S3-32
239818

547657 p12

N90-12794

TDA Progress Report 42-98

118.

August 15, 1989

Automated Monitor and Control for Deep Space Network Subsystems

P. Smyth
Communications Systems Research Section

This article will consider the problem of automating monitor and control loops for Deep Space Network (DSN) subsystems. The purpose of the article is to give an overview of currently available automation techniques. In particular, the article will consider the use of standard numerical models, knowledge-based systems, and neural networks. Among the conclusions is argued that none of these techniques alone possess sufficient generality to deal with the demands imposed by the DSN environment. However, the article will show that schemes that integrate the better aspects of each approach and are referenced to a formal system model *show considerable promise—although such an integrated technology is not yet available for implementation. The article should be of benefit to readers interested in any aspects of DSN automation as it highlights the advantages and dangers of currently available approaches to the automation problem. The article frequently refers to the receiver subsystem since this work was largely motivated by experience in developing an automated monitor and control loop for the advanced receiver.*

I. Introduction

Consider the problem of designing an automated monitor and control loop for a DSN subsystem. This article will refer to the "system" as the system being controlled and the "loop" as the hardware or software (or human) component which is used to monitor the system and control its behavior. For example, a receiver is a subsystem which is normally controlled in a relatively manual manner by human operators. This article will argue that reliance on purely manual control techniques will not be sufficient to cope with future subsystem technologies and an increasingly complex DSN ground station environment. Autonomous or semiautonomous control loops will be necessary. This article will address some basic issues regarding automated monitor and control loops: how should such loops be designed and what general principles should be used? Specifically, the problem is defined in a very general manner using a state-space model and evaluating current technologies such as knowledge-based systems and neural networks in this context. The monitor and control problem can be decomposed into four basic subfunctions: obtain sensor data, estimate system parameters from the data, make decisions based on the estimated

Table 1. Upper bound on R for the (31,15,17) RS code

u	R
16	2.74943144e−024
17	1.50775270e−024
18	4.37734673e−025
19	8.94296586e−026
20	1.44241389e−026
21	1.95423782e−027
22	2.31146419e−028
23	2.44993596e−029
24	2.37090920e−030
25	2.12446863e−031
26	1.78181347e−032
27	1.41082139e−033
28	1.06190837e−034
29	7.64152968e−036
30	5.28215447e−037
31	3.52143591e−038

Table 2. Weight distribution and its approximation for the number of decodable words of the (31,15,17) RS code over GF(32)

u	Du (exact)	$D'u$ (approximate)
9	1.998e+014	2.999e+015
10	5.232e+016	2.045e+017
11	6.348e+018	1.210e+019
12	4.817e+020	6.253e+020
13	2.613e+022	2.833e+022
14	1.113e+024	1.129e+024
15	3.970e+025	3.968e+025
16	1.231e+027	1.230e+027
17	3.364e+028	3.364e+028
18	8.111e+029	8.111e+029
19	1.721e+031	1.721e+031
20	3.200e+032	3.200e+032
21	5.196e+033	5.196e+033
22	7.322e+034	7.322e+034
23	8.822e+035	8.882e+035
24	9.178e+036	9.178e+036
25	7.967e+037	7.967e+037
26	5.699e+038	5.699e+038
27	3.272e+039	3.272e+039
28	1.449e+040	1.449e+040
29	4.647e+040	4.647e+040
30	9.603e+040	9.603e+040
31	9.603e+040	9.603e+040

parameters, and implement the decision by changing system inputs. Previously reported work on station automation [1, 2] has focused on the automation of operational procedures, namely the automation of data collection and control implementation, the first and fourth subfunctions listed above. Traditionally, the estimation and decision-making subfunctions have been performed manually by human operators. These are precisely the subfunctions on which this article focuses, i.e., this is what is referred to as automated monitor and control.

II. A Statement of the Problem

A simple general model of a DSN subsystem which one wishes to control is shown in Fig. 1. $I_o(t)$ and $O_o(t)$ are defined as the observable inputs and outputs respectively, and $I_{\bar{o}}(t)$ and $O_{\bar{o}}(t)$ as the nonobservable inputs and outputs, where t denotes time. H is the open-loop transfer function of the system, i.e., $O(t) = H(I(t))$.

In addition, there is possibly an a priori system model M. The rules of the game comprise of being given $I_o(t)$, $O_o(t)$, and M at time t, and trying to effect a particular input/output transfer function H at some time $t + \delta t$ by changing whatever subset of the inputs $I_o(t)$ one is allowed to control. This is a very general description, and, for example, could equally well describe an implementation of an adaptive control loop using classical control theory, or a human trying to perform real-time fault-diagnosis in order to restore normal operating system behavior. A key point of this article is that there is a need to be concerned with all such levels of control. The need for modeling very different control loops of this kind in an integrated manner will be identified. The function of most DSN subsystems is to deliver or process signals or data subject to some criteria. Hence, the automated loop must be able to perform a variety of functions such as performance optimization and fault detection in order to satisfy overall system performance criteria. The article will return to this point in much greater detail later. This requirement will entail that the loop has a *model*, in some form, of the system it is trying to control.

Figure 2 shows the general feedback control scheme, where G represents the control-loop function. For example, if the system can be modeled accurately by closed-form analytic solutions, then G may be implemented as a set of equations such as a phase-locked loop (PLL) or other such receiver structure. However, at the level of interest here (namely, entire subsytems such as the receiver), analytic closed-form solutions are not practical and it is traditional to close the loop manually, i.e., G is a human operator.

For two reasons the usual engineering analysis approach will not work directly on this problem: first, the open-loop system transfer function (call it H) is often nonlinear and too complex to model accurately. Secondly, the transfer function may vary over time and be a function of the input variables. Remember that this is an entire subsystem involving the interaction of hardware and software components. Hence, the system will have a set of states (possibly infinite). Let the state of the system at time t be $S(t)$. The transfer function H is then $H(S(t), I(t))$. The only time when H is known exactly may be for a subset of states S_n, which might be thought of as "normal operating states." This is precisely the case for which little or no external feedback (G in Fig. 2) is required since internal feedback in the subsystem components (for example in the PLL component) will automatically produce the desired output. One is interested of course in the other cases ($S(t) \notin S_n$), namely, spurious system behavior due to "non-normal" inputs, faulty components, etc.

As a concrete example, consider the receiver subsystem. Receivers are designed to work specifically on a known class of modulated signals at specific predicted frequencies. Hence, S_n corresponds to the case where the signal is at the correct position in the spectrum, of the correct modulation format, and the noise is bounded within expected characteristics. As is well-known, achieving optimal receiver performance even when all these assumptions are true is a very non-trivial design problem. Hence, system designers can hardly be expected to account for the large class of spurious system states that may arise during field operations, e.g., new noise characteristics, in-band radio-frequency interference (RFI), problems in other station subsystems, or system components which behave erroneously or fail. As previously mentioned, it has been traditional in field environments such as the DSN to have the human operator detect and correct spurious system behavior using his/her knowledge of the system and the environment. A point of this article is that manual control of this nature is neither feasible nor desirable in the long run.

There are two primary considerations: where to place G relative to existing subsystems and human operators and, secondly, how to implement G. This article will primarily focus on implementation issues, but will first briefly consider the issue of where to place an automated loop component relative to the existing DSN subsystem environments.

III. Basic Architectural Considerations

Consider once more Fig. 2. There are a number of possible avenues one could pursue. One could close the loop within the box (i.e., internally in the system as shown in Fig. 3) and let the the operators function as they currently do, relating inputs to outputs with little information as to the inner state of the system. This is feasible and reduces interface problems, but may actually introduce a new set of difficulties for the operator: the behavior of the internal control loop may make it very difficult for the operator to understand the behavior of the system, i.e., it is even more difficult to operate than previous "open-loop" systems.

An alternative approach is to replace the operator entirely and close the loop completely autonomously (Fig. 4). This approach is probably not necessary for DSN subsystems, since it is widely acknowledged that in any automated control system operating in uncertain environments, human intervention may always be necessary in emergency situations. A good example is that of autopilot control schemes in commercial airliners, where emergency situations require that the human pilots break the automatic control loop.

This paradigm of *semiautonomous* control (Fig. 5) introduces the important component of *situation assessment.* In airliners, for example, for the pilot to break the automated loop and effectively take over control, he must be able to assess the current state of the system or the context in which the automated control loop was acting. This issue requires that the automated system must in some sense be able to communicate at any time the current state of both the system and the control loop to a human operator. Hence, semiautonomous control entails that the control loop contains a high-level *model* of the system it is trying to control.

Another important practical point is that in order for the control loop G to be effective it must have access to all of the relevant system data and be able to control system parameters, i.e., a comprehensive sensory and effectory system must be in place. This is a fundamental requirement. Subsystems should be designed initially such that future hardware and software links to external automated control loops can be easily and cleanly implemented, e.g., hardware ports, software accessibility, etc. This holds true even if the automated loop is built into the originally designed system, since at the next level of automation these control loops will need to communicate with higher-level systems such as a monitor and control system for an entire ground station.

IV. A State-Space Model for Automated Monitor and Control

Consider in more detail the nature of the control loop. One can decompose the problem into two parts: determining the state of the system at time t (*estimation*) and effecting some control action on the system to achieve desired behavior at time $t + \delta t$ (*decision*). Monitoring the state of the system is a necessary prerequisite to controlling it, since controller actions need to be context dependent, i.e., what is done to control the system depends on the state of the system and the input variables (both observables and unobservables). This is the crux of the problem.

As described earlier one can define $S(t)$ as the state of the system at any time t, and S_n is a set of states defined as "normal operating behavior." Hence one can define the set of states $S_{\bar{n}}$ as system states indicating spurious behavior and potentially requiring corrective control action, where $S_{\bar{n}}$ is the complement of S_n over the entire set of possible system states. The purpose of the monitor/controller G can be restated as follows:

(1) At any given time t obtain a state estimate $\hat{S}$ of the true state $S(t)$ of the system, where

$$\hat{S} = f\Big(I_o(t), O_o(t), M, h(\hat{S}, I_o, O_o, t)\Big) \tag{1}$$

where $I_o(t)$ and $O_o(t)$ are the observable inputs and outputs, M is an a priori model of the system, and $h(\cdot)$ is a finite history of previous estimated states and input and output variables.

(2) If $\hat{S} \in S_{\bar{n}}$ choose a control action to achieve a specific set of outputs at time $t + \delta t$ in the future, by setting

$$I_o(t + \delta t) = g\Big(\hat{S}(t), M, I_o(t), h(\hat{S}, I_o, O_o, t)\Big) \tag{2}$$

While this state estimation and transformation model is not the only possible model, and is somewhat of a simplification, it serves the purpose of giving a good insight as to the basic nature of the problem. The first step, that of estimating the state of the system, is quite challenging in itself since the state-space model will necessarily be quite complex for any realistic system. This step must be solved before proceeding to step (2), since typically control actions are quite sensitive to context information, i.e., the state estimate $\hat{S}$. For example, an operator's actions on a receiver depend directly on what the operator perceives the current state of the receiver to be, e.g., whether or not the PLL is in lock or not.

The state-space model is useful because of its generality. In this framework a phase-locked loop is a very well-specified state estimation and control scheme operating on a signal in real-time. At the other end of the spectrum, the human operator is implementing a much more heuristic and adaptive estimation and control scheme at a much slower rate, what one might term "human real time." One is interested in control loops that operate between these two extremes. The remainder of the article will examine current technologies in the context of this state estimation and control model. The possible approaches can be divided into three broad paradigms (standard numeric algorithms, knowledge-based systems, and neural networks) not because they form a well-defined taxonomy for modeling autonomous control systems, but simply because of the widespread interest in techniques of this nature and their potential applications. A goal of the article is to clarify to which types of problems each approach is best suited, and in the process debunk some common myths.

V. Standard Analysis and Algorithms

Engineers like to model systems using differential or difference equations and then, by analysis of the mathematical model, design an estimator and controller subject to cost and performance criteria. The quality of the resulting control loop hinges critically on the accuracy of the assumed model, e.g., whether the real noise statistics fit the parametric model or not. By weakening the assumptions (e.g., by using non-parametric statistical models) one can improve the generality of the model but it then becomes correspondingly more difficult to design the control loop.

Some systems are simply impossible to model accurately using closed-form mathematical solutions alone. It is a key point of this article that DSN subsystems fall into this category. The advanced receiver subsystem, for example, will be a complex mixture of interacting hardware and software components. Clearly, at the local *component* level, existing analytical models will be quite accurate, e.g., the transfer function for an accumulator or an automatic gain control (AGC) circuit. However, one is interested in the global picture, how the components interact together to determine system behavior. Direct modeling via equations is hardly practical. For instance, how does one combine information such as "$E_s/N_0 = 9.5$ dB" and "the board for calculating FFTs is down"?

The key point to note is that variables of interest must be abstracted to the appropriate level of representation. The issue of representation is critical. The model should necessarily entail different levels of abstraction, from relatively high-level system configuration concepts (such as a board being down) down to quantitative real-time parameter estimates (such as the value of E_s/N_0). One can conclude that exact mathematical models have an important role in control loops but are not sufficiently powerful in terms of representation to meet requirements.

VI. Knowledge-Based Systems

The term "knowledge-based system" is used rather loosely to describe any system where knowledge about the system is explicitly represented in symbolic form, as opposed to being embedded in a low-level algorithm. There is an increasing realization that any system which intends to communicate with other agents, be they other pieces of software or humans, needs to be able to abstract its information to the appropriate level. In addition, explicit knowledge representation buys the ability to modify the control system, to update it, etc., in an efficient manner. The potential value of portable, easily-modifiable code that can be changed relatively effortlessly is enormous.

A. Rule-Based Expert Systems

Rule-based expert systems have proven to be very popular in recent years as solutions to the type of control problem discussed so far. Essentially, they attempt to replace or augment the human operator by a program which contains a representation of his expertise as a set of rules. The question will be "Is this a good modeling tool for solving the DSN automated monitor and control problem ?"

First consider the basic characteristics of this approach. The rules represent situation–action pairs or symptom–diagnosis pairs based on the expert's experience. Generally this approach works well in domains where experts do well and the domain is not particularly well-understood at a basic theoretical level, e.g., in many medical applications where basic cause and effect relationships are not known. Conversely, a rule-based system degrades rapidly in performance when faced with novel problems (symptom combinations not encoded as rules) which the expert has never encountered before. In artificial intelligence, this type of rule-based representation, based on experiential knowledge alone, is referred to as a "shallow" model, since it is based on simple pattern-matching without reference to any deeper causal domain theory. It is important to point out that impressive results have been achieved in real applications using rule-based systems and that the technology is available for implementation at present and is well-understood. What this article is trying to do is to identify the limitations of the approach in relation to a particular monitor and control problem.

With reference to Section IV, which defined the monitor and control problem as a two-step procedure based on the estimation function $f(\ )$ and decision function $g(\ )$, a rule-based implementation can be considered as follows: effectively the functionality of $f(\ )$ and $g(\ )$ are combined together into single condition–action pairs or rules. Hence the rules define a heuristic mapping from system parameters to control actions. In this sense the state of the system is never formally evaluated. The consequent implications for semiautonomous control and situation assessment are obvious, i.e., it will be very difficult for an external agent to take over system control in an effective manner.

Rule-based systems are not particularly appropriate in domains where any of the following hold true:

(1) The experts have limited experiential knowledge (are not really experts) or, indeed, there are no experts. This latter problem arose in the advanced receiver expert system project since the system has not been fielded in an operations environment.

(2) There exists a well-defined domain theory, at least at the component level, i.e., well-defined functional and structural properties in the system exist which are fundamental to problem diagnosis.

(3) The external environment is subject to change and, hence, novel problems are likely.

(4) Time and procedural information are critical to reasoning about the domain, e.g., in setting up a link to a spacecraft there is a definite sequence of procedural steps which occur. Rules are very inefficient at representing procedural knowledge.

System designers are *not* the types of experts that rule-based expert systems are intended to model. Their knowledge consists of causal reasoning based on first principles rather than the experiential situation–action pairs of an experienced system operator. While it is true that one can recast basic domain knowledge in the form of rules, it tends to obscure the underlying causal domain theory, i.e., it is not an efficient way to represent this type of knowledge. For a new system like the advanced receiver there is no experiential or heuristic knowledge available. This problem of designing "expert" systems without experts is quite challenging—this article will show later how the notions of model-based reasoning and learning algorithms may provide useful solutions. A practical approach might be to initially field the system with some form of intelligent model-based control loop which includes a capability for rule-based representation, and, as expertise develops, integrate experiential rules into the loop.

Given the above observations, it is clear that rule-based expert systems are not a very good overall model for the problem discussed here. It is important to keep in mind that rule-based systems were originally developed primarily by the medical community as a *consultative* off-line tool, i.e., a program with which a human interacted, answered queries, and received recommendations. Expert systems were never explicitly designed to operate in active, real-time modes in domains where expertise is scarce and reasoning from first principles is essential.

Nonetheless it is important to counter the common myth that rule-based models require dedicated hardware and software components to work and operate separately from "ordinary" software on special purpose machines. Recent trends in the application of rule-based systems have been very much focused on integrating the rule-based approach with standard software, e.g., rule bases can be called by standard procedural programs, integrated with common databases, etc. At the present time most commercial shells are available in languages such as C. Hence, the paradigm of a standard procedural control loop that can call a function to perform rule-based reasoning (to determine the value of a global context variable, perhaps), which is written entirely in a relatively low-level language such as C, and which can be executed by a single-chip microprocessor may be a much more useful paradigm than relying on the conventional rule-based approach as the basic model for control loops.

B. Model-Based Reasoning

Researchers in artificial intelligence have recognized the limitations of purely rule-based systems as outlined above and have consequently moved on to investigating more powerful knowledge representation techniques. Atkinson [3] gives a useful overview of recent AI applications for monitor and control in the aerospace domain. This is of interest since, as with the DSN subsystem problem, monitor and control of aerospace systems involves a large amount of temporal, procedural, and first-principles knowledge. The notion of *model-based reasoning* is loosely based on the idea that if a reasoning system knows the function and structure of each system component (at some appropriate level of representation), and knows how these components are connected together, then by observing certain input and output system parameters it should be able to reason from first-principles to infer system behavior, e.g., detect the most likely cause of a fault. This is an attractive paradigm for the domain discussed here. Research in this area has primarily been domain-dependent, with perhaps the most common domain being that of troubleshooting electrical circuits [4]. A particularly notable application (in a different domain) is the LES system [5] which monitors and controls the critical loading of liquid oxygen onto the space shuttle at Kennedy Space Center prior to launch—the system has been able to perform fault detection and isolation in 10 sec as compared to the 20 min taken by manual methods.

Model-based reasoning is quite different from the rule-based expert system approach and has been associated with the term "deep knowledge" or common-sense reasoning. Unfortunately, from an applications point of view, there are many unsolved problems which remain the subject of ongoing research. For example, the computational requirements can scale exponentially with the number of components under consideration. Another problem is that there has been very little applied research in this area and hence there is very little practical feedback available for the purposes of determining the appropriate level of granularity for such models. For example, for a signal-processing subsystem such as a receiver, should the model entail first-principles knowledge regarding linear systems theory, or is this too detailed? A useful overview on the current state of research in applying model-based reasoning to communication systems is given in [6].

However, despite the unresolved issues, it is important to realize that a priori knowledge of DSN subsystems often falls into this model-based category, i.e., well-defined component models of function and structure. With the advanced receiver, for example, this is very much the case, but with the scarcity of research results and applications in this area (none at all in the communication systems domain to the author's knowledge) it was not possible to apply these ideas directly to the problem at hand within the resources available. One can conclude that model-based reasoning appears promising as a useful modeling tool for the DSN domain and worthy of further attention.

C. Artificial Intelligence and Decision Theory

There has recently been a considerable amount of cross-disciplinary work relating AI with decision analysis and decision theory. Decision theory and analysis grew out of statistical decision theory and has developed a considerable basis of theoretical principles and practical techniques for modeling choice and decision in uncertain environments. Applications of this technique have typically been for decisions in economic and military environments where decision theory can render decision-making a formal statistical modeling problem rather than a subjective procedure, e.g., an oil company may wish to decide whether or not to commit resources to drilling an oil well in a particular location. More recently, this work has been applied to problems which are typically within the AI domain, e.g., fault diagnosis or reasoning with uncertainty. The key shift in focus has been the notion of using decision theory as a theoretical model for decision by an autonomous agent, rather than using it to analyze specific human decision scenarios. In a sense, one might say that this involves decision synthesis rather than decision analysis.

Whether or not decision theory can be applied to the DSN monitor and control problem remains an open question. As with model-based reasoning, research on this topic is still in relative infancy. Theoretically, since the monitor and control loop discussed in this article involves a decision element, there can be no question that decision theory is an appropriate formal model for an autonomous decision agent. There are two key benefits that could be gained from using this approach:

(1) Decision analysis emphasizes and provides many tools for the initial structuring and modeling of the problem at hand, e.g., the use of influence diagrams, a technique for structuring and identifying causal models. Using these techniques would formalize the initial model-building part of the control loop design process.

(2) The use of statistical decision theory as the basis for a rational decision agent is particularly appealing. For example, in an advanced receiver it may be possible to combine several fast Fourier transform (FFT) power spectra in a noncoherent manner. Deciding how many spectra to average over involves a tradeoff between the variance of the spectral estimate and the time taken to perform the estimate, i.e., the time delay in the control loop. Clearly this decision is very much *context*-dependent on parameters such as carrier-to-noise ratio and estimated carrier Doppler rate. For example, it is more important to minimize the time-delay when one is not in lock than when one is in lock. Utility is defined as a quantitative measure of the overall benefit or cost which accrues to the decision-maker from a particular action/outcome pair, i.e., if he/she decides on an action a_i, and event e_j subsequently occurs, $u(a_i, e_j)$ is a defined utility for this pair. For the

purposes here, utility might be defined as the variance/time tradeoff associated with each action/event pair. Rational decision-making corresponds to choosing the action a_k which maximizes one's expected utility with respect to the probability space defined over relevant events, i.e., a probability space involving context-valued variables such as in-lock/not-in-lock.

However, as with standard algorithms and analysis, decision theoretic models alone are not sufficiently rich as a knowledge representation scheme to offer a stand-alone solution. Nonetheless, there is a strong case for their inclusion as a solution component and their integration with more conventional approaches.

VII. Neural Networks

What is a neural network? One view (from an algorithm designer's perspective) is that they are highly distributed, nonlinear models for implementing function estimation, i.e., given input and output *samples* of a "black box" the neural network can reproduce the functionality of the "black box" *in general.* This view may be somewhat narrow, but generally speaking, optimization, adaptation, pattern recognition, etc., can all be suitably recast into the notion of function estimation. Hence, for example, neural networks can form the basis for computationally powerful, nonparametric statistical estimation tools. Indeed, it seems that early successful applications of this technology may be found in computationally intensive pattern recognition and signal processing problems which have proven to be difficult to model via traditional serial, linear, nonadaptive models, e.g., speech recognition, optical character recognition, nonlinear signal processing, adaptive control, etc. Typically, it seems that the less one knows about the problem solution a priori (e.g., the less one understands about the noise characteristics of the problem), the more favorably a neural solution will compare to standard alternatives such as Markov models, Bayesian classifiers, etc. The availability of dedicated hardware within the next decade will be a major factor in determining the future of neural network technology.

How do neural networks relate to the monitor and control problem in this article? One could certainly envision possible applications in terms of real-time estimation and adaptive control, particularly at the *signal* level. But as with each of the approaches we have considered earlier, a neural network would not be sufficient as an overall system model. In particular, given the current understanding of the neural approach, there is no way to incorporate a priori knowledge into the network. In terms of the earlier two-step model, the decision function $g(\ )$ is learned entirely by the network itself, with no reference to the a priori system model M. Neural networks are essentially "black boxes" in that they learn and implement input/output mappings efficiently—but their representation of this mapping is implicitly embedded in a distributed manner in terms of learned weights. Hence, for example, one cannot incorporate existing functional and structural knowledge about the subsystem discussed here into the network. One can conclude however that as a subcomponent of the overall model, say as a nonparametric signal/RFI classifier, the neural network approach shows significant promise provided the hardware becomes available.

VIII. Summary

The need for automating monitor and control functionality in DSN subsystems is becoming more apparent as technology (sensors and computational resources) improves to the extent that operators can no longer be expected to assimilate system data and make decisions in real time. This article has shown that in whatever manner one may implement such a monitor and control loop, it clearly needs to include a *model* of the system, whether it is implicitly embedded in equations or rules, or more formally represented in an explicit manner. The argument for a formal representation model revolves around three facts: first, it renders the loop design problem more tractable as it provides the reference point from which to integrate different numeric and symbolic approaches. Secondly, it increases both the life cycle and the flexibility of the designed software (making it easier to transfer prototype systems of this nature from a research laboratory to field implementation). Thirdly, it facilitates communication between the subsystem and higher authorities, be they human or automated.

It is apparent that the currently available paradigms for implementing control loops each have distinct advantages, yet, for the purposes described in this article, none are sufficiently powerful as an overall system model. It may be conjectured from these observations that successful monitor and control systems in the DSN domain will incorporate models which integrate the more appropriate aspects of each approach. One could envision a model-based representation that is run using decision-theoretic principles and that controls and calls lower-level analytic models and neural-based estimation schemes in a context-dependent manner. For example, in the receiver, computational resources could initially be focused on acquisition (using an FFT to locate peaks in the spectrum [7]), while, once lock is achieved, resources could be allocated to RFI monitoring and parameter optimization.

In the short term, the issue of model-based techniques and their integration with standard numerical models appears worthy of further investigation and application. For example, the use of object-oriented programming techniques appears to have considerable potential for implementing models of this nature. Rule-based expert systems will have their role to play, but the system implementer is advised to be wary if the domain primarily involves temporal, procedural, or first-principles knowledge. Using rule-based systems as a *component* of an overall solution seems like a much more reasonable approach, particularly given that (as discussed earlier) rule-based code can now be written in "low-level" languages such as C and consequently embedded in standard software. Neural network technology may be applicable to specific subcomponents of the estimation and control problem, particularly once dedicated hardware becomes available. Hybrid systems, which combine aspects of each approach appear potentially to be the most promising avenue, but little research and/or applications has been carried out in this arena, and hence they appear to be more of a long-term prospect.

A final point worth noting concerns that of learning algorithms, i.e., systems which can improve their performance over time. A little thought will convince the reader that, in this domain, learning may proceed much more efficiently and effectively if it is referenced to an initial system model. A very general viewpoint is that in modeling any system one can introduce an a priori bias (e.g., a set of assumptions) into the model, as in Bayesian statistics. A strong bias will only help the model if it is accurate, whereas very weak biases will make the learning problem much more difficult. A case in point for automated monitor and control of man-made systems is that the a priori bias, in the form of a functional and structural component model M, is necessarily correct and accurate. Hence, in principle, it can only improve the quality of the overall control loop and any learning componenttherein.

IX. Conclusion

Many real-time monitor and control applications will not be sufficiently well-modeled by rule-bases, neural networks, or standard algorithms *on their own.* This article reviewed the current state-of-the-art in this area and saw that more powerful representational models are in the offing. One concludes that an effective modeling technique would be one which integrated the better aspects of each approach by referencing them to an explicit functional and structural model of the system. Off-the-shelf solutions of this nature are not currently available. Hence, until such techniques are available, system developers should choose their monitor and control models carefully and recognize the limitations of each particular technology.

Acknowledgment

The author gratefully acknowledges the contributions of Gordon Bagby, Joseph Statman, Sami Hinedi, and William Hurd to this work.

References

[1] D. S. Remer and G. Lorden, "Deep Space Station (DSS-13) Automation Demonstration," *TDA Progress Report 42-57*, Jet Propulsion Laboratory, Pasadena, California, pp. 103–119, June 15, 1980.

[2] C. Foster, "Unattended Deep Space Station Tracking Station Development: Monitor and Control Technology," *TDA Progress Report 42-86*, vol. April-June 1986, Jet Propulsion Laboratory, Pasadena, California, pp. 164–170, August 15, 1986.

[3] D. J. Atkinson, *Knowledge-Based Diagnosis for Aerospace Systems*, JPL Publication 88-7, Jet Propulsion Laboratory, Pasadena, California, March 15, 1988.

[4] R. Davis, "Diagnostic Reasoning Based on Structure and Behaviour," *Artificial Intelligence*, vol. 24, nos. 1–3, pp. 347–410, December 1984.

[5] E. Scarl, et al., "A Fault Detection and Isolation Method Applied to Liquid Oxygen Loading for the Space Shuttle," *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Altos, California: Kaufmann Publishers, pp. 18–23, 1985.

[6] R. O. Yudkin, "On Testing Communication Networks," *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 5, pp. 805–812, June 1988.

[7] D. H. Brown and W. J. Hurd, "DSN Advanced Receiver: Breadboard Description and Test Results," *TDA Progress Report 42-89*, vol. January-March 1987, Jet Propulsion Laboratory, Pasadena, California, pp. 48–66, May 15, 1987.

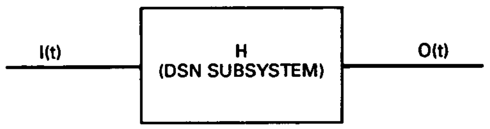

Fig. 1. Simple system model.

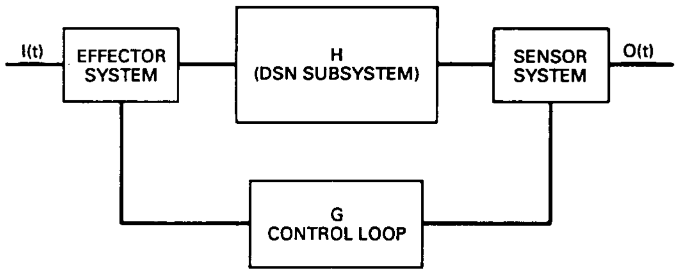

Fig. 2. System model with feedback control.

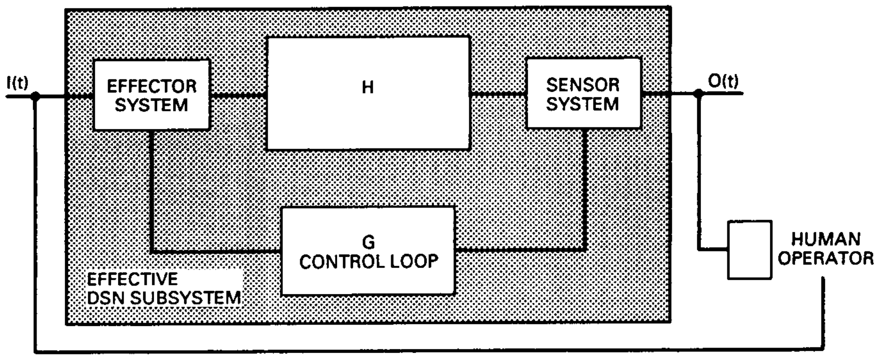

Fig. 3. Internal control loop with external manual control.

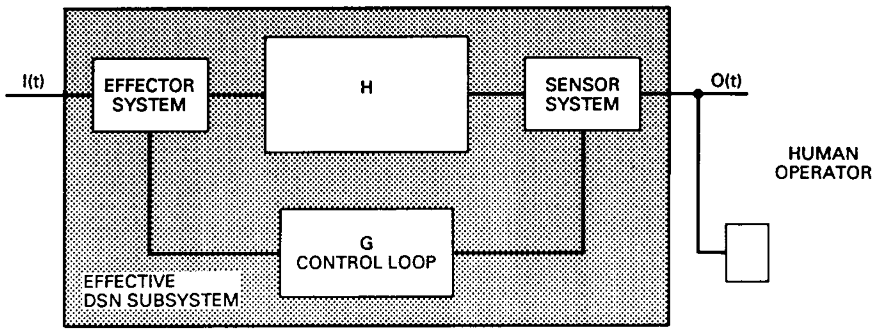

Fig. 4. Autonomous control loop.

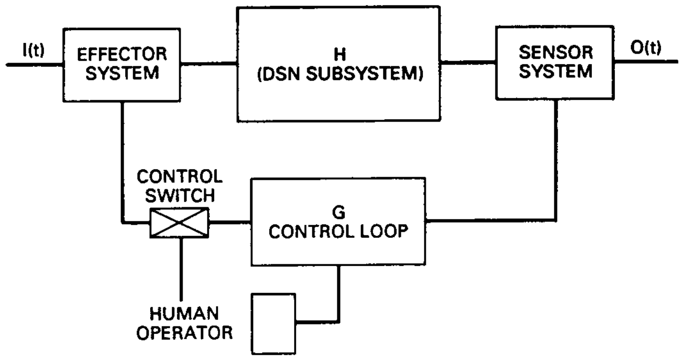

Fig. 5. Semiautonomous control loop.