N92-22719

OPERATIONS AUTOMATION USING TEMPORAL DEPENDENCY NETWORKS

Lynne P. Cooper
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109

ABSTRACT

Precalibration activities for the Deep Space Network are time- and work force-intensive. Significant gains in availability and efficiency could be realized by intelligently incorporating automation techniques. This paper presents an approach to automation based on the use of Temporal Dependency Networks (TDNs). A TDN represents an activity by breaking it down into its component pieces and formalizing the precedence and other constraints associated with lower-level activities. This paper describes the representations used to implement a TDN and the underlying system architecture needed to support its use. The commercial applications of this technique are numerous. It has potential for application in any system which requires real-time, system-level control and accurate monitoring of health, status, and configuration in an asynchronous environment.

INTRODUCTION

During precalibration (precal) of the Deep Space Network (DSN), operators configure the required subsystems (e.g. antennas, receivers, transmitters), download support data (e.g. standards and limits, pointing coordinates), and perform tests and calibration procedures. To do this, DSN operators send hundreds of directives and monitor the 1000+ response messages generated by the subsystems. Efforts have been made to automate portions of precals through software macros: creating a list of directives and then executing them through the macro rather than individually. This approach has fallen short due to two major limitations: (1) the inability to explicitly specify all possible contingencies, and (2) the lack of visibility into system status should the macro fail in-process.

Our approach is to represent the procedures as a temporal dependency network (TDN) where high-level procedures are represented as logical nodes of the network. The nodes consist of the directives needed to accomplish the task, temporal constraints, pre- and post- conditions, and local recovery information should the node "fail". The network specifies precedence relationships between nodes, any potential parallelism, and rules for recovering from global faults. The network is executed using a blackboard architecture which performs special monitoring functions so that the operator is always aware of the status of the equipment and the executing procedures.

The commercial applications of this technique are numerous. It has potential for application in any system which requires real-time, system-level control and accurate monitoring of health, status, and configuration in an asynchronous environment. This paper will first discuss the characteristics of the problem domain which led to the use of the TDN. It will then present the knowledge representation techniques used for the Temporal Dependency Network and the Domain Model. It will then discuss the system architecture of the Operator Assistant which uses the TDN to support semi-autonomous operations. Finally, it will discuss some of the lessons learned during the early prototyping stages and plans for demonstration of the technology in an operational environment.

DOMAIN ANALYSIS

DSN Link Monitor & Control

The operators at the DSN antenna complexes are responsible for setting up all of the equipment necessary to provide a communications path ("Link") between an individual spacecraft and its mission operations center[1]. In performing this function, the DSN essentially acts as a "bent pipe" -- passing through the information in a way which is philosophically similar to the way the Postal Service handles our mail. For the DSN, the goal is to deliver the data from the spacecraft (or commands to the spacecraft) as reliably as possible and for as many spacecraft as possible.

Currently, the DSN operators spend a significant amount of time performing the setup (precal) functions necessary to support a link. The Link Monitor & Control (LMC) operators initialize, configure, test, calibrate, and otherwise prepare the equipment by manually entering directives to each of the subsystems through an LMC console[2]. In order to effectively perform their jobs, the LMC operators must learn hundreds of different directives, interpret more than a thousand different messages, and evaluate 100 different information displays. This is more difficult than expected because each subsystem has its own unique "language", all directives must be entered in a cumbersome syntax through a keyboard, and the procedures themselves differ from antenna to antenna, and mission to mission. Furthermore, these procedures are not fully integrated. The spacecraft user guides provide a linear overview of the procedures necessary for their projects, but require that the LMC operators integrate other procedures available only from the subsystem handbooks. In addition, numerous changes to the procedures are received from engineering and science personnel and must be incorporated. To further complicate matters, the values needed to parameterize the directives for the specific track are also spread throughout volumes of documentation and numerous printouts of schedules and other support data.

A significant percentage of DSN operations time is spent performing precals. Reducing this operations overhead, and thereby increasing the availability of the DSN, is a major goal of the applied research presented in this paper. The two concepts which guide the effort are Positive Control and Situation Awareness.

Positive Control

In this context, positive control refers to the desire that all control actions (i.e. directives) have explicit feedback as to their effects . Under the existing LMC system, a limited set of messages report the status of directives. Unfortunately, these messages are lumped together with all of the other messages (monitor information, events, alarms, warnings, status announcements). It is up to the operator to "catch" the response as it scrolls by on a very busy text display.

Situation Awareness

There are two facets to situation awareness: (1) knowing the configuration, health, status, and readiness of the link equipment; and (2) knowing the status of the execution of the required procedures. The first of these is an information fusion problem requiring that the pertinent information be available and presentable to the operator. The second requires that the operator have an explicit high-level plan, and that the operator knows where s/he is at all times in the execution of that plan. While there is limited explicit feedback associated with the individual directives, the existing system suffers from the lack of explicit information on the "side effects" of the directives. For example, a directive which turns on the receiver will elicit a response saying that receipt of the directive has been acknowledged by the receiver. However, the operator must use other means to determine if the directive had the required effect, i.e., that the receiver did indeed turn on.

---

1 While the equipment which makes up the path is referred to as the "Link", the setup process is referred to as precalibration, and the actual beginning-to-end setup, communications, and tear-down is referred to as a "Track" or "Pass."

2 There are some subsystems which cannot be remotely controlled or monitored. For those subsystems, the operator verbally requests a technician in the actual equipment room to do what is necessary to configure, etc. the equipment.

Problem Summary

Successful automation requires that appropriate thought be given to the point at which the human operator is once again included in the system. Since DSN LMC is a complex, real-world environment, complete with ambiguity, incomplete sensor feedback, and creative combinations of circumstances which make it virtually impossible to specify all possible contingencies, our system is semi-autonomous, requiring a human operator in order to function fully -- but capable of reducing the amount of menial tasks require of those operators.

The requirements (goals) which drive our approach are:

1. Reduce manual input required of the operators (keyboard entries).
2. Provide a high-level representation of the task to be performed by integrating the necessary procedures.
3. Provide a means of easily accessing and integrating support data.
4. Parallelize the execution of the procedures.
5. Provide configuration and procedure situational awareness.
6. Support positive control (to the extent possible in the existing DSN architecture)
7. Support both autonomous and operator-in-the-loop recovery.
8. Provide fault detection, diagnosis, correction, and replanning.

KNOWLEDGE REPRESENTATION

Procedure Representation

The procedure necessary to accomplish a specific pass is represented as a Temporal Dependency Network (TDN). This is essentially a Petri Net which represents precedence relationships between procedures and which overlays time constraints. Each block in the network is a frame-type object which consists of its name, preconditions for its execution, the directives to be executed as part of the block, functions which identify how to fill in the needed parameters, time constraints (e.g. acquire the spacecraft at time, t), and the postconditions which exist after the block has been completed.

Each block represents a set of directives to be executed sequentially. The preconditions which must be met before the block can be executed are valid for the entire block. Similarly, the post conditions are valid only after completion of the entire block, although only one directive may be responsible for creating that condition. These decisions were made to simplify the execution of the TDN. In cases where existing procedures violated these requirements, the logical operations were broken down into component pieces to ensure that the pre- and post-condition representations were consistent with the design decisions described above. For example, if the existing, accepted way to perform a function requires two directives to be sent to the same subsystem sequentially (Turn on antenna hydraulics, Move antenna), but the second must wait for the first to be processed and the action invoked by it to be accomplished completely before beginning (It takes several minutes before the hydraulic system is primed and able to support moving the antenna); then, these two inter-related actions were separated into two separate blocks with explicit precedence relationships and additional constraints related to the system state.

The TDN is a network of the blocks as identified above. It presents the high-level overview of the task as an integrated whole, rather than as disassociated sets of directives. The network explicitly describes how operations of the individual subsystems must be integrated. The TDN also specifies what functions are mandatory for the given track, which are desired, and which should be done only under special circumstances. Figure 1 shows the top-level TDN for Very Long Baseline Interferometry (VLBI). Raw data from a VLBI pass is used to determine accurate positioning information for a spacecraft. Several procedures included in the network are desired, but optional. For example, the scientists analyzing the data want the operators to perform a coherency test prior to beginning the actual data collection. This function is not essential; even if it is not performed, the scientists will still get usable data. However, if the coherency test is performed, they can use the information from the test to better process the data. The TDN differentiates between the precedence relationship between this block and others in the network. Under nominal circumstances, the coherency test block would be executed. If, however, circumstances require a quicker completion of the network as a whole, that block can be bypassed with an acceptable impact on the schedule.

Domain Model

The domain model is the equipment analogue to the procedures represented in the TDN. The domain model contains a description of each piece of equipment which is used in the link. This description includes its configuration (e.g. switch settings), and performance parameters (e.g. Receiver In Lock, Signal to Noise Ratio). The domain model representation is tightly coupled to the TDN because the status of the link is a function of the operations being performed through the TDN. Therefore, the domain model can be thought of as a time series of snapshots which correspond to the expected state of the system as it responds to actions invoked by the different directives.

The domain model also represents the link equipment at higher levels of abstraction than simple configuration and performance parameters alone would allow. For example, the distinction between being just "assigned" to the link vs. "operational" in the link is made. (If equipment is *assigned*, it will accept directives from the LMC operator. When it is operational, it is actively performing its function within the link.)

OPERATOR ASSISTANT ARCHITECTURE

The architecture developed to use the TDN representation is shown in Figure 2. The architecture is interrupt-driven and interprocess communications are in the form of events and messages. Data is passed through a combination of shared memory and datagrams. The primary modules are the Execution Manager, the Response Manager, the Status Manager, the DSN Communications Interface, the Message and Event Router, the Diagnostic Module, and the User Interface. Each module (except for the Comm I/F) contains domain knowledge which is used to guide the decision making processes of the module. The role of each of these will be discussed in the following paragraphs.

Execution Manager

The Execution Manager is responsible for executing the TDN: determining which blocks to activate, setting up queues to handle the directives from the individual blocks, assessing whether preconditions have been met, and responding to any operator overrides or recovery replanning. The EM is an interrupt-driven process. Its primary function it to keep things rolling -- to ensure that the parallelism inherent in the operations procedures is exploited as fully as possible, thereby reducing the amount of time necessary to perform precals.

DSN Communications Interface

As its name implies, the Comm I/F module is responsible for interfacing the Operator Assistant to the operational DSN. Because the communications models used by the DSN are significantly different (philosophically, electronically, logically, ...) from those used internal to the Operator Assistant, this is an extremely important module. It is responsible for physically connecting to the DSN Local Area Network (LAN), accessing information meant for the actual LMC, stripping away the DSN communications protocols and reformatting the data for use by the Operator Assistant. Anyone who has built an extension to an existing system, especially an extension which uses a different design philosophy than the existing system, knows how difficult bridging the gap can be.

Message and Event Router (MER)

The MER takes the output from the Comm I/F and routes it to the appropriate modules within the Operator Assistant. There are several types of messages that the Operator Assistant needs to receive. The MER is responsible for interpreting the message type and parsing the message to extract the important information (and therefore minimize the amount of intermodule traffic). In general, the MER will take the message content and place it in shared memory, then send messages to the appropriate module(s) instructing them to view the message. This approach is currently under evaluation.

Response Manager (RM)

The Response Manager is responsible for matching the directive responses received from the subsystems with the directives sent by the Execution Manager. While most directives have single message responses, in some cases, the subsystem will respond first with an Acknowledgement or Processing message, followed by a Completed

message. In cases of multiple responses, the RM is tasked with ensuring that all responses are attributed to the appropriate directive. The RM is also responsible for detecting timeouts. A response to each directive is expected within a finite length of time (on the order of 5 seconds). If a directive has not been responded to within that limit, the RM must detect the timeout and send an event message to the Execution Manager. In many cases, these timeouts are false alarms, either the LAN lost the message carrying the response, or the time limit was set to an arbitrarily short time. The RM will incorporate knowledge on how to interpret the timeouts.

Status Manager (SM)

The Status Manager is tasked with keeping an up-to-date model of the Link. The SM has a time-tagged model of the expected status of the domain and also maintains an up-to-date model of the actual state of the link. The SM compares the two, notes discrepancies, and takes appropriate action (ignore, send to diagnostic module, ...). The status manager has to accommodate the latency effects inherent in the system. The delay times between the request for action (the directives) and their implicit and explicit effects on the Link are affected by a number of factors such as LAN utilization, subsystem health, physical conditions, and availability of personnel to manually perform a subsystem function. The SM also employs a countdown clock driven by a timed event stack, which provides a means with which to evaluate the overall progress towards completing the precalibration and to determine the probability of success and/or need for corrective action. In general, DSN operations have hard physical constraints such as a limited time window during which a particular antenna can see a spacecraft. The precalibration activities must be complete by the time the spacecraft comes into view, or valuable time may be lost. Since these specific times are known, and the TDN representation allows for estimates of how long activities take, the due times can be propagated backwards to provide a means of determining when an activity is behind schedule.

Diagnostic Module (DM)

The Diagnostic Module incorporates three tightly-coupled functions: (1) Fault Detection, (2) Diagnosis, and (3) Recovery. The other processes initiate the fault detection and diagnosis capability within the DM. The DM pulls information from the other processes as well as the Domain Model representations to accomplish these functions. It then influences the other functions through its recovery module, which contains both correction and replanning functions. Correction functions implement recognized contingency plans. Corrective actions suspend the TDN (or a part of it), jump to an external (to the TDN) procedure, perform those functions, then resume execution of the TDN. To the TDN, correction functions appear to be simple delays.

Replanning is necessary when an event occurs which invalidates a previously accomplished section of the TDN, or requires that TDN execution follow a path different from the nominal one. For example, if a piece of equipment in the link fails, but can be recovered in its pre-failure state by a simple warm start , corrective logic can be invoked. If, however, that piece of equipment fails completely and must be replaced by a brand new piece of equipment, all of the configuration and calibration activities which had taken place using the old equipment must be redone on the new equipment. This requires replanning logic. Assume further that this equipment is optional within the link and the time window constraints for acquiring the spacecraft will be violated before the new equipment is ready. Under these circumstances, the replanning mechanism would be invoked to execute an alternate path of the TDN.

User Interface (UIF)

The final, and probably most critical part of the Operator Assistant, is the User Interface. The UIF will provide the LMC operators with visibility into the automated functions as well as the ability to interrupt, override, or influence those functions, and a mechanism for interacting with all the other elements through graphical user interfaces which support a variety of presentation and input modes. One of the primary functions of the UIF is to make it easier for the operators to do their jobs and to ensure that their ability to function is not unnecessarily or arbitrarily limited by the automation meant to help them. In designing automated systems, there is an almost overwhelming temptation to "fix" individual problems, rather than to view the system as a whole. A good user interface ensures that the human element of the system is as fully integrated as the electronic and mechanical elements.

STATUS AND LESSONS LEARNED

Status

The Operator Assistant has gone through one prototyping phase and is currently being redesigned to address problems which surfaced in the earlier version. A new prototype, built to the architecture presented in this paper is in development and will be tested using DSN compatibility test facilities during the early part of 1992 and demonstrated at an operational site later in 1992. The TDN for VLBI has been reviewed and revised to accommodate additional information. The databases for the directives and their responses are complete and work is now beginning on building the domain model. Currently, the VLBI TDN is the only one in existence. During late 1992, we will conduct an extensibility analysis of the Operator Assistant approach. This analysis will address two problems: (1) how to extend the Operator Assistant to support other types of activities; and (2) how to extend the Operator Assistant to support multiple activities at the same time by one operator. In FY93, following its demonstration within the operational DSN, the Operator Assistant will be moved to the DSN experimental antenna site where it will undergo sustained testing in an operational environment.

Lessons Learned

The past 1-1/2 years of work on the Operator Assistant have yielded some significant results. The first prototype served as a proof of concept -- and also provided some insights which will make the second prototype much more effective. The following are some of the lessons learned highlights from the initial effort.

1. For our domain, a polling-type architecture was inappropriate. During the first prototype, we attempted to make use of a blackboard architecture used to support a robotic vehicle. For that domain, a polling architecture worked well because each of the sensors was continually reporting data, therefore, there was always a data value to be read at each sensor. The Operator Assistant domain, however, was much more asynchronous in nature. Processing time was being wasted polling functions which rarely had anything to report. Also, execution of the TDN was intended to be dynamic so that as preconditions were satisfied, blocks would execute. Processes were spawned and killed routinely and it was difficult incorporating the dynamics of our system into a polling architecture.

2. Integration into an existing system always has "Gotcha's". Although we began our effort with a proven communications interface already in existence and a healthy respect for the difficulties associated with connecting to the DSN -- it was even more difficult than expected -- and for "gotcha" types of reasons: The test area configuration and LAN changed; equipment was down for repair; formal Test Plans were needed afterall; the facilities were not available during normal working hours; a 6-month gap existed between actual operations of the type we wanted to automate; there were serious spacecraft problems, etc.

3. TDNs are useful as a review of operational procedures. During knowledge engineering sessions, the use of a mechanism which explicitly showed the dependencies and parallelism in operations procedures actually uncovered some errors in existing procedures. In some instances, problems surfaced because the subsystem engineers were using a linear method (a listing of directive sequences, complete with GO TO's") to represent a parallel process. Other times, the problems arose because inter-subsystem constraints were not appropriately addressed.

4. TDNs are useful for integrating different operational perspectives. Operational "magic" happens when the efforts of four different groups come together to accomplish a specific purpose: (1) the scientists that use the data; (2) the engineers who design the equipment; (3) the operators who make it all work; an (4) the technicians who fix it when it breaks. In developing the TDN, we met with people from each group -- and each individual had a unique view which enabled the TDN to have a much richer and ultimately more accurate representation.

5. Even in automated systems -- the human element must be integrated as part of the design. In laboratory test cases, it is possible to descope a domain to the point where what remainst can be fully addressed by a system you're building. That luxury doesn't exist in real-world applications. There are always surprises and ambiguities that will crash your system. Building systems that are robust enough to not die -- or at least to die gracefully -- is difficult. But it can be made easier if the human element is integrated as part of the design. For example, the first Operator Assistant prototype ran into network communications problems during a demonstration which we were powerless to overcome due to the fact that there was no interrupt capability available for the operator in that version (high on our list of requirements for the current version!).

6. Make allowances for "system slack" in what you design. In the original prototype, we planned on using the Response Manager to match up both the explicit responses for the directives and the implicit system responses. This approach would have seriously clogged the execution of the TDN because of the latency effects resident in the system - and the rather broad definition of what is nominal performance. Rather than attempting to resolve the ambiguity in absolute terms, we have since decided that we can instead relax the system by using two functions, the Response Manager and the Status Manager.

7. The work put into developing automation is a good foundation for a training system. And vice versa. An independent effort evaluating intelligent training techniques in the same domain as the Operator Assistant began several months ago. By approaching the problem from two different angles, we have set up a very rewarding exchange with the training team. Their foundation work, particularly in the area of the Domain Model is especially applicable.

8. The Absolute Truths.

a. Know and respect your user(s). Remember that they're the ones who will ultimately determine if your system is successful -- and they can help you avoid mistakes.

b. Don't automate the fun stuff. Automation is much more successful if you attack the mundane aspects of the job.

c. Be realistic in your expectations. The operations folks have heard it all (and then some) before. Don't make promises you can't keep. Build a solid foundation and then add to it.

d. Keep in touch with your users. Get their feedback as often as reasonable,

e. Don't bother your users unnecessarily. Remember that they have jobs to do. Do your homework so that you make the best use of the time they can give you.

SUMMARY

The approach to automation described in this paper centers around using Temporal Dependency Networks to represent the procedures to be automated. The commercial applications of this technique are numerous. It has potential for application in any system which requires real-time, system-level control and accurate monitoring of health, status, and configuration in an asynchronous environment.

ACKNOWLEDGEMENTS

The Operator Assistant team has benefited from the contributions of numerous operations, engineering, and scientific personnel, as well as from development team members Elmain Martinez, Juan Urista, Joe Jupin, Rajiv Desai, Lorrine Lee, and Randy Hill.

The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

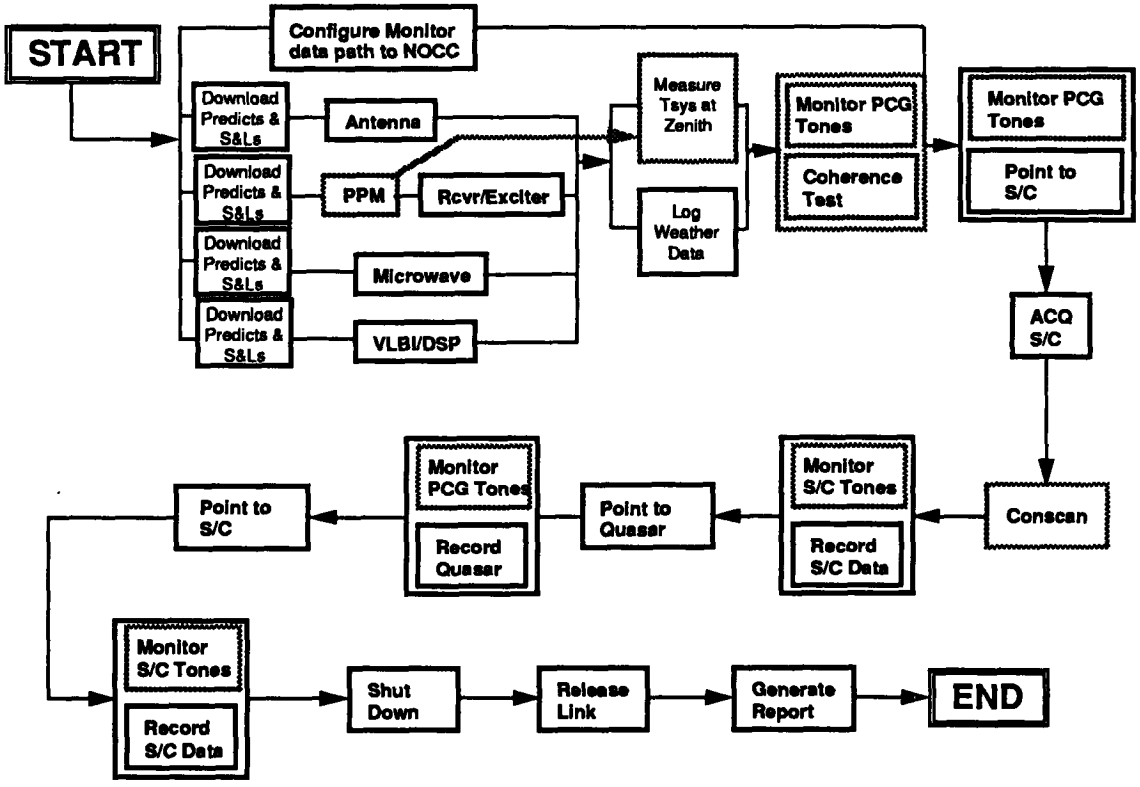

Figure 1. VLBI Temporal Dependency Network

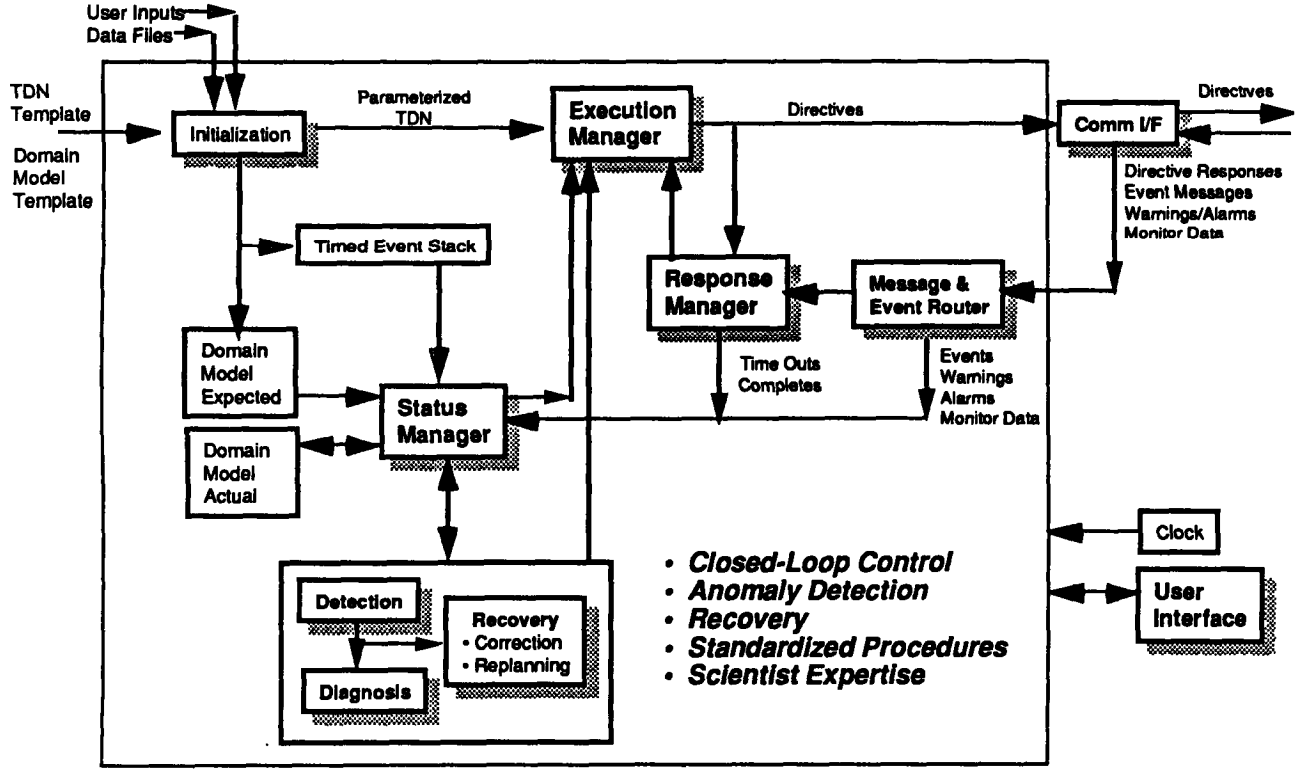

Figure 2. Operator Assistant Architecture

ELECTRONICS

(Session E3/Room B4)

Thursday December 5, 1991

- **Thermoacoustic Refrigeration**
- **Ambient Temperature Recorder**
- **Fiber-Optic Push-Pull Sensor Systems**
- **Commercial Capaciflector**

