

A Guide to Fault Detection and Diagnosis

Overview and Basic Terminology

This guide to fault detection and fault diagnosis is a work in progress. It will evolve over time, especially based on input from the LinkedIn group [Fault Detection and Diagnosis](#).

Fault detection and diagnosis is a key component of many operations management automation systems.

A "fault" is another word for a problem. A "root cause" fault is a fundamental, underlying problem that may lead to other problems and observable symptoms. (It might not be directly observable). A root cause is also generally associated with procedures for repair.

A "fault" or "problem" does not have to be the result of a complete failure of a piece of equipment, or even involve specific hardware. For instance, a problem might be defined as non-optimal operation or off-spec product. In a process plant, root causes of non-optimal operation might be hardware failures, but problems might also be caused by poor choice of operating targets, poor feedstock quality, poor controller tuning, partial loss of catalyst activity, buildup of coke, low steam system pressure, sensor calibration errors, or human error. A fault may be considered a binary variable ("OK" vs. "failed"), or there may be a numerical "extent", such as the amount of a leak or a measure of inefficiency.

A symptom is an observed event or variable value, needed to detect and isolate faults. If a symptom is the response to a question or an on-demand data request (when actively testing a system instead of just passively monitoring it), it is referred to as a test or test result.

Fault detection is recognizing that a problem has occurred, even if you don't yet know the root cause. Faults may be detected by a variety of quantitative or qualitative means. This includes many of the multivariable, model-based approaches discussed later. It also includes simple, traditional techniques for single variables, such as alarms based on high, low, or deviation limits for process variables or rates of change; Statistical Process Control (SPC) measures; and summary alarms generated by packaged subsystems.

Fault diagnosis is pinpointing one or more root causes of problems, to the point where corrective action can be taken. This is also referred to as "fault isolation", especially when emphasizing the distinction from fault detection. In common, casual usage, "fault diagnosis" often includes fault detection, so "fault isolation" emphasizes the distinction.

Other elements of Operations Management Automation related to diagnosis include the associated system and user interfaces, and workflow (procedural) support to for the overall process. Workflow steps that might be manual or automated include notifications, online instructions, escalation procedures if problems are ignored, fault mitigation actions (what to do while waiting for repairs), direct corrective actions, and steps to return to normal once repairs are complete.

Technical Resources:

- ▶ Fault Diagnosis
- ▶ Data Reconciliation
- ▶ BDAC
- ▶ Process Control
- ▶ CDG Technology
- ▶ CDG for BMC
- ▶ Integrity Technology
- ▶ GDA Technology
- ▶ Neural Nets
- ▶ MES & CIM

Diagnosis Subtopics:

- ▶ Fault Management
- ▶ Model Based
- ▶ Causal Models
- ▶ Causal Time Delays
- ▶ Fault Trees
- ▶ Compiled Models
- ▶ Uncertainty Models
- ▶ Bayesian Models
- ▶ Fault Signatures
- ▶ Neural Nets
- ▶ Procedural
- ▶ Event Oriented
- ▶ Tests
- ▶ Rules
- ▶ Multiple Faults
- ▶ Filtering
- ▶ Novel Faults
- ▶ Projects

Automated fault detection and diagnosis depends heavily on input from sensors or derived measures of performance. In many applications, such as those in the process industries, sensor failures are among the most common equipment failures. So a major focus in those industries has to be on recognizing sensor problems as well as process problems. Distinguishing between sensor problems and process problems is a major issue in these applications. Our usage of the term "sensors" includes process monitoring instrumentation for flow, level, pressure, temperature, power, and so on. In other fields such as network & systems management, it can include other measures such as error rates, CPU utilization, queue lengths, dropped calls, and so on.

In the following material, we focus mainly on online monitoring systems, based on sensor or other automated inputs, but possibly including on some manual input from end users such as plant operators. However, we also consider a broader viewpoint of diagnosis not just as a technology, but also as part of the business process of fault management. Also, diagnosis as a decision support activity rather than a fully automated operation is common in the management of large, complex operations such as those found in the process industries and network & systems management. It is also a natural fit for operations such as call centers for customer support, where a significant amount of diagnosis is done. In those cases, there may be support from a workflow engine as well as the types of tools traditionally associated with fault detection and diagnosis

Fault Management - the overall processes and life cycle of a fault

“Fault management”, “Abnormal Condition Management” (ACM), or the term trademarked by Honeywell (and hence avoided by other vendors): “Abnormal Situation Management” (ASM), all refer to overall processes associated with managing faults, from detection steps through return to normal service. Please go to the following page for a discussion of this “big picture”:

[Fault Management - the overall processes and life cycle of a fault](#)

An overview of different approaches to fault detection and diagnosis

There are many different approaches to fault detection and isolation. Because each has their strengths and weaknesses, most practical applications combine multiple approaches. In this section, we highlight some of the major differentiating factors between the different techniques.

Model based reasoning

One of the major distinctions in approaches to fault detection & diagnosis is whether or not explicit models are used, and what type of models are used. When models of the observed system are used as a basis for fault detection and diagnosis, this is often referred to as "model based reasoning". Please go to the page

[Model Based Reasoning for fault detection and diagnosis](#)

Causal models.

An important special case of model-based reasoning uses causal models. Causal models capture cause/effect information. For a review of causal models, please go to the page:

[Causal Models](#)

In physical systems, causality in reality is associated with some time delay or lags between cause and effect. This has to happen because mass or energy has to move, overcoming resistance by inertial, thermal inertia, inductance, or other physical phenomena.

This is discussed on the page

[Causal time delays and lags](#) .

Fault signatures, pattern recognition, and classifiers

Pattern recognition is a general approach that directly uses the observed symptoms of a problem and compares them to a set of known symptoms for each possible problem, looking for the best match. We can represent the “pattern”, or “fault signature”, as a vector (1 dimensional array) of symptoms for each defined fault. Please go to the following page:

[Fault signatures, pattern recognition, and classifiers](#)

Neural networks

Neural networks are nonlinear, multivariable models built from a set of input/output data. They can be used as event detectors, detecting events and trends. They can also be used as diagnostic models in model-based reasoning, or used directly as classifiers for recognizing fault signatures. For more information, please see [Neural Networks for fault detection and diagnosis](#) .

Procedural/workflow approaches - modeling the decision process rather than the observed system

Some fault detection and diagnosis is handled by creating procedures for making decisions based on the observed data. This is direct modeling of the decision process rather than modeling the system being diagnosed. Please go to the following page:

[Procedural/workflow approaches - modeling the decision process rather than the observed system](#)

Event-oriented fault detection, diagnosis, and correlation

An event represents a change of state of a monitored object. Alarms are examples of events. Diagnostics involving events can be significantly different than diagnostics involving a fixed set of variables. Please go to the following page for a discussion of event-oriented diagnostics and event correlation:

[Event-oriented fault detection, diagnosis, and correlation](#)

Passive system monitoring vs. active testing

In the case of online monitoring systems, many diagnostic techniques assume routine scanning of every variable of interest. But many times, it is preferable to request non-routine tests. Diagnosis for maintenance purposes is based on testing. For a discussion on passive monitoring vs. testing strategies, please see the following page:

[Passive system monitoring vs. active testing](#)

Rule-based approaches and implementations

Rule-based systems in most cases just implement other approaches discussed above, more as a program control mechanism than a separate diagnostic technique. For a discussion on rule-based implementations, please see the following page:

Hybrid approaches

Pattern recognition by itself does not require a model. However, input for construction of the signatures for the known failures may be based on models; for instance, as residuals from models of normal behavior. This general technique applies to static or dynamic models.

For dynamic models, the patterns can be based on predicted measurement values vs. observed values in a Kalman filter, for example. [Smartsignal](#) offers products based on an empirical process model of normal operation used for fault detection, combined with other techniques for fault isolation.

Pattern recognition can also be combined with models of abnormal behavior. For instance, in the case of the SMARTS InCharge product, the modeling was in terms of a fault propagation model (qualitative cause/effect model of abnormal behavior). But as part of the development process, this model was then used to automatically construct fault signatures - a form of compiled knowledge. At run time, diagnosis was based on matching observed data to the nearest fault signature. So the product at run time had the characteristics of a pattern matching solution.

In some cases, a qualitative model really just exists inside the application developers head. That person directly constructs the signatures based on their knowledge. So, the overall methodology is often a combination of pattern recognition with a model-based method.

Some tools such as [GDA](#), are flexible enough to support multiple approaches to fault detection and diagnosis, and also support the upfront filtering and event generation as well.

The summary paper [Experiences Using Knowledge-Based Reasoning in Online Control Systems](#) reported on the results of numerous online fault detection and diagnosis systems. One conclusion was that most applications required a mix of techniques for success.

Examples of hybrid approaches for pipeline monitoring

An example of a hybrid approach that directly uses pattern recognition on the residuals from static numerical models of a liquid pipeline is given in the [neural nets](#) section of this site.

The concept paper [Pipeline Diagnosis Emphasizing Leak Detection: An Approach And Demonstration](#) outlines an approach to pipeline leak detection that combines causal models of abnormal behavior with both static (algebraic) models and dynamic models.

Some issues in evaluating techniques and implementing them

All solutions installed as software will have issues such as the user interfaces for end users and application developers, system integration, reliability, building generic libraries describing commonly repeated components, easy modification for maintenance, computing speed and storage requirements, and so on. But there are some unique issues associated with fault detection and diagnosis.

Single fault assumption vs. multiple faults in practice

In large complex operations (such as the control centers for refineries, or for network management), there are usually multiple outstanding problems. This presents a problem for many diagnostic techniques. If it weren't for some of the partial workarounds discussed next, fault isolation techniques based on a single-fault assumption would be essentially useless in large-scale applications such as process monitoring and network management. Please go to the following page:

[Diagnosis of multiple faults](#)

Robustness: tuning to balance sensitivity vs. false positives

Fault detection and diagnosis generally involves some "tuning". A system might be tuned to be very sensitive to detecting or recognizing particular problems. But then it is also likely to declare problems when none exist. There is a tradeoff in sensitivity vs. false alarms. This is a generalization of the problem encountered when setting simple alarm thresholds. If alarm threshold limits are set too close to normal operations, problems will be caught before they become large. But, minor transient fluctuations will then result in false alarms.

Filtering

For systems using sensor data, noise is almost always an issue. The primary purpose of filtering is to reduce noise by combining multiple values of variables sampled over time so that at least some of the noise is canceled out. For a discussion, on reducing the impact of high frequency noise and "spikes" for single variables, please go to the following section:

[Filtering](#)

Robustness for sensor or model errors

A major consideration when evaluating diagnostic techniques is robustness in the presence of errors in the sensors or the model. Any system that completely believes any symptom or test input will be sensitive to errors in that input. Similarly, any system that completely believes its own models will be sensitive to errors in that model. The result in either case can be conflicting data and incorrect conclusions. The errors might just be due to transient errors in timing, but could be longer lasting.

In some sense, most diagnostic systems make approximations, such as finding the "nearest" fault signature, recognizing that there will be errors. Another way of looking at this is that most systems combine evidence. There will be some form of weighted averaging of the inputs when they are in conflict.

Event orientation and scalable architecture for large systems

Many model-based techniques require complex matrix calculations. This does not scale well for large systems. One approach is simply to break up the large diagnostic system into a set of smaller, independent ones. This results in diagnostic systems that might not account for all possible interactions in a complex system. However, as long as there are enough sensors, this might not be much of a problem.

Another fundamental approach for large-scale systems is to focus on abnormal events, and reason over the events rather than all the underlying data at once. There is a separate event generation step that converts raw data such as temperature values into an event such as a high temperature alarm. SPC (Statistical Process Control) calculations can be used to generate events as well. Event generation could also be based on calculations using complex models of normal operation, signaling deviation of the data from expected behavior. In this way, models of normal behavior can be used to generate events for use in models of abnormal behavior.

This combines the sensitivity of normal models for problem detection with fault isolation capabilities of models of abnormal operations.

In the process industries, events such as simple alarms have always been generated. In recent years, alarm management systems have also looked at the implications of combinations of alarms. In network management, significant effort was put into formulating and standardizing the roles of “agents” that typically generated events, and “managers” that processed and correlated the events. Each piece of equipment has an agent. Histories of numerical data used to generate events are mainly maintained locally by each agent. Fault diagnosis is a byproduct of “event correlation” in the manager processes. Further hierarchical levels resulted in “manager of managers” systems. All of this was an attempt to deal with large numbers of events and problems. Examples of event-oriented systems are described in the section on the [Integrity/Optegrity event-oriented technology](#). Another example of an event-oriented system was the SMARTS InCharge event correlator used in network management (This was bought by EMC and incorporated into their products.).

Event generation is only required in certain diagnostic techniques. There is a loss of information in converting what is commonly numerical information into crisp events that are simply present or absent. But just dealing with events results in simpler diagnostic calculations, so the tradeoff is often made to manage larger systems.

Treatment and propagation of unknown sensor values

Sooner or later, any (hardware) sensor will fail. After failure, its value must be treated as unknown until the sensor is repaired. Sensor values may also become unknown due to wire breaks, power loss, or network problems. The fault detection and diagnosis system must deal with this gracefully, providing the best results possible. It also should not make default assumptions. For instance, the lack of a high temperature event should not be taken to imply that the temperature is OK when the sensor value used to generate that event is unknown. While prior estimates of fault probabilities may be useful in the absence of other information, prior estimates of sensor values can lead to incorrect conclusions.

Model completeness assumptions and “novel faults”

It is difficult to model or even imagine every possible fault scenario. A fault that arises that was not considered during application development is called a “novel fault”. For a discussion of the problems this causes, please go to the page:

[Model completeness assumptions and novel faults](#)

Explanation facilities and visibility of knowledge

Plant operators and their management, like most people, hate “black boxes” that tell them what do, without being able to explain why. In the ideal system, a user should be able to find out why a particular conclusion was reached at the time it was made (either in announcing a fault or the recovery from a fault).

A related issue is visibility of all the knowledge that is built into the system. Users need confidence that all the fault possibilities they know are included, and that there aren’t errors built in that could lead to false conclusions. If the knowledge is visible in some fashion and understandable, people can formally review it, without waiting for faults at run time to see what happens. If the operation of the diagnostic system is safety-critical, such reviews by people with a wide variety of backgrounds are extremely important.

Systems based on fault propagation models, logic trees, or procedural methods like decision trees in a graphical language, can provide good explanations and transparency of the contained knowledge. (Whether they actually do or not depends on the

implementation, but the capability is there). Pattern matching methods based on explicit listing of fault signatures can also provide this transparency -- anyone can look at the faults and the associated symptoms defined in the fault signature. But fault signature matrices get large and harder to understand as the number of faults and symptoms grows. If an underlying model such as a fault propagation model is used to construct the signatures, that model may be more modular and easier to understand.

Methods based on engineering equations are less satisfying, but might be believed because their basis, assumptions, and limitations can at least be explained. Methods based on black box techniques such as statistical or neural net classifiers are the worst from the standpoint of explanation and visibility of knowledge.

Transparency for explanation might not be an issue at all for diagnosis in embedded systems, and there might not even be a direct human user. But transparency of the model knowledge can be important even for embedded diagnostic systems, so that effective reviews can be done.

Project implementation for fault detection and diagnosis

Projects to implement fault detection and diagnosis are similar to other projects for the system being monitored, such as those for process control or IT system management. For typical project implementations, please go to the following page:

[Project implementation for fault detection and diagnosis](#)

Copyright 2010-2020, Greg Stanley

External Links

Overview papers

[Challenges In the Industrial Applications of Fault Diagnostic Systems, Sourabh Dash and Venkat Venkatasubramanian](#)

[A review of process fault detection and diagnosis Part I: Quantitative model-based methods, Venkatasubramanian et al, Computers & Chemical Engineering, 27 \(2003\), pp. 293-311](#)

[A review of process fault detection and diagnosis Part II: Qualitative models and search strategies, Venkatasubramanian et al, Computers & Chemical Engineering, 27 \(2003\), pp. 313-326](#)

[A review of process fault detection and diagnosis Part III: Process history based methods, Venkatasubramanian et al, Computers & Chemical Engineering, 27 \(2003\), pp. 327-346](#)

Books

Simpson, William R., and John W. Sheppard, [System Test and Diagnosis](#), Kluwer Academic Publishers, Boston, Mass., 1994.

Other



Share this page: 