some processing performed on observed variables' values (Calado *et. al.*, 2001). Note that the human diagnostician deals with "linguistic variables" when referring to manifestations and symptoms. By default, knowledge pieces are *discrete* and *qualitative* in nature, the latter reflecting knowledge imprecision or knowledge incompleteness regarding the human diagnostician view on the (faulty) behaviour of the target system. Therefore, any processing should comply with these aspects.

8.3.2. Observed Variables

Let us consider now a computerized diagnosis system that deals with manifestations and faults with graded values of truth as described above. If the observations made upon the target system's behaviour are linguistic or binary variables, they already have a "logical meaning" – present/absent. The observations made upon the target system come to the diagnosis system from the human operator (thus meaningful) or from sensors, as numerical values, thus cognitivly neutral. To obtain a common denominator, they should undergo some processing to become manifestations, so they undergo some "intelligent encoding" indicated by Cherkassky and Lari-Najafi (1992) as being crucial in diagnosis.

The preprocessing performed by the diagnosis system on the raw acquired values depends on the observed variable's type:

a) Binary variable from digital sensor – no processing required. By default, such a variable has two values, attached to a logical meaning (e.g., present/absent, open/shut). The manifestation results immediately, and $M_r \in \{0,1\}$.
b) Continuous variable from analogical sensor/device – processing required. To obtain some discrete piece of knowledge

(manifestation with some truth value $M_r \in [0,1]$) from primary data, the continuous signal supplied by the sensor is sampled and the series of values undergoes some processing according to the current process phase.

c) Discrete variable from human operator tests – no processing required. For example, the linguistic variable "noisy" is by default a logical variable with two values; thus manifestation results immediately: $M_r \in \{0,1\}$. Note that variables like "not hot," "hot," "very hot" should be reduced to more manifestations of the same type $M_r \in \{0,1\}$.

d) Continuous variable from human operator test performed in a test-point – processing required. The numerical pointwise value, entered by the human operator, should be evaluated if normal or not. Abnormal situation results as a (discrete) manifestation, according to the current process phase (e.g., fuzzification of point-wise numerical values, obtaining a fuzzy attribute with a graded value of truth $M_r \in [0,1]$).

So, intelligent encoding depends on the type of observed variables. The specific processing brings them to a uniform representation. Knowledge incompleteness, imprecision and uncertainty, specific to human diagnostician qualitative way of thinking, come from the abstractions made on the real continuous running of the target system (Mosterman and Biswas, 2002) and from the complexity of real phenomena. These aspects of human knowledge are melted into discrete and logical representations of manifestations, both useful in the neural network approach of the diagnosis, further presented in this chapter.

8.3.3. Semiqualitative Encoding of Manifestations

Fuzzy logic deals with associating logical meanings to numbers. It copes with the qualitative way of thinking of human experts, and quantities become sets, or intervals with imprecise edges, but specific meanings. In the present approach, a manifestation is a fuzzy attribute of an observed *continuous variable V* during the process phase *P*, i.e., it is a fuzzy subset over its universe of discourse $\Omega(V)$, as shown in Figure 8.2.

8.3.3.1. Prototype Manifestations

The attributes refer to the qualitative subdomains related to the abnormal values "too low" (*lo*) and "too high" (*hi*) in the current running context. Fuzzification is chosen as the "intelligent encoding" meant for manifestations. In Figure 8.2, the subdomains between *landmarks* $Lm(no)$ - $Lm(lo)$ and $Lm(no)$ - $Lm(hi)$, respectively, refer to the qualitative subdomains of Kuipers's approach (Kuipers, 1994) on quantifying values of a variable, in qualitative physics.

Pairwise neighbour subdomains form the fuzzy attributes "too low" and "too high" for the generic manifestations *lo* and *hi* corresponding to the given variable *V* and the given process phase *P*. Note that the fuzzy attribute "normal" (*no*) refers to the range of "expected values" for the observed variable, which indicates a normal behaviour; it is essential for obtaining a closed space of causes.

The overlapped intervals of the fuzzy attributes (see Figure 8.1) reflect the knowledge incompleteness and imprecision of the human diagnostician, which is linked to the specificity of the manifestation (Turksen, 1996).

The attributes *lo* or *hi* – as triangular membership functions in the semi-qualitative representation – are *prototype manifestation* set by the human diagnostician at knowledge elicitation on the system under the diagnosis.

The effective landmarks and the fuzzy subsets for generic manifestations *lo*, *no*, *hi* are provided at elicitation time. The knowledge engineer uses deep knowledge from the domain expert to assign qualitative landmarks for each observed variable from sensors. In this case, the CAKE (Computer Aided Knowledge Elicitation) tool is useful for the human diagnostician (see Section 8.6).

The triangular membership functions of the generic manifestations fit well to the *semiqualitative representation* usually encountered by human diagnosticians (Kruse *et al.*, 1994). Due to the linear and baricentric encoding, such representation offers some advantages for logical processing in a human-like way, also for fuzzy arithmetic with ranges when assessing propagated effects (Ariton, 2003). That simple semiqualitative representation best captures the human diagnostician's knowledge on manifestations of any kind, when the system is faulty.

8.3.3.2. Handling Uncertainty on Instance Manifestations

The manifestations linked to a continuous variable (type b or d from the above classification) actually refer to the pointwise value v that enters the diagnosis system during a process phase P. After fuzzification, each attribute *lo*, *no*, *hi* gets a truth value.

The *instance manifestations* obtained reflect the uncertainty of the situation occurring when for example both truth values $\mu_{hi}(v)>0$ and $\mu_{no}(v)>0$ appear (see Figure 8.2) – the last one reflecting the opinion on "normal" behaviour of the current situation. The preprocessing block of the diagnosis system should assert, for any variable instance, the appearing manifestations and their extent (the truth value).

8.3.3.3. Types of Manifestations

The set of all instance manifestations $\boldsymbol{M}^{\boldsymbol{P}}$ for a given process phase P comprises: the instance manifestations for all sensor-observed continuous variables $\boldsymbol{M}_C^{\boldsymbol{P}}$ (truth values in [0,1]), the instance manifestations for all sensor-observed binary variables $\boldsymbol{M}_B^{\boldsymbol{P}}$ (truth values in {0,1}) and the instance manifestations for all human operator-observed variables $\boldsymbol{M}_O^{\boldsymbol{P}}$ (truth values in {0,1}).

Taking into consideration all variables of any kind, and for all process phases, will lead to the set $\boldsymbol{M}$ of all manifestations as distinct knowledge pieces. It comprises the set $\boldsymbol{MM}$ of manifestations obtained by permanent measurements through sensors mounted in the process:

$$\boldsymbol{MM} = \{ \boldsymbol{M}_C^{\boldsymbol{P}} \cup \boldsymbol{M}_B^{\boldsymbol{P}} \mid \text{for all process phases } P \} \tag{12}$$

and the set $\boldsymbol{OM}$ of manifestations obtained by human operator observations:

$$\boldsymbol{OM} = \{ \boldsymbol{M}_O^{\boldsymbol{P}} \mid \text{for all process phases } P \} \tag{13}$$

Hence, the set $\boldsymbol{M}$ of all discrete manifestations entering the diagnosis system is:

$$\boldsymbol{M} = \boldsymbol{MM} \cup \boldsymbol{OM} \tag{14}$$

and comprises all pieces of knowledge of the kind *lo*, *no*, *hi* for manifestations at continuous variables, or *present* / *absent* for binary variables.

Overall, the cardinality of the set of all observed variables is lower than the cardinality of the set of manifestations ***M***, since the sensor-observed binary variables may have two "pieces of knowledge" (i.e., one manifestation of type "present" and, afterwards, one of type "absent"), and the human operator-observed variables may have three "pieces of knowledge" (i.e., two manifestations *lo*, *hi* and one of type *no* – as "absent" or "normal"). Some "absent" manifestations are quite important in diagnosis (see below), as they require a specific type of abduction problems to be solved.

Some continuous operator-observed variables may be "measurements on the fly," i.e., they are not permanently observed by sensors, but supplied occasionally by the human operator when required, following a best next test procedure (de Kleer and Kurien, 2003). In this case, the diagnosis system should perform the fuzzification or other processing, after the operator supplies the required value. This is a usual approach to finding logical meanings for manifestations (with truth values), and the obtained unified and discrete representation will be used in the connectionist implementation for diagnosis described in the next sections.

8.3.4. Intelligent Encoding of Instance Manifestations

Depending on the source of the observation, the obtained manifestation requires more complex or simpler processing, for example when observation comes from analogical sensors or from binary sensors, respectively. In the latter case, values as close/shut are already discrete and have a meaning – thus no processing required.

For an observed pointwise value v the truth value results from regular fuzzification (Kruse *et al.*, 1994) – e.g., in Figure 8.2 the instance manifestations *hi* and *no* get truth value $\mu_{hi}(v)$ and $\mu_{no}(v)$. The representation is semiqualitative while it exhibits qualitative attributes (i.e., *lo*, *no*, *hi*) and truth (numerical) values for each. However, human diagnosticians judge manifestations for the activity as a whole, hence the instance manifestation refers to the set of values (not the pointwise one) acquired during the current process phase *P*. Thus, straight fuzzification is not suited to encode manifestations (Dubois and Prade, 1998). An appropriate processing is further used.

8.3.4.1. Instance Domain for an Observed Variable

The sampling and the conversion of the V variable during τ^P time period of the P activity produce N^P binary numbers, further denominated *instance domain* (see the solid line in Figure 8.3a). A pointwise (quantified) value v_i appears η^P_i times in the instance domain. If divided by N^P, it becomes the frequency of v_i during τ^P, with a maximum η^P_m at value v_m: $\eta^P_m = \max_i \eta^P_i$ .The value v_m is a meaningful value but it does not evoke a manifestation, while it does not refer to the entire set of values, hence a special encoding scheme is needed, which is further presented.

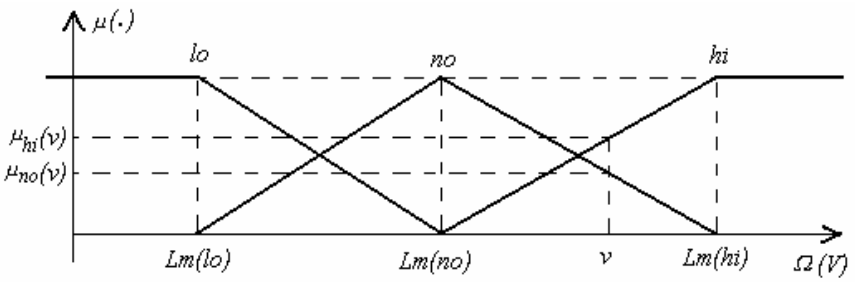

Figure 8.2. Semiqualitative representation of generic manifestations expected at a sensor-observed variable *V*.

The frequency distribution η^P_V for all values is the collection:

$$\eta^P_V = \{ \eta^P_i \mid i = 0 .. N^P \} \quad (15)$$

and the normalized frequency distribution (to the maximum η^P_m) – see Figure 8.3a – is:

$$\mu^P_V = \{ \eta^P_i / \eta^P_m \mid i = 0 .. N^P \} \quad (16)$$

8.3.4.2. Instance Membership Function for Series of Acquired Values

Instead of a pointwise value, the diagnosis system will use the normalized frequency distribution μ^P_V to assert manifestations for the variable *V* over the process phase *P*, as shown below. So, the instance domain (solid line in Figure 8.3a) may be seen as a fuzzy set in the statistical approach (as from (Kruse *et al.* , 1994)), and μ^P_V is the actual *instance membership function.*

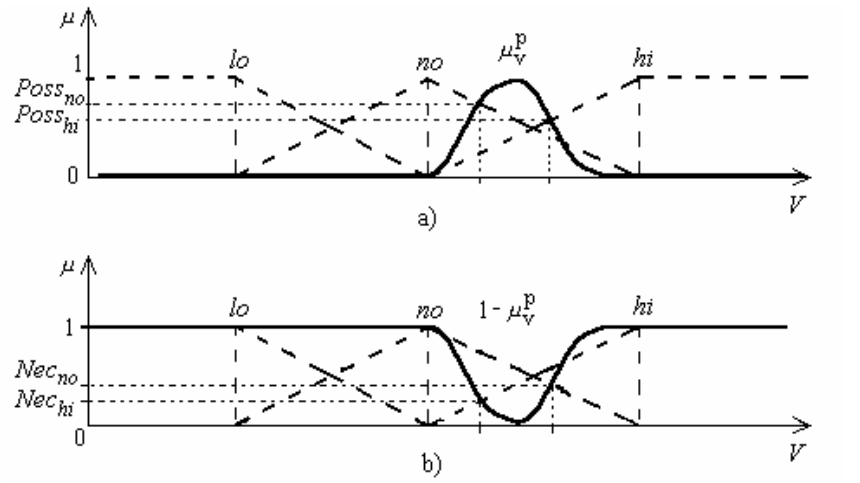

Figure 8.3. Possibility measure (a) and necessity measure (b) of the instance membership function upon the prototype manifestations for the continuous variable *V*, during the activity *P*

The instance membership function μ^P_V is not like the probability distribution p_V, while $p_V = \eta^P_i / \Sigma \eta^P_i$, thus it is obvious that $p_V \neq \eta^P_V$. On the other hand, sampling V during the period τ^P is not a random process, hence the approach is not stochastic. Frequencies do not change the proportions between the values after normalization, so that frequency distribution is scalable, but the probability distribution is not.

8.3.4.3. Instance Manifestation

The instance membership function of the observed variable V will reveal instance manifestations that appeared during the actual activity P. Manifestation is an attribute $a \in \{lo, no, hi\}$, which results from the possibility and the necessity measures (Ayeb *et al.*, 1998) of the instance membership function over the partition in Figure 8.3a:

$$\text{Poss}_V(a) = \sup_{v \in a} \mu_v^{\text{p}}, \ \text{Nec}_V(a) = 1 - \text{Poss}_V(a) = \inf_{v \notin a} (1 - \mu_v^{\text{p}}) \quad (17)$$

Inference of the instance manifestations proceeds as follows:

i) Calculate the membership function μ^P_V of the V variable's instance domain.

ii) Calculate the set $\boldsymbol{\Pi}^P_V$ of *possible manifestations*:

$$\boldsymbol{\Pi}^P_V = \{ a \mid a \in \{lo, no, hi\} \text{ and } \text{Poss}_a(\mu^P_V) > 0.5\} \quad (18)$$

iii) Calculate the set $\boldsymbol{\Gamma}^P_V$ of *necessary manifestations*:

$$\boldsymbol{\Gamma}^P_V = \{ a \mid a \in \{lo, no, hi\} \text{ and } \text{Nec}_a(\mu^P_V) > 0\} \quad (19)$$

iv) Assert which *instance manifestation* M^P_V actually occurred, applying:

$$M^P_V = \{ a \mid a \in \boldsymbol{\Pi}^P_V \cap \boldsymbol{\Gamma}^P_V \text{ and } \text{Nec}_a(\mu^P_V) \text{ is maximum from all in } \boldsymbol{\Gamma}^P_V\} \quad (20)$$

In the example from Figure 8.3, the possibility measures are: $\text{Poss}_{lo}(\mu^P_V) = 0$, $\text{Poss}_{no}(\mu^P_V) = 0.75$, $\text{Poss}_{hi}(\mu^P_V) = 0.55$ and the necessity measures are: $\text{Nec}_{lo}(\mu^P_V) = 0$, $\text{Nec}_{no}(\mu^P_V) = 0.45$, $\text{Nec}_{hi}(\mu^P_V) = 0.25$, hence the instance manifestation is *no* (see Figure 8.3b).

At elicitation time, the set of all instance manifestations $\boldsymbol{M}^P$, for a given activity P, comprises: instance manifestations for sensor-observed continuous variables $\boldsymbol{M}_C^P$ (truth values in [0,1]), binary variables $\boldsymbol{M}_B^P$ (truth values in {0,1}), and human operator-observed variables $\boldsymbol{M}_O^P$ (truth values in {0,1}).

8.4. Concepts and Structures on Normal Running

Deep and shallow knowledge, embedded in the connectionist model, comes from concepts that human diagnosticians deal with regarding the target system. However, diagnosis of real complex systems is a difficult task, while it involves a huge number of variables and events to handle, so computer-aided diagnosis is of great help.

The following section presents some principles on discriminating the concepts and their relations for the fault diagnosis following a human-like diagnosis, and using connectionist models for abduction. In that endeavor, means-end modelling approach seems best suited for the analysis and representation of

physical and functional structures. The approach makes use of bond graph models, adapted to cope with human-like qualitative view on the faulty behaviour of conductive flow systems, and also to the modular way of thinking when isolating faults.

8.4.1. Means-End Abstractions of Physical and Functional Structures

Real systems are multifunctional systems, while they perform many functions at the same time. Functions refer to tasks performed by modules and components toward specific utilities envisaged by the artefact. Each module performs a sequence of activities, and all modules perform activities in parallel – each module one activity at a time, during the given slice of time in the whole installation running. As a term, "multifunctional" is introduced in (O'Brien, 1970) on complex systems' safety, and it is used in fault diagnosis in (Okuda and Miyasaka, 1991; Shibata *et al.*, 1991).

Most encountered systems in technical or economical domains are *conductive flow systems* (CFSs) (Cellier, 1995) – i.e., they transport matter, energy and information as flows passing through pipe-like paths. Through the effects propagation, same effects may appear at many faults, located at faulty and non-faulty units. In such cases, the human diagnostician deals with primary and secondary effects, i.e., effects located at the faulty component and effects spread to nonfaulty components, respectively.

Means-end modelling approach is a view on artefacts from the utility perspective: the ends (concrete goals of the artefact) are those structuring the means (functional structures) supported by physical components. In (Larsson, 1992) a component performs a "flow function" (and a module a network of flow functions) – acting upon the flow.

8.4.1.1. Multifunctional Systems

A multifunctional system (MFS) under the diagnosis is the 5-tuple $\langle \boldsymbol{C}, \boldsymbol{G}, \boldsymbol{S}, \boldsymbol{T}, \boldsymbol{H} \rangle$:

$\boldsymbol{C}$ is the set of all physical components, each component meant as the final location for fault isolation, each completing certain functions;

$\boldsymbol{G}$ is the set of functions components may accomplish;

$\boldsymbol{S}$ is the set of ends, each end characterized by performance of a certain utility that the system must accomplish;

$\boldsymbol{T}$ is the set of time durations in accomplishing (each of all) ends;

$\boldsymbol{H}$ is the set of modules, each module h_i comprising a subset C_i of components and accomplishing a subset S_i of ends.

An elementary end s_{ik} is achieved during (and corresponds to) an *activity* – from the Discrete Event System abstraction of the h_i module's running. A module may accomplish more ends. For example, a hydraulic conveyor executes four activities corresponding to the four ends of the actuator (the hydraulic cylinder): still left, move left-to-right, still right, move right-to-left, each being a function of the actuator component.

The set of modules $\boldsymbol{H}$ is a disjunctive partition upon the set $\boldsymbol{S}$ of ends, each module accomplishing a specific subset of ends S_i but only one end s_{ik} at a time. In the example above, the module comprises components as pipes, control valve,

damper, hydraulic cylinder. The ends are the "move" or "stay still" services, and the durations in accomplishing those ends are either specified – e.g., the expected duration for each movement of the piston, or derived – e.g., the stay-still duration (between movements). The relations between cardinalities $|\boldsymbol{S}|=|\boldsymbol{T}|$ and $|\boldsymbol{S}|>|\boldsymbol{H}|$ hold; in other words, each end has a certain duration (in normal and abnormal situations) and a module exhibits at least two activities (idle/active) to a certain end.

8.4.1.2. Multifunctional Conductive Flow Systems

Multifunctional conductive flow system (MCFS) is the 7-tuple $\langle \boldsymbol{C},\boldsymbol{G},\boldsymbol{S},\boldsymbol{T},\boldsymbol{U},\boldsymbol{H}, \leq\leq\rangle$:

$\boldsymbol{C}$, $\boldsymbol{G}$, $\boldsymbol{S}$, $\boldsymbol{T}$ are as above;

$\boldsymbol{U}$ is the set of flow types; a certain flow type u_t is processed by components of a module toward a specific end by means of specific functions of components;

$\boldsymbol{H}$ is as above, but restricted to the subset C_i of components that act upon the same flow type u_t .

$\leq\leq$ is the weak upstream relation taking place between components c_{ij}, and between modules h_i along the flow paths in the conductive flow system.

The (matter/energy) flow conduction is ruled by specific laws that are not captured in the definition above but will be discussed later (see Section 8.4.2) in the discrimination of primary from secondary effects at faults.

Note that upstream relations of neighbour components depend on the activity; for example, the "hydraulic cylinder" has an upstream relation with a component when the piston moves left-to-right (filling its left chamber) and downstream relation with the same component when the piston moves right-to-left (filling its right chamber).

In the proposed approach, MCFS appears as a multiple layered structure of conductive flow systems, each of them handling a certain type of flow and acting toward some definite ends on the same set of components. For example, the "mineral oil flow" in the hydraulic installation of a rolling mill plant is an auxiliary flow beside the "long steel plate flow" meant for the (main) technological end – plate extrusion.

8.4.1.3. Means-End Abstraction on Functions

Each component c_{ij} fulfills a certain flow function during a certain activity, upon a certain type of flow u_t, but it may fulfill simultaneously more flow functions, each upon different flow types "passing" through the component. For example, a control valve in a hydraulic system may complete a "barrier" flow function (when blocking the flow for "piston stay-still" end) or a "transport" flow function (when letting through flow for "piston move" end); on the other side, the control valve always exhibits a "transport" flow function for the electric current through the control coil of the valve.

Each module $h_i \in \boldsymbol{H}$ achieves a certain end by means of the functions g_{ij} specific to the components in the set C_i of the given module. Other aspects of the flow functions follow:

a) the component c_{ij} fulfills a unique "flow function" upon a certain flow type, during a certain activity of the module h_i (according to Larsson (1992));

b) the end s_{ik} of a module is accomplished by the set C_i of components by means of the "network" of "flow functions" (Larsson, 1992);
c) the module is actually a functional unit, comprising only components that process the same u_t flow type (in the presented approach).

8.4.1.4. Qualitative View on Flow Functions

The detailed flow functions (transport, barrier, distribution, etc.) in (Larsson, 1992) may be reduced to three qualitative functions, sufficiently relevant for the diagnosis task, while it is somehow simpler and more qualitative than the control task. In (Opdahl and Sindre, 1994) three orthogonal operational facets of real-world systems are proposed, as in Table 8.1.

Table 8.1. Functional orthogonal facets of real-world systems

Concept	*Process*	*Flow*	*Store*
Activity	Transformation	Transportation	Preservation
Aspect	Matter	Location	Time

The concepts refer to physical or chemical processing (see *Process*), the space location change (see *Flow*) and the time location change (see *Store*), i.e., time delay.

The activities associated with the three concepts suggest three primary flow functions, suited to the qualitative modelling of components' faulty behaviours. For each concept in Table 8.1 the corresponding primary flow function is:

i) flow processing function (*fpf*) – like chemical or physical transformation of the piece of flux (to a certain utility);
ii) flow transport function (*ftf*) – like space location change of the piece of flux (by pipes, conveyors, etc.);
iii) flow store function (*fsf*) – like time delay of the piece of flux, by accumulation of mass or energy in some storing or inertial components.

A real component achieves several primary flow functions, but solely one during a given activity. Note that components that directly accomplish ends of the target system, fulfill processing (*fpf*) and store (*fsf*) primary flow functions; most components fulfill transport (*ftf*) primary flow function. Flow function's misbehaviour is easily associated with some generic anomalies that may appear at faults (see Section 8.5.2).

8.4.2. Bond-Graph Modelling and MCFS's Structures

Conductive flow modelling of real systems observes Kirchkoff's laws, no matter the type of flow (matter, energy or information). Bond graphs are appropriate and general modelling tools for conductive flow systems, with the great advantage of Kirchkoff's laws applied in a modular way, and not for the whole system as in the

classical way (Cellier, 1995; Mosterman *et al.*, 1995). Moreover, bond-graph modelling offers general concepts useful for behavioural abstractions of the flow functions for every type of flow (see below).

8.4.2.1. Modularisation by Bond Graph Junctions in the Target MCFS

Bond graph modelling deals with flow power variables: the intensive (pressure like) and the extensive (flow-rate like) variables, called *effort* (*e*) and *flow* (*f*), respectively (Cellier, 1995).

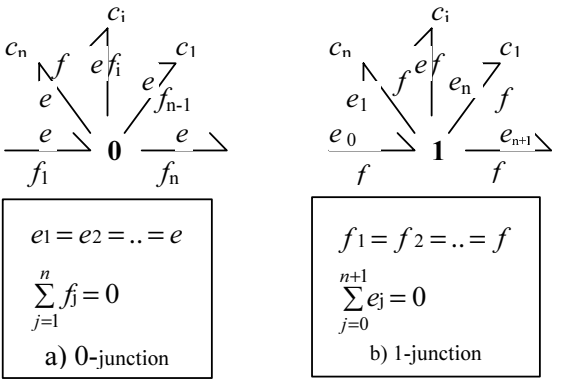

Figure 8.4. The bond graph 0-junction (a), and 1-junction (b).

Components, along flow paths in CFS, form bond graph junctions:

- type 1 junction – that corresponds to a loop of interconnected components,
- type 0 junction – that corresponds to a node of interconnected components.

Each junction's common variables are: effort in 0-junction and flow in 1-junction; the noncommon power variable is specific to each component and all enter a sum (e.g., the flow in the 0-junction), as in Figure 8.4a,b.

In the present approach, the 1-junction corresponds to a given activity of a module, i.e., the 1-junction is the bond graph model of the activity, so it may play the role of the "module" – in the multifunctional abstraction (Ariton, 2003). The 1-junction is already a network of flow functions – complying with the means-end point of view.

The conclusions above are useful in knowledge elicitation of modules, during MCFS hierarchical decomposition. In this view, the 0-junction is the interconnection of modules, and the structure of the whole target system is made of 0-junctions.

8.4.2.2. Primary Flow Functions and Bond-Graph Components

The large generalization specific to the bond graph approach is synthetically illustrated in the tetrahedron of state in Figure 8.5 (Cellier, 1995). Variables on flow conduction may have specific meanings to specific domains: the effort *e* may correspond to force (in mechanics), to voltage (in electricity), to pressure (in

hydraulics), flow *f* may correspond to velocity, to current, to volume flow rate (in the respective domains). Other general concepts in bond graph modelling approach are: the generalized momentum *p* (momentum in mechanics, flux in electricity, etc.), and the generalized displacement *q* (distance, charge, etc.).

The presented approach extensively uses the concepts of *bond graph components*:

- power flow components: Resistance *R*, Capacitance *C*, Inductance *I*, corresponding to dissipative, storage and inertial elements, respectively;
- power transfer components: transformer *TR* (effort-effort and flow-flow ratios) and gyrator *GY* (effort-flow and flow-effort ratios).

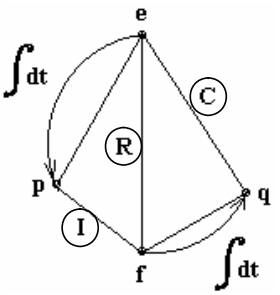

Figure 8.5. The tetrahedron of state and the bond graph components *R*, *C*, *I*.

Components of MCFSs have projections on bond-graph and means-end perspectives:

- *R* component corresponds to transport function (*ftf*);
- *C* and *I* components correspond to storing function (*fsf*);
- *TR* and *GY* components correspond to processing function (*fpf*).

This result is useful in the faulty behaviour modelling (see Sections 8.5.1 and 8.5.2) and in the hierarchical decomposition of the target system toward components (see Section 8.6.2.1).

8.4.2.3. Upstream Relations between Modules and Components

The bonds (half-arrows in Figure 8.4) indicate the flow but do not refer to the upstream/downstream relations between components. Those relations are important in locating the effects along flow paths (see Section 8.5.3.3).

In Figure 8.4 the indices $j \leq 1 \leq n$ show the components' upstream order between components c_{i1}, c_{i2}, $c_{i3} \in h_i$ (belonging to the same module) and direct neighbours (which input / output ports are directly coupled). Neighbour modules also exhibit upstream relations.

The upstream relation is strong (<<) at 1-junction: $c_{i1} << c_{i2} << c_{i3}$ when the order of two neighbours is strict, while they are output-input coupled and the flow strictly gets out from one component and gets in the neighbour one.

The upstream relation is weak (≤≤) at 0-junction: $c_{i1} \leq\leq c_{i2} \leq\leq c_{i3}$ when two neighbour components' ports are input-input or output-output coupled, so for both neighbour components the flow either gets out or gets in the coupled ports.

The two bonds of indices 0 and n+1of the 1-junction indicate effort at the input and at the output of the series of components, and actually represent links to the upstream and downstream 0-junctions, respectively.

8.5. Concepts and Structures on Faulty Running

Elicitation defines knowledge pieces (some of them discriminated above) but also prepares corresponding data for further processing. The chapter introduces knowledge pieces related to faulty behaviour and their representation for the computational model.

8.5.1. Generic Faulty Behaviour of CFS's Components

Following the above approach, the faulty behaviour of components of the target CFSs is conceived as human-like symptoms attached to various faults of the real components:

- Faults in *R* component affect the transport function (*ftf*); manifestations refer to *R* parameter changes, and the symptoms refer to propagation of power deviations along the paths in the system (discussed in Section 8.5.2.2).
- Faults in *C* and *I* components affect the storing function (*fsf*); manifestations refer to changes in time delays in the process running.
- Faults in *TR* and *GY* components affect the processing function (*fpf*); manifestations and symptoms are specific to each end of flow processing.

Faults may occur at any components but only *R* components are involved in power propagation along the system. Consequently, deviations of the power variables *e* and *f* propagate from the faulty component to other components, where they indirectly affect specific parameters – for example the delay for *C* and *I*, or the transferred effort and flow for *TR* and *GY*.

An important conclusion is drawn from the statements above: the anomalies of *R* bond-graph components are primary effects, and they provoke secondary effects by means of flow power variables deviations propagated throughout the flow path in the target CFS. Another important conclusion, from the point of view of diagnosis, is that the discrimination of primary effects from secondary effects leads to fault isolation.

The *TR* and *GY* components correspond to actuators in the target system, and they decouple flows or modules. Hence, the two components are, usually, the final components in the network of flow functions, i.e., they are components at the border between two modules. For example, the carrier of a conveyor is not part of the module, while the hydraulic cylinder is a transformer from the effort of the

mineral oil towards displacement (of the carrier). Actually, the carrier and its load are part of another module, decoupled by the hydraulic cylinder (as a transformer). So, the b) item from the Section 8.4.1.3 is observed.

8.5.2. Anomalies Related to Primary Flow Functions

Anomaly is a piece of knowledge indicating a class of abnormal behaviours; it is another word for symptom, which is used in the present approach to restrict the meaning of the symptom to a deviation from the expected behaviour of one of the three primary flow functions defined above. The anomaly is located at the faulty unit, i.e., it is a "primary effect." This way, the fault isolation procedure benefits from some additional information useful when the location of the fault is of concern.

8.5.2.1. Anomalies and Primary Flow Functions

Flow process anomaly, flow store anomaly and flow transport anomaly are disorders of respective flow functions, located at the faulty component or module:

a) Process anomaly (*AnoP*) appears at the actuator components – bond-graph gyrator *GY* or transformer *TR* components. Process anomalies refer to abnormal values of performance parameters of the end envisaged.
b) Store anomaly (*AnoS*) appears at storage or inertial components – bond-graph capacitance *C* and inductance *I* components. The store anomaly refers to abnormal values of the time delay appearing at faults in storage or inertial elements.
c) Transport anomaly (*AnoT*) appears at dissipative component, in the bond-graph view resistance *R* components. In fault diagnosis literature and practice "leakage" or "clogged pipe" are usual terms for such anomalies.

8.5.2.2. Transport Anomalies

Ariton (2003) introduces four orthogonal transport anomalies that completely cover the faulty behaviour of a component involved in the flow transport, namely:

d) *Obstruction* (*Ob*) – consists in change (increase) of the transport *R* parameter of a component, without flow path modification (e.g., clogged pipe).
e) *Tunnelling* (*Tu*) – consists in change (decrease) of the transport *R* parameter of a component, without flow path modification (e.g., broken-through pipe).
f) *Leakage* (*Le*) – consists in *structure* changing (output flow too low) of a flow transport component, involving flow path modification.
g) *Infiltration* (*In*) – consists in *structure* changing (output flow too high) of a flow transport component, involving flow path modification.

Transport anomalies are orthogonal (see Figure 8.6): inside the pair and between pairs. In Figure 8.6 the axes' names indicate the "main" power flow variable for the pair, the one mainly involved in the effect at the respective pair of transport anomalies. Note that the effort for *Ob/Tu* pair is meant at the input, and the flow for *In/Le* pair is meant at the output of the given flow transport unit (component or module), so the signs depicted in Figure 8.6 are specific to those situations.

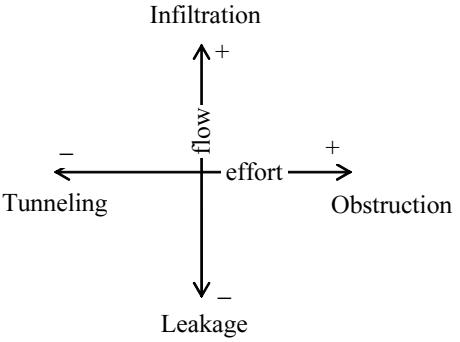

Figure 8.6. Orthogonal transport anomalies.

Solely, one transport anomaly may appear at a time vis-à-vis a faulty component.

The four transport anomalies are effective concepts in the qualitative modelling of faulty behaviour and in effects propagation. As later shown, transport anomalies are of seminal importance in the discrimination of primary effects from secondary ones, in detection and isolation of faults.

Various components in real systems are involved in flow transport, i.e., they act as *R* bond-graph components and may exhibit transport anomalies at faults.

The transport anomalies *Ob*/*Tu* are symptoms similar to events as "clogged paths" or broken-through paths, and *In*/*Le* are symptoms similar to flow exchange with the environment. The first pair observes the (expected) flow balance equations, while the second does not. Transport anomalies play a central role in fault detection, while they have the meaning of "primary effects" – i.e., effects located at the faulty component (or module). Asserting a transport anomaly means detecting a fault and also isolating the fault – while the transport anomaly location is asserted.

Process anomalies *AnoP* and store anomalies *AnoS* may appear as *secondary effects* when induced by the flow power deviations propagated through components with flow transport functions, along the flow paths, while the deviations appeared at the location of a transport anomaly *AnoT* that occurred as a *primary effect*.

8.5.3. Qualitative Deviations Induced by Transport Anomalies

The following study focuses on deviations of the effort e and the flow f of bond-graph power variables at faulty and nonfaulty bond-graph R type components.

8.5.3.1. Qualitative Behaviour of R Components

The qualitative relation between the power variables for a nonfaulty component is $e = M^{+} \cdot f$, according to the general qualitative Ohm's law (Struss, 1997). The flow variables' deviations from expected values at the input port comply:

$$\Delta e = M^{+} \cdot \Delta f \tag{21}$$

where M^{+} is a class of increasing monotonic functions (according to qualitative physics and notations from (Kuipers, 1994)). Δe and Δf refer to power variable finite deviations (due to some external causes of the nonfaulty component. The qualitative relation Eq. 21 also holds for the flow variables at the output port (note that no concern exists in the extent of the relation).

8.5.3.2. Power Deviations at Faulty and Non-faulty R Components

As presented in Section 8.5.2.2, the faulty flow transport components induce one of the four orthogonal symptoms (transport anomalies) shown in Table 8.2.

The deviations of effort and flow variables from the expected (normal) values are specific to R bond-graph component for the given transport anomaly (*Ob*/*Tu*, *In*/*Le*). The deviations' signs (i.e., the qualitative values) simply result from the affected parameters of R and of the main variable in the context of the transport anomaly.

Table 8.2. Flow power variables' deviations at input and output ports of R bond-graph components for each transport anomaly occurrence

Transport anomaly	"Main" variable deviation for the anomaly class	Effort deviation at the input (output) ports	Flow deviation at the input (output) ports	Qualitative effort-flow relations
Obstruction (*Ob*)	$e_{in\text{-}out} > 0$	$\Delta e_{(in)} > 0$ ($\Delta e_{(out)} < 0$)	$\Delta f_{(in)} < 0$ ($\Delta f_{(out)} < 0$)	M^{-} (M^{+})
Tunneling (*Tu*)	$e_{in\text{-}out} < 0$	$\Delta e_{(in)} < 0$ ($\Delta e_{(out)} > 0$)	$\Delta f_{(in)} > 0$ ($\Delta f_{(out)} > 0$)	M^{-} (M^{+})
Infiltration (*In*)	$f_{out} > 0$	$\Delta e_{(in)} > 0$ ($\Delta e_{(out)} > 0$)	$\Delta f_{(in)} < 0$ ($\Delta f_{(out)} > 0$)	M^{-} (M^{+})
Leakage (*Le*)	$f_{out} < 0$	$\Delta e_{(in)} < 0$ ($\Delta e_{(out)} < 0$)	$\Delta f_{(in)} > 0$ ($\Delta f_{(out)} < 0$)	M^{-} (M^{+})

As shown in the last column of Table 8.2, the qualitative relation between the deviations of flow variables at the input port is:

$$\Delta e = M^{-} \Delta f \tag{22}$$

where M^{-} is a class of negative monotonic (decreasing) functions. It seems that the relation does not comply with the general Ohm's law; note that Eq. 22 refers to

deviations from *expected values*, so it is not the Ohm's law in question but variables' deviations.

Equations 21 and 22 are the basis of the qualitative modelling for the effects' propagation along the flow paths in the conductive flow system.

8.5.3.3. Signatures of Qualitative Deviations at Flow Transport Anomalies

The transport flow function reflected by *R* bond-graph generic component is involved in the propagation of flow power and also in propagation of the deviations of the flow power variables when faults occur. The propagated flow power deviation reaches a neighbour nonfaulty component involved in the flow transport, and affects the effort (at input port) and the flow (at output port) values depending on the bond-graph junction they share.

Table 8.3 presents the signatures of manifestations for the effort and flow variables corresponding to each transport anomaly and to each type of bond-graph junction. The signatures are patterns expressed in terms of qualitative deviations (*lo* – "too low" and *hi* – "too high") for the flow variables at a nonfaulty component sharing the same bond-graph junction with the faulty one. Note that both (faulty and nonfaulty) components are flow transport (*R* bond-graph) components; hence they are both involved in the flow power deviation's propagation (from the *AnoT* "cause" location).

Table 8.3. Signatures of the transport anomalies as effort-flow manifestations at the input-output ports (respectively), in each type of bond-graph junction

Transport anomaly (*AnoT*)	1-**junction** 1>>2>>3 shadowed item is AnoT (the faulty component)		**0-junction** 1 >> 2 ≤≤ 3 4 fault downstream (of Kirchkoff's node)		**0-junction** 1 >> 2 ≥≥ 3 4 fault upstream (of Kirchkoff's node)	
	1 >> 2	3 << 2	1 >> 2	4 ≥≥ 2	2 << 1	3 ≤≤ 1
Obstruction (*Ob*)	*hi-lo*	*lo-lo*	*hi-hi*	*hi-lo*	*lo-hi*	*lo-lo*
Tunneling (*Tu*)	*lo-hi*	*hi-hi*	*lo-lo*	*lo-hi*	*hi-lo*	*hi-hi*
Infiltration (*In*)	*hi-lo*	*hi-hi*	*hi-lo*	*hi-lo*	*hi-lo*	*hi-hi*
Leakage (*Le*)	*lo-hi*	*lo-lo*	*lo-lo*	*lo-lo*	*hi-hi*	*lo-lo*

If the flow power deviation reaches the location of GY/TR bond-graph (actuator) component, or of C/I (store/inertial) bond-graph component, a secondary effect appears, expressed by the *AnoP* or *AnoS* anomalies. Those effects actually reflect the *AnoT* anomaly propagated as power flow deviations along the flow paths throughout the target system.

Manifestations at nonfaulty components are expressed in terms of qualitative deviation of the effort – at the input port, and of the flow – at the output port, in pairs (*hi-lo*, *lo-lo*, etc.), and they result from the qualitative relations of the flow power variables at faulty (Eq. 22) and nonfaulty (Eq. 21) components (Ariton,

2003), in the corresponding behaviour contexts (the triplet: junction type, upstream relation, transport anomaly).

The signatures with manifestations at the components upstream/ downstream from the faulty one are specific to the transport anomaly (*AnoT*) and the junction type; the only exceptions are Tunnelling and Infiltration in 0-junction (column 3 of the Table 8.3), which cases should be decided based on relations in neighbour 1-junction(s). Note that weak relations ($\leq\leq$ / $\geq\geq$) are equivalent for the meant study of qualitative signatures.

8.6. Knowledge Elicitation and the CAKE Tool

Diagnosis performed by human experts involves deep knowledge and shallow knowledge on a real target system comprising many modules and components, many activities, many faults, manifestations and symptoms.

It is difficult to manage the huge amount of information if no adequate instrument exists, i.e., a Computer Aided Knowledge Elicitation (CAKE) tool. Such a tool assists the human diagnostician in the knowledge acquisition phase and in managing the information on the concrete target system. Therefore, the knowledge acquisition is performed more easily and the computational model is easily adapted to specific situations on the place. The CAKE (software) tool takes the place of the knowledge engineer, who is the essential human expert in the design phase of a dedicated diagnosis system. So, human diagnosticians and human operators do not need a knowledge engineer to build their own diagnosis system (for the target system) but they simply put all the information into it guided by the software tool.

8.6.1. Elicited Concepts with the Aim of Fault Diagnosis

The concepts' representation involves a combination of models presented above and concisely noted below, along with their role and use:

a) *Means-end* modelling of hierarchical structures for the multifunctional aspect:
   i. role – identifies deep knowledge on physical and functional structures (components and simplified functions, modules and ends);
   ii. use – define behavioural patterns at faults based on proposed primary flow functions.

b) *Discrete event* modelling of the running context for the multifunctional aspect:
   i. role – identifies deep knowledge on activities toward ends of modules;
   ii. use – determines current activity of a module and its time limits.

c) *Bond-graph* modelling of components for the flow conduction aspect:
   i. role – identifies deep knowledge on flow conduction as bond-graph junctions and components;

ii. use – associates functions to bond-graph components and generic anomalies observing effects propagation.

d) *Qualitative* modelling of concepts and relations for the faulty behaviour:
   i. role – describes deep knowledge on faulty behaviour: faults (at component level), symptoms (as generic anomalies), observations and manifestations (with prototype and instance attributes);
   ii. use – detects faults (by instance manifestations and symptoms) and hierarchically isolate faults (at module and then component levels) by recognizing cause-effects as from deep and shallow knowledge of human diagnosticians.

The models follow the human expert's common view on diagnosis: items a and b cover the discrete view on the structure and the behaviour in normal situations, while item d covers the discrete view on the behaviour in faulty situations. Item c. covers the continuous view on fault effects propagation by flow conduction. The paper proposes a qualitative view on faulty behaviour of components and a procedure to assert primary effects from the propagated (secondary) effects.

The data on real running of the target system have a close representation to the human diagnostician's view, through:

e) *Fuzzy logic* – for the "intelligent encoding" of observations to manifestations:
   i. role – encodes "prototype manifestations" as meaningful intervals according to the deep knowledge of human diagnostician;
   ii. use – obtains "instance manifestations" from the actual values collected from sensors during installation running.

The diagnosis follows modular and incremental procedures, carried out by:

f) *Inference engine* – for fault detection and sequential diagnostic refinement:
   i. role – detects abnormal behaviour (symptoms) and sequentially performs diagnosis for temporal sliding windows and for newly observed variables;
   ii. use – locates a transport anomaly at module level, then starts the neural network recognition process for further fault isolation.

g) *Artificial Neural Networks* – for recognition of the faults:
   i. role – embeds shallow knowledge from practice and experiments as links between manifestations and symptoms to faults;
   ii. use – isolates faults by recognizing patterns of manifestations and anomalies.

The diagnostic is obtained by recognizing patterns of manifestations and symptoms associated with faults. Items e to g are computational models that emulate the human diagnostician's way of acting, and directly embed human

concepts in their native form. The diagnosis proceeds incrementally, following the sequence of activities of the modules during the target system's running and adding new observation meant to refine the diagnostic.

The knowledge pieces for diagnosis involve a large amount of data that should enter the diagnosis expert system (Patton *et al.*, 2000). Each concept addresses a set of specific information:

- module – name, ends, activities, specific set of components, up-stream relations to neighbour modules, junctions and signatures for each transport anomaly identification, nonspecific observations (e.g., mud);
- activities – code, next activity, time limits;
- component – name, primary flow function and bond-graph component for each activity of the host module, set of specific faults, component and module located manifestations;
- fault – name, (deep knowledge) links from manifestations and anomalies of the flow function in the host component, abductive relations to causes from the target system or environment, (shallow knowledge) links from other manifestations in the host module;
- manifestation – name, source type (sensor or human operator observations), prototype attributes and ranges of values (specific to the activity of the host module), abductive relations to causes;
- anomaly – type (*AnoP*, *AnoS*, *AnoT*), host component or module, end parameters values for abnormal behaviour, etc.

Knowledge elicitation will provide data for building the structures of ANN blocks (e.g., data on layers of neurons for manifestations and faults, for the abductive links between them, for training with patterns). Knowledge elicitation provides data for the inference engine of the diagnosis expert system: the series of activities for each module, order of 0-junction for which signatures of neighbour modules identify the transport anomaly, etc.

The knowledge pieces enter the Knowledge Base for consistency checking and for storing concrete data in the appropriate representation. After elicitation, the training of ANN blocks follows, then the diagnosis expert may be generated as a dedicated software for the given target MCFS.

All knowledge pieces, presented in previous sections, are specific knowledge structures that the CAKE tool deals with. The structures refer, for example, to the physical and functional units of the target system, to the systems interconnected with the targeted one, to all situations that may disturb or originate faulty situations.

The feasibility condition, meant for the computational model of the fault diagnosis system, is to assure closed spaces for causes and effects. Abnormal behaviours are not only caused by faults at components but also by any other abnormal situation inside the target system or coming from outside. To cope with such cases, the concept of *disorder* is introduced. Disorder refers to any cause that will induce an abnormal situation: human operator mishandling (e.g., ill tuning, infringement of technological rules, etc.), ill state of matter or energy flows (e.g.,

the quality), abnormal conditions in the environment (e.g., too hot or too cold), and negative influences from the neighbour systems.

Fault diagnosis deals with various aspects of the target system, each of them identified as a subsystem:

a) *Physical Subsystem* – refers to all physical units (e.g., modules and components) and hierarchical structures (e.g., the whole installation and the modules) as means for achieving the ends of the system. Regarding the diagnosis, they represent the locations for faults.
b) *Functional Subsystem* – refers to all functional units (primary flow functions) and hierarchical structures (process phases and activities), which actually achieve the ends of the system. Regarding the diagnosis, they represent locations for the behavioural aspects of the target system.
c) *Behavioural Subsystem* – refers to all concepts related to the abnormal running of the target system: observations, manifestations, symptoms and faults, along with their links.
d) *Operational Subsystem* – refers to the human operator actions that may provoke an abnormal situation.
e) *Flow Subsystem*(*s*) – all types of matter or energy flow that may induce abnormal situations (e.g., the "foaming oil" in a hydraulic installation).
f) *Environment* – refers to all systems out of the diagnosis contour (i.e. the target system): the ambient atmosphere, the mounting conditions, and the neighbour systems.

All knowledge pieces become entities related to each other that should be indicated by the knowledge engineer and should enter the computational model for fault diagnosis, as further presented. The structures of knowledge pieces are further presented in the entity-relationship diagrams that follow.

8.6.2. Elicitation Aspects on Normative and Faulty Models

The normative model consists of physical and functional structures that support the ends' achievement. They comprise entities specific to their corresponding subsystems, presented in the previous sections.

The diagrams in Figures 8.7 and 8.8 are UML representations of entities relations elicited for the corresponding subsystems. Having in mind fault diagnosis, in each diagram will appear the two entities *Disorder* and *Fault* – the last inheriting the first one. The dashed ellipses indicate borders of the other subsystems.

8.6.2.1. The Physical Subsystem

The entities involved in the Physical Subsystem are `Component` (the entire set ***C***), `Module` (the entire set ***M***), and `Installation`; all of them are locations of faults. However, there are disorders that may produce similar effects as faults, which are located in other systems (Flow, Operational or Neighbour systems).

The discrimination of the physical units proceeds from the means-end view (as MFS) and from the bond-graph view (as CFS), following the hierarchy of physical/functional units. For each flow type u_t, the knowledge engineer should

assert the end of the modules, then the primary flow functions of the comprised components along with the associated bond-graph generic component. So,

- Modules – result from ends (and activities) accomplished towards products / services achieved, and correspond to bond-graph 1-junctions.
- Components – result from primary flow functions completed in each activity, and correspond to certain bond-graph components.

Fault *isolation granularity* is the extent of the decomposition of the physical structures into components, hence the cardinality of ***C***. The fault isolation granularity reflects human diagnostician's troubleshooting pragmatism regarding the sufficient location of disorders for their removal; it also reflects the incompleteness of human knowledge on physical structures and on the environment. Usually, a component may exhibit more faults, so ***C*** induces a disjunctive partitioning over ***F***.

The discrimination of physical components – sufficient for fault isolation – follows the hierarchical structure of the target system, and proceeds to a combined decomposition observing the physical and the functional structures:

i. from the entire `Installation` – which is also the whole `Process`,
ii. decomposition proceeds to `Modules` – each referring to a `Subprocess` with two or more `Activities`,
iii. then each `Activity` is decomposed in primary `Flow Functions` – each being attached to a `Component`.

The relations between entities – with the corresponding multiplicity attached to each relation – are illustrated in Figure 8.7 and they represent:

- Association «`loc`» (located to) directs to the location of the `Disorder`;
- Dependency «`evo`» (evokes) directs to the anomaly evoked by the `Disorder`;
- Inheritance `Fault` from `Disorder`;
- Composition of `Component` to `Module`, and to `Installation`.

The physical units (in the physical structures) present hierarchical relations and also upstream >> (strong) and ≥≥ (weak) relations, depending on the bond-graph junction the physical units enter. Upstream and downstream relations appear in the diagrams representing the bond-graph junctions of the target system, for each combination of activities of the participating modules, and for the components inside the module. While specific, those diagrams are not shown here.

8.6.2.2. The Functional Subsystem

The functional structure is also a hierarchical structure: activities (of each module) comprise flow functions (of each component) and each flow function is linked to a specific faulty behaviour. All knowledge on physical and functional structures is deep knowledge, while it comes from human experts' acquaintance with the domain and with the design issues of the target MCFS.

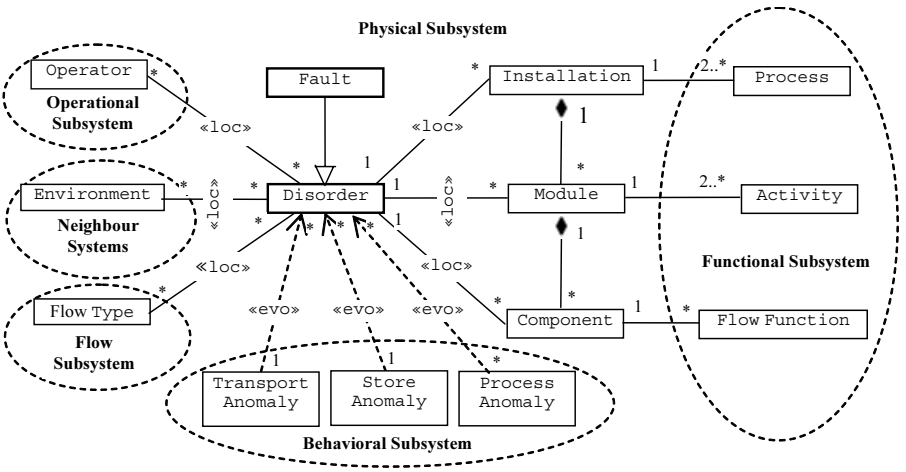

Figure 8.7. The UML diagram for entities and relations of the Physical Subsystem.

The entities of the Functional Subsystem are: `Process` and `Subprocess` (as general concepts related to the running of the whole `Installation` and of each `Module`). `Activity` is defined in Section 8.4.1.1, and – from the means-end point of view – corresponds to the network of flow functions for the components that leads to a certain (processing) end of the module. The `Process phase` is the current set of activities existing at a moment during the whole installation running. The `Operational Mode` indicates a state of the `Component` that leads to a primary flow function or to another, depending on the control action meant for the components (e.g., valve is open or shut). The fact that a `Disorder` depends on the `Activity` it appears, is represented by the constraint {and} upon the respective relations (note that {} stands for {and}, reduced because of the limited space).

8.6.2.3. The Behavioural Subsystem

The human diagnostician's view on manifestations and symptoms concerns:

i) deviations of the observed variables from the expected ("normal") values – where observations may refer to ends, effort and flow variables, linguistic values from human operator;

ii) deviations of functions that lead to abnormal ends – anomalies in the end's accomplishment, in the flow store or flow transport;

iii) propagation of the effects from the fault location – deviations of flow variables appear as primary effects (transport anomalies) and provoke secondary effects.

Entities on the faulty behaviour come from the deep knowledge of human experts in the domain and on the target MCFS, as presented in Section 8.5.

The relation «evo» indicate that a `Manifestation` evokes a `Disorder`, while «rev» indicates that an `Observation` reveals a `Manifestation`.

The {and} constraints between respective dependencies and associations indicate that the `Disorder` is specific to the `Anomaly` *and* the `Activity` that appear.

A causal relation that has an explanation represents deep knowledge. Relations that come from experiments or practice represent shallow knowledge that is embedded in the Artificial Neural Network (ANN) blocks. Shallow knowledge is embedded into the diagnosis expert system during the training procedures, based on known patterns acquired from practice (off line) or from experiments (on-line).

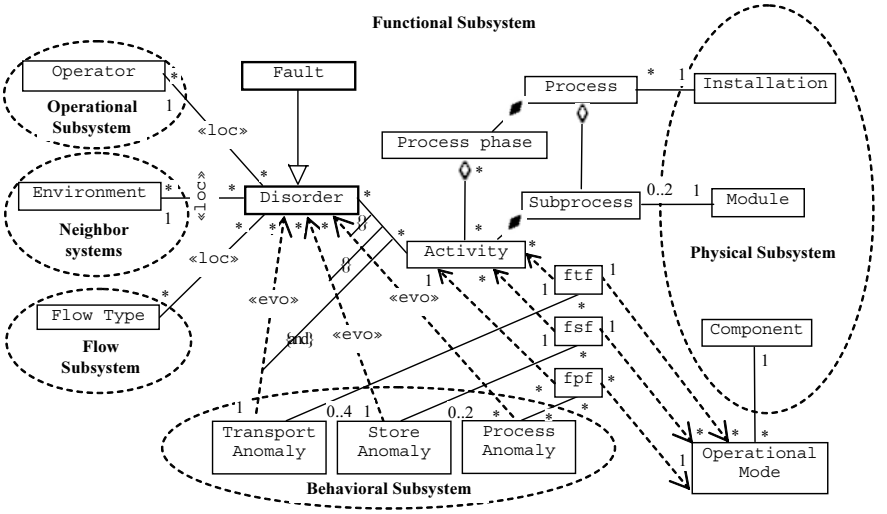

Figure 8.8. The UML diagram for entities and relations of the Functional Subsystem.

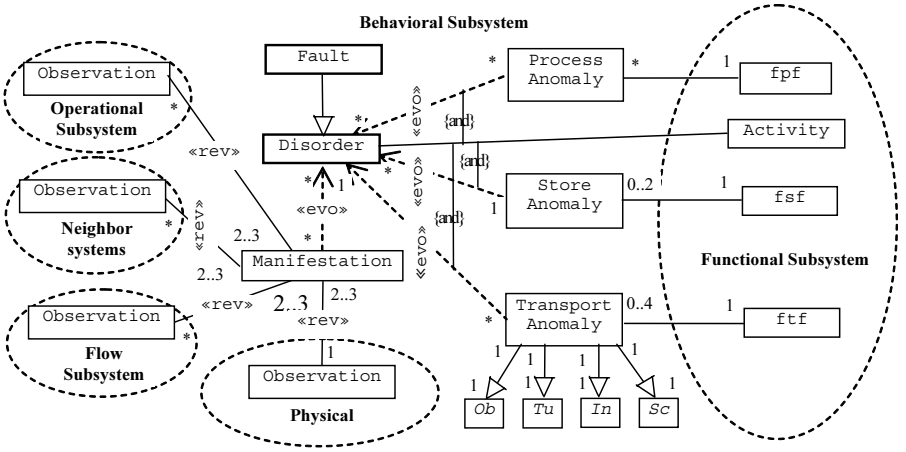

Figure 8.9. The UML diagram for entities and relations of the Behavioural Subsystem.

8.6.3. The CAKE Tool

Knowledge elicitation and knowledge acquisition are assisted by the Computer Aided Knowledge Elicitation (CAKE) software tool, which actually replaces the knowledge engineer who is involved in the design and implementation of the diagnosis expert system (Ariton and Baciu, 2002).

Knowledge elicitation proceeds by asking the operator about entities, values and relations, namely, on specific concepts of the subsystems in the target system. Knowledge elicitation activity consists of three phases: the top-down phase – which performs means-end discrimination of modules to components in the normative model, then the bottom-up phase – for collection of specific data on the faulty model, and finally the join phase – for establishing relations between all entities.

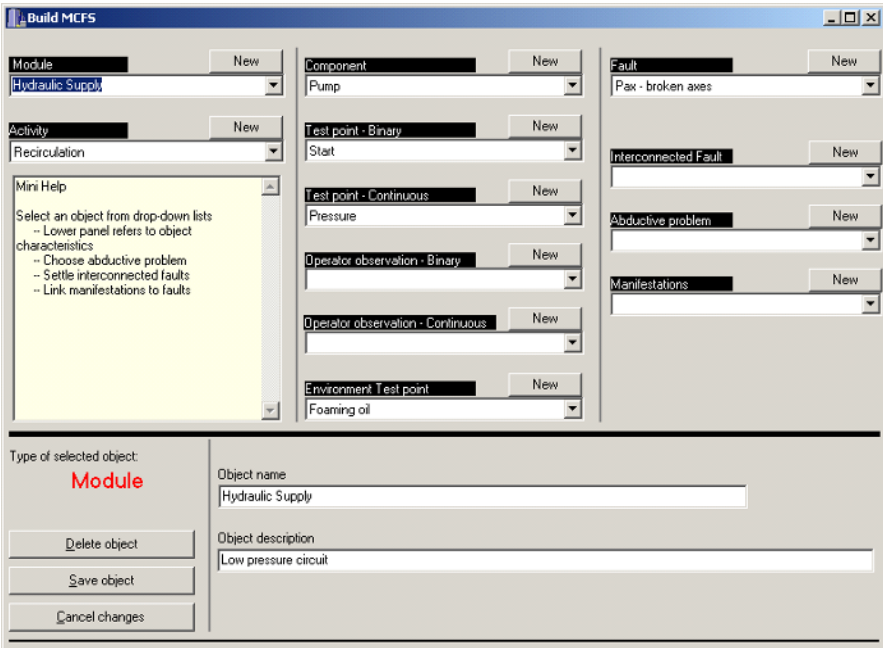

Figure 8.10. Screenshot for the CAKE screen for knowledge acquisition.

The top-down phase scans the layered structure of flows in the target MCFS, considering each flow type and "asking" for: modules (with activities and junction types), components (with flow functions and bond-graph components), faults and observed variables along with manifestations attached. The functional structure results from the functions attached to each physical unit: for each module – ends and activities they accomplish, for each component – the appropriate flow functions and the corresponding bond-graph generic component for each activity.

The bottom-up phase scans in the reversed order the physical and the functional structures, attaches faults to components, performs intelligent encoding

of manifestations, attaches manifestations to appropriate faults (from the shallow knowledge), and finally attaches anomalies to faults (from the deep knowledge).

The join phase puts together the existing modules in the respective bond-graph junctions (as from deep knowledge), attaches signatures to each junction, and indicates specific tasks for the inference engine (e.g., the order of bond-graph junctions to scan for transport anomalies).

The knowledge acquisition in the three phases refers to all knowledge pieces and relations for the target MCFS. The information is stored in the CAKE tool's Knowledge Base, which is specific to the target system. This way, data are prepared for the generation of a dedicated diagnosis application. Figure 8.10 shows a screenshot of the CAKE tool for MCFS building involved in the second phase.

The result of the knowledge acquisition is the complete description of the target system as text and data stored in the knowledge base. Following the text description and the knowledge base, the CAKE tool generates the code for a dedicated diagnosis expert system. The "Fault Isolation" (neural) blocks are later trained with faults-manifestations and faults-symptoms patterns, based on previously collected data from practice and/or experiments.

8.7. Fault Diagnosis System by Abduction

As already shown, the human diagnostician combines deep and shallow knowledge on the target system, and then isolates faults following hierarchical decomposition and incremental procedures in refining the diagnostic (i.e., finally locating the fault). The deep knowledge is more compact and it rapidly reduces the searching space based on laws from the domain ("explanations"). However, deep knowledge captures only general causal links and hardly refers to the diversity of effects and causes in the real running. So, shallow knowledge comes to describe the detailed behaviour in the uncertain and incomplete context of the complex real system.

8.7.1. Diagnosis Expert System's Structure

In Figure 8.11 is depicted the block structure of the Diagnosis Expert system and the place of the CAKE tool – which, actually, is not part of the diagnosis system. The diagnosis approach mainly focuses the knowledge regarding the faulty behaviour of the target MCFS, while knowledge regarding the normative model is only meant for the physical and the functional structures that will support the behavioural model in locating anomalies and faults.

All knowledge enters the "Knowledge Base" block, which in the proposed approach is simply a data base, while the normative and the faulty models are sets of behavioural units with parameters and links between them.

The "Knowledge Base" is the central block of the diagnosis expert system; data structures come from the "Knowledge elicitation" block (the CAKE tool included).

The actual data (values) come from the "Target MCFS" through the "Data acquisition and pre-processing" block which performs scanning, sampling and

intelligent encoding of data from sensors and from human operators; data channels are depicted as ⇨ in Figure 8.11.

The "Incremental diagnosis" block is the inference engine of the expert system; it controls the other blocks through control channels (depicted as simple arrows → in Figure 8.11). The inference engine's tasks are presented in Section 8.7.2.3.

The "Fault isolation" blocks are Artificial Neural Networks (ANN) dedicated and trained each for a given module faults recognition, based on patterns of manifestations and anomalies. The ANN blocks are connectionist models for abduction of faults from effects that embed deep knowledge on "abductive problems" of causes and effects (see (Ariton and Palade, 2004)), and also shallow knowledge on effects-to-causes pattern relations.

The "Human operator interface" block interacts with the human operator by asking and providing operator observations to "Data acquisition and pre-processing" block (arrow ↷ in Figure 8.11) and displays the diagnostic.

The "Knowledge elicitation and acquisition" block provides the knowledge (see ⇨ in Figure 8.11) for the "Knowledge Base" block, prototype manifestations for the "Data acquisition and pre-processing" block and faults-manifestations patterns for the ANN blocks. The "Knowledge elicitation and acquisition" is the CAKE tool (see Section 8.6) and it is actually the subject of the present work.

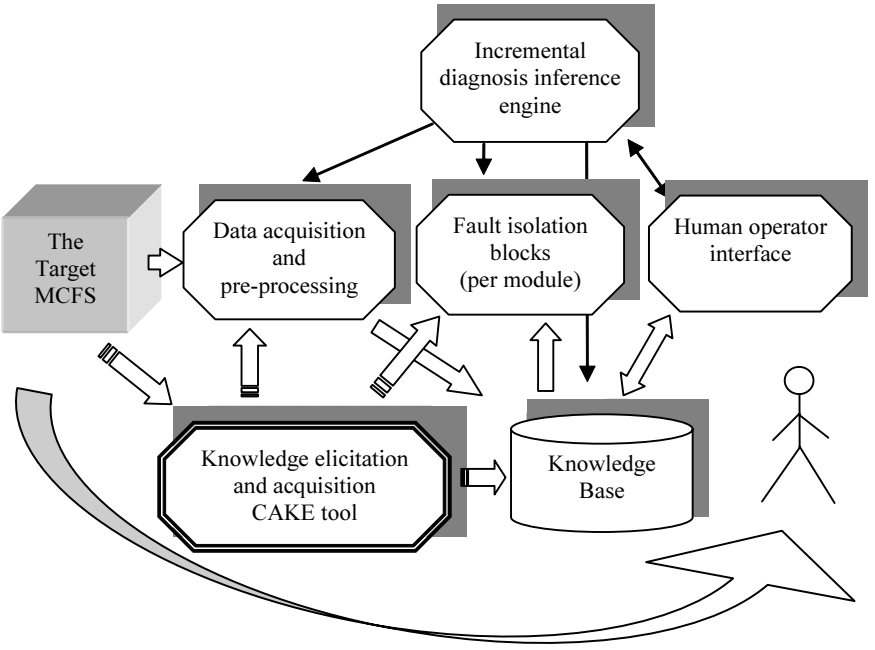

Figure 8.11. Diagnosis expert system and the place of the CAKE tool.

8.7.2. Modular and Incremental Diagnosis

Diagnosis proceeds by locating faults hierarchically, like the human diagnostician does:

- first discriminating the module with a transport anomaly,
- then recognizing fault(s) inside the module.

The transport anomaly is detected using signatures of manifestations on effort and flow variables at each module's input/outputs – see Table 8.3 – which leads to isolation of the faulty module. The existence of the transport anomaly is a confirmation of the faulty state and valuable information for further isolation of the concrete faults inside the module.

At the module level, it is possible to proceed the same way, i.e., to locate the faulty component detecting it by signatures of power variables' deviations. However, it is hardly the case that effort and flow are measured at every component in real installations. So, at the module level, fault isolation is performed by recognizing "pattern faults" from "pattern manifestations and symptoms," based on a dedicated ANN block provided for the module.

8.7.2.1. Parallel Processing for Modular Diagnosis

Manifestations (i.e., *lo*, *no*, *hi* linguistic values at the observed variables) and symptoms (i.e., process, store and transport anomalies) are input neurons and the faults are the output neurons of the ANN. All concepts have the appropriate representations as presented above: discrete (i.e., linguistic) knowledge and logical meanings (i.e., truth values). This way, the abductive reasoning of the human diagnostician may be described by the connectionist model proposed in Section 8.2.3.

The main advantage of the presented connectionist approach in diagnosis is the embedding of the human diagnostician's shallow knowledge by ANN training, using manifestations-to-faults patterns as from the actual behaviour of each module in the target system. It is worth mentioning that it is unrealistic to use a unique ANN block for an entire real system, while it deals with enormous numbers of combined causes and training patterns. By using the modular approach presented, the combination of manifestations-to-faults patterns is drastically reduced, and the fact that the human expert's shallow knowledge usually refers to the module level, even experiments on site or in laboratory conditions are conducted at the module level.

8.7.2.2. Testing Policy as an Abduction Problem Solving

The testing policy aims to indicate the best next test the result of which allows the optimal diagnostic refining, in other words the shortest path (as steps) to the diagnostic.

The "next best test problem" can be formulated as an abduction problem, and it can be solved in the same way as the diagnosis itself, i.e., as a connectionist implementation of plausibility and relevance of the next test to follow. Testing is performed stepwise, and takes part of the sequential diagnostic refinement.

The testing procedure requires human operator observations, but only a few are useful given the current situations (faults occurred, process phase, etc.), and given the entire set ***OM*** of human observed manifestations (see Section 8.3.3.3).

The current set of instance manifestations used at the training phase of the ANN block includes those observed by human senses (or portable measurement devices) and they should be provided as required at the time of diagnosis. In reverse, the embedded knowledge may be used to find out which is the plausible and relevant observation that the human operator should supply to advance the optimal diagnosis.

In this way, the next best test is obtained as the solution of the abduction problem solving, using plausible and relevance criteria as follows:

- *plausibility*(*P_CRITERIA, EFFECTS, CAUSES*) – whose outcome is the set of operator-observed manifestations *OM* (hence variables to be tested), based on the set of manifestations joined with the set of plausible faults obtained at the current step in the diagnosis.
- *relevance*(*R_CRITERIA*, *OM*) – whose outcome is the set of relevant operator observations out of the plausible ones, that satisfy *R_CRITERIA*.

The abduction problem is solved by means of a neural network implementation, and indicates the most plausible and relevant operator observation (if the competition is strict), or a set of observations (if the competition is relaxed), for which the human operator will supply data.

8.7.2.3. Incremental Processing for the Diagnostic Refining

The inference engine of the expert system with the same name, sequentially and repeatedly fulfills the following tasks:

i) Start data acquisition from the Target MCFS by means of the "Data acquisition and pre-processing" block, which also performs the „intelligent encoding."

ii) Identify the activities of all modules, during the current process phase (note that a process phase lasts between any two transitions of activities for any of the modules entering the same 0-junction).

iii) Detect faults – by identifying process and store anomalies.

iv) Detect transport anomalies and the faulty module – by identifying signatures of manifestations of effort and flow variables from Table 8.3.

v) Isolate fault(s) inside the faulty module(s), by means of manifestations and anomalies patterns, applied at the inputs of "Fault isolation block per module"; recognize fault using the dedicated ANN for the module.

vi) Evaluate the truth value of the "faulty" state versus the "normal" state for the entire target system.

vii) If "faulty" is greater than "normal" but no diagnostic exists (i.e., truth value of all activated faults is under a given threshold) ask human operator for additional observations and go to step i.

viii) If a diagnostic exists ("normal" and "unknown" included) and no further additional observations requested, display the diagnostic.

The inference engine cycle is standard but embedded knowledge and data are specific to the target system under the diagnosis.

8.7.3. Aspects of the Sequential Diagnosis

In the presented approach, sequential diagnosis involves three aspects:

a) Abduction by plausibility and relevance proceeds stepwise: first, plausible causes are obtained through feed-forward activations according to instance manifestations; second, the relevant faults are discriminated from the relevance groups, each group as a specific modularisation of faults, one modularisation applied at a time.
b) Process phases arise one after another, each process phase exhibiting specific plausibility criteria; consequently, the connectionist abduction is performed according to the (expected) current process phase.
c) Additional observations required from and supplied by human operator get into the diagnosis system, until no test is required – i.e., until the diagnostic is obtained (even if it is "no fault" or "unknown fault").

For aspects a and b above, an example of sequential diagnosis is presented in the previous section; item c refers to the next best test policy formulated as an abduction problem, and solved by plausibility and relevance implemented by neural networks. Note that "unknown fault" that occurs in the real running is finally decided by the human operator of the diagnosis system – when a faulty situation exists but no diagnostic provided.

8.7.3.1. Diagnosis by Plausibility and Relevance Criteria Sequentially Applied

Let us consider the diagnosis performed for a process phase P. After applying the plausibility criteria *P_CRITERIA* upon the set of *EFFECTS*, the set F^* of all plausible causes is deducted (i.e., the set comprising all causes with a positive activation). The "mass activation" of plausible faults is, by notation, $\sum F_i$, as given in Eq. 23, where F^* is the set of plausible causes.

$$\sum F_i = \sum_{F^*} F_i = \sum_{F} F_i \tag{23}$$

The sum is performed over the set F^* of plausible causes but it actually is the same if performed over the entire set F of causes, while nonplausible causes exhibit zero activation. So, the computational procedure always deals with the entire set F of causes, hence simple implementation.

Applying the relevance criteria *R_CRITERIA* upon the set of *CAUSES* will increase the activation of a plausible and relevant cause F_i^*, according to the layer's weight W^L (see Section 8.2.5.2). The increase will affect the numerical value of F_i^*, according to the mass activation $\sum F_i$ of all faults and to the weight W^L of the current modularisation layer:

$$\Delta F_i^* = W^L \cdot \frac{F_i}{\sum_i F_i} \tag{24}$$

In the proposed approach, the order of the relevance criteria applied is important, because the activation mass changes accordingly. The best order is the one of increasing weights W^L, so the activation mass $\sum F_i$ is updated only once, before the current layer processing. Each layer induces a graded increase of respective cause(s) activation, the last layer of modularisation inducing the highest increase.

After applying all relevance criteria, the relevance of faulty situation is determined (see Section 8.2.5.3) and the diagnostic is issued as the most relevant causes resulted, including faults with activation over the doubt level.

8.7.3.2. Testing Policy by Plausibility and Relevance Criteria Sequentially Applied

The next test is required after each diagnostic obtained. The diagnosis system "asks" the human operator to provide a certain variable value; he or she supplies the value, and so diagnosis based on plausibility and relevance restarts.

The most plausible and relevant operator-observed variable(s), for the given situation, result as an abduction problem solving according to Section 8.2.3. The next best observation (i.e., test) is indicated by the ANN block provided for each module, based on current faults and instance manifestations activated.

Now, the activation of plausible fault F_i^* changes according to new manifestations provided and, additionally, the activation is affected by the weight W^O attached to the operator-observed variable provided at the current step:

$$\Delta F_i^* = W^O \cdot \frac{F_i}{\sum_i F_i} \tag{25}$$

The human diagnostician sets up weights for the operator-observed manifestations according to the deep knowledge in the domain, provided $\sum W^O = 1$ for the set of operator-observed manifestations in the relevance group. The human operator supplies the observed values (manifestations) in the reverse order of weights W^O. That is, the values of the most important variables are provided first.

In the economy of the diagnosis by next best test, the most important role is played by Eq. 11, which starts the next test procedure if the *FAULTY* situation prevails over the *NORMAL* one. It is possible to stop asking for new operator-observed variables if a predefined faulty situation threshold is surpassed, e.g.,

$$\frac{\sum_{i=0}^{n-1} F_i}{n \cdot F_n} > \alpha \tag{26}$$

where $\alpha = 9$ means that the faulty situation is 90% certain as the normal one.

8.7.4. Neural Network Architecture for Diagnosis and Testing

The neural network architecture for diagnosis using a testing policy comprises two neural networks, each dedicated to the abduction problem solving: one for the diagnosis – DNN (Diagnosis Neural Network), the other for indicating the next observations to be made – TNN (new Test Neural Network), as shown in Figure 8.12.

Both neural network blocks contain feed-forward links for plausibility, between the input and the output neurons; for DNN, between input neurons of the type *OM* (Operator-observed Manifestations), *MM* (permanent Monitored Manifestations), *SY* (SYmptoms detected) and output neurons *F* (Faults); for TNN, between *F* (Faults) and *OM* (Operator-required Manifestations – identical as set with the Operator-observed Manifestations set). Forward links are represented as arrows between input and output layers of neurons, and competition links are represented by horizontal arrows between the output neurons (*F* and *OM*, respectively).

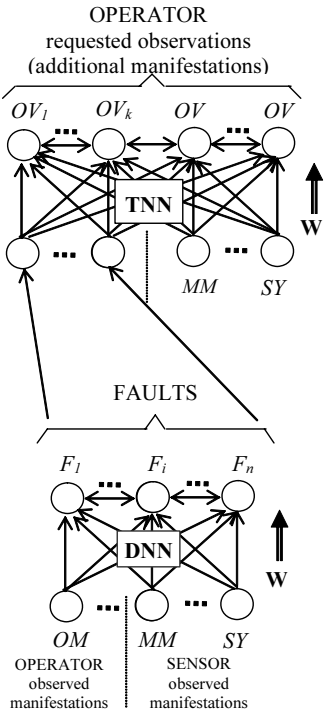

Figure 8.12. Neural network architecture for fault diagnosis by abduction, with additional observations from human operator.

The input of the DNN block consists of permanent observed manifestations *MM* and operator-observed manifestations *OM* – the last ones

passing through the network as long as they are triggered and supplied by the operator. Plausible faults at the output of the DNN block become inputs of the TNN block, along with the current manifestations *MM*, to produce most plausible and relevant observations to be tested by the human operator. The entire set of neuron outputs of the DNN block are passed to the input of the TNN block, while only plausible faults get activated and count for the abduction towards the next test indicated to the operator.

Diagnosis proceeds stepwise. At each step, the observations become first manifestations by "intelligent encoding," then most plausible and relevant faults result by abduction at the output of the DNN block, along with required operator-observed variables indicated at the output of the TNN block. The set of most plausible and relevant faults at each step is a partial result with attached values of *FAULTY* and *NORMAL* situations, as from Eq. 14. The final diagnostic is obtained when the *FAULTY* situation surpasses a given threshold and no operator observations are required. Depending on the number of relevant faults resulting from competition, single or multiple fault diagnosis is in concern.

The "closed world assumption" is satisfied if all situations that may appear during the diagnosis have a result; hence the "no fault" (NORMAL) as well as "unknown fault" (UNKNOWN) neurons appear in the output layer of the DNN neural network block. The processing for plausibility and relevance roughly corresponds to general phases in diagnostic reasoning: "hypotheses generation" and "hypotheses discrimination," respectively.

The neural network is the core of the diagnosis expert system, and it deposits the deep and shallow knowledge of the human diagnostician. The way the diagnosis proceeds also complies with the human diagnostician's way of acting, i.e., it is performed sequentially, applying plausibility and relevance criteria step by step, until the final diagnostic is obtained.

8.8. Case Study on a Hydraulic Installation in a Rolling Mill Plant

The case study is performed on the simple hydraulic installation shown in Figure 8.13. It comprises two hydraulic cylinders (for a carrier and a brake), two control valves, the mineral oil tank, the pump with a pressure valve, and two long pipes. To master the complexity of the installation under diagnosis, the installation was divided into modules: the Hydraulic supply (containing tank, pump and pressure valve) and two driving modules (containing control valve, cylinder, damper – Drossel) – the Hydraulic brake and the Hydraulic conveyor.

8.8.1. Knowledge Elicitation

The information regarding the physical and the behavioural subsystems consists of knowledge pieces presented in Table 8.4. The whole set of disorders considered consists of: faults, the *NORMAL* situation, and the nonconformities at flow, human operator and neighbour systems.

For each component, the numbers of faults are: 2 at the tank, 4 at the pump, 3 at the pressure valve, 2 at the pipes, 3×2 at the control valves, 2 at the damper, 2×2 at the cylinders.

There exist 6 disorders that refer to nonconformities: 2 for the mineral oil (i.e., "too many suspensions" and "foamy oil"), 1 at the environment ("too hot"), 3 for operating errors (*Olu* "no oil in the tank" – see below, "carrier load too heavy," "pump velocity ill tuned"). So, the disorders consist of 23+1 faults (*NORMAL* added), and 6 nonconformities, i.e., $|\boldsymbol{F}| = 30$.

Line 4 in Table 8.4 shows the types of manifestations and the number of data according to the activities in line 2; for example, the number of the fuzzy attributes for the Supply module is (6 variables)·(3 landmarks)·(2 activities) = 36 manifestations of type *lo*, *no*, *hi*.

The measured manifestations refer to $|\boldsymbol{MM}| = 48$ pieces that are variables expressed as single neurons (for the binary variables), or triple neurons (for the continuous variables with *lo*, *no*, *hi* attributes), each neuron with a graded value of truth. The observed variables come from analogical sensors for 2 input/output flow-rates, 3 input/output/damper pressures, 4 temperatures (control valves, pump and tank), from contacts for 4 operator commands (brake on/off, carrier on/off), for 5 positions (of type left/right, open/shut) of the two pistons and of the pressure valve. The 4 durations of the pistons' movements (left/right – for the two cylinders) enter also as measured manifestations.

In the set of the $|\boldsymbol{OM}| = 14$ operator-observed variables, there are 5 of type "noise" (2 for the pump, 3 for the pressure and the control valves), 6 "oil leakage" (all except the damper) and also there are 3 anomalies outside the hydraulic system (brake/carrier mechanical blockage, no pump power).

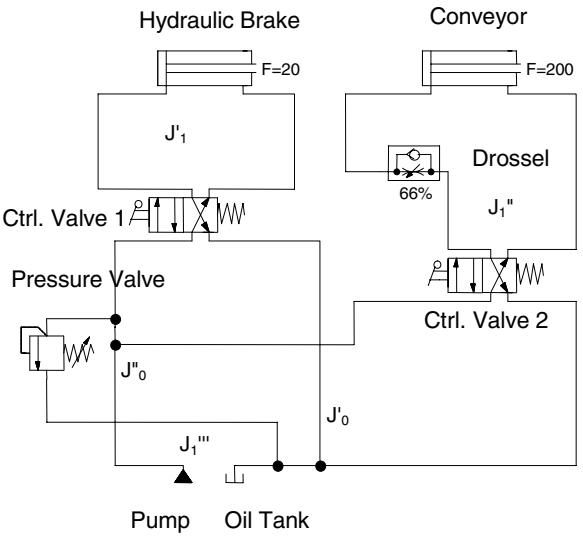

Figure 8.13. Hydraulic installation under elicitation case study.

Running contexts of the target hydraulic installation refer to each discrete position or motion of the pistons in the two cylinders, as well as to the two states of the control valve. So, we find the activities for each of the three modules: 2 activities for the supply module and 4 activities for each driving module. The total number of process phases is 2+4×4=18. Even for such a simple installation, the numbers of process phases is quite large, provided that for each of them the knowledge engineer should develop experiments to assess the specific manifestations and their links to faults, hence plausibility criteria and the DNN block training. Instead, each module's specific behaviour was studied separately when faulty. The simulated faults and the manifestations that had appeared were collected for each separate module, concerning only the 2 activities of the supply module and the 4 activities of each driving module, respectively.

Table 8.4. Inventory of the knowledge pieces involved in the fault diagnosis

Module / Entity	Hydraulic Supply	Hydraulic Brake	Hydraulic Conveyor
1. Components	pump, tank, pressure valve, pipes	control valve, cylinder	control valve, self, cylinder
2. Activities / Faults	2 / 11	4 / 5	4 / 7
3. Sensors (observed variables)	Analogical 6, Digital 7	Analogical 5, Digital 8	Analogical 3, Digital 8
4. Manifestations	Fuzzy 6·3·2, Binary 7·2	Fuzzy 5·3·4, Binary 8·2	Fuzzy 3·3·4, Binary 8·2
5. AnoP, AnoS, AnoT	2, 1, 4	2, 2, 4	2, 2, 4

A total number of 155 (fuzzy and binary) manifestations result, hence 888 manifestations-to-faults and 255 anomalies-to-faults links get established. If faults and manifestations were considered for the entire installation (as in "classic" ANN-based diagnosis – i.e., without modularisation), 32 combinations of activities result, hence (6·3+5·3+3·3)·32=1344 knowledge pieces for manifestations, which require 30·1344=40320 manifestation-to-faults links, and 9600 anomalies-to-faults links.

Using the modularisation in presented approach, just for the simple hydraulic installation, the data volume is (1344+40320)/(888+255)=36 times less for the modularised approach than using a unique ANN block for the entire installation. In the case of a more complex installation, the ratio is much bigger, and embedding deep knowledge in the links between faults and manifestations is quite impossible. While the knowledge acquisition is rather difficult even for the modularised scheme, the CAKE tool comes to assist the human diagnostician in managing the elicitation and the data volumes, also in yielding the data structures for a dedicated diagnosis system.

8.8.2. Neural Blocks for Physical Modules

The diagnosis was meant for three distinct process phases, namely, the one with the control valve open (for faults at the supply module) and those with moving pistons (for the two driving modules). No symptoms were considered on the installation

behaviour. The faults–to-manifestations patterns, used in the training of the DNN and TNN neural network blocks for each module, were partly acquired from human diagnostician practice, partly from experiments.

Again, the modularisation represents an advantage in the implementation of the diagnosis system. So, instead of considering the process phases for the whole installation as the running contexts (which determine the specific faulty behaviour), it is now possible to consider only the activities of modules interconnected in the same bond-graph junction. Furthermore, the neural sites for the abduction problems were easier to build separately for each module.

The structure of the neural network block for the supply module is depicted in Figure 8.14, where:

- Faults are: *Pax* (pump – axis broken), *Pai* (pump – clogged admission), *Pne* (pump – ill joints), *Puz* (pump – worn out), *Tne* (tank – worn-out filter);
- Manifestations are: *P1* (oil pressure at the tank outlet: "too low" *lo*, "too high" *hi*), *D1* (oil flow rate at the tank outlet: "too low" *lo*, "too high" *hi*), *T2* (oil temperature "high" in the tank);
- Manifestations requested from the human operator are: *Z1* (whistling noise at pump), *Z2* (jerky noise at pump), *M1* (oil mud at pump), *M2* (oil mud at tank);
- Nonconformities from flow and from human operator: *Uim* (dirty oil), *Usp* (foaming oil), *Olu* (tank empty), *Otm* (pump angular velocity ill tuned).

As shown in Figure 8.14, there are two monotonic faults (*Pne* and *Puz*), two monotonic operator-observed variables (*Z1* and *Otm*), and two conjunction sites (for *Pax* and *Otm*). The negation sites for operator-observed variables prevent further demand of the variables already requested and supplied.

8.8.3. Plausibility and Relevance

For each running context, the plausibility links between faults and manifestations were set up according to the human diagnostician's deep knowledge, but also systematically linking all manifestations to faults in a module.

The neural network model used for the DNN and TNN blocks is the perceptron; it supports the feed-forward plausibility criteria with modified structure, suited to abduction problem solving (see Figure 8.14).

Plausibility criteria refer to different abduction problems implemented as neural sites and to trained faults-to-manifestations patterns from simulated experiments, on each target module, for fault, "normal" and "unknown" cases.

Competition is added over the set of fault neurons regarding the following relevance grouping:

1. faults at the same component (physical structure scope) – minimum cardinality criterion;
2. faults which are obvious only in specific activities of the respective module (e.g., control valves "blocked parallel" and "blocked crossed" are obvious only when the piston is moving in the hydraulic cylinder);

3. faults in the module provoking leakage and those provoking clogged symptoms for mineral oil flow.

For each functional module i, it corresponds to a neural network module with two blocks, DNN^i and TNN^i. Additional relevance criteria discriminate between the diagnostics at module level, in order to issue the diagnostic at the level of the whole installation. The relevance criteria at installation level are based on symptoms and on Eq. 11; the relevance criterion of minimum cardinality was considered.

The training of the neural network block DNN^1 (associated with module 1 – the oil supply module presented above) is performed using the standard learning algorithm for the perceptron. The NORMAL situation for the entire supply module is trained using normal values (see Figure 8.15): *no* for P1 and D1, and normal states of the other manifestations. The UNKNOWN situation is trained by means of patterns, randomly generated but consistent with those used for plausibility of faults and the normal situation.

As it is difficult to gather all the necessary details for the NN structures for all modules, a CAKE (Computer Aided Knowledge Elicitation) instrument was build and used to describe and automatically generate the DNN and TNN structures at the module level.

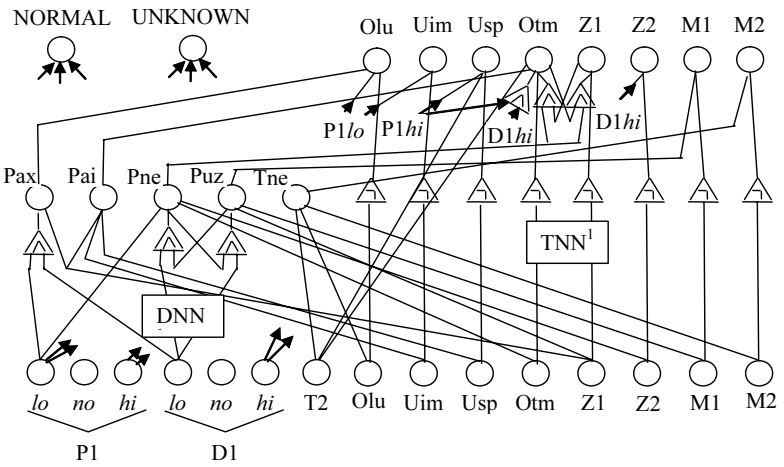

Figure 8.14. The neural network structure for the first (oil supply) module for diagnosis.

8.8.4. Sequential Diagnosis for the Supply Module

Figure 8.16 illustrates the four steps in which the diagnosis regarding the supply module is performed, with respect to a fault that occurred in the pump, namely, *Pai* (short name for the fault "oil tank pipe is clogged"). Each window shows a step during the diagnosis refinement, including the partial diagnostic and the operator-observed variables required from the human operator. For the simulated fault – marked by x – the diagnostic is obtained in four steps, after eliminating other causes

– see in the third window *Olu* (short name for the non-conformity "oil tank empty").

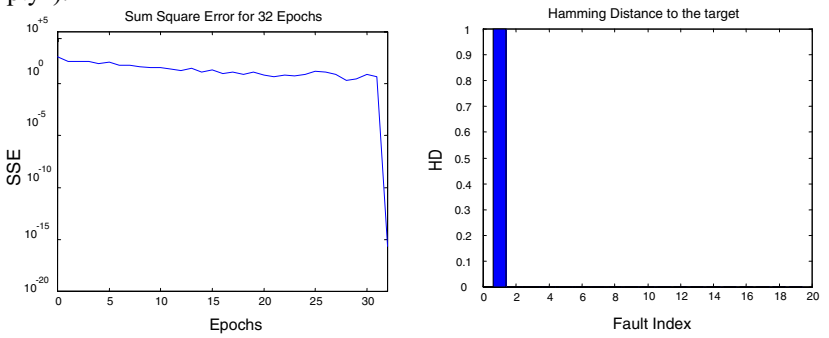

Figure 8.15. Training of the NORMAL situation for the oil supply module, in 32 epochs.

The Y axis indicates the truth value of a specific item from the X axis, which shows discrete knowledge pieces from 0 to 50.

The three sections of the X axis represent: the 31 disorders mentioned above, the 6 nonconformities (in the section "Non-cf"), and the 14 observations needed from the human operator. Faults' truth values, as resulted from the diagnosis, are indicated as bars at the index position of each fault (0 to 30).

The 6 nonconformities and the 14 operator-observed manifestations are also indicated as bars, but their meaning is now a demand to the human operator, i.e., a confirmation required for a possible nonconformity indicated as a bar, or a value required from the operator for the observed variable indicated as a bar, at its specific index on X axis. As a response, the human operator has to indicate if that environment nonconformity is present, or the current value for the operator-observed variable, respectively. In the sections for nonconformities and for operator observations, the height of a bar indicates how stringent is the respective item, so the human operator may choose the highest one(s) for supplying the confirmation or the value.

Additional observations required from the human operator in the current step appear in the *Non-cf.* section and in the *Operator Observations* section on the X axis. The window in each step shows the current diagnostic. Activated observations from the human operator decrease to 0 after the value is supplied.

The diagnostic is strongly dependent on the coverage of faulty behaviours for each module with faults or classes of faults. The data on the behaviour of the hydraulic installation come from simulated experiments. The diagnosis system always produced a diagnostic in a finite number of steps, and the average accuracy of the diagnosis was 96%. Additional observations supplied by the human operator require some steps in the diagnostic refinement that hinders real-time diagnosis.

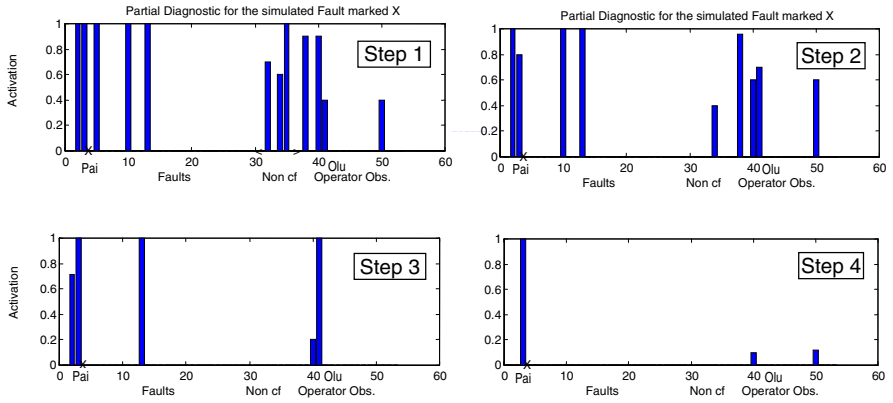

Figure 8.16. Sequential diagnosis in 4 steps for the fault *Pai*, with additional Operator Observations.

8.9. Conclusions

Fault diagnosis of complex systems involves deep and shallow knowledge of human diagnosticians, since diagnosis in the reallife deals with incomplete, imprecise and uncertain knowledge on the behaviour of target systems. The aim of the chapter is to describe a diagnostic system that emulates the human diagnostician's way of acting, in order to build dedicated diagnosis systems for concrete target systems. The automated fault diagnosis is based on computational intelligence models: fuzzy and possibilistic logic, artificial neural networks.

The chapter focuses on the fault diagnosis of artefacts often met in industry (and not only), that executes more functions at the same time based on conductive flows of matter and energy, i.e., multifunctional conductive flow systems (MCFSs). The proposed MCFS abstraction is close to the human diagnostician's way of conceiving entities and relations on physical, functional and behavioural structures.

Diagnosis reasoning is intrinsically abductive reasoning. The chapter presents the abduction by plausibility and relevance, in a connectionist approach. Plausibility criteria become feed-forward links from manifestations to faults – as from the shallow knowledge acquired in practice or experiments. Relevance criteria become competition between the elements of various groups of causes (be they faults or other kind of disorders), put together according to the deep knowledge on physical, functional and behavioural structures of the target system.

In order to solve all types of abduction problems (according to Bylander *et al.*, 1991), specific architectural features are added to the neural network. The features refer to plausibility criteria and affect the feed-forward links between manifestation and fault neurons, also between fault neurons. This way, the abduction problem solving is straightforward and easier implemented in various neural network types than other approaches (e.g., Ayeb *et al.*, 1998).

Deep knowledge refers to physical and functional structures, as means for achieving the ends of the target system. Also, deep knowledge refers to the sets of faults, manifestations and symptoms along with some behavioural hints regarding primary and secondary effects useful for locating faults. Shallow knowledge refers to (unexplained) links of faults to manifestations or to symptoms, from the human diagnostician's practice or experiments.

The embedding of the deep and shallow knowledge requires appropriate representations of physical, functional and behavioural concepts, observing the discrete and qualitative nature of human knowledge. In this respect, means-end and qualitative modelling approaches are adapted to obtain a unified representation of various behavioural entities. The faults' effects propagation is modelled using four orthogonal transport anomalies related to the bond-graph model of components and bond-graph junctions for modules for the entire target system.

The concepts and relations involved in human-like diagnosis get appropriate representations by computational intelligence paradigms. All concepts and relations enter the connectionist models of the abduction problem solving, and their representation is also meant for the systematic knowledge acquisition on concrete target systems. All knowledge pieces involved in fault diagnosis enter appropriate elicitation models addressing human diagnosticians' way of acting, and lead to structures useful for the computational model of the diagnosis system.

The decision on the next best test, aiming the diagnostic refining, is also seen as an abduction problem, and it is solved based on plausibility and relevance criteria in the connectionist implementation. The diagnosis on the whole is performed as a sequential application of plausibility and relevance criteria, applied incrementally, and completed with new tests until the final diagnostic is found.

Fault diagnosis of real systems involves a great amount of data. Therefore, knowledge acquisition, knowledge representation and data management tasks require appropriate tools to assist human diagnosticians in building the diagnosis system. The Computer Aided Knowledge Elicitation (CAKE) software tool assists the human diagnostician, or even the human operator, in the design and generation of the dedicated diagnosis system for the concrete target system envisaged. So, the CAKE tool replaces the knowledge engineer and the software designer. Moreover, specific knowledge on the concrete target system is embedded in the diagnostic expert system, exploiting the human diagnostician's practice and knowledge on the running conditions of the target real system.

The case study on a hydraulic installation of a rolling mill plant gives examples on the knowledge elicitation process and on the diagnostic expert system building and running.

References

1. Ariton V, Ariton D (2000) A General Approach for Diagnostic Problems Solving by Abduction. In: Proceedings of IFAC-SAFEPROCESS, Budapest, Hungary, pp. 446-451
2. Ariton V (2001) Abstraction Levels for the Fault Isolation in Multifunctional Conductive Flow Systems. In: Proceedings of the 9th IFAC/IFORS/IMACS/IFIP/

Symposium on Large Scale Systems-Theory and Applications, Bucharest, Romania, pp. 386-391
3. Ariton V, Baciu C (2002) Knowledge Elicitation and Case Tool for Fault Diagnosis in Multifunctional Conductive Flow Systems. In: Proceedings of SCI2002 - 6th World Multiconference on Systemics, Cybernetics and Informatics, Orlando, Florida, USA, July 14-18, vol. XXII, pp. 345-350
4. Ariton V (2003) Deep and shallow knowledge in fault diagnosis. In: Palade V, Howlett RJ, Lakhmi J (eds) Knowledge-Based Intelligent Information and Engineering Systems, 7th International Conference, KES 2003, Oxford, UK, September 3-5, Proceedings, Springer-Verlag, pp.748-755
5. Ariton V, Palade V (2004) Human-like fault diagnosis using a neural network implementation of plausibility and relevance. Neural Computing & Applications (Springer-Verlag) 14(2):149-165
6. Ayeb B, Wang S, Ge J (1998) A Unified Model for Abduction-Based Reasoning. IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans 28(4):408-424
7. Bocaniala CD, Sa da Costa J, Palade V (2004) A Novel Fuzzy Classification Solution for Fault Diagnosis. International Journal of Fuzzy and Intelligent Systems 15(3-4):195-206
8. Bocaniala CD, Sa da Costa J, Palade V (2005) Fuzzy-based refinement of the fault diagnosis task in industrial devices. International Journal of Intelligent Manufacturing 16(6): 599-614
9. Bylander T, Allemang D, Tanner MC, Josephson JR (1991) The Computational Complexity of Abduction. Artificial Intelligence 49:25-60
10. Cherkassky V, Lari-Najafi H (1992) Data Representation for Diagnostic Neural Networks. IEEE Expert 7(5):43-53
11. Cellier FE (1995) Modelling from Physical Principles. In: Levine WS (ed) The Control Handbook. CRC Press, Boca Raton, pp.98-108
12. Calado JMF, Korbicz J, Patan K, Patton RJ, Sa da Costa MG (2001) Soft Computing Approaches to Fault Diagnosis for Dynamic Systems. European Journal of Control 7(2-3):248-286
13. Cordier MO, Dague P, Dumas M, Lévy F, Motmain J, Staroswiecki M, Travé-Massuyès L (2000) AI and Automatic Control Approaches of Model-Based Diagnosis: Links and Underlying Hypotheses. In: Proceedings of IFAC-SAFEPROCESS, Budapest, Hungary, pp. 274-279
14. Davis R (1993) Retrospective on "Diagnostic Reasoning Based on Structure and Behaviour". Artificial Intelligence 59:149-157
15. Dubois D, Prade H (1998) Possibility Theory: Qualitative and Quantitative Aspects. In: Gabbay DM, Smets P (eds) Handbook of Defeasible Reasoning and Uncertainty Management Systems, vol 1, pp. 120-159, Kluwer Academic Publishers, New York
16. de Kleer J, Kurien J (2003) Fundamentals of Model-Based Diagnosis. Proceedings of IFAC-SAFEPROCESS, Washington, USA, pp. 1-12
17. Konolige K (1992) Abduction Versus Closure in Causal Theories. Artificial Intelligence 53:255-272
18. Kruse R, Gebhardt J, Klawon F (1994) Foundations of Fuzzy Systems. John Wiley & Sons, New York

19. Kuipers BJ (1994) Qualitative Reasoning: Modelling and Simulation with Incomplete Knowledge. MIT Press, Cambridge, MA, USA
20. Larsson JE (1992) Knowledge-based methods for control systems. PhD Thesis, Lund
21. Mosterman PJ, Biswas G (2002) A Hybrid Modelling and Simulation Methodology for Dynamic Physical Systems. In: SIMULATION: Transactions of the Society for Modeling and Simulation International, 78(1):5-17
22. Mosterman PJ, Kapadia R, Biswas G (1995) Using bond graphs for diagnosis of dynamical physical systems. In: Proceedings of the Sixth International Conference on Principles of Diagnosis, pp. 81-85
23. O'Brien T (1970) Reliability of Multifunction Structures. New York University
24. Okuda K, Miyasaka N (1991) Model based intelligent monitoring and real time diagnosis. In: Isermann R (ed) Preprints of SAFEPROCESS '91
25. Opdahl AL, Sindre G (1994) A taxonomy for real-world modelling concepts. Information Systems 19(3): 229-241
26. Palade V, Patton RJ, Uppal FJ, Quevedo J, Daley S (2002) Fault diagnosis of an industrial gas turbine using neuro-fuzzy methods. In: Proceedings of the 15th IFAC World Congress, 21–26 July, Barcelona, pp. 2477–2482
27. Patton RJ, Frank PM, Clark RN (2000) Issues of Fault Diagnosis for Dynamic Systems. Springer-Verlag, London
28. Peng Y, Reggia J (1990) Abductive Inference Models for Diagnostic Problem Solving. Springer-Verlag, London
29. Schurz G (2002) Models of Abductive Reasoning. TPD Preprints Annual 2002, no.1, University of Düsseldorf, Germany
30. Shibata B, Tateno S, Tsuge Y, Matsuyama H (1991) Fault diagnosis of the chemical process utilizing signed directed graph. In: Isermann R (ed) Preprints of Fault Detection Supervision and Safety for Technical Processes - SAFEPROCESS '91, pp.381-386
31. Struss P (1997) Model-based and qualitative reasoning: An introduction. Annals of Mathematics and Artificial Intelligence 19: 355-381
32. Turksen IB (1996) Non-Specificity and Interval-Values Fuzzy Sets. Fuzzy Sets and Systems 80:87-100
33. Uppal FJ, Patton RJ, Palade V (2002) Neuro-Fuzzy Based Fault Diagnosis Applied to an Electro-Pneumatic Valve. In: Proceedings of the 15th IFAC World Congress, 21–26 July, Barcelona, Spain, pp. 2483-2488