Automated Safety Monitoring: A Review and Classification of Methods

Yiannis Papadopoulos¹ & John McDermid²

¹Department of Computer Science, University of Hull, Hull, HU6 7RX, UK
e-mail: Y.I.Papadopoulos@hull.ac.uk
²Department of Computer Science, University of York, York YO10 5DD, UK
e-mail: jam@cs.york.ac.uk

Yiannis Papadopoulos holds a degree in Electrical Engineering from Aristotle University of Thessaloniki in Greece, an MSc in Advanced Manufacturing Technology from Cranfield University and a PhD in Computer Science from the University of York. In 1989, he joined the Square D Company where he took up a leading role in the development of an experimental Ladder Logic compiler with fault injection capabilities and a simulator for the Symax programmable logic controllers. Between 1994 and 2001 he worked as a research associate and research fellow in the department of Computer Science at the University of York. Today he is a lecturer in the department of Computer Science at the University of Hull where he continues his research on the safety and reliability of computer systems and software. Dr. Papadopoulos is the author of more than 20 scientific journal and conference papers. His research engages with a body of work on model-based safety analysis and on-line system diagnosis and contributes to that body of work with new models and algorithms that capture and usefully exploit the hierarchical organisation of the structure and behaviour of the system. His work currently develops through extensive technical collaborations with the European Industry, mainly with DaimlerChrysler, Volvo and Germanischer Lloyd.

John McDermid studied Engineering and Electrical Science at Cambridge, as an MoD-sponsored student, then worked at RSRE Malvern (now part of DERA) as a research scientist. Whilst at RSRE he obtained a PhD in Computer Science from the University of Birmingham. He spent five years working for a software house (Systems Designers, now part of EDS) before being appointed to a chair in Software Engineering at the University of York in 1987. In York, he built up a new research group in high integrity systems engineering (HISE). The group studies a broad range of issues in systems, software and safety engineering, and works closely with the UK aerospace industry. Professor McDermid is the Director of the Rolls-Royce funded University Technology Centre (UTC) in Systems and Software Engineering and the British Aerospace-funded Dependable Computing System Centre (DCSC). The work in these two centres was a significant factor in the award of the 1996 Queen’s Anniversary Prize for Higher and Further Education to the University. Professor McDermid has published over 200 papers and 7 books. He has also been an invited speaker at conferences in the UK, Australia and South America. He has worked as a consultant to industry and government in the UK, Europe and North America, particularly advising on policy for the procurement and assessment of safety critical systems.
Abstract. The role of automated safety monitors in safety critical systems is to detect conditions that signal potentially hazardous disturbances, and assist the operators of the system in the timely control of those disturbances. In this paper, we re-define the problem of safety monitoring in terms of three principal dimensions: the optimisation of low-level status information and alarms, fault diagnosis, and failure control and correction. Within this framework, we discuss state of the art systems and research prototypes, and develop a critical review and classification of safety monitoring methods. We also arrive at a set of properties that, in our view, should characterise an effective approach to the problem and draw some conclusions about the current state of research in the field.

1. Introduction

Failures in safety critical systems can pose serious hazards for people and the environment. One way of preventing failures from developing into hazardous disturbances in such systems is to use additional external devices that continuously monitor the health of the system or that of the controlled processes. The role of such safety monitoring devices is to detect conditions that signal potentially hazardous disturbances, and assist the operators of the system in the timely control of those disturbances.

Traditionally, safety monitoring relied primarily on the abilities of operators to process effectively a large amount of status information and alarms. Indeed, until the early 80s, monitoring aids were confined to hard-wired monitoring and alarm annunciation panels that provided direct indication of system statuses and alarms. Those conventional alarm systems have often failed to support the effective recognition and control of failures. A prominent example of such failure is the Three Mile Island nuclear reactor (TMI-2) accident in 1979 [1], where a multiplicity of alarms raised at the early stages of the disturbance confused operators and led them to a misdiagnosis of the plant status. Similar problems have also been experienced in the aviation industry. Older aircraft, for example, were equipped with status monitoring systems that incorporated a large number of instruments and pilot lights. Those systems were also ineffective in raising the level of situational awareness that is required in conditions of failure. Indeed, several accidents have occurred because pilots ignored or responded late to the indications or warnings of the alarm system [2].

The problems experienced with those conventional monitoring systems helped to realise that it is unsafe to rely on operators (alone) to cope with an increasing amount of process information. They highlighted the need of improving the process feedback and the importance of providing intelligent support in monitoring operations. At the same time, the problems of conventional hard-wired systems motivated considerable work towards automating the safety monitoring task and developing computerised safety monitors. In this paper, we explore that body of work on safety monitoring through a comprehensive review of the relevant literature.

We discuss the different facets of the problem, and develop a critical review and classification of safety monitoring methods. We also arrive at a set of properties that, in our view, should characterise an effective approach to the problem. Our presentation is developed in four sections. In section two, we identify and analyse what we perceive as the three key aspects of the problem: the optimisation of a potentially enormous volume of low-level status information and alarms, the diagnosis of complex failures, and the control and correction of those failures. We then use this trichotomy of the problem as a basis for the methodological review, which occupies the main body of the paper (sections three to five). Finally, we summarise the discussion, enumerate a number of properties that an effective safety monitoring scheme should have, and draw some conclusions about the current state of research in the field.

2. Nature of the Problem and Goals of Safety Monitoring

During the 80s and early 90s a series of significant studies in the aerospace, nuclear and process industries1 re-assessed the problem of safety monitoring and created a vast body of literature on

---

1 For studies on requirements for safety monitoring in the aerospace sector, the reader is referred to [2-6]. For similar studies in the nuclear and process industries, the reader is referred to [7-12].
requirements for the effective accomplishment of the task. In this paper, we abstract away from the detailed recommendations of individual studies, and we present a structured set of goals, which, we believe, effectively summarises this body of research. Figure 1 illustrates these goals and the hierarchical relationships between them.

The first of those goals is the optimisation of status monitoring and alarm annunciation. We use the term optimisation to describe functions such as: the filtering of false and unimportant alarms; the organisation of alarms to reflect priority or causal relationships, and the integration of plant measurements and low-level alarm signals into high-level functional alarms. The rationale here is that fewer, and more informative, functional alarms could help operators in understanding better the nature of disturbances. Functional alarms, though, would generally describe the symptoms of a disturbance and not its underlying causes, but the problem here lies in that complex failures are not always manifested with unique, clear and easy to identify symptoms. The reason for this is that causes and their effects do not necessarily have an exclusive one to one relationship. Indeed, different faults often exhibit identical or very similar symptoms.

The treatment of those symptoms typically requires fault diagnosis, that is the isolation of the underlying causal faults from a series of observable effects on the monitored process. Over recent years considerable research has been focused on the development of automated systems that can assist operators in this highly skilled and difficult task. The principal goal of this research is to explore ways of automating the early detection of escalating disturbances and the isolation of root failures from ambiguous anomalous symptoms.

The detection and diagnosis of failures is followed by the final and more decisive phase of safety monitoring, the phase of failure control and correction. Here, the system or the operator has to take actions that will remove or minimise the hazardous effects of failure. In taking such actions, operators often need to understand first how low-level malfunctions affect the functionality of the system. Thus, there are two areas in which an automated monitor could provide help in the control and correction of failures: the assessment of the effects of failure, and the provision of guidance on appropriate corrective measures.

An additional and potentially useful goal of advanced monitoring is prognosis, the prediction of future conditions from present observations and historical trends of process parameters. In the context of mechanical components, and generally components which are subject to wear, prognosis can be seen as the early identification of conditions that signify a potential failure [13]. At system level, the objective is to make predictions about the propagation of a disturbance while the disturbance is in progress and still at the early stages. Such predictions could help, for example, to assess the severity of the disturbance and, hence, the risk associated with the current state of the plant. Prognosis is, therefore, a potentially useful tool for optimising the decisions about the appropriate course of action during process disturbances. In this paper, we deal with two aspects of the safety monitoring problem that are closely related to prognosis, the early detection and diagnosis of disturbances and the assessment of the functional effects of low-level failures. We wish to clarify though that the issue of prognosis itself is out of the scope of this paper.

![Diagram](image_url)

**Figure 1.** Goals of advanced safety monitoring
3. Optimisation of Status Information and Alarms

The first key area of the safety monitoring problem identified in the preceding section is the optimisation of the status information and alarms produced by the monitoring system. Operational experience shows that disturbances in complex systems can cause an enormous, and potentially confusing, volume of anomalous plant measurements. Indeed, in traditional alarm annunciation panels this phenomenon has caused problems in the detection and control of disturbances. In such systems, alarms often went undetected by operators as they were reported amongst a large number of other less important and even spurious alarms [9]. The experience from those conventional monitoring systems shows that a key issue in improving the process feedback is the filtering of false and unimportant alarms.

False alarms are typically caused either by instrumentation failures, that is failures of the sensor or the instrumentation circuit, or normal parameter transients that violate the acceptable operational limits of monitored variables. An equally important factor in the generation of false alarms is the confusion often made between alarms and statuses [14]. The status of an element of the system or the controlled process indicates the state that the element (component, sub-system or physical parameter) is in. An alarm, on the other hand, indicates that the element is in a particular state when it should be in a different one. Although status and alarm represent two different types of process information, statuses are often directly translated into alarms. The problem of false alarms can then arise whenever a particular status is not a genuine alarm in every context of the system operation.

Another important issue in improving the process feedback appears to be the ability of the monitoring system to recognise distinct groups of alarms and their causal/temporal relationships during the disturbance. An initiating hazardous event is typically followed by a cascading sequence of alarms, which are raised by the monitor as the failure propagates in the system and gradually disturbs an increasing number of process variables. However, a static view of this set of alarms does not convey very useful information about the nature of the disturbance. The order in which alarms are added into this set, though, can progressively indicate both the source and future direction of the disturbance (see, for example, the work of Dansak and Roscoe [15,16]). Thus, the ability to organise alarms in a way that can reflect their temporal or causal relationships is an important factor in helping operators to understand better the nature of disturbances.

An alternative, and equally useful, way to organise status information and alarms is via a monitoring model which relates low-level process indications to functional losses or malfunctions of the system (see for example the work of Kim, Modarres, and Larsson [17-19]. Such a model can be used in real-time to provide the operator with a functional, as opposed to a purely architectural view of failures. This functional view is becoming important as the complexity of safety critical systems increases and operators become less aware of the architectural details of the system.

A number of methods have been proposed to address the optimisation of the process feedback during a disturbance. Their aims span from filtering of false alarms to the provision of an intelligent interface between the plant and the operator that can organise low-level alarm signals and plant measurements into fewer and more meaningful alarm messages and warnings. Figure 2 identifies a number of such methods, and classifies them into three categories using as a criterion their primary goal with respect to safety monitoring. A discussion of these methods follows in the remainder of this section.

![Figure 2. Classification of alarm optimisation methods](image-url)
3.1 Detection of Sensor Failures

A primary cause of false and misleading alarms is sensor failures. Such alarms can be avoided using general techniques for signal validation and sensor failure detection. Sensor failures can be classified into two types with respect to the detection process: coarse (easy to detect) and subtle (more difficult to detect). Coarse failures cause indications that lie out of the normal measurement range and can be easily detected using simple limit checking techniques [20]. Examples of such failures are a short circuit to the power supply or an open circuit. Subtle failures, on the other hand, cause indications that lie within the normal range of measurement and they are more difficult to detect. Such failures include biases, non-linear drifts, and stuck at failures.

One way to detect such failures is to employ a hardware redundancy scheme, for example, a triplex or quadruple configuration of components [21]. Such a system operates by comparing sensor outputs in a straightforward majority-voting scheme. The replicated sensors are usually spatially distributed to ensure protection against common cause failure. Although hardware replication offers a robust method of protection against sensor failures, at the same time it leads to increased weight, volume and cost.

An alternative, and in theory more cost-effective, way to detect subtle failures is by using analytical redundancy, in other words relationships between dissimilar process variables. State estimation [22] and parity space [23-25] techniques, employ control theory to derive a normally consistent set of such relationships, detect inconsistencies caused by sensor failures and locate these failures. Kim and Modarres [26] propose an alternative method based on analytical redundancy, in which the diagnosis is achieved by parsing the structure of a sensor validation tree, and by examining in the process a number of sensor validation criteria specified in the tree structure. These criteria represent coherent relationships among process variables, which are violated in the context of sensor failures. Such relationships are formulated using deep knowledge about the system processes, for example, pump performance curves, mass and energy conservation equations and control equations.

Overall, a number of methods based on analytical redundancy have been developed to detect and diagnose sensor faults [27]. Experience, however, from attempts to apply these methods in industrial scale has shown that the development of the required models is an expensive process which often results to incomplete and inaccurate results [28]. More recently, research has been directed to the concept of self-diagnostics, in which sensor failures are detected not from their effects on the process but through self-tests that exploit the manufacturer’s own expertise and knowledge of the instrument [29]. The state of the art in this area is the SEVA approach developed at Oxford University [30,31]. In SEVA, sensor self-diagnosis is supplemented with an assessment of the impact of detected faults on measurement values. If a measurement is invalid, a SEVA sensor will attempt to replace that measurement with a projection of historical values of the given parameter. Together with the measurement value, SEVA sensors also provide information about uncertainty in the measurement as well as detailed device specific diagnostics for maintenance engineers. A number of prototype SEVA sensors have been developed and the concept is now reaching maturity and the stage of commercialisation.

3.2 State-Alarms Association

Another cause of false and irrelevant alarms is the confusion often made between statuses and alarms [14]. A seemingly abnormal status of a system parameter (e.g. high or low) is often directly translated to an alarm. This direct translation, however, is not always valid. To illuminate this point, we will consider a simple system in which a tank is used to heat a chemical to a specified temperature (see Figure 3). The operational sequence of the system is described in the GRACET notation (see for example, [32]) and shows the four states of this system: idle, filling, heating and emptying. Initially the system is idle, and the operational sequence starts when the start cycle signal arrives. A process cycle is then executed as the system goes through the four states in a sequential manner and then the cycle is repeated until a stop signal arrives.

Let us now attempt to interpret the meaning of a tank full signal in the three functional states of the system. In the filling state, the signal does not signify an alarm. On the contrary, in this state the signal is expected to cause a normal transition of the system from the filling state to the heating state. In the heating state, not only does the presence of the signal not signify an alarm, but its absence should generate one. The tank is now isolated, and therefore, the tank full signal should remain hi as long as the system remains in that state. Finally, in the emptying state, the signal signifies a genuine alarm. If tank full remains while the system is in this state then either Pump B has failed or the tank outlet is blocked.
This example indicates clearly that we need to interpret the status in the context of the system operation before we can decide if it is normal or not. One way we can achieve this is by determining which conditions signify a real alarm in each state or mode of the system. The monitor can then exploit mechanisms which track down system states in real time to monitor only the relevant alarms in each state. The concept was successfully demonstrated in the Alarm Processing System [14]. This system can interpret the process feedback in the evolving context of operation of a nuclear power station. The system has demonstrated that it significantly reduces the number of false and unimportant alarms.

3.3 Alarm Filtering Using Multiple Stage Alarm Relationships

Multiple stage alarms are often used to indicate increasing levels of criticality in a particular disturbance. A typical example of this is the two stage alarms employed in level monitoring systems to indicate unacceptably high and very high level of content. When the actual level exceeds the very high point both alarms are normally activated. This is obviously unnecessary because the very-high level alarm implies the high level alarm. The level-precursor relationship method that has been employed in the Alarm Filtering System [34] extends this concept and uses more complex hierarchical relationships between multiple stage alarms to suppress precursor alarms whenever a more important alarm is generated. Many alarms could be filtered using this method alone in a typical process plant.

3.4 Identifying Alarm Patterns

An alarm pattern is a sequence of alarms which is typically activated following the occurrence of an initiating hazardous event in the system. Alarm patterns carry a significantly higher information content than the set of alarms that compose them. A unique alarm pattern, for example, unambiguously points to the cause of the disturbance, and indicates the course of its future propagation. The idea of organising alarms into patterns and then deriving diagnoses on-line from such patterns was first demonstrated in the Diagnosis of Multiple Alarms System at the Savannah River reactors [15]. A similar approach was followed in the Nuclear Power Plant Alarm Prioritisation program [16], where the alternative term alarm signature was used to describe a sequence of alarms which unfold in a characteristic temporal sequence.

A shortcoming of this approach is that the unique nature of the alarm signature is often defined by the characteristic time of activation of each alarm. The implication is that the “pattern matching” algorithm must know the timing characteristics of the expected patterns, a fact which creates substantial technical difficulties in the definition of alarm patterns for complex processes. An additional problem is that the timing of an alarm signature may not be fixed and may depend on several system parameters. In the cooling system of a nuclear power plant, for example, the timing of an alarm sequence will almost certainly depend on parameters such as the plant operating power level, the number of active coolant pumps, and the availability of service water systems [35].
### 3.5 Organising Alarms Using Alarm Trees

An alternative way to organise alarms is the *alarm tree* [7]. The tree is composed of nodes, which represent alarms, and arrows that interconnect nodes and denote cause-effect relationships. Active alarms at the lowest level of the tree are called *primary cause alarms*. Alarms which appear at higher levels of the tree (effect alarms) are classified in two categories: important (uninhibited) alarms and less important (inhibited) alarms. During a disturbance, the primary cause alarm and all active uninhibited alarms are displayed using the causal order with which they appear in the tree. At the same time, inhibited alarms are suppressed to reduce the amount of information returned to the operator. Beyond real process alarms based on measurements, the tree can also generate messages based on such alarms. These messages, for example, may relate a number of alarms to indicate a possible diagnosis of a process fault. Such diagnostic messages are called *deduced or synthesised* alarms.

Alarm trees have been extensively tried in the UK Nuclear industry. Plant-wide applications of the method, however, have shown that alarm trees are difficult and costly to develop. It took ten man-years of effort, for example, to develop the alarm trees for the Oldbury nuclear power plant [9]. Operational experience has also indicated that large and complex alarm trees often deviate from the actual behaviour of the plant and that they do not appear to be very useful to operators. In evaluating the British experience, Meijer [36] states that the “most valuable trees were relatively small, typically connecting three or four alarms”.

### 3.6 Synthesising Alarms Using Logical Inferences

A more flexible way of organising alarms and status information is by using logic and production rules [37]. A production rule is an *If-Then* statement that defines an implication relationship between a prerequisite and a conclusion. Production rules can be used to express logical relationships between low level process indications and more informative conclusions about the disturbance. Such rules can be used in real time for the synthesis of high level alarms. Rule-based monitors incorporate an inference engine which evaluates and decides how to chain rules, using data from the monitored process. As rules are chained and new conclusions are reached, the system effectively synthesises low-level plant data into higher level and more informative alarms.

To illustrate the concept we will use the example of a pump, which delivers a substance from A to B (Figure 4). One of the faults that may occur in this system is an inadvertent closure of the suction valve while the pump is running. In such circumstances, the operator of a conventional system would see the status of the pump remain set to *running* and two alarms: *no flow* and *low pressure at the suction valve*. On the other hand, a rule-based system would call a simple rule to synthesise these low-level data and generate a single alarm that does not require further interpretation. This alarm (*suction valve inadvertently closed*) certainly conveys a more dense and useful message to the operator.

It is worth noticing that in generating this alarm, the system has performed a *diagnostic* function. Indeed, rule-based systems have been extensively applied in the area of fault diagnosis. The idea of using logic and production rules for alarm synthesis has been successfully applied in the Alarm Filtering System [34], an experimental monitor for nuclear power plants.

### 3.7 Function-Oriented Monitoring

The status monitoring and alarm organisation methods that we have examined so far share a fundamental principle. Monitoring is performed against a model of the *behaviour of the system in conditions of failure*. Indeed, the different models that we have discussed (e.g. alarm patterns, alarm trees) are founded on a common notion of an expected anomalous event, a signal which signifies equipment failure or a disturbed system variable.

![Figure 4. Production rule and transfer pump](image)
A radically different approach is to monitor the system using a model of its normal (fault free) behaviour. The objective here is to detect discrepancies between the actual process feedback and the expected normal behaviour. Such discrepancies indicate potential problems in the system and, therefore, provide the basis for generating alarms and warnings.

A simple monitoring model of normal behaviour is the electronic checklist. An electronic checklist associates a critical function of the system with a checklist of normal conditions which ensure that the function can be carried out safely. In real-time, the checklist enables monitoring of the function by evaluation of the necessary conditions. Electronic checklists have been widely used in the design of contemporary aircraft. Figure 5 illustrates an example from the Electronic Instrument System of the MD-11 aircraft [38]. Before a take-off, the pilot has to check assorted status information to ensure that he can proceed safely. The function “safe take-off” is, therefore, associated with a list of prerequisite conditions. If a take-off is attempted and these essential conditions are not present, the monitor generates a red alert. The checklist is then displayed to explain why the function cannot be carried out safely.

Although electronic checklists are useful, they are limited in scope. To enable more complex representations of normal behaviour, Kim and Modares [17] have proposed a much more powerful model called the Goal Tree Success Tree (GTST). According to the method, maintaining a critical function in a system can be seen as a target or a goal. In a nuclear power plant, for example, a basic goal is to “prevent radiation leak to the environment”. Such a broad objective, though, can be decomposed into a group of goals which represent lower-level safety functions that the system should maintain in order to achieve the basic objective. These goals are then analysed and reduced to sub-goals and the decomposition process is repeated until sub-goals cannot be specified without reference to the system hardware. The result of this process is a Goal Tree which shows the hierarchical implementation of critical functions in the system. A Success Tree is then attached to each of the terminal nodes in the Goal Tree. The Success Tree models the plant conditions which satisfy the goal that this terminal node represents.

The GTST contains only two types of logical connectives: AND and OR gates. Thus, a simple value-propagation algorithm [26,39] can be used by the monitor to update nodes and provide functional alarms for these critical safety functions that the system fails to maintain. The GTST has been successfully applied in a number of pilot applications in the nuclear [18], space [40,41] and process industries [42]. Later on in this paper we return to this model to explore how it has been employed for on-line fault diagnosis.

### 3.8 Hierarchical Presentation of Status Information

An additional aspect of the problem of optimising status information is presentation. This aspect of the problem involves many issues such as the design of appropriate media for input and display, the user-interface design and other issues of man-machine interaction. Many of those issues are very much system, application and task dependent. Different interfaces, for example, would be required in a cockpit and a plant control room. We have identified however, and we discuss here, a fundamental principle that has been used in a number of advanced systems to enhance the presentation of status information. This is the concept of layered monitors or, in other words, the idea of hierarchical presentation of the process feedback.

<table>
<thead>
<tr>
<th>Critical function</th>
<th>Checklist</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safe take – off</td>
<td>Slab not in green band</td>
</tr>
<tr>
<td></td>
<td>Slats not in take-off position</td>
</tr>
<tr>
<td></td>
<td>Flaps not in take-off position</td>
</tr>
<tr>
<td></td>
<td>Parking brake on</td>
</tr>
<tr>
<td></td>
<td>Spoilers not armed</td>
</tr>
</tbody>
</table>

**Figure 5.** Checklist of essential items for take-off [38]
Operational experience in safety critical systems shows that operators of such systems need much less information when systems are working than when they are malfunctioning [2]. To achieve this type of control over the amount of information provided to the operator, many contemporary systems employ a layered monitor. Under normal conditions, a main system display is used to present only a minimum amount of important data. When a subsystem malfunction is detected, more data is presented in a subsystem display, either automatically or on request. This approach is used in systems such as the Electronic Instrument system of Douglas MD-11 [43] and the Electronic Centralised Aircraft Monitor of Airbus A320 [44,45]. In both systems, the aircraft monitor consists of two displays.

The first display is an Engine/Warning Display, which displays the status and health of the engines in terms of various parameters. The engine status is the most significant information under normal flight and it is always made available to the pilot. The second part of the display is an area dedicated to aircraft alerting information. The second display is called the Subsystems Display and can display synoptics of the aircraft subsystems (Electrical, Hydraulic, Fuel, Environment etc.). A particular synoptic can be selected from a control panel by pressing the corresponding push-button. Part of this display is an area where the system generates reports about failures, provides warnings about consequences and recommends corrective measures. Figure 6 illustrates how the MD-11 electronic instrument system reports a hydraulic fluid loss in one of the three hydraulic systems.

Initially, an alerting message appears in the main (engine) display. The hydraulic system push-button on the control panel (HYD) is also lighted. When the button is actuated, the monitor brings up the synoptic of the hydraulic system. The synoptic shows that there is low hydraulic fluid in system A. It also displays the new status of the system after the automatic resolution of the fault by inactivation of the system A pump. The consequences of the fault are reported and some actions are recommended to the pilot.

The idea of a layered monitor can also be found in the nuclear industry [46]. In a similar fashion, top level displays are used to monitor important parameters which provide indication of the overall plant status. Low level displays provide a focus on subsystems. When a process anomaly is detected, the monitoring system generates appropriate messages to guide the operator from main displays to appropriate subsystem displays.

4. On-line Fault Diagnosis

If faults and their effects had an exclusive one to one relationship then complex failures would always have exhibited unique and easy to identify symptoms. In practice, however, more than one fault often causes identical or very similar symptoms. The control of such failures requires on-line fault diagnosis, in other words the isolation of the underlying causal faults from a series of observable effects on the monitored process.

Fault diagnosis is a demanding task for system operators. It is performed under the stressful conditions that normally follow a serious failure, and the results have to be timely in order to allow further corrective action. The task also requires a wide range of expertise which typically include a functional understanding of components and their interrelationships, knowledge of the failure history of the system, and familiarity with system procedures [47].

![Figure 6. A hydraulic failure reported by the MD-11 EIS [38]](image-url)
The complicated nature of the task has motivated considerable research towards its automation. The common goal of this research has been the development of monitors that can perform diagnoses, by effectively combining on-line *observations*² of the system and some form of *coded knowledge* about the system and its processes. Beyond this common goal though, there is an interesting, and not yet concluded, debate concerning the type of *coded knowledge* that would be most appropriate for fault diagnosis. Should this knowledge be a set of problem-solving heuristics generated by an expert diagnostician, or should it be based on explicit models of the system? And if we assume the latter, which type of model would be more suitable for diagnostic reasoning: a model of normal behaviour of the system, for example, or a model of disturbed behaviour?

This debate has played an important role in the evolution and diversification of research on automated fault diagnosis. In the context of this paper, the debate concerning the type of knowledge and its representation provides an observation point from which we can see diagnostic methods converging and forming “families”, and then those families of methods diverging as different approaches to the same problem. Figure 7 classifies diagnostic methods using as a criterion the type of knowledge they use as a basis for the diagnostic reasoning.

At the first level of this classification we identify three classes of methods, the *rule-based experiential/heuristic* approach, *model-based* approaches and *data-driven* approaches. Within the model-based approach, we make a further distinction between two classes of diagnostic methods: those that rely on *models of disturbed behaviour* and those that rely on models of the *normal behaviour* of the system. Models of disturbed behaviour include fault propagation models that are typically used in reliability engineering and qualitative causal process graphs such as the digraph. Models of normal behaviour include models that can be used to represent the functional decomposition of the system, models of system topology and component behaviour, and qualitative simulation models. Finally, the third class of methods in Figure 7 includes all those methods that infer process anomalies from observations of historical trends of process parameters. We collectively characterise those methods as *data-driven* approaches. A further discussion of the above methods follows in the remainder of this section.

---

² The term *observations* here is used in a general sense and may include facts which are derived through a dialogue with the operator.
4.1 Rule-based Expert Systems

Diagnosis here is seen as a process of observing anomalous symptoms and then relating these symptoms to possible malfunctions using rules that describe relationships between faults and their effects. This approach, which is often termed as "shallow reasoning" [48], has originated from early work in medical diagnosis and the well-known MYCIN expert system [49]. Similar diagnostic monitors for engineered systems started to appear in the early eighties. These systems reason about process disturbances by operating on a knowledge base that contains two categories of knowledge: (a) declarative knowledge in the form of statements about the system, e.g., its initial conditions or configuration (facts) and (b) procedural knowledge in the form of production rules. Production rules typically describe functional relationships between system entities and causal relationships between failures and effects. Such rules are formed from empirical associations made by expert troubleshooters, or they are derived from analysis of plant models and theoretical studies of plant behaviour.

The theory underlying rule-based systems is well documented in the literature (see, for example, [50,51]). In this paper, we assume some basic knowledge of this theory to discuss the application of such systems in fault diagnosis. Diagnosis here is achieved with either forward or backward chaining of rules. In a typical diagnosis with forward chaining, a deviation of a monitored system parameter will trigger and fire a production rule. When the rule is fired, its consequent (conclusion) is considered a new fact that in turn may trigger other rules which anticipate this fact as one of their antecedents (prerequisites). The derived facts will then trigger a chain of rules and the process is repeated until the most recently derived facts cannot trigger any other rules. These facts describe the causes of the disturbance.

Another way to perform diagnosis is by testing fault hypotheses [52]. A fault hypothesis is typically a malfunction that operators or the monitor hypothesise to be true on the basis of some abnormal indications. Testing fault hypotheses requires a different inference mechanism called backward chaining. In backward chaining, the knowledge base is first scanned to see if the hypothesis is a consequent of a rule. The antecedents of the rule then formulate the next set of hypotheses. Hypothesis testing continues recursively until some hypothesis is false or all hypotheses are verified with real-time observations, in which case the system confirms the original fault hypothesis. An example of a diagnostic monitor which incorporates both inference paradigms is REACTOR [53,54], an experimental monitor for nuclear power plants. FALCON [55,56] is a similar prototype rule-based system that identifies the causes of process disturbances in chemical process plants.

One of the problems of early expert systems was their inability to record history values and, in consequence, their inability to perform temporal reasoning. This limitation has prohibited the application of early rule-based systems to many diagnosis problems, where it is important to watch both short-term and long-term trends of system parameters. In more recent developments, though, we can see temporal extensions to the rule-based approach [57]. RT-WORKS, for example, is a rule-based expert system shell which also uses a frame representation language [58] to enable the definition of object classes. Each class has attributes, called frame slots, which can be used to represent monitored variables. Each frame slot can point to buffers where a series of values and their associated time tags are kept. An instance of the class is therefore defined not only in terms of its present state, but also in terms of its history.

Figure 8 illustrates an instance of a hypothetical BATTERY frame class. The system allows production rules to reference the attributes of such frames and the inference engine to retrieve/insert history values in the buffers. Rules can also reference a number of statistical and other mathematical functions, which allow the inference engine to calculate trends and reason about past and future events.

![Figure 8. An instance of a hypothetical BATTERY frame [58]](image)
A number of diagnostic monitors have adopted this hybrid rule/frame-based approach as a more powerful framework for knowledge representation. Ramesh et al, for example, have developed a hybrid expert system that is used to analyse disturbances in an acid production plant [59]. RESHELL is a similar expert system for fault diagnosis of gradually deteriorating engineering systems [60].

An additional problem in diagnosis is the possible uncertainty about the implication relationship between observations made and conclusions reached in the course of the diagnosis. This uncertainty is often inevitable, especially when the diagnosis exploits production rules that represent empirical associations about which we can only have a certain level of confidence. There can also be uncertainty about the antecedents or consequents of production rules when, for example, the verification of the facts that they anticipate is not a straightforward task but requires human judgement. Probably the most common way of representing this uncertainty is the use of certainty factors [49]. A certainty factor is a heuristic index (number) which associates a level of confidence with a rule, an antecedent or a consequent. Systems using such factors can arrive at one or more diagnostic conclusions with a degree of certainty. This is achieved as the inference engine uses certainty factors to continuously update belief values about antecedents or consequents that are encountered in the course of the inference process. The technique was first described in the MYCIN project [49] and it was later applied in other systems such as the mineral exploration expert system PROSPECTOR [61].

Rule based systems have demonstrated several benefits of separating knowledge from reasoning. They have demonstrated, for example, that the same principles can be applied to solve diagnostic problems in a number of domains, which span from medical diagnosis to troubleshooting of electromechanical devices. Rule-based systems can also perform both fault diagnosis and fault hypothesis testing, by operating on the same knowledge base and switching to the appropriate inference mechanism. Recently, rule-based diagnostic systems have been successfully combined with neural networks, an emerging and promising technology for the primary detection of symptoms of failure in industrial processes. QUINCE [62,63], for example, is a hybrid system that relies on neural networks to detect the primary symptoms of disturbances and rules that reflect expert knowledge to relate those symptoms to root faults. In this system, a neural network is trained to predict the values of monitored parameters. Provided that the network is trained on patterns that reflect normal conditions of operation, the prediction error of the network can be used as an indicator of how unexpected new values of the monitored parameters are. This in turn can be used to signal anomalies in the system. A set of observed such anomalies form a symptoms vector which is then multiplied by a fault matrix, in which expert rules that describe causal links between faults and symptoms have been encoded. The result of this operation indicates the most likely underlying fault that can explain the observed symptoms. This system has been used for monitoring of vibration harmonics and fault diagnosis in aircraft engines.

In general, the application of rule-based systems in engineering has been very successful in processes that can be described using a small number of rules. However, attempts of wider application to complex systems have shown that large rule-based systems are prone to inconsistencies, incompleteness, long search time and lack of portability and maintainability [18,64]. From a methodological point of view, many of these problems can be attributed to the limitations of the production rule as a model for knowledge representation [65]. Production rules allow homogenous representation and incremental growth of knowledge [66]. At the same time though, simple implication relationships offer analysts very limited help in thinking about how to represent properties of the system, or in thinking about how to use such knowledge for reasoning about the nature of disturbances. These problems underlined the need for more elaborate models for knowledge representation and created a strong interest towards model-based diagnostic systems.

### 4.2 Fault Propagation Models

The first monitor with some diagnostic capabilities that employed a fault propagation model was STAR [67], a small experimental system for the OECD Halden Reactor in Norway. STAR made use of Cause Consequence Diagrams (CCDs) to model the expected propagation of disturbances in the reactor.

CCDs represent in a formal, logical way the causal relationships between the status of various system entities during a disturbance. They are typically composed of various interconnected cause and consequence trees. A cause tree describes how logical combinations of system malfunctions and their effects lead to a hazardous event. On the other hand, a consequence tree starts with the assumption that a hazardous event has occurred and then describes how other conditions and the success or failure of
corrective measures lead the system to safe or unsafe states. Figure 9 illustrates an example cause consequence diagram.

CCDs are used in STAR as templates for systematic monitoring of the system. The monitoring model is updated every five seconds with fresh process feedback, and every event in the CCD that can be verified with real-time data is checked for occurrence. An analysis module then examines the model for event chains and by extrapolation of these chains it determines possible causes of the disturbance and possible subsequent events. The EPRI-DAS [36] system uses a similar model, the *Cause Consequence Tree* (CCT), to perform early matching of anomalous event patterns that could signify the early stages of hazardous disturbances propagating into the system. The Cause Consequence Tree is a modified Cause Tree in which delays have been added to determine the precise chronological relationships between events.

Despite some methodological problems such as the difficulties in the development of representative CCDs and problems of computational efficiency [9], the principles that these early model-based systems have introduced are still encountered in contemporary research. The safety monitors recently developed for the Surry and San Onofre nuclear power plants in the USA [68], for example, also employ a fault-propagation model to assist real-time safety monitoring. This is a *master fault tree* [69] synthesised from the plant Probabilistic Risk Assessment (PRA).

Varde et al [70] have proposed an alternative way of exploiting fault trees for diagnostic purposes. They describe a hybrid system in which artificial neural networks perform the initial primary detection of anomalies and transient conditions and a rule-based monitor then performs diagnoses using a knowledge base that has been derived from the fault trees contained in the plant PRA. The knowledge base is created as fault trees are translated into a set of production rules, each of which records the logical relationship between two successive levels of the tree.

A traditional limitation of fault propagation models has been the difficulty of representing the effects that changes in the behaviour or structure of complex dynamic systems have in the failure behaviour of those systems. To illustrate this point we will consider, for example, the fuel system of an aircraft. In this system, there are usually a number of alternative ways of supplying the fuel to the aircraft engines. During operation, the system typically switches between different states in which it uses different configurations of fuel resources, pumps and pipes to maintain the necessary fuel flow. Initially, for example, the system may supply the fuel from the wing tanks and when these resources are depleted it may continue providing the fuel from a central tank. The system also incorporates complex control functions such as fuel sequencing and transfer among a number of separated tanks to maintain an optimal centre of gravity. If we attempt to analyse such a system with a technique like fault tree analysis we will soon be faced with the difficulties caused by the dynamic character of the system. Indeed, as components are activated, deactivated or perform alternative fuel transfer functions in different operational states, the set of failure modes that may have an adverse effect on the system changes. Almost inevitably, the causes and propagation of failure in one state of the system are different from those in other states. In that case, though, how can such complex state dependencies be taken into account during the fault tree analysis, and how can they be represented in the structure of fault trees?

![Figure 9. Cause Consequence Diagram (CCD)](image-url)
To address this problem Papadopoulos and McDermid [71] introduce in the centre stage of the assessment process a dynamic model that can capture the behavioural and structural transformations that occur in complex systems. In its general form, this model is a hierarchy of abstract state-machines (see right hand side of Figure 10) which is developed around the structural decomposition of the system (see left hand side of Figure 10). The structural part of that model records the physical or logical architectural decomposition of the system into sub-systems and basic components, while the state machines determine the behaviour of the system and its sub-systems. The lower layers of the behavioural model identify transitions of low-level sub-systems to abnormal states, in other words states where those sub-systems deviate from their expected normal behaviour. As we progressively move towards the higher layers of the behavioural model, the model shows how logical combinations or sequences of such lower-level subsystem failures (transitions to abnormal states) propagate upwards and cause failure transitions at higher levels of the design.

As Figure 10 illustrates, some of the transition conditions at the low-levels of the design represent the top events of fault trees which record the causes and propagation of failure through the architectures of the corresponding sub-systems. A methodology for the semi-mechanical construction of such models is elaborated in [72]. According to that methodology (see also [73,74]) the fault trees which are attached to the state-machines can be mechanically synthesised by traversing the structural model of the system and by exploiting some form of local component failure models, which define how the corresponding components fail in the various states of system.

As opposed to a classical fault tree or a cause consequence diagram, the proposed model records the gradual transformation of lower-level failures into system malfunctions, taking into account the physical or logical dependencies between those failures and their chronological order. Thus, the model not only situates the propagation of failure in the evolving context of the system operation, but also provides an increasingly more abstract and simplified representation of failure in the system. Furthermore, the model can be mechanically transformed into an executable specification, upon which an automated monitor could operate in real-time. In [72], Papadopoulos describes the engine of such a monitor, in other words a set of generic algorithms by which the monitor can operate on the model in order to deliver a wide range of monitoring functions. These function span from the primary detection of the symptoms of disturbances though on-line fault diagnosis to the provision of corrective measures that minimise or remove the effects of failures. This concept has been demonstrated in a laboratory model of an aircraft fuel system.

![Figure 10. Placing Fault Trees in the Context of a Dynamic Behavioural Model](image-url)
4.3 Qualitative Causal Process Graphs

The second class of diagnostic methods that rely on a model of disturbed behaviour is formed by methods which use qualitative causal process graphs. These graphs generally provide a graphical representation of the interactions between process variables in the course of a disturbance (although in the general case those interactions may also be valid during normal operation). Such models include the digraph [75-77], the logic flow-graph [78,79] and the event graph [80-82].

A digraph models how significant qualitative changes in the trend of process parameters affect other parameters. The notation uses nodes that represent process parameters and arrows that represent relationships between these parameters. If a process variable affects another one, a directed edge connects the independent and dependent variable and a sign (-1, +1) is used to indicate the type of relationship. A positive sign indicates that changes in the values of the two variables occur in the same direction (e.g. an increase will cause an increase), while a negative sign denotes that changes occur in the opposite direction. A diagnostic system can exploit these relationships to trace backwards the propagation of failure on the digraph and locate the nodes that have deviated first as a direct effect of a fault.

We will use a simple example to demonstrate the basic principles underlying the application of digraphs in fault diagnosis. Figure 11 illustrates a heat exchanger and part of the digraph of this system. In this system, the hot stream follows the path between points one and two (1,2) and the cooling stream follows the path between points three and four (3,4). Since there are no control loops, any disturbance in the input variables (the temperature and flow at the inputs of the hot stream \([T1, M1]\) and the cooling stream \([T3, M3]\)) will cause a disturbance in the output temperature of the hot stream \((T2)\). The digraph determines this relationship of \(T2\) with the four input parameters. It also anticipates the possibility of a leak in the cooling system, through a normally closed valve for example. This event is represented as the flow of leaking coolant \(M5\). Finally, the digraph defines the relationships between the leakage \((M5)\), the pump speed \((P6)\) and the coolant flow \((M3)\).

Let us now assume that the diagnostic monitor observes the event "excessive temperature at the output of the hot stream", which we symbolise as \(T2(+1)\). The task of the monitor is to locate one or more nodes, termed primary deviations [76], that can explain the event \(T2(+1)\) and any other abnormal event in the digraph. From the relationships between \(T2\) and its adjacent nodes the system can determine that the possible causes of \(T2(+1)\) are \(T1(+1), M1(+1), T3(+1)\) or \(M3(-1)\). If all the process variables were observable, the system would be able to select a node in which to proceed with the search. If \(M3(-1)\) was true, for example, the set of possible causes would change to \([P6(-1), M5(+1)]\). If we further assume that \(P6(-1)\) was true, the system would be able to locate the cause of the disturbance to a single primary deviation.

In practice though, only a small proportion of digraph nodes is monitored and, hence, the diagnosis is likely to be ambiguous and to contain a set of more than one possible primary deviations [64]. In the DIEX expert system [83], for example, every time that a node deviates the system initiates a backward search on the digraph to determine a set of potential primary deviations. This search is terminated when a boundary of the digraph is reached, a normal node is met or the same node is encountered twice. With each new abnormal indication, a new backward search is initiated starting from the disturbed node, and the resulting set of candidate nodes is intersected with the existing set of primary deviations to update the diagnosis. To complete the diagnosis, the system reports a list of potential faults that have been statically associated with each primary node.

![Figure 11. Heat exchanger system and digraph](image)
Uleirich and Powers [84] have proposed a variant of this approach, where the diagnostic model is a fault tree that has been semi-mechanically generated from a system digraph. Let us use again the example of the heat exchanger, this time to illustrate the fault tree synthesis algorithm. The heat exchanger system is an open loop system. In a digraph like this, which does not contain closed control loops, the fault tree will have only or gates [46]. This is a valid generalisation which reflects the fact that open loop systems do not have control mechanisms that can compensate for disturbances in the input variables. Any single disturbance is, therefore, likely to propagate and cause the deviation that represents the top event of the fault tree.

To construct a fault tree for the top event $T_2(+1)$ we will first visit the nodes that affect $T_2$ to determine the deviations that can cause the top event. These deviations {$T_1(+1), M_1(+1), T_3(+1), M_3(-1)$} form the next level of the fault tree and they are connected to the top event with an OR gate. If a node is independent (e.g. $T_1$), the corresponding event {$T_1(+1)$} is a primary deviation and becomes a terminal node in the fault tree. When a node is dependent, the search continues in a recursive way until all nodes connected to a dependent node are themselves independent. Figure 12 illustrates the fault tree synthesised for the heat exchanger by application of this algorithm. It can be noticed that nodes in this fault tree contain abstract qualitative descriptions. For the diagnosis though, it is essential to replace at least some of these descriptions with expressions that can be monitored. The solution proposed by Uleirich is to attach to each root node of the fault tree an AND gate which determines the verification conditions for this event. Fault diagnosis is then accomplished via on-line monitoring of the minimal cut-sets derived from this tree. This method has been demonstrated on a single closed control loop of a mixed-heat exchanger system.

We must point out that the simple recursive algorithm that we have described can only be applied on systems without control loops. Lambert has analysed the digraph of a generic negative feedback loop system and derived a generic fault tree operator for the synthesis of fault trees for systems with such closed loops [46]. Lapp and Powers have also developed algorithms for digraphs with negative feedback and negative feed-forward loops. The synthesis of fault trees from digraphs has demonstrated benefits from formalising, systematising and automating the development of fault trees. However the application of this method in fault diagnosis presents many difficulties, mainly because the system under examination should be modelled so as to fall into one of the categories for which a synthesis algorithm has been developed.

A traditional limitation of the digraph approach had been the difficulty in developing digraphs for large process plants. To overcome this limitation, Chung [85] has proposed a method for the automatic construction of large plant digraphs from smaller digraphs of basic plant units. In this work, the plant digraph is synthesised from the topology of the plant and a library of models describing common plant units such as valves, pipes, pumps and vessels [86-87].

A number of other variations of the basic digraph approach have been proposed to enhance the expressiveness of the model and extend its application in complex systems. Kramer and Palowitch, for example, have developed an algorithm to convert a digraph into a concise set of production rules [88]. Such rules can be integrated with heuristic rules to achieve a more flexible and potentially powerful scheme for fault diagnosis.

![Figure 12. Automatically generated fault tree for heat exchanger](image-url)
Guarro et al propose an extended digraph notation called the *logic flowgraph* [78]. The notation maintains the basic units of nodes and edges to represent process variables and their dependencies, but introduces an extended set of symbols and representation rules to achieve a greater degree of flexibility and modelling capability. Continuous and binary state *variables*, for example, are represented using different symbols. *Logical gates* and *condition edges* can also be used to show how logical combinations of conditions can affect the network of variables. Yau, Guarro and Apostolakis have recently demonstrated the use of *dynamic flowgraphs* as part of their methodology for the development of a flight control system for the Titan II Space Launch Vehicle [79, 89].

The Model-Integrated Diagnostic Analysis System (MIDAS) [80] uses another type of qualitative causal graph, the *event graph*, to perform monitoring, validation of measurements, primary detection of anomalies like violations of quantitative constraints and diagnostic reasoning. In the event graph, nodes still represent qualitative states of system parameters. However, edges do not simply represent the directional change between qualitative states. They are indicative of events and can depict relationships between different states of the same variable (*intravariabe* links) or relationships between states of different variables (*intervariable* links). In MIDAS, such event graphs can be built semi-automatically from system digraphs. Although the general problem of diagnosis of multiple faults remains intractable, the system can diagnose multiple faults provided that each fault influences a different set of measurements [83]. Thus, it overcomes the hypothesis of a single fault present in the process, which has seriously limited the application of traditional digraph-based approaches to diagnosis [81].

Tarifa and Scenna [90] attempt to address the difficulties arising in the development of large digraphs and the associated problems of efficiency of search algorithms, by proposing a method for structural and functional decomposition of the monitored process. They have reported a successful, in terms of performance and computational cost, application of this method to a multistage flash desalination process plant [91].

Digraph-based approaches and their derivatives are process oriented, in the sense that, they focus on process variables and exploit models of the propagation of disturbances between those variables. Within this framework though, the modelling and safety monitoring of a system with complex control and operating procedures becomes extremely difficult. Indeed, the application of these approaches has been confined to fairly simple automated plant processes. However, we should note that the application of process di-graphs in combination with other methods, statistical techniques for example [92], is still investigated and some interesting results are reported in the literature, new efficient algorithms for the diagnosis of multiple faults [93] for instance. For more information on these developments the reader is referred to the work carried out in the Laboratory for Intelligent Process Systems at Purdue University.

### 4.4 Models of Functional Decomposition

Let us now turn our attention to those diagnostic methods that rely on a model of the normal behaviour of the system. One model that we have already introduced in section 3.7 is the Goal Tree Success Tree (GTST). A GTST depicts the hierarchical decomposition of a high-level safety objective into sets of lower-level safety goals and logical combinations of conditions that guarantee the accomplishment of these goals. The typical structure of a GTST is illustrated in Figure 13.

GTSTs are logical trees which contain only *AND* and *OR* gates. Simple value-propagation algorithms can therefore be used by a monitor to update nodes of the tree that are not observable and to provide alarms for the critical functions that the system fails to maintain. On the other hand, simple search algorithms can be applied on the tree structure to locate the causes of functional failures. GOTRES [94], for example, incorporates an inference engine which can perform a depth first search on the tree to determine the reasons why a goal has failed.

---

3 Information on this work can currently be found in *www.lips.ecn.purdue.edu*. This work is also linked to the *Abnormal Situation Management Joint Research and Development Consortium* led by Honeywell, formed in 1992 to develop technologies that would help plant operators in dealing with abnormal situations. Beyond Honeywell, the consortium also comprises Purdue and Ohio State Universities as well as a number of major oil and petrochemical companies.
An advancement of GDTRES is FAX [18], an expert system which uses a probabilistic approach and a breadth first search algorithm to update belief values in the tree structure. The system is, thus, able to decide which sub-goal is more likely to have failed and should be further investigated when the process indicates that a parent goal has failed. FORMENTOR [95] is another real-time expert system for risk prevention in hazardous applications which uses value propagation and search algorithms on a GTST to deliver monitoring and diagnostic functions. The GTST has been successfully applied in a number of realistically complex pilot applications in the nuclear [18], space [40] and petrochemical industries [96-97].

An alternative model of functional decomposition is the Multilevel Flow Model (MFM). Like the GTST, the MFM is a normative model of the system [98], a representation of what the system is supposed to do (system goals) and the functions that the system employs to reach these goals. The MFM defines three basic types: goals, functions, and components. The model is developed as a network of goals which are supported by a number of system functions which in turn are supported by components as Figure 14 illustrates. The notation defines a rich vocabulary of basic mass, energy and information flow functions and their semantics. Beyond this decomposition of goals and functions, the notation also allows the description of the functional structure of the system. Each function in the hierarchy can be represented as a network of connected flow sub-functions.

Larsson has described an algorithm which produces a “backward chaining style” of diagnosis using MFM models [19]. Triggered by a failure of a goal, the monitor hypothesises failures of functions, examines the networks of flow sub-functions for potential candidates and ultimately arrives to diagnoses of component malfunctions. Larson argues that the MFM is particularly suitable for diagnosis with hard real-time constraints. By exploiting the hierarchy of goals and functions it is possible to produce a rough diagnosis quickly and then use any additional time to localise the fault with more precision.
4.5 Diagnosis from First Principles

A number of diagnostic methods have adopted an approach to diagnosis known as ‘diagnosis from first principles’. Such methods use a model of the system structure and normal behaviour of its components to make predictions about the state of various system parameters. Diagnosis here is seen as a process of interaction between prediction and observation of the physical system, which is driven by discrepancies between predicted values and actual measurements. The theoretical foundation of this approach is Reiter’s [99] original diagnosis algorithm and its corrected version [100]. Reiter uses a model of system structure and component behaviour to derive a description of the system expressed in logic. Together with a set of observations of the system, this description provides the basis for computing a minimal set of possible hypotheses about component malfunctions that can explain observed deviations from the predicted (normal) behaviour.

Davis and Hamscher [65] identify three stages in model-based diagnosis: hypothesis generation, hypothesis testing and hypothesis discrimination. Given a discrepancy, the first step in the diagnostic process is to identify which components might have misbehaved and caused that discrepancy (hypothesis generation). These components are called suspects. In the second stage of the process (hypothesis testing), we can eliminate a number of suspects, by hypothesising their failure and checking in the model if this condition is consistent with other observations of the system. Finally, if more than one hypothesis survives testing, we need to determine what additional observations of the system would be required to allow further discrimination among the remaining suspects (hypothesis discrimination).

In generating hypotheses, one can use the structure of the system or properties of components to constrain the number of suspects. Consider, for example, the model of an electronic circuit, which consists of three multipliers and two adders (Figure 15), and assume that an incorrect output (e.g. $F=10$) has been observed. In this type of model one can safely assume that to be a suspect a component must be connected to the deviation. Application of this rule in this example reduces the suspects to ADD-1, MULT-1 and MULT-2.

We can further reduce the number of suspects, by exploiting measurements that indicate more than one discrepancy between the model and the system. If the observations were $F=10$ and $G=10$, for example, then searching backwards from F would have generated the set of suspects [ADD-1, MULT-1, MULT-2], while tracing back the dependencies from G would have generated the set [ADD-2, MULT-2, MULT-3]. Assuming a single point of failure, we can locate the suspects in the intersection between the two sets (MULT-2). This scheme is easily expanded to deal with multiple failures, where in the general case each hypothesis would be a set of components that could possibly account for the observed anomalies [65].

Let us now turn to the second stage of the diagnostic process and examine hypothesis testing. There are two main approaches to this task: constraint suspension and assumption-based truth maintenance systems. In constraint suspension [101], the behaviour of each component is modelled as a set of constraints. These constraints define all the possible relationships between the inputs and outputs of the device, or stated more generally, the inferences that can be drawn about the state of one terminal if we know the state of the other terminals. Figure 16, for example, illustrates an adder and the constraints that describe the normal behaviour of this component.

![Figure 15. Multiplier and adder circuit [65]](image-url)
By modelling each component in this way, we effectively transform the model of the system into a network of constraints where we can propagate values in any direction, backwards or forwards, using relationships between component inputs and outputs. Assuming that we can supply the network with a set of measurements, we can then use the network to predict the state of other non-observable terminals within the model. *An important property of this process is that when some of the input measurements are abnormal the predicted values are inconsistent.* These inconsistencies arise as the predictive engine propagates both normal and abnormal values and, inevitably, fires from different directions constraints which generate different values for the same input or output terminal.

In hypothesis testing we assume that the behaviour of the suspect (normal or faulty) is unknown. This is represented by a suspension of the suspect’s constraints [65]. We then supply the current observations of the system and let the rest of the network propagate values. If at the end of the propagation process the network is still inconsistent, even with the suspects constraints suspended, this indicates that the problem lies somewhere else, in other words, that a component other than the (suspended) suspect is misbehaving. In those circumstances, the current hypothesis is rejected. If, however the network is consistent, we can safely conclude that the component was the source of the original inconsistency. Hence, the current hypothesis has survived testing, and the component remains a suspect. The more observations that we supply to the network, the more hypotheses are rejected and the diagnosis becomes more precise.

Some systems such as GDE (General Diagnostic Engine) [102] provide a single mechanism for hypothesis generation and testing. The underlying technology in GDE is an *assumption-based truth maintenance system* [103], a predictive engine that propagates not only values, but also the assumptions that these values carry with them. Since all values are generated from a model of correct behaviour, the assumption underlying each value is that all components involved in the calculation of the value are functioning correctly. To illustrate this let us take the example [104] of the standard adder-multiplier circuit illustrated in Figure 17.

By propagating the values given for the inputs and output F, a truth maintenance system will determine that either \( X=6 \) with the assumption that \( \text{MULT}-1 \) is working or \( X=4 \) with the assumption that \( \text{MULT}-2 \) and \( \text{ADD}-1 \) are working. The conflicting values for \( X \), though, imply that at least one of the three components implicated in the assumptions must be malfunctioning.

In GDE, whenever such *conflicts* are encountered, the system generates a set of suspects by taking the union of the assumptions underlying the conflicting predictions. The diagnosis is achieved, as the propagation of constraints generates more conflicts, and the sets of assumptions are then intersected to identify suspects that can explain all the conflicts. This mechanism can generate diagnoses of both single and multiple faults. GDE has been tested successfully in many examples in the domain of troubleshooting digital circuits [104].

![Figure 16. A model of the behaviour of an adder](image)

![Figure 17. Assumption based Truth Maintenance System [104](image)
Another system that performs “diagnosis from first principles” is KATE [105], a model-based system developed for NASA’s Systems Autonomy Program at Kennedy Space Centre. KATE has been used for monitoring and troubleshooting of simple electromechanical systems. However, the system is not suitable for diagnosis of dynamic systems with complex behaviour [12]. This fact is a reflection of the general problems of diagnosis from first principles. Constraint propagators, for example, rely on snapshot measurements and do not permit temporal reasoning, something essential for the analysis of disturbances that exhibit complex and variable symptoms. There are several other difficulties with this approach, for example, difficulties in representing and reasoning about complex behaviour and issues concerning the adequacy and efficiency of predictive engines. For a discussion of these problems the reader is referred to [19,65,66].

### 4.6 Qualitative Simulation

One possible approach to monitoring and diagnosis of dynamic, continuous systems is via qualitative modelling and simulation [106]. A continuous system is typically represented in control theory as a set of quantitative, first order differential equations.

In qualitative modelling, this representation is transformed into a set of qualitative constraints, which in Kuipers’ work are expressed in the language of a tool called QSIM [106]. Some of the constraints in QSIM represent equalities: for example, \( \text{MULT}(k,x,y) \) is directly equivalent to \( y=ka \), and \( \text{DERIV}(f_x,f_y) \) is equivalent to \( f_x = f_y / dt \). Constraints though can also represent other qualitative relationships between physical parameters: \( M'(x,y) \), for example, indicates that \( y \) is monotonically increasing with \( x \). Figure 18 illustrates how a quantitative differential equation is converted into a set of such constraints.

QSIM incorporates an algorithm which can simulate a system represented as a set of qualitative constraints. Given the description of the system and a set of initial states, the algorithm can predict qualitative values (increasing, stable, decreasing) and quantitative ranges (e.g. [3.4..5.6]) for the immediate successor state of each parameter. MIMIC [107] is a system which uses the predictive power of QSIM for fault diagnosis of dynamic devices. Triggered by an anomalous observation of the physical system, MIMIC first generates a set of fault hypotheses. For each hypothesis, MIMIC evokes and initialises a QSIM model of the system under the current hypothesis. If the observations of the system match the predictions of the model, the model and the corresponding hypotheses are retained. Otherwise, they are discarded.

One problem with this approach is that the number of QSIM models that need to be specified \textit{a priori} is equal to a potentially large number of plausible faults. An additional drawback is that unanticipated faults are not modelled and, therefore, go undetected [108]. To overcome those problems, Ng [109] has proposed a modified version of Reiter’s algorithm which integrates qualitative simulation with the theory of diagnosis from first principles. Ng retains Reiter’s model of structure and component behaviour as a basis for the representation of the system. However, the normal behaviour of each component is now specified as a set of qualitative constraints and, hence, the constraint propagators used by Davis and deKleer are replaced by the predictive engine of QSIM. Ng has demonstrated his approach on three small processes: a proportional temperature controller, a pressure regulator and a toaster.

\[
\frac{d^2 u}{dt^2} - \frac{du}{dt} + \arctan ku = 0
\]

\[
\begin{align*}
  f_1 &= \frac{du}{dt} \\
  f_2 &= \frac{df_x}{dt} \\
  f_3 &= ku \\
  f_4 &= \arctan f_3 \\
  f_5 &= f_2 + f_4 = 0
\end{align*}
\]

\[
\begin{align*}
  &\text{DERIV}(u, f_1) \\
  &\text{DERIV}(f_x, f_2) \\
  &\text{MULT}(k, u, f_3) \\
  &\text{DERIV}(f_x, f_4) \\
  &\text{ADD}(f_2, f_4, f_5)
\end{align*}
\]

**Figure 18.** Transformation of a differential equation into a set of qualitative constraints [106]
4.7 Data driven approaches to fault detection and diagnosis

In the preceding sections we have shown that one way to construct a predictive model of normal behaviour for diagnostic purposes is from in-depth knowledge about the system and its processes. An alternative way to construct such a model is from empirical data collected during the operation of the system. In that case the reconstructed model of the process should fit the collected historical data to a function that relates different parameters of that process, output parameters for example to input parameters and control settings. The theoretical advantage of this approach is that the development of the predictive model does not require deep knowledge of the system. The challenge of course here is to define an appropriate fitting function that generates accurate predictions.

In univariate data-driven monitoring methods, the fitting function usually relates one dependent variable (e.g. output parameter) to a number of independent variables (e.g. input parameters). The output of the model here is the value of the dependent variable. In simple systems, the fitting function can be a single-order linear polynomial with co-efficients that correspond to the empirical data. In more complex systems, though, higher order non-linear functions may be required. In such systems, therefore, it is often difficult to arrive at an accurate model. An additional limitation of univariate modelling is that the quality of the prediction of the dependent variable is usually strongly contingent to the quality of the measurements of the independent variables.

Univariate methods are now being replaced by more powerful multivariate approaches in which statistical methods are applied on sets of historical data to create models that can more accurately predict the values of both dependent and independent variables in a process. Principal component analysis [110] and partial least squares [111] represent the principal examples of such techniques. A vast body of literature reports on how discrepancies between the values predicted by those techniques and actual parameter values can be used in real-time to determine and locate possible disturbances in plant processes.

Another approach to data-driven fault detection is that of qualitative trend analysis [112]. There are two steps in this approach. The first is the identification of trends in measurements and the second is the interpretation of trends into fault scenarios. Qualitative parameter trends can be constructed from a series of primitives, each of which describes the qualitative state of a parameter in a window of time. Dash and Venkatasubramanian [113] define five such primitives: parameter value stable, increasing with an increasing rate, increasing with a decreasing rate, decreasing with an increasing rate and decreasing with a decreasing rate. In qualitative trend analysis, a sequence of such primitives forms trends that can then be related to fault scenarios. The primary identification of primitives is achieved by simple calculation of first and second derivatives or alternatively by using neural networks with the advantage that such networks can learn from examples and tolerate noise. The qualitative abstraction of data achieved in this approach coupled with techniques for the recognition of trend patterns can provide a way for dealing with large amounts of process data [114].

5. Failure control and Correction

Failure control and correction is the final and probably most decisive stage of safety monitoring, where operators (or an automated system) have to take actions that will remove or minimise the effects of failure. Like fault diagnosis, this task is also performed under stressful conditions and requires a wide range of expertise which include at least a functional understanding of the system and a working knowledge of any published remedial procedures.

State of the art aircraft monitors, such as the A320 Electronic Centralised Aircraft Monitor [44] and the Engine Indicating and Crew Alerting System on the Boeing 757 [115], provide some degree of assistance in this task. The design of these systems reflects a common underlying philosophy. For each warning that the system generates, there is an associated corrective procedure. In its simplest form, a corrective procedure is a set of actions that should be performed without further investigation. Such procedures are usually displayed electronically by the monitor. More complex procedures, which define conditional sequences of actions, are represented as decision trees, and they are typically listed in the pilot's Quick Reference Handbook. Figure 19 illustrates a sample page of the McDonald-Douglas MD11 aircraft handbook [2], which describes the procedure for an avionics compartment overhear alert.
Several of the experimental monitors that we have examined in other domains follow similar approaches to failure control and correction. FORMENTOR [95], for example, relates nodes of the monitoring model (GTST) that represent significant alarms to FMEAs that provide corrective measures for these alarms. Varde et al [70], on the other hand, have developed a system which generates corrective measures from event trees. Event trees are typically employed in safety analysis to determine the effect of a succession of responses to an initiating hazardous event.

These approaches work in so far as there are no conflicts between the procedures associated with each alarm or initiating event. Problems arise, though, in cases where multiple failures or propagating disturbances cause symptoms that require conflicting remedies. An example of such a conflict is a condition where a high fuel temperature in an aircraft coincides with an engine stall. The procedure to cure high fuel temperature requires that the thrust on the associated engine is increased to aid cooling, while the corrective measure for an engine stall is to reduce thrust. Hill [116,117] has proposed a model, called the Goal Ordered Search Tree (GOST), that can accommodate such conflicting safety goals, and a method to use such a model for dynamic generation of optimum corrective measures. The GOST forms an aircraft status metric, by means of which the system can assess various procedures and decide which configuration optimises the status metric and therefore system safety. These procedures are optimal in the sense that they maximise system safety by securing as many as possible of the most critical functions.

The GOST is a network of abstract nodes that represent aircraft critical functions and nodes that represent equipment status. Each node has a binary value (1,0), which shows whether the state represented by the node is achieved or not. A second attribute indicates the priority (or criticality) of the function that the node represents. Nodes are linked with directed arrows and logical connectives (and, or) which define how statuses are propagated in the network of nodes. A solid arrow carries the status of the source node, while a dashed arrow propagates the negative of this status. Let us now see how such a
model can be used for dynamic resolution of potential conflicts between safety goals. Figure 20 illustrates part of a simplified GOST for an aircraft [116]. The example shows five safety goals (ALL-ENGINES, ENGINE1-SAFE, ELECTRICS, FUEL-SAFE and DOMESTICS) and their respective priorities (20,40,10,30 and 5). The first failure that we will consider in this example is a high fuel temperature.

This is represented by the node FUEL-HI-TEMP changing value from 0 to 1. The network shows that this condition causes loss of the FUEL-SAFE goal. This problem can be corrected by either disconnecting the inertial drive generator (IDG=0) or switching off the galley which will cause FUEL-ACTIONS=1 and therefore FUEL-SAFE=1. The first procedure will result in loss of ELECTRICS, while the second in loss of DOMESTICS. Because DOMESTICS has a lower priority, the second procedure is selected and the network is reconfigured as illustrated in Figure 21.

![Figure 20. Example GOST in initial configuration [116]](image1)

![Figure 21. The GOST after the first corrective action [116]](image2)

![Figure 22. The GOST after the second corrective action [116]](image3)
The second failure that we will consider in this example is an engine stall (ENGI-STALL=1). As we can see, this condition causes loss of the ENGINE1-SAFE goal. According to the network, to ensure the safety of the engine, the pilot can either switch off the engine (ENGINE-1=0) or reduce the engine thrust below the stall margin (THRUST-LOW=1). Switching the engine off directly denies the goal ALL-ENGINES, while reducing thrust indirectly denies the goal ELECTRICS⁴. The priority of ELECTRICS is lower and, therefore, the second option is chosen and, therefore, THRUST-LOW is selected. GALLEY off is not required any more to maintain FUEL-SAFE and, thus, the galley is switched on to restore the goal DOMESTICS. The new configuration is illustrated in Figure 22.

Although in this example Hill demonstrates a very attractive idea, that of dynamic synthesis of optimum non-conflicting corrective measures, the method has a number of limitations. Firstly, there is an assumption that the statically defined priorities of goals remain valid in all circumstances throughout the flight [117]. Secondly, the method requires the binary representation of equipment statuses and, hence, is unable to generate corrective measures based on analogue settings. This problem brings us to a more general issue concerning the application of the method, that of how to address the difficulties arising from the scale and complexity of a system and how to apply the proposed concept in a realistic environment.

6. Conclusions

In this paper, we developed a framework which defines the problem of safety monitoring in terms of three principal dimensions: the optimisation of low-level status information and alarms, fault diagnosis and fault correction. Within this framework, we have discussed state of the art systems and research prototypes, placing an emphasis on the methodological features of these systems. In the course of this discussion we extended our basic framework by identifying, in each of the three areas of the problem, the main methodological approaches and their derivatives.

Our analysis confirms that safety monitoring is a complex and multifaceted problem. Perhaps more importantly, it draws on previous research on requirements and methods to identify the various aspects of a potentially effective solution to the problem. In Table 1, we summarise a set of properties that our research suggests should characterise an effective safety monitoring scheme. These properties cover a wide range of goals, which span from the early primary detection of the symptoms of a disturbance to the synthesis and provision of non-conflicting corrective measures.

<table>
<thead>
<tr>
<th>Table 1. Properties of an effective safety monitoring scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aspects of a potentially effective scheme for safety monitoring</td>
</tr>
<tr>
<td>early detection of the symptoms of a disturbance</td>
</tr>
<tr>
<td>detection of sensor failures and the effective validation of measurements</td>
</tr>
<tr>
<td>correct interpretation of statuses and alarms in the evolving context of the system operation</td>
</tr>
<tr>
<td>ability to organise and synthesise low level status information and alarms exploiting their temporal, causal, functional and logical relationships</td>
</tr>
<tr>
<td>provision of a hierarchical and functional, as opposed to a purely architectural, view of failure</td>
</tr>
<tr>
<td>timely and unambiguous diagnosis of malfunctions</td>
</tr>
<tr>
<td>the synthesis and provision of non-conflicting corrective procedures.</td>
</tr>
</tbody>
</table>

⁴ ELECTRICS is sacrificed as the IDG is switched off to maintain FUEL-SAFE, that would otherwise have been denied because of THRUST-LOW. Indeed THRUST-LOW causes FUEL-ACTIONS=0 and therefore FUEL-SAFE=0. Thus, to maintain FUEL-SAFE (priority=30) and ALL-ENGINES (priority=20) we sacrifice ELECTRICS (priority=10) switching off the IDG.
We have seen that model-based monitors can address one or more of these requirements by operating on functional, causal, logical, structural and behavioural models of the monitored processes. These monitors are more likely to be consistent, and they provide better diagnostic coverage than “experiential” expert systems, because the model building and validation processes aid the systematic collection and validation of the required knowledge about the monitored process. There are, however, a number of open research issues concerning the model-based approach.

We have seen, for example, that within the model-based approach, there are two classes of diagnostic methods: those that rely on models of the failure behaviour of the system and those that rely on models of normal behaviour. In the past, it has been argued that models of failure behaviour do not perform as well as models of normal behaviour. Davis and Ng [118,108], for instance, claim that a method which relies on a model of normal behaviour can detect and diagnose any fault that will cause a discrepancy between observations of the system and the model predictions. The authors argue that explicit fault models, on the other hand, are likely to be incomplete or carry incorrect assumptions about how the system may fail, in which case unanticipated or incorrectly anticipated faults will go undetected.

A counter-argument here is that a model (indeed, any model including models of normal behaviour) is nothing but a set of assumptions, which suggests that normal models may also be incorrect, incomplete or insufficiently detailed. In the context of monitoring though, such inappropriate models will also generate wrong predictions and in turn cause false and misleading alarms. This observation simply highlights an important truth, that the quality of monitoring and the inferences drawn by model-based monitors is always strongly dependent on the completeness and correctness of the underlying model. Formal verification techniques, as well as simulation and testing under failure conditions can, of course, be applied to increase our confidence on the underlying model. The validation of the monitoring model, however, is clearly an issue that creates much wider scope for further research.

Another problem with model-based approaches is the difficulty in scaling up to large systems or systems with complex behaviour. Indeed, large scale often makes it difficult to achieve representative and consistent models of the monitored system. Simplifying assumptions can usually help to reduce the complexity of the monitoring model. Such assumptions though must be made with care so that they do not compromise the detection and diagnostic ability of the monitor. In addition, there are problems in representing and reasoning about complex behaviour and difficulties concerning the efficiency of some of the current predictive programs and diagnostic algorithms.

Research is under way to address these issues and improve existing model-based methods. The recent plant wide applications of fault propagation models, for example, are particularly encouraging, and demonstrate the real possibility of transforming fault propagation models from offline tools for understanding risk to operational tools for controlling this risk in real time [119]. At the same time, the fact that there are still many monitoring and diagnostic problems which are too hard to describe using current modelling technology highlights the significance of approaches that employ expert diagnostic knowledge, statistical analysis of data or machine learning, and suggests that further exploration of the potential for combining these approaches with model-based methods would be particularly useful in the future.

**Acknowledgements**

This study was carried out in the context of a European Commission funded project called SETTA (IST contract number 10043). We would like to thank the anonymous reviewers for pointing out omissions and for generally helping us with their comments to improve the initial version of this paper.

**References**