

NASA/TM—1998-208804



Model-Based Diagnosis in a Power Distribution Test-Bed

E. Scarl
The Boeing Company, Huntsville, Alabama

K. McCall
Marshall Space Flight Center, Marshall Space Flight Center, Alabama

National Aeronautics and
Space Administration

Marshall Space Flight Center

September 1998

Available from:

NASA Center for AeroSpace Information
800 Elkridge Landing Road
Linthicum Heights, MD 21090-2934
(301) 621-0390

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161
(703) 487-4650

TABLE OF CONTENTS

I.	INTRODUCTION	1
II.	MODEL-BASED DIAGNOSIS	2
III.	RODON	3
	A. Constraint Satisfaction	3
	B. Rodon's Interval Arithmetic	4
	C. The Rodon Graphical Interface	4
	D. Constraint Propagation for Diagnosis	5
	E. Behavior Suspension in Rodon	6
	F. Hierarchical and Dynamic System Representation in Rodon	6
	G. Solutions to Nonlinear and Simultaneous-Equation Constraints in Rodon	7
	H. New Rodon Release Announced	8
	I. Fault Recording for Continued Monitoring Past Failure	8
IV.	SSM/PMAD—A POWER DISTRIBUTION TEST-BED	9
	A. NASA's SSM/PMAD	9
	B. Structure of SSM/PMAD	9
	C. Modeling SSM/PMAD	9
	D. A Comment on Single-Fault Modeling and Debugging	12
	E. Test Fault Scenarios	12
V.	OTHER COMMENTS ON RODON	14
	A. Documentation	14
	B. Constraint Solver	14
	C. Debugging Facility	14
	D. Preparation Program	15
	E. Graphical Interface	15
	F. System Hierarchy	15
VI.	CONCLUSIONS	17
	REFERENCES	19

UNUSUAL TERMS

Associational diagnosis—a diagnosis based on a set of rules rather than a system model.

Computational compaction—in a model, an entire higher-level system is represented as a single function that is somehow more compact than the composition of all its lower-level functions.

Constraint suspension—the removal of one or more constraints from a constraint network, replacing each with a logical “true.”

Disjunct—a term in a logical “or” expression.

Multidirectional network—a constraint network in which values can be propagated both from an object’s inputs to outputs and from its outputs to inputs.

Overcurrent—a condition in an electrical power system when current exceeds the maximum allowed.

Undervoltage—a condition in an electrical power system when voltage falls below the minimum allowed.

NONSTANDARD ABBREVIATIONS

GDE	general diagnostic engine
GUI	graphical user interface
LEO	low-Earth orbit
MSFC	Marshall Space Flight Center
PDCU	power distribution and control unit
PMAD	power management and distribution
PP	preparation program
RDE	R.O.S.E. diagnostic engine
R.O.S.E.	Reasoning Over Systems in their Entirety
RPC	remote power controller
SSM	Space Station Module

TECHNICAL MEMORANDUM

MODEL-BASED DIAGNOSIS IN A POWER DISTRIBUTION TEST-BED

I. INTRODUCTION

NASA missions are vulnerable to hardware failure. In low-Earth orbit (LEO), physical retrieval and repair have been demonstrated, although such missions are expensive, infrequent, and often dramatic. Beyond LEO, direct access is generally not possible, and all diagnostic and recovery operations have been achieved through telemetry. NASA has maintained high levels of interest in automated fault detection, isolation and recovery, and has sponsored considerable innovative effort to develop powerful and robust maintenance aids. One of the most promising directions, called "model-based diagnosis," has been sponsored by NASA at the Kennedy Space Center¹ and its other Field Centers, and has been under development at industrial and academic laboratories for over a decade.

Similarly, Boeing's principal business is the design and manufacture of aerospace platforms that must operate autonomously in distant and isolated environments with high reliability. Boeing's manufacturing facilities are also increasingly automated and, even though shop floor machinery is not isolated from maintenance facilities, there is increasing dependence on larger, more complex and interdependent systems which make downtime increasingly expensive.

Several commercial ventures promoting diagnostic systems, including those using model-based reasoning, have not been successful in the marketplace. At this time, we are aware of only one commercially available system featuring model-based diagnosis: It is called "rodon" and is being developed and sold by Reasoning Over Systems in their Entirety (R.O.S.E.) GmbH in Heidenheim, Germany. (Note: The vendors often spell "rodon" in all lower case and with a Greek delta in place of the "d" since the word rodon means "rose" in Greek. We will write "Rodon" in this report.)

Both NASA and Boeing have an ongoing interest in tracking new commercial model-based technology to prevent or resolve failure in autonomous systems. A joint evaluation effort was therefore established as a Space Act Agreement to determine the capabilities and promise of a particular product and thereby the current capability of commercially available model-based reasoning.

II. MODEL-BASED DIAGNOSIS

The term “model-based” may be unfortunate, since virtually all types of diagnostic reasoning use something that might be considered a model. In diagnosis, the term “model-based” means there is at least a simulation model that can predict the target system’s normal behavior. Stochastic and discrete-event types of models are excluded because their purpose is to predict behavior over a set of representative inputs and environments; they do not necessarily foretell what the system’s behavior will be in response to some specific environmental input vector. The model-based diagnosis has been established as a field of investigation for over a decade.²⁻⁴

The “direct simulation” models used in a model-based diagnosis explicitly represent important components and subsystems: the way these are connected (structure), and the way their parameters’ values are determined by their inputs and internal states (behavior). With such models, one can match expectations against performance, both quantitatively and qualitatively. The advantages of such behavioral models have been understood for some time^{5, 6} and include:

- Easy accommodation to design changes
- Ready diagnosis of sensor failures
- The ability to operate with or without explicit fault models
- The ability to diagnose unpredicted types of failures
- The ability to continue uninterrupted operational diagnosis after such failures
- The potential to support other needs such as system analysis, documentation, and the automation of fault recovery and incremental redesign.

Unfortunately, the ability to produce generally suitable modeling environments has been more elusive than expected over the last 10 to 15 yrs. Several systems have been developed under industrial and NASA sponsorship and in academia, but commercial shells have generally not been available. There have been difficulties in modeling complex component behaviors and complex interactions between behaviors. Sometimes component behaviors may be unknown or unpredictable to desirable levels of accuracy. Even when sufficient models are available, we have not had access to environments allowing satisfactory speed.

This report summarizes the experience with one commercial shell in its application to the Space Station Module/Power Management and Distribution system (SSM/PMAD) hardware and software tested at the Marshall Space Flight Center (MSFC).

III. RODON

The Rodon shell is being developed by a startup corporation called R.O.S.E. Although it is based in Heidenheim, Germany, under the direction of Dr. Werner Seibold, it is supported in the United States by American-based scientists and engineers.

A. Constraint Satisfaction

The core of Rodon is a constraint satisfaction engine that determines component networks' behavior. For each class of components, a behavior is specified as an algebraic equation. The component's inputs and outputs, their associated units, and any named constants, are defined along with any number of equations which relate these items. A Rodon definition for a model of a two-input multiplier might be:

Constants:	None
Inputs:	A = volts B = volts
Outputs:	C = volts FAULT-STATE = integer
Normal Behavior:	FAULT-STATE = 0 and $A * B = C$
Fault Behavior:	FAULT-STATE = 1 and $A * B < C$, or FAULT-STATE = 2 and $A * B > C$, or FAULT-STATE = 3 and $A * B \neq C$, and ($A = 0$, or $B = 0$, or $C = 0$).

There are several points worth noting. The expression " $A * B = C$ " is an equation, not an assignment statement, as in a procedural language like C or FORTRAN. It could just as well have been written " $A = C / B$," or " $A * B / C = 1$." In fact, the primary effect of declaring "A" to be an input rather than an output is merely the graphical representation for this multiplier will draw the port for "A" on the left rather than the right side of its icon.

"FAULT-STATE" is just another variable. Since there are no constraints on it other than equating it to a different value in different disjuncts, the value of FAULT-STATE serves as a marker for which disjuncts succeed. Notice that disjuncts need not be mutually exclusive. Lexical order is unimportant, and more than one disjunct may succeed.

The power of this constraint propagation engine was evident when we examined the problem of extracting flag bits from an integer word in the SSM/PMAD data. R.O.S.E. personnel suggested the following function:

$$\text{WORD} = A0 + 2*A1 + 4*A2 + 8*A3 + 16*A4 + 32*A5 + 64*A6 + 128*A7 + 256*A8 ,$$

where WORD is the only “output” and each of the nine “inputs” was constrained only to be Boolean ($=\{[0] [1]\}$). Given WORD as an integer in [0 255], simulation quickly sets the inputs A0 through A8 to the correct binary representation of WORD’s value! Of course, Rodon’s constraint solver has no algorithmic knowledge of binary/decimal conversion, but it did assign correct values to all 10 ports given the value of only 1 of them. We did not actually use this method for flag bit extraction, which was more conveniently integrated into NASA’s C-language interface code. However we were impressed by Rodon’s capacity to use a general purpose constraint solver to effectively perform a job normally reserved for a specialized algorithm.

B. Rodon’s Interval Arithmetic

Rodon effectively operates on a single data type—sets of numeric intervals of arbitrary precision. This turns out to be quite powerful, with intervals being used to express data types, operating ranges, and uncertainties. For example, the expression:

$$A = \{[- 5] [6 8.32] [10 20[]20 +]\} \text{ and } \dots$$

means that A is constrained to have a value that is either less than or equal to 5, between 6 and 8.32 inclusive, or any number greater than 10 except for the number 20. Note that the negative and positive signs denote negative and positive infinity, respectively.

This release of Rodon is based on constraint suspension as described below. Behaviors can be suspended manually by the user, or automatically by the diagnostic process. While constraint functions are restricted to be linear, there are facilities for automated piecewise linearization and interpolation. Iterative search mechanisms supplement the constraint solver when local propagation fails.

C. The Rodon Graphical Interface

A graphical user interface (GUI) allows graphical system construction. Component instances are created from their generic types in a class hierarchy. These appear as rectangles, created large enough to accommodate names and port tabs for their inputs and outputs, and may be labeled with a graphical icon. The interface seems adequate, although it seems fair to say that the Rodon system’s development has emphasized its functional capabilities rather than its graphical interface. R.O.S.E. expressed an expectation that larger application projects would want to develop their own application specific graphics.

The graphical interface was often perplexedly slow in drawing and redrawing the screen. It seems that this may have been due to interactions with our local X-Windows service. R.O.S.E. was handicapped in helping us from a distance because we were using a platform different than R.O.S.E.'s principal develop and delivery platform (Sun Sparcstation 10 and 20 models). Demonstrations on Sun hardware did not appear to suffer from these problems.

D. Constraint Propagation for Diagnosis

Simulation proceeds by seeking a solution to all system constraints, including those inherited from component behaviors and those associated with connections as tolerances or values. Parameters on connections may be set by assigning values to them manually or by receiving values for them from sensors during active system monitoring. Such values have full status as constraints and may not be made less precise or be shifted by constraints propagated elsewhere. The constraint satisfaction process is one of successively adding further restrictions to possible values. A set of values is recorded which can be assigned to each parameter without violating any of the system constraints examined thus far. When this set becomes empty for any parameter, as would occur if its value is constrained, belonging to two nonoverlapping interval sets, a global failure is signaled. Naturally, this cannot promise any definite information about the root "cause" of the constraint satisfaction failure, but sometimes knowing the location (the parameter) at which the failure was discovered is helpful to the user in understanding the situation.

When constraint propagation failure occurs, the user may invoke Rodon's diagnoser to determine possible causes for the problem. In automated monitoring mode, the diagnoser is started automatically.

For diagnostic purposes, a failure of constraint propagation usually means that the sensor values are consistent with a component's new and unknown behavior, rather than the behavior which the component was designed to perform. However, there are other possible interpretations:

- The behaviors of one or more components have been defined inconsistently with each other, and so diagnosis is a tool for eliminating inconsistencies from design.
- Measurements are consistent with actual working component behaviors. In this case, the diagnostic process is a tool for debugging the model.
- Measurements represent goal states rather than physical readings, and diagnosis is a search for components whose behaviors can be altered to achieve those measurements. In this case, the diagnostic process is a tool for goal-directed redesign.
- Measurements represent goal states rather than physical readings, and diagnosis is a search for inputs or environmental values which can be altered to achieve those measurements. In this case, the diagnostic process is a tool for a form of closed-loop control.

In this work, and in the current development of Rodon, the first mentioned interpretation is assumed, as is appropriate for monitoring an operational system with a verified model.

E. Behavior Suspension in Rodon

Failure to find a solution means that the system is overconstrained, and so Rodon looks for constraints to remove, or suspend, in accord with the constraint suspension approach introduced by Randy Davis and Walter Hamscher in the early 1980's.² Each component or subsystem is tested, one at a time, by suspending all of its constraints. If the system constraint network can now be satisfied while a component's constraints are suspended, then that component is retained, being a plausible suspect as the cause of the problem.

When failure modes (fault behaviors) are specified for a component, then constraint suspension means not just removing the component's normal behavior constraints, but replacing them by the fault behavior constraints. There may be several alternative fault behaviors, as shown in the multiplier example above, or even an unspecified but labeled fault behavior saying only that "FAULT-STATE =1."

Rodon contains an interesting innovation in "normal + fault" suspension, which is invoked by the diagnoser or through manual selection. This replaces the normal behavior by a disjunction of the normal and fault behaviors. This disjunction is then propagated like any other behavior except when constraint satisfaction is achieved. Rodon then keeps track of whether it was the normal and/or the fault parts of the composite function that were used in achieving a successful solution. During simulation, components with "normal + fault" suspension are treated as follows:

- Suppose only the normal function contributed to successful propagation. That is, the assumption of at least one disjunct from the normal function leads to system consistency, but there is no disjunct from the fault function that leads to system consistency. In this case the component is no longer suspect and its icon is green.
- If only the fault function contributed to successful propagation, the component is confirmed as a suspect and its icon is red.
- If both the normal and the fault functions can lead to successful propagation, then no conclusion can be reached about this component as a cause of the original inconsistency, and its color (blue) is unchanged.

Through this mechanism, it was noted that Rodon is capable of multiple fault diagnosis as a side effect of simulation under "normal + fault" suspension, even though the diagnoser itself did not take full advantage of this capability and was advertised as being for single faults only.

F. Hierarchical and Dynamic System Representation in Rodon

There is provision in Rodon for hierarchical system modeling, wherein an entire system can be represented by a single icon, as a subsystem within other systems. It is our understanding that there is no compositional compaction or approximation available for representing the behavior of subsystems in Rodon. Nevertheless, hierarchical subsystems are used to advantage within diagnosis. The subsystem has all of its constraints simultaneously suspended as if it were a single complex component, and only if system-wide constraint satisfaction is thereby achieved does the display expand the subsystem and test its components individually. Given that all constraints must be satisfied in the test of each suspension,

there may be great savings if many large subsystems are ruled out at the higher diagnostic levels, obviating repeated (useless) suspensions of each of their components.

Rodon contains provisions for dynamic system modeling through its use of “time layers,” designed to represent the system at successive time intervals. In the adopted “\$-subscript” notation, $X_{\$1}$ might mean the value of X at the current time while $X_{\$2}$ holds the value at the next time slice (say, time dt later). The zero subscript on $dt_{\$0}$ indicates that it is a global value which is constant in all time layers. As an example, a damped harmonic oscillator is given as:

Constants: C, K

Inputs: $dt = \text{time}$

Outputs: $x = \text{displacement}, v = \text{velocity}$

Normal Function: $v_{\$1} = (x_{\$2} - x_{\$1})/dt$ and $(v_{\$2} - v_{\$1})/dt + C*v_{\$1} + K*x_{\$1} = 0$.

(This is slightly simplified; for example, the equations actually need to be entered in a form that is solved for the highest time-layer ($\$2$) variables.) This models a second-order differential equation with only two time layers and produces x and v values for damped oscillations as time progresses.

This time layer mechanism can be used to create a table of computed values for graphical display (graphing tools are included in Rodon), or to represent disjoint noninteracting versions of the same system (if no constraints connect any $\$1$ with any $\$2$ variables).

G. Solutions to Nonlinear and Simultaneous-Equation Constraints in Rodon

A significant restriction on the Rodon implementation which we used is that it accepts only linear constraints; i.e., a constraint will not propagate properly if it contains a term like $x*x$. This is partly compensated by supplying tools to facilitate the conversion of nonlinear functions to a piecewise linear representation. Clearly, this restriction is due to the difficulties of automatically solving the constraints for all their variables. In fact, one can use more general constraints for simulation if one abandons the generation of a multidirectional network and, therefore, of diagnosis. We expect there will eventually be facilities for supplying user-defined inverses.

Another problem, pointed out at least as far back as 1980,⁷ is that some simple systems, like two resistors in series, cannot be readily solved through local constraint propagation alone. A single resistor with a known voltage imposed upon it will immediately yield the current via Ohm’s Law, but with two resistors in series, a local application of Ohm’s Law cannot yield the global current. Students typically learn rules for combining resistances to reduce this problem to a soluble form. A more general approach is to enable the solution of simultaneous equations. This is what Rodon has done through an add-on package referred to as “interval bisection,” which in effect solves the equations through the application of a numeric Newton’s method. It is surprising how infrequently the need arose for this bisection routine in actual practice. (It was not used by our model of the SSM/PMAD power distribution and control unit (PDCU).)

H. New Rodon Release Announced

The R.O.S.E. diagnostic engine (RDE)—a new multiple-fault diagnostic engine—does not use constraint suspension; rather it possesses an architecture closer to that of a general diagnostic engine (GDE)³ using a truth maintenance engine. Order of magnitude improvements are claimed in both speed and required memory, as well as lower computational complexity on the size of the system's constraint population. Unfortunately, RDE was announced too late to be incorporated into this study.

I. Fault Recording for Continued Monitoring Past Failure

The ability to continue monitoring after-fault diagnosis is an essential advantage of model-based methods. Associational diagnostic methods^{8 9 10} can operate beyond a failure only with difficulty, since the knowledge bases would require modifications to reflect the new (and, in general, unpredicted) condition of the system. A model-based diagnoser has the potential of doing this automatically, by substituting diagnostically isolated fault modes in place of the component's normal function, or by simply leaving the component's constraints suspended. When we received Rodon, it did not include a facility for recording failures. This meant that each subsequent data snapshot would cause a full repeat of the diagnosis, leading to Rodon's rapidly falling behind in its monitoring task. After all, snapshot intervals are normally chosen just far enough apart so that monitoring is assured of keeping up with the unfaulted system. Diagnosis naturally is a slower process and, in principle, need not be rushed since it could be done offline or by another processor. Another serious consequence of not recording failures is that subsequent failures become increasingly probable and will then be seen as multiple faults, confounding a machine designed to analyze one fault at a time. Rodon's previous applications had never required fault recording since they would normally stop operations to repair any fault before proceeding. Pausing until repairs are complete is not an option in many aerospace applications, like a launch in progress or a platform in deep space. Therefore, we suggested to R.O.S.E. that fault recording be investigated. They responded immediately by supplying an option to record any fault which has been isolated by an unambiguous diagnosis. The situation is more complex when a diagnosis is ambiguous, since monitoring must keep alternate possible faults alive in different possible worlds, rejecting those which contradict later observations.¹¹ We believe this is a plausible and not overly difficult extension to the current Rodon facility, particularly if they take advantage of the truth maintenance system used by the new Rodon release.

IV. SSM/PMAD—A POWER DISTRIBUTION TEST-BED

A. NASA's SSM/PMAD

The SSM/PMAD system is a test-bed for the automated planning and control of power, originally designed to shadow the power distribution system on the *International Space Station*. In fact it was initially a high-frequency alternating current system and later converted to 120 V direct current (dc) after the Space Station did so. Although it is not presently maintained in lock step with current Space Station design, it is a useful breadboard with sophisticated load planning, load prioritization, load shedding, and self-diagnostic capabilities. It has been used as a test-bed for several studies in diagnosis and control, some just recently published.

B. Structure of SSM/PMAD

SSM/PMAD contains two parallel PDCU's which are cross-linked to provide redundant power service to the system's loads. Within the PDCU's, and the load centers through which they supply power to end loads, the primary type of control device is the "remote power controller" (RPC), also referred to as a "smart switch." RPC's contain voltage and current sensors, and will disconnect themselves ("trip") when the voltage is too low, the current too high, or the power consumption too great. The voltage is compared to the fixed 120 V dc supply, while the limits on current can be dynamically changed during operation. There are also simple switches at the source of the PDCU's, with no automatic tripping capability.

C. Modeling SSM/PMAD

Due to the limited time and resources available to this project, the Rodon evaluation was restricted to the modeling of one of the PDCU's down to the RPC level, and representing the downstream load centers as simple resistive loads.

This work uses the existing device drivers and data storage system to obtain online data. It does not make direct use of SSM/PMAD's load planning and related facilities. When Rodon is used for monitoring, the user must build a Preparation Program (PP) which supplies telemetry from the system being monitored to Rodon. The SSM/PMAD software, residing on a separate workstation, archives all ethernet transactions with the test-bed control computers in several large files. Our PP, named RIF (Rodon Interface, also named in honor of government downsizing), parses the archive files, attempts to construct coherent snapshots of SSM/PMAD's state, and pipes the snapshots to Rodon. Rodon analyzes the snapshots in succession, each of which contains a complete set of test-bed variables. Data were flexibly transmitted in one of several alternative modes:

- From archived files without delay.
- From archived files while emulating the original timing.
- From live files actively receiving data from the breadboard. This was all achieved by a C-language interface program written by the NASA team members.

SSM/PMAD's software includes diagnostics which effectively run in parallel with Rodon's, and the two diagnosers in principle check each others' conclusions. In practice, Rodon typically responded to faults not detected by the existing SSM/PMAD diagnoser. We originally had planned to represent RPC trips by using time layers to model the fact that an undervoltage or overcurrent at time \$1 meant the RPC would be tripped and its current would be zero at time \$2. After some experimentation, it was determined that RPC's hardware control logic responded too quickly for us to see the data containing anomalous voltages or currents before the RPC trip cut them off. Rodon's models therefore could look only at the static problem of ensuring that there were no anomalous power values, or that the tripped RPC was consistent with its loads being disconnected. Thus, the Rodon model and the existing SSM/PMAD's diagnostics had little overlap in the types of problems they handled, and in practice, may be quite complimentary.

The RPC is the most complex single component modeled. Shown below is a normal behavior model, after editing for simplicity and readability:

```
(FAULT-STATE = 0) and (I1 = I2)                                and
(I1 <= MaxLoadCurrent and MaxLoadCurrent < MaxSwitchCurrent)  and
((OverCurrent = On or UnderVoltage = On or FastTrip = On) -> SwitchState = Trip)  and
((Command = Off and SwitchState = Off and I1 = 0)                or
(Command = On and SwitchState = On and V1 = V2)                  or
(Command = On and SwitchState = Trip)                             or
(Command = Unavail and SwitchState = Unavail))                  and
((SwitchState = Trip or SwitchState = Unavail) -> I1 = 0).
```

This model says that:

- Current is always conserved through the RPC.
- Current is within the dynamic limits presently in force.
- If any of the anomalous conditions are flagged, then the RPC must be tripped. (FastTrip is similar to Overcurrent but responds more quickly and at higher current levels.)
- The RPC's voltage and current are consistent with its state:
 - If commanded Off, the RPC's current must be zero.
 - If commanded On, then either:

- The switch is On and the voltage drop across the RPC must be zero.
- The switch is tripped and its current is zero.
- If Tripped, the RPC's current must be zero.
- If Unavailable (as if removed from the breadboard), the RPC's current must be zero.

Note that there is no need to place any constraints upon the current or voltage other than those required by the switch state. For example, when the switch is Off its current must be zero, but there is nothing that needs to be, or should be, said about its voltages. Shown below is the fault behavior of the RPC:

(FAULT-STATE = 1 and (Command = Off or SwitchState = Off) and I1 > 0) ; stuck ON

or

(FAULT-STATE = 2 and SwitchState = On and V1 <> V2) ; fails to turn ON

or

FAULT-STATE = 3.

Text following a semicolon is a comment. The first two fault states are fairly self-descriptive. If the third, unconstrained fault state were not given, then the RPC could only be recognized as faulty if its parameters matched one (or both) of the first two fault states. Even if the constraint network could not be satisfied by the normal behavior, if failure modes are defined but also do not lead to consistent state, then the component is considered exonerated of all suspicion. Since these first two fault states are not held to exhaust all possible RPC failure types, the third case should be included to permit the RPC to be judged faulty when neither normal behavior nor the specifically given fault modes yield network consistency. There is value in including the two specific fault types if their possibility is of interest to the user. For example, if the diagnosis gives FAULT-STATE = {[1][3]}, then the user immediately knows that a "failure to turn On" has been ruled out (naturally [3] will be present as a possibility whenever the RPC is implicated, since it subsumes all more specific fault modes). Rodon is expected to be adapted in the future to continue monitoring beyond a diagnosis by replacing the normal behavior by the fault behavior with unsuccessful disjuncts eliminated.

Rodon's constraint and interval semantics mean that the V1 <> V2 test is not as useful as it appears, since two parameters with real data will virtually always test successfully as unequal. Two interval sets X and Y will test unequal unless they are precisely identical, because there will exist a valid possibility that the actual values of X and Y are different. For example, the test ({[2.1 2.5]} <> {[2.1 2.52]}) yields the mixed result {[0][1]} (where [0] means False and [1] means True), because it is possible for the second variable to have certain values, such as 2.51, which the first variable cannot. Under this interpretation of interval sets, only equality is a useful relation since the equality of two variables interval sets is false whenever they cannot overlap.

D. A Comment on Single-Fault Modeling and Debugging

It can be argued that multiple-fault analysis is unnecessary for the model-based diagnosis of operational systems, provided the mean time between failures is long compared to the interval between data snapshots, and as long as the model is reconciled with each fault after its diagnosis—achieved by suspending or replacing its constraints. While this is quite correct, it is nevertheless true that single-fault diagnosis lacks usefulness when debugging a newly written model, even though the model may be of a fully functional and operational system. A new model is like a new and untested piece of hardware off the assembly line, and may well have multiple “faults” under the interpretation of “faults as model defects” (i.e., normal behaviors in the model which are inconsistent with physical components’ actual normal behaviors). Single-fault diagnosis was of little help in debugging the initial model. The best procedure was to use large initial tolerances on all important parameters. This can be done manually in Rodon, and repeated on all parameters involved in unexpected conflicts that prevent system constraint satisfaction. This is a case in which it is helpful for the simulation to display the location at which conflict was discovered, even though this information may be misleading during operational failures. After fault-free simulation succeeds in finding a consistent solution, the tolerances may be reduced to useful levels by trial and error. Faults may then be introduced to guide the tightening of tolerances until they can be detected.

E. Test Fault Scenarios

Three types of faults were physically present or introduced into the SSM/PMAD breadboard, and were successfully diagnosed in uninterrupted operational sequence.

1. Sensor Faults

Sensor faults are typically difficult for associational diagnosers, since those would need special associations installed in advance of each failure to perform diagnoses not dependent upon that sensor’s data. Moreover, special “meta-tests” must be run prior to their standard diagnostic rules to determine which sensor might be failed and therefore which diagnostic strategy to invoke. As shown in previous work,^{1, 12} model-based diagnosis avoids these problems by seeking global consistency without any special status for sensors. This was demonstrated for two different sensor conditions:

- **Hard sensor failure:** An ammeter on the PDCU’s main power feed happens to be genuinely and permanently failed stuck at zero. This fault is not observable before power is turned on, but is immediately evident thereafter. Expectations derived from other sensors, the RPC’s being turned On, and the load resistances, all propagated constraints on the current that were inconsistent with a zero reading. This was immediately spotted, unambiguously diagnosed, and recorded.
- **Soft sensor failure:** A voltmeter above one of the RPC’s was induced to give a different reading by placing a dc voltage source in opposition to the supplied voltage across the sensor. This led to the sensor reading of about 70 V instead of 120 V. The only voltage hard-coded into the model is the output of the power supply feeding the PDCU’s input cable. Propagation of this voltage and the readings of other sensors led to a conflict at the affected sensor which, again, was immediately spotted, unambiguously diagnosed, and recorded.

2. Soft Load Fault

The resistance corresponding to one of the Load Centers was increased from 30 to 60 Ω , thus reducing the current in that load by about half. The model of the Load Centers is given as:

Constants: R = resistance

Inputs: I1 = current, V1 = volt

Outputs: FAULT-STATE

Normal function: FAULT-STATE = 0 and I1=V1/R.

A particularly wide tolerance is assigned to the resistance, to allow for likely load variation. The current in this load was just low enough to be outside the considerable tolerances that had accumulated through system propagation, and the resulting fault was also unambiguously diagnosed and recorded.

3. Soft Internal PDCU Failure

A resistive shunt to ground was attached to the hot wire of one of the cables between an RPC and the PDCU's ammeter downstream of it. The RPC's have their own internal ammeters, so it was immediately noticed that their current readings were in conflict. The current difference was outside the tolerances on the ammeters, although well inside the tolerance propagated from the load. Rodon could therefore not tell by looking elsewhere in the system which ammeter was correct, and the resulting diagnosis was ambiguous. Both ammeters remained suspect, as well as the (correct diagnosis of) cable between them. The cable's model is given below (note that "[+]" here means +infinity):

Inputs: IIN = current, VIN = volt

Outputs: IOUT = current, VOUT = volt, RSHORT = resistance, FAULT-STATE = integer

Normal function: (FAULT-STATE = 0 and IIN >= 0 and IOUT >= 0 and IIN = IOUT and VIN = VOUT and RSHORT = [+]) ; infinite shunt resistance

Fault function: ((FAULT-STATE = 1 and IIN = 0 and IOUT = 0 and RSHORT = [+]) ; broken cable
or
(FAULT-STATE = 2 and VIN = 0 and VOUT = 0) ; hard fault to ground
or
(FAULT-STATE = 3 and VIN = VOUT and IIN = IOUT+VIN/RSHORT ; soft fault to ground
and RSHORT = [4 100])).

The diagnosis for the cable indicated FAULT-STATE = {[3]} and RSHORT = {[36.3 100]}. The latter is consistent with the actual shunt of 50 Ω . This fault yielded an ambiguous diagnosis, so it was not recorded and therefore was repeated until the shunt was removed.

V. OTHER COMMENTS ON RODON

This section includes some comments on the version of Rodon that was used.

A. Documentation

It is clear that the English system documentation needs considerable improvement. Markups of certain sections of the existing documentation are being sent to R.O.S.E. under separate cover. First and foremost, a good index would have overcome some of the problems associated with the current erratic nature of the documentation's content and organization. The menu-driven organization is good, but the explanations are not always helpful. There is no question that a keystroke example for each facility would be enormously useful. R.O.S.E. is taking the documentation problem seriously and is reissuing radically updated versions.

B. Constraint Solver

The heart of Rodon, the constraint solver, was powerful and robust. With the possible exception noted in the next paragraph, we were not able to find any flaw in its advertised operation. Its primary shortcoming, for our purposes, was not handling nonlinear constraints.

One possible problem noted, however, either in the solver or more specifically in the time layer facility, is the peculiar dependence of the results of the damped harmonic oscillator example upon the selected time step. For $x'' + cx' + kx = 0$, the condition for undamped oscillation appeared to be $c*dt = 0.2$, rather than $c = 0.0$. This may be simply an artifact of the numerical solution of the difference equation with different time steps which we expect to be resolved in subsequent releases.

C. Model Debugging Facility

As we developed the model of the SSM/PMAD test-bed, it became apparent that Rodon needed some improved model debugging facilities. If the diagnoser was triggered by a failure in telemetry that was known to be normal, it usually involved multiple constraint failures, and no list of suspect components was produced. Manually comparing telemetry to constraints was necessary to locate the modeling errors. Likewise, if known abnormal telemetry was passed without comment by the diagnoser, it could be quite time-consuming to locate erroneous constraints. Of course, if it is known what is wrong with a specific snapshot of telemetry data, anticipating the constraints it will violate is generally possible.

For the former problem, the ability to detect multiple failures would help greatly. For either problem, a facility which would make model development much easier would enable the developer to examine individual constraints or subconstraints (disjuncts) within propagated constraints, their status (pass or fail), and the telemetry associated with each. For example, the constraint solver could highlight successful clauses in the model editor.

D. Preparation Program

On a Unix platform, Rodon is designed so that the Preparation program, which fetches and formats telemetry for Rodon, can write the telemetry directly to its standard output (e.g., stdout in a C program). To get the formatted telemetry, Rodon spawns a shell in which to run the PP, and pipes in the telemetry. This feature makes it easy to develop the PP independently of the system model.

E. Graphical Interface

As noted previously, the graphical interface was the weakest part of the Rodon system used. Some of the difficulties may have been due to the use of a nonstandard environment (an HP rather than a Sparcstation) which made it difficult for the R.O.S.E. developers to keep us supplied with their most recent code, and made it hard for them to check out some of the interface problems we reported (notably, slow refresh rates).

Port names were often confusing, and it would have helped to be able to assign different length labels to different ports. Units were not as useful as they might have been. Since they appeared to be used only for reporting clarification, they should not have been mandatory during object definition. It would be beneficial to have a more compact syntax for assigning the same unit to multiple ports. It also seemed odd that units could not be assigned to constants.

In general, the supplied interface is below the current standard of domain-dependent schematic representations. We understand that this is a reasonable place for them to defer development since users in major domains will develop their own interfaces. Nevertheless, the uniformly rectangular boxes and port tab connectors did not help system visualization. Junctions that simply set parameters to being equal (such as voltage connectors) were done nicely, but more complex junctions (such as those needed to add currents) were represented in ungainly fashion as separate components.

One could overlay icon graphics onto rectangular components, but an icon editor was not provided, and the supplied set had limited applicability, so this feature was not used in this evaluation. Despite the above comments, the point-and-click graphics editor is useful as supplied and is expected to be the basis for more domain-specific development of schematic interfaces.

F. System Hierarchy

The current modeling hierarchy is a very good beginning, and is headed towards a fully developed object-oriented representation, but it needs further development. We recommend that when a change is made in the model definition of a component, the change should be automatically inherited by every instance of that component in the system. It might also be desirable to be able to override such a feature, but we believe this would be the desired outcome in 95 percent of all situations. Rodon includes a search facility which aids in locating instances.

VI. CONCLUSIONS

Overall, the internal machinery worked well. Constraint propagation and solution was quick and consistent. The options supplied are useful and generally exploit well the capabilities offered by model-based diagnosis. The user interface is not as well developed, reflecting the development team's focus on internal machinery, and the expectation that major users may be developing their own interfaces. The faults introduced into the SSM/PMAD test-bed and diagnosed by Rodon were not detected or diagnosed by SSM/PMAD's hardware or software diagnostics. There were several reasons why SSM/PMAD did not identify these failures, but the principal reason was the lack of a reference model with sufficient accuracy to allow the dynamic discrimination of acceptable from unacceptable measurements. Rodon's ability to successfully use such a model is an indication of the importance and strength of this approach. We note that improved documentation is required, and has been under development.

While Rodon performed well under our test scenario, it must be remembered that this scenario was not challenging in size or complexity, since the limited duration of the project prevented further exploration. We have every reason to believe that the Rodon constraint engine would handle considerably larger and more complex systems, although compilation (generation) times might grow burdensome. Larger numbers of components would have stressed the interface, which was already painfully slow. But we understand that this was an artifact of using a nonstandard machine. (We used an HP-745, while almost all Rodon development was done on Sparcstations). Had we used a Sparcstation, or had R.O.S.E. personnel been locally available, we believe that the interface problem would have been resolved.

We must note that we did not master all of the facilities that Rodon offers. Some of the shortcomings noted here may be a result of the limitations in our familiarity with the system.

The recent introduction of the new RDE version of Rodon promises significant improvements in power and capability (e.g., multiple faults scenarios).

In summary, we judge that the Rodon system, in its current state of development, succeeds in implementing model-based diagnosis, perhaps as the only commercially available shell. We hope that this development continues along the promising lines already being pursued, and we see it as a basis for making model-based diagnosis more widely understood and used.

REFERENCES

1. Scarl, E.A.; Jamieson, J.R.; and New, E.: "Deriving Fault Location and Control from a Functional Model." *Proceedings of the Third IEEE Symposium on Intelligent Control*, Arlington, VA, 1988.
2. Davis, R.: "Diagnostic Reasoning Based on Structure and Behavior." *Qualitative Reasoning About Physical Systems*, MIT Press, Cambridge, MA, pp. 347–410, 1985. Also in *Readings in Model-Based Diagnosis*, Morgan Kaufmann, San Mateo, CA, pp. 376–407, 1992.
3. de Kleer, J.; and Williams, B.C.: "Diagnosing Multiple Faults." *Artificial Intelligence*, 32(1), pp. 97–130, 1987. Also in *Readings in Model-Based Diagnosis*, Morgan Kaufmann, San Mateo, CA, pp. 100–117, 1992.
4. Reiter, R.: "A Theory of Diagnosis from First Principles." *Artificial Intelligence*, 2(1), pp. 57–95, April 1987.
5. Davis, R.; and Hamscher, W.: "Model-Based Reasoning: Troubleshooting." *Readings in Model-Based Diagnosis*, Morgan Kaufmann, San Mateo, CA, pp. 3–24, 1992.
6. Scarl, E.A.: "Monitoring and Diagnosis: Stress from Weakened Environmental Knowledge." *Proceedings of IEEE International Conference on Robotics and Automation*, Sacramento, CA, April, 1991.
7. Sussman, G.J.; and Steele, G.L.: "Constraints: A Language for Expressing Almost-Hierarchical Descriptions." *Artificial Intelligence*, 14(1), pp. 1–39, 1980.
8. Pople, H.: "The Formation of Composite Hypotheses in Diagnostic Problem Solving." *Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI-77)*, Cambridge, MA, pp. 1030–1037, 1977.
9. Shortliffe, E.H.: *Computer-Based Medical Consultations: MYCIN*, North-Holland, NY, 1976.
10. Torasso, P.; and Console, L.: *Diagnostic Problem Solving: Combining Heuristic, Approximate, and Causal Reasoning*, Van Nostrand Reinhold, 1989.
11. de Kleer, J.; and Williams, B.C.: "Diagnosis with Behavioral Modes." *Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI-89)*, Detroit, MI, pp. 324–330, 1989.

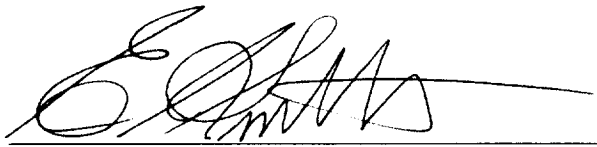
12. Scarl, E.A.: "Diagnostic Tolerance for Missing Sensor Data." *Proceedings of 1989 Goddard Conference on Space Applications of Artificial Intelligence*, Goddard Space Flight Center, Greenbelt, MD, pp. 213–220, May 1989. Republished in *Telematics and Informatics*, 6(3/4), November/December 1989.

APPROVAL

MODEL-BASED DIAGNOSIS IN A POWER DISTRIBUTION TEST-BED

E. Scarl and K. McCall

The information in this report has been reviewed for technical content. Review of any information concerning Department of Defense or nuclear energy activities or programs has been made by the MSFC Security Classification Officer. This report, in its entirety, has been determined to be unclassified.

A handwritten signature in black ink, appearing to read 'E.C. Smith', is written over a horizontal line.

E.C. SMITH
DIRECTOR, ASTRIONICS LABORATORY

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operation and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503				
1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE September 1998	3. REPORT TYPE AND DATES COVERED Technical Memorandum		
4. TITLE AND SUBTITLE Model-Based Diagnosis in a Power Distribution Test-Bed			5. FUNDING NUMBERS	
6. AUTHORS E. Scarl* and K. McCall				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) George C. Marshall Space Flight Center Marshall Space Flight Center, Alabama 35812			8. PERFORMING ORGANIZATION REPORT NUMBER M-898	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, DC 20546-0001			10. SPONSORING/MONITORING AGENCY REPORT NUMBER NASA/TM-1998-208804	
11. SUPPLEMENTARY NOTES Prepared by Astrionics Laboratory, Science and Engineering Directorate. *The Boeing Company, Huntsville, Alabama This work was performed under a joint NASA/Boeing Space Act Agreement, EB01-BSDG-001				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified-Unlimited Subject Category 61 Nonstandard Distribution			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) The Rodon model-based diagnosis shell was applied to a breadboard test-bed, modeling an automated power distribution system. The constraint-based modeling paradigm and diagnostic algorithm were found to adequately represent the selected set of test scenarios.				
14. SUBJECT TERMS model-based diagnosis, Rodon, constraint networks, SSM/FMAD, space power, automation			15. NUMBER OF PAGES 28	
			16. PRICE CODE A03	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT Unlimited	