
The Space Congress® Proceedings

1988 (25th) Heritage - Dedication - Vision

Apr 1st, 8:00 AM

# Real-Time Fault Management for Large-Scale Systems

H. Biglari
*Boeing Aerospace Company P.O. Box 1470 Huntsville, Alabama 35807*

C. Cheng
*School of Electrical Engineering Georgia Institute of Technology Atlanta, Georgia 30332*

G. Vachtsevanos
*School of Electrical Engineering Georgia Institute of Technology Atlanta, Georgia 30332*

# Real-Time Fault Management for Large-Scale Systems

H. Biglari

Boeing Aerospace Company
P.O. Box 1470
Huntsville, Alabama 35807

C. Cheng, G. Vachtsevanos

School of Electrical Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332

## Abstract

Apriori knowledge of failure modes of a system is an indispensable information for design of robust decentralized hierarchical control schemes. In particular inclusion of system faults as part of the process under control provides greater flexibility for self diagnosis and maintenance of real-time systems. By assigning discrete states to the process under control, an "artificial consciousness" can be created within the controller which allows the controller to exercise selective actions for each given discrete state. This concept has been implemented to control the utility systems of the Space Station Laboratory Simulator.

## Introduction

Although in many situations "a stitch in time saves nine", it is an underestimate when it comes to spacecraft fault management. It is well known that a prompt fault management for a spacecraft saves the lives of the onboard crew members. Case in point is the dramatic 1985 explosion of the space shuttle Challenger which indeed was preventable if proper fault management techniques were exercised. Complex and life critical large scale systems (and space station is no exception) need not generate myriad of alarms and expect the operator or the crew members to acknowledge them on a timely manner to prevent potential failures. What really is needed is a systematic fault detection, isolation and recovery scheme.

This paper attempts to describe a methodology for fault management, which could not have been implemented without the recent advances in hardware and software technology. Although the concept of state machines [2] is very mature and it has been successfully utilized for design of digital computers, only recently researchers are beginning to apply the technique directly to process control (i.e no conversion of state diagrams to sequential codes by a programmer, which itself results in software faults and thus defeats the purpose).

The environment that this state machine can be realized is described next.

## ICON Based Application Generator

The advent of microcomputer technology along with new developments in computer languages have given rise to a number of highly sophisticated intelligent controllers and work stations that can assist the control engineer in the design of control strategies with virtually no direct intervention from the software engineers [7]. Although, general purpose conventional computer languages such as Fortran, Pascal, C, Ada, etc. have been evolved to serve a number of disciplines, it has become evident that each discipline has its own peculiarities. In particular the sequential nature of conventional programming languages does not easily lend themself to development of control algorithms. These facts have led the computer scientists to develop computer languages that are more

suitable for domain specific problems. The domain of the problem being specific allows the computer scientist to design the system to be data driven. A very familiar example of a data driven system is the spread sheet which has found enormous popularity among nonprogrammers.

The Application Generator provides an ICON based environment where the control engineer can graphically represent his control strategies and processes under control. Furthermore, the control engineer is capable of testing and executing his control algorithms without the intervention of programmers. This apparent decoupling of two disciplines results in higher productivity and drastic reduction of cascading faults due to hardware/software and process interactions.

The Application Generator performs a "knowledge capture" in this domain specific environment and converts the result to a global database which is used by the kernel residing in the controller (kernel is a program that does not change even if the process under control is modified). Steps for this " knowledge capture" are presented next

### Representation Of Process Under Control

It is clear that the process under control must be described to the intelligent controller before the controller takes any control action. This knowledge includes, logical partitioning of the process under control into a number of subprocesses, sensor and actuator relative locations within a subprocess, sensor and actuator characteristics, and process variable limits. For greater efficiency and productivity, ICONs are used extensively to generate the graphical representation of the process under control. Figure 1 shows the logical partitioning of lab module simulator subsystems.

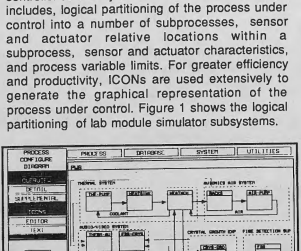


Figure 1 Subprocesses of Space Station Lab Module Simulator

### State Programming Concept

Within the context of the Application Generator , the state programming is a graphical declarative knowledge representation which allows the control engineer to designate directly discrete states to each subprocess within the overall process under control. In addition, the control engineer defines the corresponding transition conditions. Thus, both the normal states and the abnormal states of a subprocess are declared by the process control engineer which gives an "artificial consciousness" to the overall system. This awareness enables the controller to exercise selective actions for each given discrete state. Furthermore, the concept is naturally suitable for fail-safe realization of systems with stringent reliability requirements.

### Representation Of Process control strategies

The control strategies, whether logical or analytical, must be defined for each state of a given subprocess. For example, due to certain failures, the system goes into an emergency state, which may call for different order observers and a different control law to avoid further propagation of failures. ICON based graphical editors drastically reduce the huge task of data entry for this phase of development.

### The Potable Water System - An Example

This example illustrates the application of the Fault Detection and Isolation (FDI) algorithm for the Potable Water System (PWS) of the Space Station using a prototype domain specific Application Generator. On this expert process control system four types of faults (leaky pipe, stuck valve, pump failure, and breached filter) have been considered. These are crucial faults which affect either the quality or the quantity of the potable water reclaimed on the Space Station. An Hp-9836 desktop computer and HP3497 data acquisition system were used to simulate the PWS as well as selected fault conditions which are, in turn, monitored by the workstation.

The PWS is designed to reclaim potable water from waste water which is produced by the crew members and the station operation. Figure 2 depicts the Space Station PWS. It is composed of two input waste water accumulators, one pump module, five particle filters, one carbon filter, one cation resin bed, one anion resin bed, two microbials, and two output potable water storage

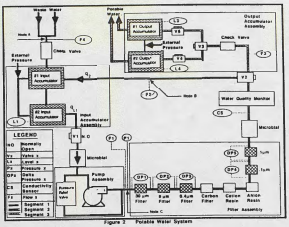


Figure 1 Potable Water System

accumulators. A Water Quality Monitor (WQM) samples the water and provides information as to the quality of treated water.

The output of the WQM can be simply a binary signal indicating acceptable or unacceptable quality according to a set of values for water quality standards. The WQM system can provide useful information for component failure detection purposes. The component might be the microbial, carbon filter, or ion resin exchanger.

The PWS is operating in two modes : Mode 1 is the filling mode when water quality is acceptable , while Mode 2 is called the recirculating mode which occurs when water quality is not acceptable. In addition the PWS has two configurations: Configuration 1 uses the first output accumulator as the water source while the second accumulator is being filled. Configuration 2 uses the first accumulator for filling and the second as water source. The decision for mode switching is a function of the signal provided by the WQM and conductivity sensors, while the decision for configuration switching is a function of the level of the output accumulators. A full complement of flow, level, pressure, and conductivity sensors provide information about the state of the PWS to the intelligent controller and ultimately to the central processor. Since Mode 1 and Mode 2 can be operational only one at a time, the state equations for both modes and both configurations can be combined by introducing two switches: Switch 1 ($S_1$) and Switch 2 ($S_2$). The switches modify the components of the A matrix of the state space model. The definition of the switches and the state space representation of the PWS model is given as:

Analog state vector:

$$X^T = [x_1, x_2, \ldots, x_6]^T \quad (1)$$

$$U^T = [u_1, u_2, u_3]^T \quad (2)$$

The discrete states are defined by two switches:

$$S = [S_1, S_2] \quad (3)$$

Where:

$$S_1 = \begin{cases} 1 \rightarrow \text{Mode 1 (water quality ok)} \\ 0 \rightarrow \text{Mode 2 (recycle)} \end{cases} \quad (4)$$

$$S_2 = \begin{cases} 1 \rightarrow \text{Output accumulator 1 is being depleted} \\ 0 \rightarrow \text{Output accumulator 1 is being filled} \end{cases} \quad (5)$$

The state space model of the PWS is defined by:

$$\dot{X} = A X + B U \quad (6)$$

Where:

$$A = \begin{bmatrix} \frac{-R}{I_f} & 0 & 0 & 0 & \frac{1}{I_f} & 0 & 0 \\ \frac{S_1}{2A} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{S_1 S_2}{A} & 0 & 0 & 0 & 0 & 0 & 0 \\ -1+\frac{R\tau}{I_f} & 0 & 0 & 0 & \frac{-\tau}{I_f} & 0 & 0 \\ 0 & 0 & 0 & \frac{K_p}{\tau_p} & \frac{-1}{\tau_p} & 0 & 0 \\ S_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{S_1(1-S_2)}{A} & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & \frac{1}{2A} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{-(1-S_2)}{A} & 0 & 0 & 0 & \frac{-S_2}{A} \end{bmatrix}^T$$

Where:

q : Flow rate of the system
$I_f$ : Inertia of the fluid mass

**PWS Fault Detection Schemes**

The goals of the Intelligent Control (IC) design are to develop a fault detection and isolation scheme which: (1) maximizes the sensitivity to component failure , and (2) minimizes the rate of the failure detection-false alarms. Usually these two design goals involve conflicting criteria and the IC design is called upon to optimize the tradeoff.

The proposed methodology exploits a combination of signal validation techniques based upon a parity space representation, analytic redundancy, and artificial intelligence [1]. These tools guarantee its sensitivity to subtle failure modes that are transparent to most techniques.

Since the prototype Application Generator used provides only for single-input single-output and two-inputs single-output mathematical ICONS, It is impossible to manipulate array arithmetic operations and time series analysis using the particular configuration. The proposed parity space techniques usually involve a large number of operations and cannot be applied in this case due to the limitations imposed by the present state of Application Generator technology.

The fault decision logic is simply a binary type and the Intelligent Controller produces a status report detailing the fault type and its location through a color display and a speech synthesizer. The controller finally decides as to what appropriate control action must be taken in order to isolate the fault and avoid a catastrophic event.

For purposes of illustrating the adopted methodology, let us consider one of the critical PWS fault conditions, that of a leaky pipe segment. It is obvious that a leaky pipe may cause severe damage to vital station components, as well as bring about a reduction in the available water resource. Among basic assumptions we consider that all sensor data have been validated with only simple faults to be detected at any time. Furthermore, the design of the Intelligent Controller is based upon a knowledge of the noise bias characteristics of the sensors and the noise and error propagation behavior of the analytic models.

An analytic measurement is expressed as some functional dependence on values of variables, which are obtained from process measurements. The latter are always associated with measurement uncertainties and bias errors. In general, an analytic measurement may be expressed as $f(x_1, x_2, ..., x_n)$. If $\varepsilon = [\varepsilon^1, \varepsilon^2, ..., \varepsilon^n]$ represents the error bound vector corresponding to the direct set of measurements $\{x_1, x_2, ..., x_n\}$, then , under worst case conditions, the analytic estimate of the error threshold may be obtained as:

$$\varepsilon_f = \varepsilon^1 \left|\partial f/\partial x_1\right| + \varepsilon^2 \left|\partial f/\partial x_2\right| + ... + \varepsilon^n \left|\partial f/\partial x_n\right| = \sum_{i=1}^{n} \varepsilon^i \left|\partial f/\partial x_i\right| \quad (7)$$

Or simply:

$$\varepsilon_f = \langle \varepsilon, \nabla f \rangle \quad (8)$$

Digital processing of data is performed by both the Global and the Intelligent Controller, while the actual data to the actuators and from the sensors are analog. Analog signals must be converted to digital form during the sampling process. Because of the finite word length inherent in digital signal processing, the quantized errors introduced by the Analog to Digital conversion process can be expressed by the following equations:

$$S_q = (0.5)^{k+1} \times 100\% \quad (9)$$

where:

$S_q$ : Relative quantization error

$k$ : Number of bits of the A/D device

In our particular implementation $k$ was 11 bits which implies a quantization error of less than .03% which was an order of magnitude smaller than sensor and actuator accuracies, and therefore can be ignored without any significant impact on the overall error threshold due to combined sensor/actuator accuracies.

The use of error theshold for the purpose of leak detection is described next.

**Leak Detection for Pipe Segment 1**

Figure 2 shows pipe segment 1 which is represented by solid lines and cut set {A, B, C}. In this segment F2 is a flow sensor for $q_2$, F1 is a flow sensor for $q_1$ and F4 represents $q_4$. A mass conservation principle dictates that a leak occurs when there is a discrepancy between the net

| | |
|---|---|
| R | : Resistance of the system |
| P | : Pressure raised by the pump |
| A | : Area of the accumulator |
| $h_{in}$ | : Level of the input accumulator |
| $h_{out}$ | : Level of the output accumulator |
| $h_{use}$ | : Level of the user accumulator |

Input vector U is :

| | |
|---|---|
| $u_1 = q_{in}$ | : Flow rate of input waste water |
| $u_2 = q_{ref}$ | : Reference flow |
| $u_3 = q_{use}$ | : Flow rate of water being used |

And the states are defined as:

$x_1 = q$

$x_2 = h_{in}$

$x_3 = h_{out}$

$x_4 = V$ (Pump motor voltage)

$x_5 = \Delta P$

$x_6 = Q$ ; $(Q = q)$

$x_7 = h_{use}$

The remaining coefficients are constant process parameters. Since the transition matrix $\Phi(A)$ is a function of $S_1$ and $S_2$ , any change of state by these switches will require rapid matrix inversion (by Pade approximation of order (1,1) or namely bilinear transformation), other multi-step techniques such as (Runge Kutta) also require inordinate amount of CPU time which exceeds the computer processing power. Therefore, the modified Euler algorithm with variable integration time step was adopted to propagate the state equation in real-time using an HP9836 computer.

Figure 3 depicts the control structure for the simulation studies. The function of each block is described below.

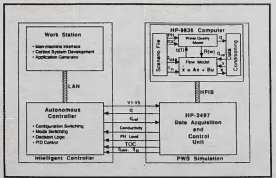


Figure 3 Control Structure for Real-Time Simulation Studies

**(1) Intelligent Controller:**

The Intelligent Controller is designed to perform the following three tasks:

- Sensor data management
- Sensor data validation
- Fault detection and identification

The Intelligent Controller monitors the system and provides diagnostic messages to the maintenance crew when a fault is detected. Also the controller constantly minimizes the errors between the process variables and the commanded set points.

**(2) Global Controller:**

The Global Controller (the workstation computer) receives the down linked signals from the Intelligent Controller and monitors the system. Furthermore, it performs the following tasks:

- Generates set points which are transmitted to the Intelligent Controller
- Monitors the system states
- Provides Man-Machine Interface

The Global Controller communicates with the Intelligent Controller and other auxiliary processors thru a Local Area Network.

**(3) Flow Model:**

The flow model simulates the dynamics of the PWS. The model receives information about system resistance from the water quality model. The state space representation of this model is given by Equation (3) which is propagated in real-time using the HP9836 and is transmitted to the HP3497 data acquisition system and finally to the intelligent controller.

**(4) Water Quality Model (Particle Model):**

The water quality model simulates the flow resistance of the PWS. Since the nominal system flow rate is very small (in the order of micro (meter cube)/sec), we may assume that the flow resistance of the system is concentrated at the particle filters. The resistance presented by piping and other components is so small that it can be neglected.

inflow and the net outflow (the discrepancy should be greater than the error threshold). Then, for segment 1, we have:

Inflows:
$q_4$ : Direct measurement of the flow sensor F4
$q_2$ : Direct measurement of the flow sensor F2

Net inlfow:

$$Q_{NI1} = q_2 + q_4 \quad (10)$$

Out flows:

q : Direct measurement of the flow sensor F1
$q_{L1}$ : Analytic measurement of level sensor L1

$$q_{L1} = (2A)(dL1/dt) \quad (11)$$

Where
A = Area of one input accumulator

Net outflow:

$$q_{NO1} = q + q_{L1} \quad (12)$$

Applying the conservation law to find discrepancy:

$$q_{d1} = |Q_{NI1} - q_{NO1}| = |q_2 + q_4 - q - q_{L1}| \quad (13)$$

The measurement error associated with the above equations can be derived as the maximum error criterion. For maximum flow , $q_i$ is the nominal flow, i.e. 1.82 litters/hour, then:

$$\Delta q_{d1}\ 1.82 \times 0.1 \times 4 = 0.0073 \text{ litters/hour} \quad (14)$$

As it is pointed out earlier, the uncertainty analysis was not implemented due to the limitations imposed by the prototype Application Generator. For binary decision without false alarming, the measurement error (reasonable tolerance level) is taken as 0.15 litters/hour (from the simulation results) which is considered to be the threshold for all leaky pipes.

Now we can state the decision rule for fault in segment 1:

If $q_{d1} > 0.15$ litters/hour, then pipe segment 1 is leaky

If $q_{d1} < 0.15$ litters/hour, then pipe segment 1 is not leaky

Similar procedures are used to define faults in other pipe segments. Typical FDI simulation results are shown in Figure 4. The fault detection algorithm monitors the system at all times through the sensor data management system. The fault simulation module is triggered by the specific fault introduced from the keyboard. Since the PWS is a series configuration, the only way to isolate a leaky pipe segment is to stop the pump and terminate, temporarily, system operation.

Actual implementation is described next.

**Implementation**

The commercial Application Generator used was an ICON-1000 a product of Data Acquisition System of Boston Mass . The work station of this intelligent control system uses an IBM-XT with enhanced touch sensitive display, while the controllers are based on Motorola 68000.

The overall process to be controlled is shown in Figure 1. The PWS is a subsystem within the overall lab module simulator and it is logically partitioned into 5 subprocesses. These

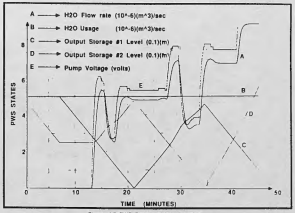

Figure 4.0 PWS State variable evolution

subprocess are grouped inside the dashed box labeled PWS. Due to space limitations only the OUT_ACCU (output accumulators) subprocess will be described here. This subprocess is expanded in Figure 5. Similarly all of the remaining 15 subprocess are expanded, (but are not shown here). The expanded subprocesses show the component interaction and sensor-actuator location. The corresponding state diagram for each subprocess is entered next. Figure 6 depicts the states of the OUt-ACCU subprocess. The MAINTAIN state is fail-safe state where the operator is notified about the nature of the problem. The transition definition from NORMAL to MAINTAIN is shown in Figure 7.

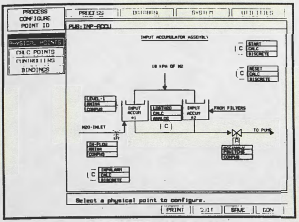

Figure 5 INP-ACCU Subprocess Expnaded

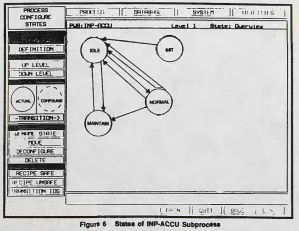

Figure 6 States of INP-ACCU Subprocess

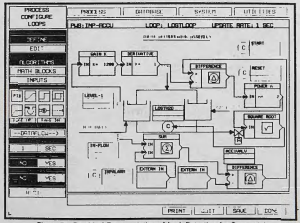

Figure 7 Graphical Representation of Leak Detection for Segment 1

## Concluding Remarks

The Intelligent Controller described in this paper uses a simplified fault detection and isolatoin algorithm in conjunction with an ICON-based environment to represent a subsystem - in this case the Potable Water System of the Space Station's common module - the control strategies associated with the subsystem as well as the fault management techniques. The utility of the approach is demonstrated with the PWS example. Obvious advantage of the Application Generator include increased design flexibility and reliability. The study has also revealed some disadvantages of existing technology referring primarily to the limited availability of tools for representing large-scale dynamic systems which, in turn, limits considerably the application domain. Future efforts must focus on the development of more powerfull state machines capable of expanding their domain of applications within the specific domain of process control.

## REFERENCES


[1] G. Vactsevanos and K. Davey, "Fault Diagnostics for the Space Station Thermal Control System Using a Hybrid Analytical/Intelligent Approach", Proc. of 1987 IEEE symposium on Intelligent Control, Philadelphia, Pennsylvania, Jan 1987.

[2] F. C. Hennie, Finite State Models for Logical Machines, John Wiley, (1968).

[3] J. D. Erickson, "Manned Spacecraft Automation and Robotics", PROCEEDING OF THE IEEE, VOL 75, NO. 3, MARCH 1987, pp. 417-426.

[4] A. K. Mok, "Fundamental Design Problems of Distributed Systems for the Hard-Real-Time Environment", Ph.D Dissertation, MIT, May 1983.

[5] E. J. Henley, H. Kumamoto, Designing for Reliability and Safety Control, Prentice Hall, (1985).

[6] P. K. Lala, Fault Tolerant & Fault Testable Hardware Design, Prentice Hall, (1985).

[7] M. Rabb, "New Personal Computer Graphics Software Drives Process Control Strategy", Control Engineering, January 1987, pp. 92-94.

[8] H. Morris, "Design Station Uses AI for Factory Cell Control", Control Engineering, September 1987, pp. 116-117.