

Fuzzy Logic Based Real-Time Predictive Process Fault Analysis

Richard J. Fickelscherer¹, Douglas H. Lenz¹, Daniel L. Chester²

¹ FALCONEER Technologies LLC, 5820 Main St., Suite 102,
Williamsville, NY 14221

²Department of Computer & Information Sciences, University of Delaware,
Newark, DE 19716

Key Words: Control Technologies, Fuzzy Logic, Artificial Intelligence,
Modeling, Fault Analysis, Sensor Validation

Abstract

Operating processing plants is inherently risky due to potential abnormal situations caused by equipment or sensor failures or out of control process conditions. When serious single or multiple simultaneous faults occur, this situation can directly lead to accidents, releases and injuries. Process Fault Analyzers are computer programs that continuously monitor process sensor data to determine the current operating status of those sensors and the underlying process. Any faults or failures so determined are presented to the process operators in real-time in order to maximize the time available to effectively respond to those problems. If no problems are discovered those sensor measurements are classified as validated.

The fuzzy logic based real-time diagnostic method described here is called the Method of Minimal Evidence. It uses first principle or statistical models, correlations and experiential heuristics to define relationships between particular measured sensor data and assumed unmeasured process variables that describe normal process operation. It continuously evaluates those relationships with real time data to determine which close and which do not. It also exhaustively combines those relationships together to form additional but novel relationships, which are also continuously evaluated. The resulting patterns of these evaluations are interpreted with a general-purpose fuzzy logic diagnostic rule to determine the certainty associated with each of the potential fault hypotheses. This algorithm is general enough to diagnose both single and multiple faults directly. A software product incorporating this method has now been demonstrated to be competent in a world-class industrial chemical manufacturing process owned and operated by FMC.

Copyright 2004 by ISA – The Instrumentation, Systems and Automation Society.

Presented at ISA EXPO 2004; <http://www.isa.org>

Introduction

Process Fault Analyzers are computer programs that can monitor process operations in order to identify the underlying cause(s) of operating problems. A general method for creating process fault analyzers for processing plants has been sought ever since the incorporation of computers into process control (Fickelscherer 1990). The motivation has been the enormous potential for improving process plant operations in terms of safety and productivity. Automated process fault analysis should help process operators 1.) prevent catastrophic operating disasters such as explosions, fires, meltdowns, toxic chemical releases, etc.; 2.) reduce downtime after emergency process shutdowns; 3.) eliminate unnecessary process shutdowns; and 4.) maintain better quality control of the desired process products.

A wide variety of logically viable diagnostic strategies currently exist for automating process fault analysis (Venkatasubramanian, et al. 2003A, 2003B, 2003C, 2003D). However, automated fault analysis is still not widely used within the processing industries. This is mainly due to these following limitations: 1.) prohibitively large development, verification, implementation, or maintenance costs of these programs; 2.) inability to operate a program based upon a given diagnostic strategy continuously on-line or in real-time; and 3.) inability to model process behavior at the desired level of detail, thus leading to unreliable or highly ambiguous diagnoses. Subsequently, an effective method for producing automated process fault analyzers is still being actively sought. This paper presents an algorithm and associated software that overcomes many of these shortcomings.

Evaluating models of normal process operation with current process data is the most promising and powerful means for directly identifying underlying process operating problems. Doing so generates an unimpeachable source from which to logically infer the current state of the process being modeled. Consequently, this was the approach followed in the development of FALCONEER™ IV. It continuously does this model evaluation and then automatically infers the underlying cause(s) for any models that do not close. This inference is actually a fuzzy logic calculation based on our Method of Minimal Evidence (MOME) algorithm and is the heart of our diagnostic rule engine. Performing this inference automatically on-line enables these fault analyzers to perform "intelligent supervision" of the daily operations of their associated process systems.

Method of Minimal Evidence Overview

The **Method of Minimal Evidence (MOME)** is a model based diagnostic strategy for developing optimal automated process fault analyzers (Fickelscherer 1990, 1993). It was derived from our experience in developing *FALCON*, a real-time, on-line process fault analyzer for an commercial-scale Adipic Acid plant that DuPont operates in Victoria, Texas (Rowan 1989, 1992). It provides a uniform framework for examining models of normal process operation and their associated assumptions. When evaluated

Copyright 2004 by ISA – The Instrumentation, Systems and Automation Society.

Presented at ISA EXPO 2004; <http://www.isa.org>

with process data, each process model generates a residual that gives evidence of the actual process state. Significantly large residuals directly indicate the invalidity of one or more of those models' associated assumptions. It is possible to identify faults by comparing the patterns of these residuals with those expected to occur during a fault situation. Sensor data validation occurs when specific faults are shown to be not occurring and is thus just the flip side of the fault analysis. Fault Analyzers based on MOME have highly structured knowledge bases that directly optimize their overall competence along with their diagnostic resolution and sensitivity. MOME can be directly used to diagnose many multiple fault situations, to determine the optimal placement of process sensors to facilitate fault analysis, and to determine the optimal division of a large process system for distributing fault analyzers. The basic logic behind MOME is as follows:

Once a target process system has been selected for automated process sensor validation and fault analysis, as many linearly independent models describing the normal operating behavior of that process system as possible should be derived. These models should be based upon the most fundamental understanding of normal process behavior available, limited only by the specific type and frequency of the process data being collected. The resulting set of models should constitute a highly accurate description of the target process system's normal operating behavior.

All unevaluated process models describing normal process operation can be characterized as functions of the following quantities:

$$0 = f(i) \text{ (model } i \text{ modeling assumption variables, time)} \quad (1)$$

Where modeling assumption variables = specific sensor measurements
or
 standard or extreme values of specific parameters or unmeasured variables

Once the modeling assumption variables are instantiated with actual process sensor data or standard or extreme values of specific parameters or unmeasured variables, a residual results (**i.e, r(i) below**) which is just a function of process sensor noise and any currently occurring modeling assumption variable deviations, e.g.,

$$r(i) = f(i)(\text{sensor noise, modeling assumption variable deviations}) \quad (2)$$

If the **residual (i.e, r(i))** of an evaluated model is **significantly high or low** (significantly higher or lower than zero), then it can be inferred that at least one or more of the possible modeling assumption variable deviations that could cause such a residual value is occurring.

If the **residual** (i.e, $r(i)$) of an evaluated model is **not significant**, then either: (1) there are no modeling assumption variable deviations; (2) one or more such deviations are occurring but at magnitudes or rates of change which are below the sensitivity of that model to discriminate such deviations; or (3) two or more significant assumption variable deviations are interacting in an opposing fashion.

Briefly, the **MOME Strategy** for fault analysis compares patterns of residual behavior expected to occur during the various possible assumption variable deviations with the patterns of those residuals currently present in the process. It uses the minimum unique patterns required for correctly doing this analysis, allowing for many of the possible multiple assumption deviations to be directly identified. This is important because this methodology does not discern only process faults, but all possible process operating events, fault and non-fault events alike. In other words, all assumptions necessary to derive a model are potential diagnoses whether they are about the absence of faults or about some other non-fault event (e.g., process is not at steady state, etc.). The logic is designed to venture diagnoses only if it is highly certain of the underlying problem. This conservative behavior is advantageous because it should not confuse its users with incorrect diagnoses at times when the actual process operating state is in flux.

The **patterns of expected residual behavior** resulting from applying this method (e.g., the **diagnostic rules**) contain the minimum patterns required to diagnose each of the possible fault situations. This method directly maximizes the sensitivity of the fault analyzer for these various faults, maximizes the resolution (discrimination between various possible faults and non-fault events) of that analysis, and optimizes its overall competence when confronted with multiple assumption deviations. The strategy can also be used to directly determine the optimal placement of process sensors for performing fault analysis and the optimal division of large process systems for distributing process fault analyzers.

Method of Minimal Evidence Detailed Description

As described, MOME is based on the evaluation of process models of normal operation. The pattern of residuals that result is interpreted to determine any underlying assumptions that are currently not holding. More generally, such a residual process model may be represented generically as follows:

$$residual = f(x_1, x_2, \dots, x_n) \quad (3)$$

where x_1, x_2, \dots, x_n are process variables, i.e., parameters that define the state of a process at any given moment, and f is a function of those parameters that computes, for example, a balance of energy or mass in a control volume.

Because the sensors that measure process variables may not be 100 percent accurate or provide exact readings; because the process models may not be perfect models of the relationship between the process variables; and because random perturbations may occur, it is empirically observed that the residuals are not always zero, though they are usually close to zero when the process is operating normally. The mathematical model of sensor validation and predictive fault analysis (“SV&PFA”) used in the program requires that all residuals be zero, on average, when a monitored process is normal. Therefore, a calculation is made from historical plant data of the average value of each residual and that average value is subtracted from the corresponding residual process model.

In practice, then, each function f , above, will behave like a statistical random variable having a mean value β and a standard deviation σ . The mean value $\beta = \beta_0 \rho$ and the standard deviation $\sigma = \sigma_0 \rho$, where β_0 and σ_0 are constants and ρ is either 1 or a process variable that is the definitive measure of the production level of the process being monitored. Usually, β is just a constant value, but sometimes it is the product of a constant times a process variable whose value determines the level of production at which the process is operating.

The generic residual process model above can be replaced with a primary residual process model defined as:

$$r = f(x_1, x_2, \dots, x_n) - \beta \quad (4)$$

which has a mean value of zero and a standard deviation of σ . The equation defining r is referred to herein as a primary residual process model. The program examines the values of such (adjusted) residual process models and, among other things, infers from the pattern of deviations from zero which sensors are faulty or which other parts of the process may be faulty.

In more generic terms, if a plant engineer provides the formula $f(\dots)$ as the formula for a residual process model under ideal conditions, and the formula *mean* for the average of $f(\dots)$ over time based on historical plant data, and the formula *sigma* for the standard deviation of $f(\dots)$ over time, the program generates the primary residual process model:

$$r = f(\dots) - \text{mean} \quad (5)$$

which has the property that the average of r is expected to be zero. These are preferred forms for these formulas, but any formula that can be expressed in the mathematical language of FALCONEER™ IV is allowed. The formula *sigma* is not used in the primary residual process models, but is used to calculate certainty factors, as discussed below.

Primary residual process models are distinguished from certain linearly dependent residual process models automatically generated by our program. Such additional models

are referred herein to as secondary residual process models, and are computed as follows: Suppose that r_1 and r_2 are primary residual process models and both contain a common variable v . If both residual models are linear functions of v , they may be combined algebraically to remove the terms containing v . The standard deviation for this secondary residual process model can be directly computed from its two parent model standard deviations.

The MOME algorithm computes certainty factors to identify faults and/or validate underlying assumptions. As used below, a “fault” is a pair consisting of a process variable v and a direction d , and is designated as $\langle v, d \rangle$. The value of d can be either *high* or *low*, which are in turn defined by

$$high = 1 \quad (6)$$

$$low = -1 \quad (7)$$

Any number of different functions for computing certainty factors from residuals may be used. A commonly used function, the Gaussian function, is defined as:

$$Gauss(x, sigma) = e^{-(x/sigma)^2/2} \quad (8)$$

where $sigma$ is the standard deviation of the process variable x .

When this program monitors a process, it reads real-time sensor data, computes the associated primary and/or secondary residual values and their standard deviations, and then calculates three certainty factors for each residual value, as needed. Let r be one of the residual values and let $sigma$ be its standard deviation. Residual r is expected to be zero, but often it is not. If it is only a little off from zero, the program considers it to be satisfactory, but the farther away from zero it gets, there exists less confidence that it is satisfactory. Using the Gaussian function for example, the certainty factor for r being satisfactory is represented in this embodiment as:

$$cf(r, sat) = Gauss(r, sigma) \quad (9)$$

If r is much greater than zero, it is considered to be high, that is, higher than it is supposed to be. The certainty factor for r being high is represented as:

$$cf(r, high) = 1 - Gauss(r, sigma) \text{ if } r > 0 \\ = 0 \text{ otherwise} \quad (10)$$

Similarly, if r is much less than zero, it is considered to be low. The certainty factor for r being low is represented as:

$$cf(r, low) = 1 - Gauss(r, sigma) \text{ if } r < 0 \\ = 0 \text{ otherwise} \quad (11)$$

As stated above, for purposes of calculating certainty factors for faults, $\langle v, d \rangle$ signifies a fault; that is, v is a process variable and d is a direction, either *high* or *low*. To compute the certainty factor that a fault is present, the certainty factors for the primary and/or secondary residuals are examined to find evidence for the fault. If r is a residual, r provides evidence for fault $\langle v, d \rangle$ when it has deviated from zero in a direction that is consistent with variable v deviating in the direction d . For example, if $\partial r/\partial v$ is greater than zero, then both v and r can be expected to go high (or low) at the same time. If, however, $\partial r/\partial v$ is less than zero, then v and r can deviate in opposite directions. The certainty factor for r in the appropriate direction is then the strength to which r can provide evidence for the fault. One strong piece of evidence for the fault is enough to strongly conclude that the fault is present, unless there is also strong evidence that it is not present.

The evidence for fault $\langle v, d \rangle$ is this set of certainty factors for all relevant residuals:

$$evidence\text{-for}\text{-fault}(\langle v, d \rangle) = \{cf(r, sign(\partial r/\partial v)d) \mid (\partial r/\partial v) \neq 0 \text{ and } r \text{ is a primary residual}\} \quad (12)$$

In some applications, the certainty factor for any residual, primary or secondary, may be included in this set. The strength of the evidence for the fault is the maximum of the values in this set.

Similarly, if a residual deviates in the opposite direction from what is expected when the fault is present, that deviation is evidence against the fault being present. The evidence against fault $\langle v, d \rangle$ is this set of certainty factors for all relevant residuals:

$$evidence\text{-against}\text{-fault}(\langle v, d \rangle) = \{cf(r, -sign(\partial r/\partial v)d) \mid (\partial r/\partial v) \neq 0\} \quad (13)$$

Certainty factors for both primary and secondary residuals may be included in this set. The strength of the evidence against the fault is the maximum of the values in this set. If that value is subtracted from one, the strength to which this evidence is consistent with the fault being present is determined.

An additional consideration is significant in evaluating a certainty factor for a fault. Some residuals are not functions of v and so are not expected to deviate from zero when the fault $\langle v, d \rangle$ is present. The secondary residual process model that was formed by eliminating v from two primary residual process models is such a residual. It is relevant to evaluating the presence of the fault, so this secondary residual is expected to have a high certainty factor of being satisfactory when the fault involves v . Also, if two primary residual process models were combined to generate a secondary residual process model by eliminating some variable other than v , and one of these primary residuals is a function of v but the other is not, it is expected that the primary residual that is not a function of v is satisfactory. This primary residual is considered relevant to the fault as well.

Some primary residual process models may not be functions of v and are not combined with any models that are. These are considered to be not relevant to the fault $\langle v, d \rangle$. Another fault can be present and cause them to deviate from zero, but this will not affect the assessment for fault $\langle v, d \rangle$. This allows a diagnosis of the presence of several single faults that happen not to interact with each other. In addition, r may be a function of v , but at the moment, $(\partial r / \partial v) = 0$. The neutral-evidence for fault $\langle v, d \rangle$ is this set of certainty factors for all relevant residuals:

$$\text{neutral-evidence}(\langle v, d \rangle) = \{cf(r, sat) \mid r \text{ is relevant as neutral-evidence for } v\} \quad (14)$$

The strength of this evidence is the minimum of the set because if any one of the residuals that are supposed to be satisfactory is in fact high or low, that weakens the evidence for the fault $\langle v, d \rangle$.

The certainty factors in these three sets, *evidence-for-fault*, *neutral-evidence*, and *evidence-against-fault*, are considered as fuzzy logic values, and are combined using a common interpretation of fuzzy “AND” as the minimum function, fuzzy “OR” as the maximum function, and fuzzy “NOT” as the complement function (1 minus the value of its argument). For finite sets, the quantifier “SOME” is just the “OR” of the values in the set, so it is equivalent to taking the maximum of the set. Similarly, for finite sets, the quantifier “ALL” is just the “AND” of all the values in the set, so it is equivalent to taking the minimum of the set. Putting this all together, the certainty factor for fault $\langle v, d \rangle$ is defined in this embodiment as:

$$\begin{aligned} cf(\langle v, d \rangle) = & \text{SOME}(\text{evidence-for-fault}(\langle v, d \rangle)) \text{ AND} \\ & \text{ALL}(\text{neutral-evidence}(\langle v, d \rangle)) \text{ AND} \\ & \text{NOT}(\text{SOME}(\text{evidence-against-fault}(\langle v, d \rangle))) \end{aligned} \quad (15)$$

If this value is above a threshold, it is displayed as a possible fault.

Regarding the display of a fault $\langle v, d \rangle$: If v is a measured variable, the sensor value for that variable was substituted for the variable in computing all the primary and secondary residual values. If $d = \text{high}$, a conclusion is drawn that the sensor reading is higher than the true value for that process variable. If $d = \text{low}$, a conclusion is drawn that the sensor reading is lower than the true value for that process variable. In either case, a conclusion is drawn that the sensor is at fault. If $cf(\langle v, d \rangle)$ is about zero for both cases, $d = \text{high}$ and $d = \text{low}$, then the sensor reading has been validated.

In the case of an unmeasured variable v , such as a leak, a high certainty factor for $\langle v, \text{low} \rangle$ means that the assumed value of v , which can be viewed as the reading from a virtual sensor, is low compared to the actual value. In order to display a conclusion about the actual value of the unmeasured variable, the program displays a message that v is high in this case (*i.e.*, there is a positive leak out of the process). Similarly, if the certainty factor for $\langle v, \text{high} \rangle$ is high, it displays a message about v being low. If neither of these

cases apply, a conclusion is drawn that the real value of v is about equal to its assumed value.

The fuzzy logic rule above may be generalized to sets of faults by redefining what counts as evidence for the set, evidence against the set, and what counts as neutral-evidence. An inference may be drawn that a set of faults is present when no subset of them may be inferred to be present. In particular, this means there must be at least one residual value for each fault deviating in the direction that the fault can cause. This leads to the following general fuzzy rule of this methodology: Let

$$fault\text{-}set = \{ \langle v_1, d_1 \rangle, \dots, \langle v_n, d_n \rangle \} \quad (16)$$

Then

$$cf(fault\text{-}set) = SOME(evidence\text{-}for\text{-}fault(\langle v_1, d_1 \rangle)) AND \quad (17)$$

$$\vdots$$

$$\vdots$$

$$SOME(evidence\text{-}for\text{-}fault(\langle v_n, d_n \rangle)) AND$$

$$ALL(neutral\text{-}evidence(fault\text{-}set)) AND$$

$$NOT(SOME(evidence\text{-}against(fault\text{-}set)))$$

The component evidence sets are defined as follows:

- The set $evidence\text{-}for\text{-}fault(\langle v_i, d_i \rangle)$ is the set of $cf(r, sign(\partial r/\partial v_i)d_i)$ such that r is a primary residual and $(\partial r/\partial v_i) \neq 0$. It is the same set that was used for single faults.
- The set $neutral\text{-}evidence(fault\text{-}set)$ is the set of $cf(r, sat)$ such that r is relevant to one or more of the faults in $fault\text{-}set$ as neutral-evidence and such that $(\partial r/\partial v) = 0$ for all the variables v in $fault\text{-}set$.
- The set $evidence\text{-}against(fault\text{-}set)$ is the set of $cf(r, -sign(\partial r/\partial v)d)$ such that $(\partial r/\partial v) \neq 0$ for at least one variable v in $fault\text{-}set$ provided that $-sign(\partial r/\partial v)d$ has the same value for all such variables v in $fault\text{-}set$. (In other words, all the faults that can influence r must influence it in the same direction; if two faults can influence r to deviate in opposite directions, we can learn nothing about $fault\text{-}set$ from that residual.)

The generalized rule may be limited to compute only certainty factors for pairs of faults.

Conclusion

Using models to perform data validation and fault analysis in real time proves the assertion that, “**Models are the means by which data can be converted to meaningful information**” (Kramer & Mah 1994). These models are based on a fundamental understanding of normal operating behavior of the given process system. They thus generate an unimpeachable source of information for logically inferring conclusions

about the process being modeled. Automatically performing this inference after each update of process sensor data allows such programs to continuously perform “intelligent supervision” of the daily operations of their associated process systems. Due to their nature (i.e., these programs expect normal operating behavior), all excursions from normal operating behavior will be scrutinized fully to uncover all plausible explanations for such behavior. The resulting fault analyzer is thus a “diligent watchdog” of its associated process system.

FALCONEER™ IV automatically generates the SV&PFA diagnostic rules based on MOME directly from the underlying Primary models of normal process operation. All of the development and maintenance effort to create and upkeep such fault analyzers can now be directed at the derivation and improvement of the Primary models of normal process operation (the declarative knowledge) in the database. Since all the Primary models are linearly independent, they can be added, improved or deleted as need be and then the entire program recompiled to include these improvements. Consequently, with FALCONEER™ IV, the fault analyzer can be incrementally improved with minimal effort as the process system’s operating behavior becomes better understood or the process system changes, allowing these programs to easily evolve along with its associated process system. It thus makes these programs almost self-creating and self-maintaining.

Consequently, the diagnostic rule engine eliminates the need to explicitly derive patterns of model satisfactions and violations expected to occur when the associated modeling assumption variable deviation occurs: all that is required is to derive all possible Primary models and determine and classify all the various modeling assumption variables associated with those models. This greatly simplifies the effort required to implement and maintain these fault analyzers in practice (Fickelscherer, et al. 2003). In the case of FMC’s Tonawanda, New York Electrolytic Sodium Persulfate plant (Figure 1), it was the difference between maintaining 24 Primary Models, 101 Secondary Models (i.e., all possible linearly dependent models derived from those Primary Models), and 1050 single fault diagnostic rules (altogether requiring approximately 18,000 lines of Visual Basic code to implement) and just maintaining the 24 Primary Models (requiring approximately 2000 database entries), over a two orders of magnitude reduction in code to create and maintain, not even counting the multiple fault analysis also being done automatically. This innovation allows these programs to be implemented and maintained by the same process engineers responsible for the daily operations of their processes.

References

Fickelscherer, R. J., "Automated Process Fault Analysis," Ph.D. Thesis, University of Delaware, Newark, DE, 1990.

Fickelscherer, R. J., "A Generalized Approach to Model-based Process Fault Analysis", Proceedings of the 2nd International Conference on Foundations of Computer-Aided Process Operations, ed. By D. W. T. Rippin, J. C. Hale, and J. F. Davis, Austin, TX, CACHE, 1993, pp. 451-456.

Fickelscherer, R. J., Lenz, D. H. and Chester, D. L., "Intelligent Process Supervision via Automated Data Validation and Fault analysis: Results of Actual CPI Applications", AIChE 2003 Spring National Meeting, New Orleans, LA, April 2003.

Kramer, M. A. , "Malfunction Diagnosis Using Quantitative Models and Non-Boolean Reasoning in Expert Systems," AIChE Journal 33, 1987, pp. 130-147.

Kramer, M. A. and Mah, R. S. H., "Model-Based Monitoring", Foundations of Computer-Aided Process Operations II, ed. by D. W .T. Rippin, J. C. Hale and J. F. Davis, Austin, TX, CACHE, Inc., 1994, pp. 45-68.

Rowan, D. A. and Taylor, R. J., "On-Line Fault Diagnosis: *FALCON* Project," Artificial Intelligence Handbook, Instrument Society of America 1989, Vol. 2, pp. 379-399

Rowan, D. A., "Beyond *FALCON*: Industrial Applications of Knowledge-Based Systems," Proceedings of the International Federation of Automatic Control Symposium, ed. by P.S. Dhurati, Newark, Delaware, USA, 1992, pp. 215-217

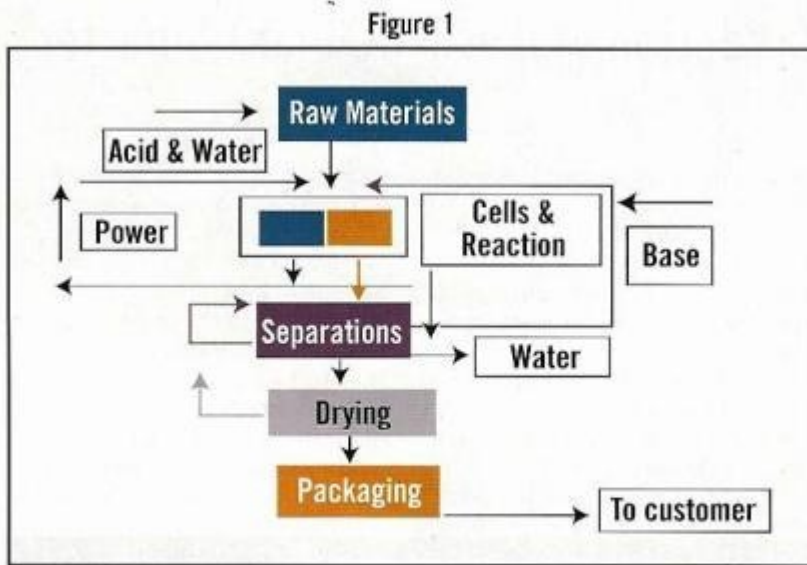
Venkatasubramanian, V., Rengaswamy, R., Yin, K., and Kavuri, S. N., "A Review of Process Fault Detection and Diagnosis Part 1: Quantitative Model Based Methods", Computers and Chemical Engineering, Vol. 27, 2003A, pp. 293-311.

Venkatasubramanian, V., Rengaswamy, R., Yin, K., and Kavuri, S. N., "A Review of Process Fault Detection and Diagnosis Part 2: Qualitative Models and Search Strategies", Computers and Chemical Engineering, Vol. 27, 2003B, pp. 313-326.

Venkatasubramanian, V., Rengaswamy, R., Kavuri, S. N., and Yin, K., "A Review of Process Fault Detection and Diagnosis Part 3: Process History Based Methods", Computers and Chemical Engineering, Vol. 27, 2003C, pp. 327-346.

Venkatasubramanian, V., "A Review of Process Fault Detection and Diagnosis: Past, Present and Future" submitted to Computers and Chemical Engineering, 2003D.

FIGURE 1. ELECTROLYTIC SODIUM PERSULFATE PROCESS



The process that FMC is optimizing, using a Honeywell DCS and Falconeer sensor validation software, is outlined here.

Source: Falconeer LLC