Transactions on Information and Communications Technologies vol 1, © 1993 WIT Press, www.witpress.com, ISSN 1743-3517

Qualitative model based diagnosis of a continuous process

R. Leitch, Q. Shen, G. Coghill, M. Chantler &
A. Slater
Intelligent Automation Laboratory, Department of Computing and Electrical Engineering, Heriot-Watt University, Edinburgh EH14 4AS, UK

ABSTRACT

We show how qualitative simulation can be used to monitor and diagnose faults in the behaviour of dynamic physical systems. Our system synchronously tracks the evolution of the real system and iteratively searches for faults based on a characterisation of the modelling space. A particular innovation is the use of priority measures to rank the multiple behaviours predicted by the simulation and to use these in establishing the most likely fault. The operation of the system is illustrated by application to an experimental process rig that consists of continuous and dynamic thermal and flow processes.

INTRODUCTION

The benefits of using Model Based Reasoning for diagnosing faults on physical systems are now well established. However very little consideration has been given to the types of model that should be used in a given situation. The majority of work on Model Based Diagnosis has been investigated using static real-valued models and consistency maintenance techniques to relate the modelling primitives (constraints) to conflicts obtained between the physical observations and the predicted state of the model. In many practical cases, however, predicting the dynamic behaviour of the system is crucial to establishing the cause of the fault as the incipient fault may only be observable during the transient behaviour between equilibria. Detecting and diagnosing such faults often allows corrective or repair action to be taken before major damage occurs or perhaps even preventing the development of a hazardous situation. Further, in many practical situations real-valued models are not readily available, either because of a lack of the fundamental operation of the system or because extensive modelling effort is not justified. In these circumstances qualitative modelling and simulation techniques can often be more applicable. Indeed qualitative models result in diagnostic systems that are inherently more robust than numerically based systems. Strangely, the development of Model Based Diagnosis using qualitative dynamic models has

only been pursued during the last few years. In this paper we overview our approach to Model Based Diagnosis based on our Fuzzy Qualitative Simulation System and present experimental results in applying the approach to a laboratory scale process rig.

DYNAMO: USING QUALITATIVE DYNAMIC MODELS IN DIAGNOSIS

To determine why a physical system has not worked correctly compared with its design intention, it is useful to know how it was supposed to work in the first place. This simple observation underlies recent development of model-based diagnostic techniques [1], within which observations from the physical system being diagnosed indicate what that system is actually doing whilst predictions made from an explicit structural model of the system indicate what that system is intended to do. Such an approach to diagnosis is, therefore, crucially dependent upon the use of the explicit structural models of the physical systems. In order to avoid potential great complexity in system modelling and diagnostic reasoning qualitative models are usually utilised for behavioural prediction or simulation. Viewing this, within this section, we first present a brief overview of the Fuzzy Qualitative Simulation algorithm, FuSim [7], with a method for prioritising possibly multiple qualitative behaviours predicted by FuSim to increase the efficiency in performing the diagnostic reasoning task, and then describe a model-based diagnostic system that uses FuSim for behavioural predictions.

Fuzzy Qualitative Simulation and Behaviour Prioritisation

Clearly, the choice of representation of physical quantities plays a critical role in qualitative simulation [8, 10]. All qualitative modelling techniques describe quantities with a small set of symbols, called qualitative values, which are abstracted from the underlying field that the variables of a physical system take values from. In FuSim a qualitative value of a system variable is a fuzzy number chosen from a subset of normal convex fuzzy numbers [7]. This subset, called the fuzzy quantity space, is generated by an arbitrary but finite discretisation of the underlying numeric range of the variable. For computational efficiency, such qualitative values are characterised by the 4-tuple parametric representation of their membership functions within the implementation of the algorithm [7]. A 4-tuple fuzzy qualitative value, [a, b, α, β], as shown in figure 1, is defined as

$$\mu_A(x) = \begin{cases} 0, & x < a - \alpha; \\ \alpha^{-1}(x - a + \alpha), & x \in [a - \alpha, \ a]; \\ 1, & x \in [a, \ b]; \\ \beta^{-1}(b + \beta - x), & x \in [b, \ b + \beta]; \\ 0, & x > b + \beta; \end{cases}$$

FuSim adopts a constraint-centred ontology for system-modelling. A model is derived from an underlying differential equation representation or from direct application of first order energy storage mechanisms. The sets of possible values which system variables can take are thus restricted by either algebraic, derivative or function relational constraints amongst the variables. More specifically, the algebraic operations performed within a fuzzy quantity space are simple arithmetic operations allowed within the set of fuzzy numbers. A derivative constraint simply reflects that the qualitative value of a variable's magnitude must be the same as that of another variable's rate of change. Functional relationships within FuSim are represented by fuzzy relations [7], thereby allowing imprecise and/or partial numerical information on functional dependencies between variables to be exploited if indeed such information is available.

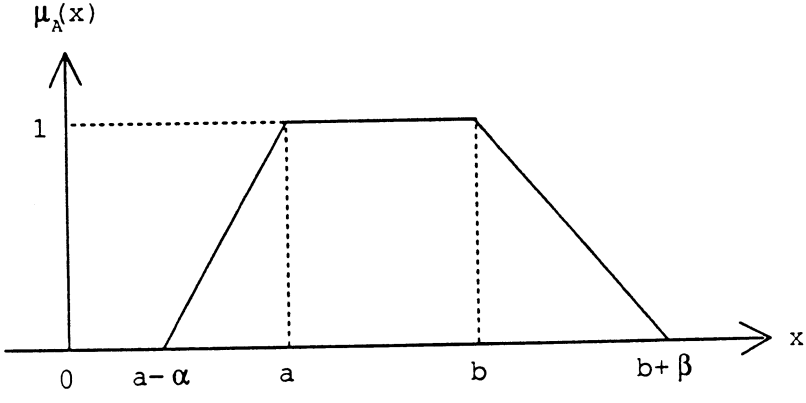

Fig. 1. A 4-tuple Fuzzy Qualitative Value

Based on such a qualitative representation of values and constraints, FuSim takes as input a set of system variables, a set of constraints relating the variables (as the system model), and a set of initial values for the variables, and produces a tree of states with each path representing a possible behaviour of the system as output. In fact, FuSim first generates a set of transitions from one qualitative state description to its possible successor states by exploiting the continuity of the system variables. Further restrictions on these possible successor states are then imposed by checking for consistency with the definition of the constraints and the consistency between constraints which share an argument -- called constraint filtering, and information on the rates of change of the system variables held as part of the fuzzy qualitative state -- called temporal filtering. In addition, other knowledge about the system such as energy conservation may be used to produce so-called global filters [10]. Importantly, associated with each

sequence of states, i.e., each path of the output tree, FuSim also generates a sequence of temporal intervals to indicate how long the system may persist within a particular state. This is a distinct advantage of FuSim over other qualitative simulation techniques. Especially, when used for diagnosing dynamic systems, FuSim is thus able to show which particular portion of the predicted behaviour should be matched by an observation at a particular time point or during a particular time interval [6].

It should be noticed that, in present approaches to qualitative simulation, except for FuSim, the worst case solution is always assumed. That is, all theoretically possible successor states of a present state are maintained and, in fact, propagated with equal status. This, of course, leads to the generation of spurious behaviours that discredits qualitative simulation in the eyes of application engineers, in that, such multiple behaviour predictions are at variance with the uniqueness of the behaviour of the physical world. However, attaching an uncertainty measurement to those potential successor states, and hence a commitment to an associated behaviour, allows prioritised generation of behaviours or an efficient mechanism for their use within applications e.g. diagnosis. This allows a progressive approach to reasoning that first generates or utilises the 'most likely' behaviour and only progresses to other less committed behaviours if the behaviours considered fail to meet the purpose of the application system. Thus, ultimately, the soundness of the algorithm is still retained, however, a significant improvement in efficiency is possible by exploring the higher priority behaviours first.

An algorithm for prioritising behaviours in qualitative simulation has been developed by the authors [5]. In summary, the determination of the state priority of system variables can be carried out as follows, where, informally, a constrained variable is the single variable on the one side of a given constraint while the constraining variables are those appearing on the other side, and the metric $D(s,\hat{s})$ measures the distance between two qualitative states s and $\hat{s}$ [5, 7]:

1) For each variable x find all constraints relating to it.
2) When x is the constrained variable within a constraint, find the N propagated qualitative states of the x, $\{\hat{s}_i \,|i = 1,2,\ldots,N\}$, by operating on the values of constraining variables; where N is the number of the tuples of the predicted possible states associated with this constraint; find the distances between s_i and $\hat{s}_i$ respectively:

$$\{D(s_i,\hat{s}_i)\,|i = 1,2,\ldots,N\};$$

then, attach each resulting distance measure to its corresponding predicted possible state of x and go to 4).

3) When x is a constraining variable in a constraint, for each of its predicted states (within the tuples of the possible states associated with that constraint), find the propagated state of the constrained variable (but not the x) within the constraint and then the distance between this propagated state and its respective predicted state; attach so resulting distance to the predicted possible state of the constraining variable x.

4) Redo 2) or 3) until all the predicted states associated with all the constraints that are related to x have been attached with distance measures.

5) Prioritise the states of x such that $\rho(s_i) = j, i = 1,2,\ldots,M,\ M \leq N,$ $0 \leq j \leq N$, if

$$D_i = \min\{\{D_k \,|k = 1,2,\ldots,M\} - \{D_k \,|k < j\}\},$$

where $D_i, i = 1,2,\ldots,M$, is the distance label attached with state s_i, M is the total number of possible states of the variable x associated with all the constraints, and r(.) indicates the priority level of the state s_i.

Iterative Search Based Diagnostic System

This sub-section describes a particular implementation of the model-based diagnostic systems following the iterative search based approach [6]. We present the diagnostic system by explaining the functionalities of different modules utilised within the system and indicating the methods to realise these modules. As shown in figure 2, at the most schematic level, this diagnostic system consists of four fundamental sub-systems: a behaviour predictor that predicts the expected system behaviour; a discrepancy generator that generates the discrepancies between predictions and observations; a candidate proposer that produces fault candidates based on the discrepancies; and a diagnostic strategist that controls and co-ordinates the complete diagnostic process. Each of these sub-systems are explained in order below.

Behaviour Predictor The central idea of the model-based approach to diagnosis is the use of an explicit model of a physical system's structure. Based upon this model, the predictor accomplishes its task by predicting the expected behaviour of the system via a behaviour simulation or constraint propagation algorithm, regarding the dynamic or static property of the physical system being diagnosed respectively [4].

The reliable diagnosis of faults on continuous dynamic systems requires that the model of the system synchronously tracks the evolution of the observations from the physical system. To achieve this the behaviour simulator used within

the diagnostic system must produce estimates of the durations associated with the respective states and also minimise the spurious behaviours generated. From the previous review of FuSim, it can be seen that FuSim has several features useful for synchronising the model evolution with the observations: 1) the temporal duration of the qualitative states is given; 2) stronger functional constraints can be utilised; 3) a reduction in the spurious behaviours is possible as a product of 1) and 2); 4) qualitative behaviours generated are prioritised with respect to their possibility to reflect the underlying real behaviour; and 5) fuzzy sets allow the subjective element in system modelling to be incorporated and reasoned with in a formal way. Further, a distance metric for detecting and generating discrepancies between two fuzzy qualitative values is ready for use [6, 7]. Having taken a notice of these, FuSim is thus utilised as the behaviour predictor within the iterative search based diagnostic system presented herein.

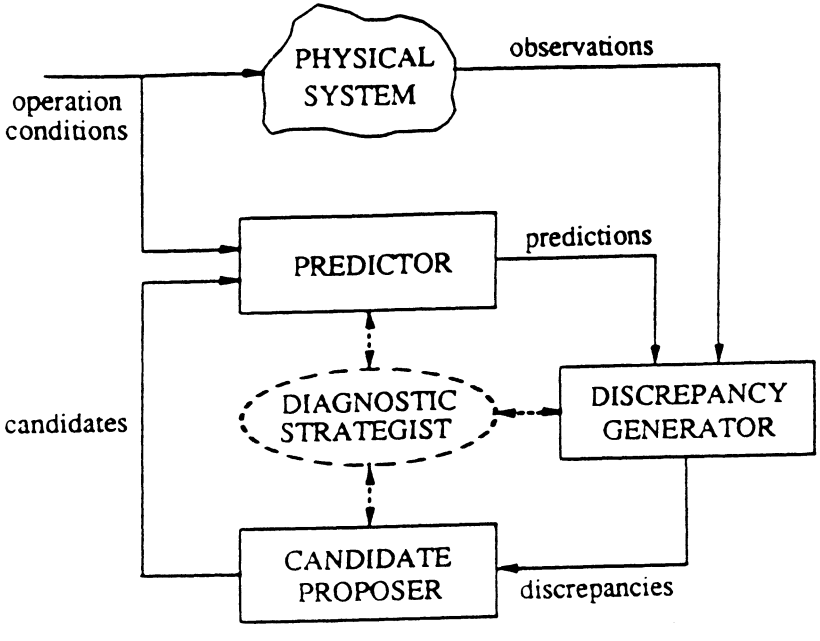

Fig. 2. Iterative Search Based Diagnostic System

The outline of the FuSim algorithm has been given in the preceding sub-section while details about it can be found in [7]. We shall, therefore, not further re-describe how the predictor generates a system's behaviour.

Discrepancy Generator Diagnosing a physical system crucially relies upon the discrepancy detection between the observed and predicted behaviours of the system. The task of the discrepancy generator is to decide on whether or not

there is indeed an inconsistency between a prediction and an observation and, if so, to generate the discrepancy. Since both the interpreted predicted and observed values are represented by the 4-tuple fuzzy numbers, the prediction and the observation can generally be denoted as $P = [p_1, p_2, p_3, p_4]$ and $O = [o, o, 0, 0]$, respectively. The discrepancy generation is then realised by applying a set of rules to P and O, deduced by the use of the a distance that satisfies the three basic properties of a discrepancy 'metric' [6]. Informally, the a distance is the shortest distance between two underlying real numbers with their memberships equal to or greater than the a and each belonging to a different fuzzy number. The discrepancy generation rules are listed in the following, where a reflects the desired degree of matching, which may be given by default or independently assigned by the diagnostic strategist on-line:

(1) If O intersects with P, use the sub-rules below:

(1-1) If $o \in [p_1, p_2]$ (see figure (3.a)), there is no inconsistency between P and O, P and O are matched;

(1-2) If $o \in (p_2, p_2 + p_4(1-\alpha)]$ (see figure (3.b)), there exists a minor inconsistency but P and O are still said to be matched because the a distance between the two equals 0;

(1-3) If $o \in (p_2 + p_4(1-\alpha), p_2 + p_4]$ (see figure (3.c)), there is an inconsistency between P and O and the a distance, $o - p_2 - p_4(1-\alpha)$, is regarded as a type I discrepancy.

(2) If O does not intersect with P (see figure (3.d)), there is an inconsistency between P and O and the a distance, $o - p_2 - p_4(1-\alpha)$, is regarded as a type II discrepancy.

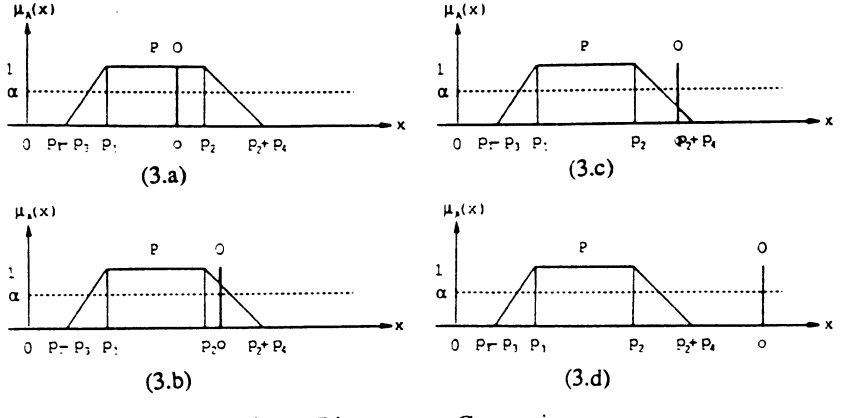

Fig. 3. Discrepancy Generation

Notice that, symbolically, both rule (1-3) and rule (2) have the same a distance in representation. However, they have rather different implications in

that, within rule (1-3), the a distance is detected under the condition that O intersects with P, whilst this is not the case with rule (2). Also, rules (1-2), (1-3), and (2) give the method to detect and generate discrepancies when o is greater than p_2. For the other situation, where o is smaller than p_1 the corresponding discrepancy generation rules are obtained by symmetry.

It is clear that the discrepancy generator accomplishes its task by comparing observations with predictions. For dynamic systems, of course, a technique which is able to compare predictions against observations over time is required. In fact, for coherent detection the comparison between the observed behaviour and the predicted behaviour must be made at the same system state, i.e., at the same absolute time (point or duration). This is because that, the system model used must operate synchronously with the natural evolution of the physical system. This brings a problem to the use of traditional qualitative simulators (e.g. QSim [3]) as the behaviour predictors because of the need to guide and control the comparison between the observations and the predictions. Temporal information becomes essential to the maintenance of synchronous behaviour. Without this it is impossible, without resorting to heuristics, to control the evolution of the models, including spurious behaviours and fault models [6]. It is in this respect that FuSim provides important advantages. This is reflected from the fact that FuSim produces a temporal duration sequence associated with the state sequence and the possible behaviours generated are prioritised, thereby enabling an effective and efficient discrepancy detection method to be developed.

The rule used to guide and control the discrepancy detection is directly deduced from knowledge of both the temporal durations and the priorities of the qualitative states and can be stated as follows:

> Treat the current observation, $OBS(t_0)$, as the initial state of the model and the time when the observation is made as the initial temporal point. From the next observation $OBS(t_1)$, generate the simulated behaviour from $OBS(t_0)$ until the temporal upper bound of a qualitative state meets or covers t_1. If there is more than one such state are generated prioritise these states. Next, compare the highest prioritised state with $OBS(t_1)$. Redo this process if they are matched; else, compare the state with the second highest priority with $OBS(t_1)$ and so on. If all such states do not match $OBS(t_1)$ discard the current model.

Notice that, in the above rule, discarding a model simply implies that this model is inconsistent with the current working condition of the physical system.

Transactions on Information and Communications Technologies vol 1, © 1993 WIT Press, www.witpress.com, ISSN 1743-3517

Illustratively, the way to track a model, or to compare predictions against observations, can be depicted by figure 4 and explained in the following.

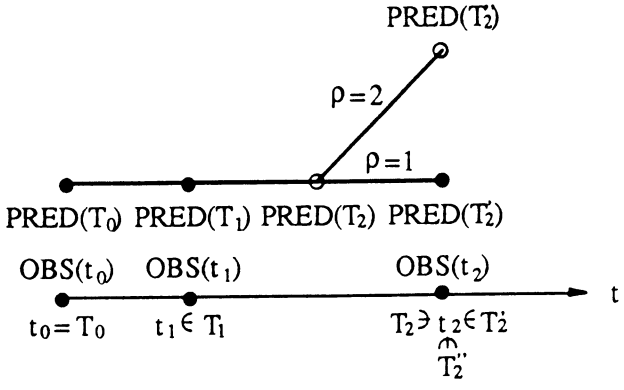

Fig. 4. Prioritised Behaviour Tracking

Without losing generality, suppose that the first two observations from the physical system are $OBS(t_0)$ and $OBS(t_1)$, FuSim then uses $OBS(t_0)$ as the initial state, $PRED(T_0)$, and starts running the simulation of the model. If it generates the possible next state, say, $PRED(T_1)$, which matches $OBS(t_1)$ under the condition that $t_1 \in T_1$, no discrepancy is detected and hence the discrepancy generator waits for another observation $OBS(t_2)$ to be available. After this, FuSim continues predicting the successor state $PRED(T_2)$, however, this state's temporal information indicates that for any $t \in T_2, t < t_2$. Thus, FuSim keeps making further predictions from $PRED(T_2)$ and results in both $PRED(T_2')$ and $PRED(T_2'')$ as possible next states. Using the technique for state prioritisation it follows that $\rho(PRED(T_2')) = 1$ and $\rho(PRED(T_2'')) = 2$. Now that $PRED(T_2')$ matches $OBS(t_2)$ and $t_2 \in T_2'$, the model being used remains as validated and the process of prediction and comparison recurs, starting from the matched prediction $PRED(T_2')$. Importantly, if there is no priority information on these two predicted states, both of them have to be compared with the observation $OBS(t_2)$ and hence the discrepancy generation process would become less efficient than it is. Actually, if $PRED(T_2'')$ is checked first, though $t_2 \in T_2''$, $PRED(T_2'')$ does not match with $OBS(t_2)$ and no further predictions will be made following the branch beginning at the $PRED(T_2'')$ whatever later observations are obtained. This comparison has, therefore, been made

unnecessarily. In the event that $PRED(T_2')$ also conflicts with $OBS(t_2)$, the current model will then be discarded.

Clearly, due to the temporal information in FuSim only a small number of predictions starting from certain current state(s) are generated and, also, the discrepancy generator makes the comparison between a prediction and an observation only when the prediction is the last one generated with respect to the latest sampling time.

The present discrepancy generation mechanism tracks a model without distinguishing if it is the normal-behaviour model, a modified model against particular model variation direction, or a known fault model. Therefore, if the model currently being tracked is the normal-behaviour one, an on-line discrepancy generator built in this way actually performs system-monitoring with an identical structure. Once a discrepancy is detected the diagnostic process is activated and the model of normal behaviour is adjusted along potential model variation directions. The monitoring task then becomes a fault identification task. How to hypothesise the possible faults is, of course, accomplished by the candidate proposer discussed in the following.

Candidate Proposer The final diagnoses, namely the faults returned by the diagnostic system, come from the candidate proposer under the condition that the predictions from the behaviour predictor match the observations. The task of fault candidate generation is achieved by exploiting discrepancies through an iterative search process performed within defined model sub-spaces characterised by particular model variation directions or modelling dimensions [9]. This process can be described as the following: Start with system-monitoring based on the normal-behaviour model of the physical system, when certain discrepancies are generated (by the discrepancy generation rule (1-3) or (2)) these discrepancies initiate the modifications and/or adjustments of the model within different sub-spaces of model variations characterised by the underlying modelling dimensions. Each such modified (or adjusted) model will be treated as a candidate to be further evaluated as illustrated in the model-tracking process. Which model dimensions to search is, of course, under the control of the diagnostic strategist. From this viewpoint, the candidate proposer maps discrepancies onto particular fault hypotheses. However, as can be seen later, the fault hypotheses herein do not necessarily imply those pre-defined off-line but usually obtained from the on-line diagnostic process through modifying the normal-behaviour model of the physical system.

In order to perform the mapping between the discrepancies and the candidates the proposer stores the modelling dimensions to be used to build the system model and a set of rules, called the search rules, to guide the search against these dimensions. If fault models are known *a priori* they can be used to

initialise the search at the corresponding point in the model space. This focuses the search at the most likely point and iteration starts from there, greatly enhancing the efficiency of the search process. However, fault models are not essential and search can be initialised anywhere in the model space. The search rules currently used reflect diagnostic heuristics and can be stated as follows, where modifying a model means to vary that model with respect to a particular dimension chosen by the diagnostic strategist:

1) Start with normal-behaviour model: if the discrepancy generated is type I modify this model; and if the discrepancy generated is type II choose the most likely fault model as the modified model, i.e., the candidate unless there do not exist any known faults. In case where no fault models exist, the type II discrepancy also initiates the modification of the normal-behaviour model.

2) Start with modified model: if the discrepancy resulting from the modified model is less than that generated previously, namely the a distance is shorter than the previous one, maintain this model as the candidate and substitute the previous discrepancy for the current one; if the discrepancy vanishes, i.e., the predictions obtained from this modified model match the observations, return the model as a final diagnosis; if the discrepancy is unchanged further modify this model; and if the discrepancy becomes larger than before discard this model as a possible candidate.

It is important, however, to notice that for simplicity the above rules are presented without indicating which direction of the modelling dimension is to be modified. In practice, without knowing fault models, the modifications of the normal-behaviour model will be performed against one direction first and, if no diagnosis is returned from this direction the search process will be redone against the inverse direction, starting again with the normal-behaviour model. If the first modified model is that of a known fault, further modifications of it will be executed in both directions by viewing the fault model as an initial 'normal-behaviour' model, thereby searching in the same way as the situation where no fault models exist.

It is clear that the candidate proposer generates candidates by making use of both the type and the size of the discrepancies to guide the identification of faults. The candidate generation process is, in fact, a systematic search process performed within the space of possible model variations and diagnosis is refined over time through the iterative search process. This avoids the need for explicit fault models and even allows the diagnostic process to start with an estimated system model of normal behaviour. However, knowledge about faults can be effectively utilised to restrict the search space as reflected in the search rules if indeed such knowledge is available, a technique we term fault-guided search.

This reflects an objective that, in real applications, fault models (if any) are constructed and/or provided for more severe or more obvious possible faults. If there is a type II discrepancy there is a heavy distortion of the physical system from the designed behaviour and, thus, the most likely fault model should be tested first. An interesting but extreme case resulting from the proposed diagnostic system is that, if each model sub-space consists of only normal-behaviour model and fault models the candidate proposer will degenerate into a decision tree or a look-up table as used in [2].

Diagnostic Strategist The diagnostic strategist controls the entire diagnostic process based on the on-line evaluation of the diagnostic performance, in connection with other knowledge resources such as domain-dependent heuristics and resource limit. With respect to each of the other three sub-systems within the current iterative search based diagnostic system, the functionality of the strategist can be summarised as follows:

1) Decide on the levels of abstraction and commitment of the quantity space for system modelling and data interpretation and determine which model to be used as the system model for normal behaviour simulation;

2) Select the a value for discrepancy detection and generation;

3) Choose appropriate modelling dimensions to search for candidate generation.

Although important, it can be seen from the fault candidate generation process that, the diagnostic strategist does not affect the central mechanism of iterative search. In essence, the diagnostic reasoning process merely involves the behaviour predictor, the discrepancy generator, and the candidate proposer. We shall, therefore, leave the concrete realisation of the strategist as an important future work.

APPLICATION TO A CONTINUOUS PROCESS

A realistic application, based on a laboratory scale "Process Rig System", as depicted in figure 5, is given to demonstrate how a physical system is modelled within FuSim, and to show how the iterative search based diagnostic system using FuSim determines faults. The process rig is an experimental system in which the behaviour of a heat input and extraction process from flowing water can be examined. It consists of several components: a tank, a sump, a pump, a radiator and a heater.

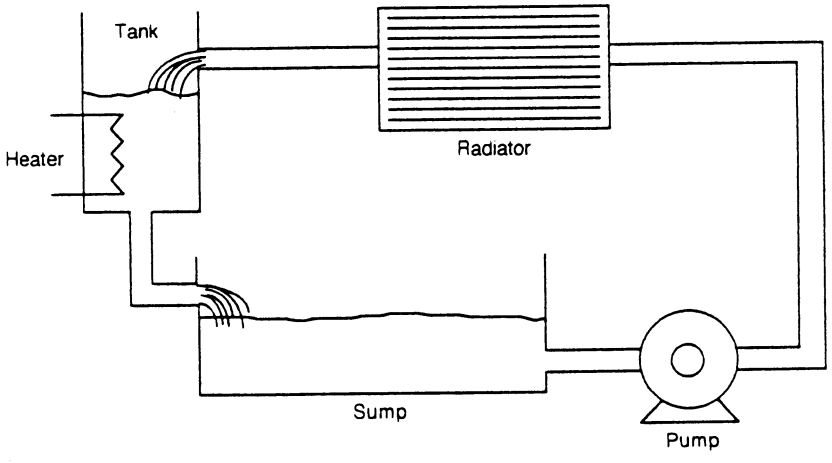

Fig. 5. The Process Rig

System Modelling

The physical process may be modelled as a flow process and a thermal process. In the present example only the mass flow loop (the block diagram of which is shown in figure 6) will be discussed. The model of such a system consists of seven variables: the flow of fluid into the tank, *i*, the height of fluid in the tank, *h*, the volume of fluid in the tank, *v1*, the flow of fluid from the tank, *o*, the rate of change of the height of fluid in the tank, *v1'*, the volume of fluid in the sump, *v2*, and the total volume in the system, including the radiator, v_T. For simplicity, the following set of 4-tuple parametric fuzzy numbers is chosen to form the fuzzy quantity space:

$$Q_F = \{ [-1, -1, 0, 0.1], [-0.9, -0.75, 0.05, 0.15], [-0.6, -0.4, 0.1, 0.1] \\ [-0.25, -0.15, 0.1, 0.15], [0, 0, 0, 0], [-.15, 0.25, 0.15, 0.1], \\ [0.4, 0.6, 0.1, 0.1], [0.75, 0.9, 0.15, 0.05], [1, 1, 0.1 0] \},$$

and is abbreviated to $Q_F = \{ -t, -l, -m, -s, 0, s, m, l, t \}$; with each value corresponding to a perceived meaning, for instance, *-t* denoting *negative top* and *s* indicating *positive small.*

Suppose that the normal model assumes free flow of fluid from the tank. Thus, the physical system can be characterised by the following fuzzy qualitative model:

$$h' = i - o; \quad V_1 = V_T - V_2; \quad deriv(h) = h'$$

$$\begin{pmatrix} vl\text{-}h & -r & -l & -m & -s & 0 & s & m & l & t \\ -r & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -l & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -m & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -s & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ s & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ m & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ l & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ t & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} h\text{-}o & -r & -l & -m & -s & 0 & s & m & l & t \\ -r & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -l & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -m & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -s & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ s & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ m & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ l & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ t & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The first three equations establish the ordinary algebraic and derivative relationships holding amongst the variables of the system. The functional relations between *vl* and *h*, and *h* and *o* (the fourth and fifth constraints) are weak, but stronger than monotonic operators, represented as degenerated fuzzy relations [7].

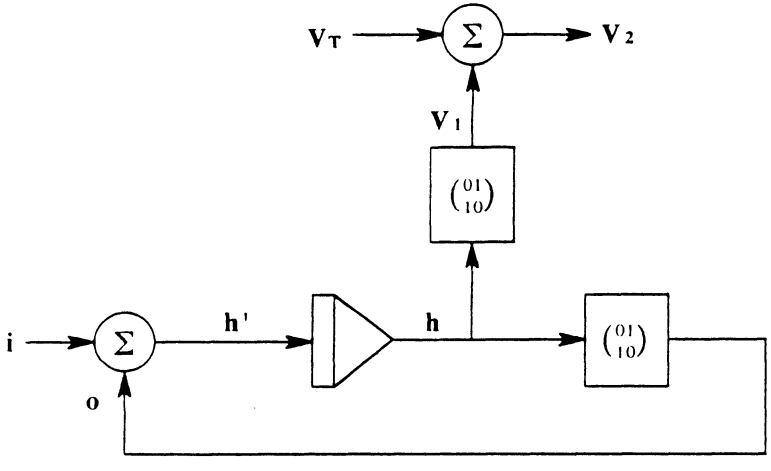

Fig. 6. The Block Diagram of the Flow Loop of the Process Rig

Monitoring and Diagnosis

Suppose that the diagnostic system starts monitoring the physical system from its initial operating condition, where the observed variable *h* has an initial value of *positive small* and the input flow of fluid into the tank, *i*, is assumed to be also *positive small* and kept as constant. When an observation is obtained at $t_1 = 1.9$ from the observable variable *h* such that *OBS(1.9)* = 0.2, where the

observed values are normalised with respect to the underlying numerical ranges of the variables. Based on the above normal-behaviour model, FuSim predicts the immediate next state from the initial one, *s*, which matches with the observation. This indicates that the physical system is performing normally. In the same way, a further observation *OBS(3.8)* = 0.6, also guides the system model to make the next prediction to be checked against the observation, providing synchronous tracking of the normal-behaviour model. However, from the model of normal behaviour FuSim generates a state with *s* as the successor state, resulting in a type II discrepancy. Figure 7 shows the predicted and measured values for the observed variable plotted against time. The small dark rectangle represents the measured value, the short dashes represents the α-cut [7] of the predicted value and the 'T' shapes: the extremities of the predicted fuzzy number. It can easily be seen from this diagram that the value of the observed variable is outside the furthest extremity of the predicted fuzzy number, indicating a type II discrepancy.

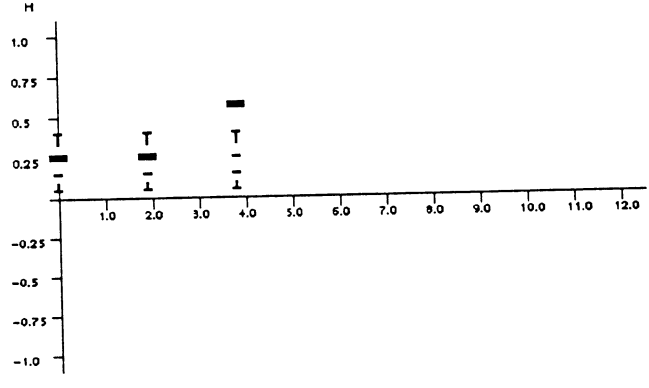

Fig. 7. The Monitoring Process

Now that a discrepancy between the current observation and prediction has been detected and, hence, that the diagnostic system assumes the physical system to be `faulty', the current normal-behaviour model is discarded, and the original monitoring task becomes a one of diagnosis . For simplicity, assume that the diagnostic strategist only requires the candidate proposer to search the model sub-space characterised by a dimension corresponding to the relation strength [9] between the height of fluid in the tank and the output flow. Within this dimension, if there were a known fault model (reflecting a fault condition) then this model would be tried first. However, in this case no fault models exist and so the candidate proposer attempts to modify the existing model against the current modelling dimension to find a diagnosis. Along this relation strength dimension there can exist several degenerated fuzzy relations corresponding the degree to which the relation strength has changed form *normal*. In this case the

first modified model represents a *small* deviation from the normal relationship. An initial state is created by means of this model for the time at which the discrepancy occurred. The state value matches the observation at $t_3 = 3.8$ so the process continues. A further observation is made at $t_4 = 5.7$ which yields OBS(5.7) = 0.8, invoking a prediction with the revised model. Due to qualitative ambiguity a set of states are produced by the prediction whose absolute times all match the observation time These states are then prioritised; their values and absolute times are shown below:

Value:	*s*	*s*	*m*	*s*	*s*
Absolute time:	[4.6 7.1]	[4.6 7.1]	[4.8 7.8]	[4.9 8.1]	[4.9 8.1]

It can be seen that four of the predicted values for the observed variable are the same. This is because a separate state is created if there is an ambiguity in any variable of the system. These prioritised states are checked against the measured value for the observed variable, one by one, in order of priority; and they are all found to fail. The diagnostic process proceeds by modifying the relationship to be a *medium* deviation from normal and repeating the above procedure. Once again there is no match, and the modification is discarded. The diagnosis continues by trying a *large* modification to the relationship. This time the prediction with priority 1 has the value, *l*, and an absolute time of [4.7 6.5], both of which match the measurement. Now we have a model which matches the observations at two time points. Therefore, the model with a *large* change in the degenerate fuzzy relation between *h* and *o* is returned as the candidate. This corresponds to returning a large blockage in the output from the tank as the fault within the physical system. Figure 8 shows the complete temporal evolution of this diagnostic process. Alternatively, it may continue tracking this possible fault to add to the confirmation that the physical system is indeed suffering from this fault.

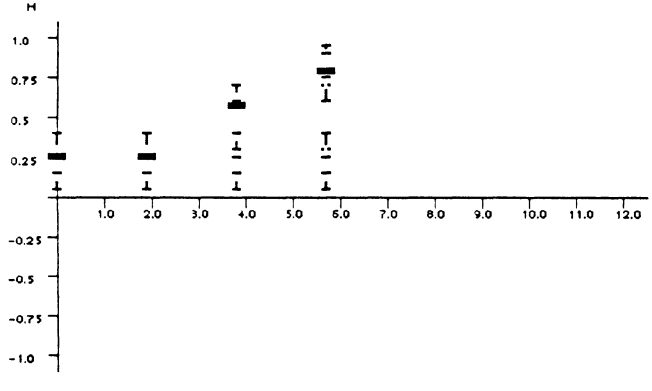

Fig. 8. The Diagnostic Process

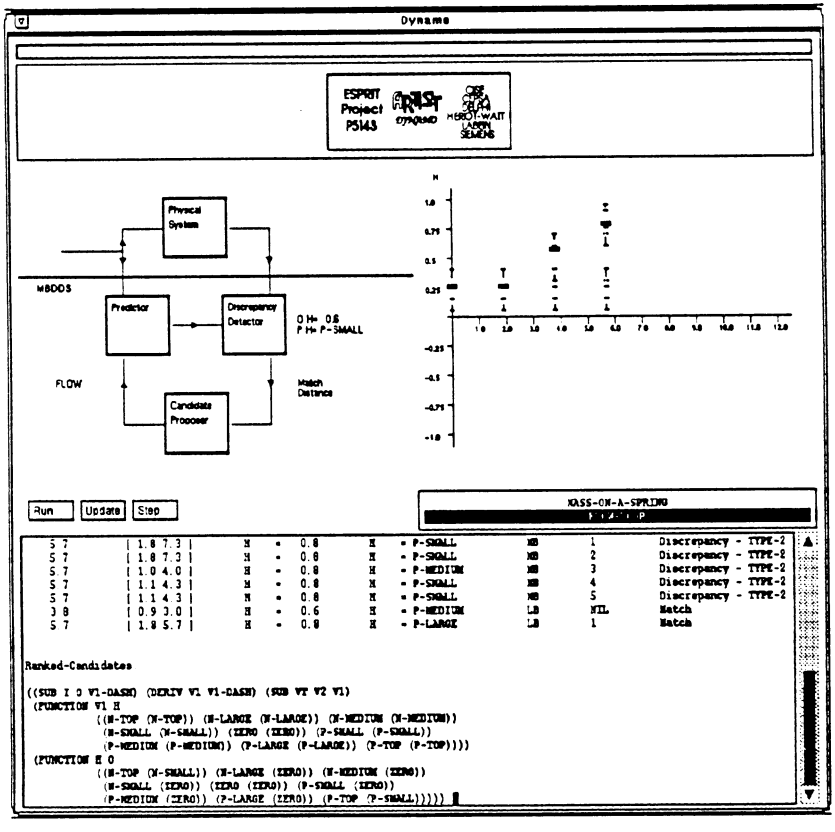

Fig. 9. The Screendump of the Monitoring and Diagnosis of the Process Rig

This example has been implemented in Common Lisp under the LispWorks environment on a SUN-4 Sparcstation II; the output window of the process is shown in figure 9. The upper left portion of the window contains a schematic diagram of the diagnostic process and displays the name of the current model along with the present measured and predicted values of the observed variable. The upper right quadrant displays the temporal evolution of the observed variable, both measured and predicted (the measured value is displayed as a point and the predicted value as a fuzzy interval). The lower half of the display lists the important entities of the monitoring and diagnostic process, which are: the sampling time, the predicted interval, the measured and predicted value(s) of the observed variable, the priority of the prediction, the name of the model and the type of the discrepancy. Once a diagnosis has occurred the candidate is printed in this section of the window.

CONCLUSION

Using qualitative simulation techniques within a Model Based Diagnostic System offers many advantages.

- The model is less precise so that it can accurately represent the available knowledge
- The dynamic model allows for the synchronous tracking of the evolution of the process
- Assigning priorities to the behaviours supports a trade-off between efficiency and completeness

The presented example, whilst laboratory based, is very realistic; coping with many practical problems found in industrial applications. The possibilities for extension to other applications seem to be boundless. At present we are interested in exploring the differences and similarities between our approach and the Control Engineering approaches to Model Based Diagnosis based on real-valued differential equation models.

Acknowledgement
The work described in this paper has been partially undertaken in Esprit project ARTIST (P5143) by the following partners: Cise, Cepsa, Delphi, Heriot-Watt University, Labein, and Siemens. The Authors wish to acknowledge the contribution of all members of the project team whilst taking full responsibility for the views expressed in this paper.

REFERENCES

[1] R. Davis and W. Hamscher, Model Based Reasoning: Troubleshooting, *Exploring Artificial Intelligence*, pp. 297-346, 1988.

[2] D. Dvorak and B. Kuipers, Model-Based Monitoring of Dynamic Systems, *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Vol. 2, pp. 1338-1343, 1989.

[3] B. J. Kuipers, Qualitative Simulation, *Artificial Intelligence*, Vol. 29, pp 289-338, 1986.

[4] R. R. Leitch, M. J. Chantler, Q. Shen, and G. M. Coghill, A Preliminary Specification Methodology for Model Based Diagnosis, *Annals of Mathematics and Artificial Intelligence*, 1993.

[5] R. R. Leitch and Q. Shen, Prioritising Behaviours in Qualitative Simulation, *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, 1993.

[6] R. R. Leitch and Q. Shen, Finding Faults with Model Based Diagnosis, *Proceedings of the 2nd International Workshop on Principles of Diagnosis*, pp 121-134, Milan 1991.

[7] Q. Shen and R. R. Leitch, Fuzzy Qualitative Simulation, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 23, No. 4, 1993.

[8] Q. Shen and R. R. Leitch, On Extending the Quantity Space in Qualitative Reasoning, *Artificial Intelligence in Engineering*, Vol. 7, No. 3, pp. 167-173, 1992.

[9] Q. Shen and R. R. Leitch, Application Studies of Fuzzy Qualitative Simulation, *Mathematics of the Analysis and Design of Process Control*, 1992.

[10] D. Weld and J. de Kleer, *Readings in Qualitative Reasoning about Physical Systems*, 1990.