DESSY: Making a Real-Time Expert System Robust and Useful

Sherry A. Land*
Jane T. Malin
NASA Johnson Space Center
Intelligent Systems Branch - ER22
Houston, TX 77058
(713) 483-2064

Donald R. Culp
Rockwell International
Remote Manipulator Section - DF44
Houston, TX 77058
(713) 483-0891

ABSTRACT

As the complexity and expected life-span of modern space systems continue to increase, the need for real-time data monitoring and failure analysis becomes more critical to their successful operation. DESSY, or DEcision Support SYstem, is a joint effort by the Intelligent Systems Branch/ER2 and the Remote Manipulator System (RMS) Section/DF44 to develop an expert system for the monitoring of the Payload Deployment and Retrieval System (PDRS). DESSY users, the RMS flight controllers, are provided with user interface enhancements and automated monitoring of system state (physical orientation) and status (operational health). Currently, a DESSY prototype for the Manipulator Positioning Mechanism (MPM) and Manipulator Retention Latches (MRL) of the PDRS has been developed and successfully demonstrated during the STS-49 and STS-46 missions.

Expert systems for monitoring real-time operations must not only accurately represent domain knowledge, but also address the challenges of using unfiltered real-time data as input. This paper describes the methods and design strategies developed to overcome problems with real-time data in the NASA Mission Control Center. Types of data problems addressed are (1) loss of data, (2) erratic data and (3) data lags and irregularities during state transition. Methods used to handle data problems include rule disabling for ignoring data when data quality is uncertain, context-sensitive bounded pattern recognition for minimizing incorrect conclusions based on bad data, and graceful recovery through system correction when reliable data returns. This combination of methods with an object-based modular DESSY design assures a robust program capable of lengthy periods of uninterrupted use in operations.

I. INTRODUCTION

As modern space systems become more advanced, the need for intelligent computer-assisted support during operations continues to grow. The support required usually begins with real-time data monitoring and includes failure diagnosis and possibly recommended actions. As a result intelligent software must continue to evolve to meet these demands. In this paper we discuss the development of the DEcision Support SYstem (DESSY), an expert system which aids flight controllers by performing real-time data monitoring and intelligent evaluations of system hardware through assessments of telemetry data patterns and transitions. Although we are documenting many aspects of the development process, we focus this discussion on the most challenging issue we faced - dealing with unreliable real-time telemetry data. We briefly address other development topics including knowledge base organization, object and rule structure and user interface enhancements.

II. THE DESSY EXPERT SYSTEM

Project Background

The DESSY project is a joint effort by the Intelligent Systems Branch/ER2 and the Remote Manipulator System (RMS) Section/DF44 of Johnson Space Center to develop an expert system for monitoring the Payload Deployment and Retrieval System (PDRS). The PDRS is made up of several subsystems, each of which will correspond to a DESSY module. Currently, a DESSY prototype for the Manipulator Positioning Mechanism (MPM) and Manipulator Retention Latches (MRL) of the PDRS has been developed and successfully demonstrated during the STS-49 and STS-46 missions. MPM's are pedestals that roll the shuttle arm in and out of the payload bay. MRL's are latches which latch the arm in place when it is not being used. The next subsystem being undertaken is the Remotely Operated Electrical Umbilical (ROEU), a system to provide an electrical interface between the orbiter and a payload while the payload is in the payload bay. Other RMS subsystems to be implemented as DESSY modules include the End Effector, Arm-Based Electronics, and Displays & Controls Panel.

C-C

Software Design

DESSY is implemented in a commercial real-time expert system development environment. Its object-oriented data structures represent the RMS system design and its rules capture the logic of system operations and failures. Modularity within the software separates rules for data monitoring, state transition detection, failure diagnosis, expert system control and user interface. This partitioning of rules allows the enabling or disabling of appropriate groups of rules when necessary, as in the case when unreliable telemetry data enters the system, or for changing system contexts. For example, as the MPM/MRL system reaches a new configuration in nominal operations, state transition rules monitor this change of state and DESSY changes to a new context. As a result, the appropriate set of diagnostic rules is enabled for the duration of this context.

The DESSY software design is further modularized in that each RMS subsystem will be implemented as a separate software module, able to act independently of the others. Each subsystem module will have a rule set and display developed specifically for that RMS subsystem. A summary display called the Integrated Status Display will provide an overview of all subsystem states and statuses. Figure 1 shows how each of the planned subsystem modules fits in the overall DESSY design.

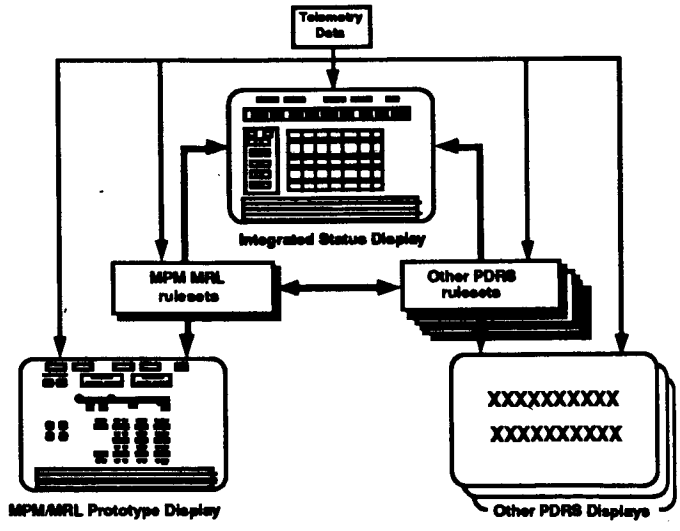

Figure 1. The DESSY Design

Careful object and rule design, along with appropriate modularity, both in subsystem partitioning and in the separation of rules according to rule functionality, has played an important role in the successful development of DESSY. However, even if the software provides objects and rules that correctly capture the functionality and logic of the physical system, failure to address the issues concerning real-time data as input can lead to disappointing system behavior. In the remaining section of this paper we discuss problems observed and solutions developed for working with real-time data. It is these methods that have contributed most to the robustness of DESSY.

III. WORKING WITH REAL-TIME DATA

Although many expert system projects deliver a knowledge base which accurately represents the domain of the problem, they generally fail to successfully address the challenges of using unfiltered real-time data. We present several types of problems that occur when using real-time data as input to a monitoring expert system. Although all telemetry sensor data sent to DESSY is binary, these problems could occur with any data format. The data problems addressed include (1) loss of data, (2) erratic data and (3) data lags and irregularities during state transitions. Each of these problem types, discovered during DESSY development and testing, is discussed.

Following the overview of data problems, we examine the solutions developed to account for these problems. Table I lists each of the data problems with their applicable solutions. These methods, in the order of increasing sophistication and potential benefit to real-time operations, include rule disabling to ignore data when data quality is uncertain, context-sensitive bounded pattern recognition for minimizing incorrect conclusions based on bad data that has entered the system, and graceful recovery through system correction when reliable data returns. Each method works independently to prevent or correct erroneous expert system conclusions resulting from bad or missing data. It is their combination, however, that assures a robust program capable of lengthy periods of uninterrupted use in operations.

Data Problem	Solution Methods
loss of data	rule disabling, graceful recovery
erratic data	pattern recognition, graceful recovery
data lags and irregularities during state transition	pattern recognition, graceful recovery

Table I. Data Problems and Solutions

Types of Data Problems

Loss of Data

The most common and well understood data problem is a loss of data. Data loss usually takes the form of a complete loss of signal (LOS) and may be either expected, i.e., the shuttle enters the Zone of Exclusion (ZOE) where there is no telemetry downlink, or it may be unexpected. Unexpected data loss may occur as a complete or partial loss of data, usually due to ground processing or computer hardware problems.

Although LOS seems like a simple concept to account for, hardware implementation of telemetry processors can complicate the situation. Depending upon how a particular telemetry processor handles periods of LOS, the monitoring system may receive an inactive state for all data values, no data values at all, a static frame of the last data values, or as in the case of the DESSY data source, the Real Time Data Systems (RTDS), the last 4-5 seconds of data may be repeatedly replayed until the signal returns. Yet another possibility is that nonsense data is received during periods of LOS. The first step in dealing with loss of data due to signal loss is to find out the form of data that will be received during this time.

Once it is understand what LOS will mean for a monitoring expert system, it must be determined how that system is going to detect it. One possibility is to compare actual data format with expected format for each data frame received to determine quality. In the DESSY project we were fortunate enough to have a data quality measurement from the telemetry processor. OI-Quality indicates the telemetry processor's assessment of data quality for each data frame. It was discovered, however, that OI-Quality did not always reliably reflect true data quality. Often there seemed to be a lag between the data quality drop and OI-Quality's reflection of this drop. At other times, even when the lag did not appear, the low quality indication occurred in the same data frame that included bad or missing data, making it impossible to filter the data or to alert the system of its presence. Inevitably, bad data due to an LOS situation would periodically enter the system.

Later in the paper we present the methods developed for dealing with LOS problems, such as disabling of rules and recovery methods. These methods apply to both the case when LOS is detected in time to prevent problems from occurring and when bad data due to LOS enters the system.

Erratic Data

Erratic data is unstable and does not meet expected behavior for a given operational context. Erratic data may occur at anytime, but is most likely to be seen immediately before or after an LOS or at the time of state transitions. Erratic data is characterized by frequent flipping of bits (in the binary case) in a particular data set. Bit flipping occurs when a data value toggles or flips between values of 0 and 1. Of course this signature may also result from intermittent sensor failures, and that possibility should not be ruled out. For large sets of data, however, it is much more likely that any bit flipping is due to bad telemetry data, rather than bad sensors.

For DESSY, periods surrounding an expected LOS seem to be a common time for erratic data to appear. As the shuttle moves in or out of a satellite's range, the telemetry link has a period of degradation, during which the OI-Quality has not yet dropped. The result is often a significant amount of bad data entering DESSY. Because there has been no previous low quality indication, DESSY has no clue that a data frame from this scenario contains degraded data.

The second situation when erratic data often occurs is near the beginning and end of a state transition, when DESSY frequently encounters bit flippings of data. Although this is an instance of erratic data, this scenario is discussed in the following paragraphs concerning data behavior during state transitions. Later we discuss our use of context-sensitive bounded pattern recognition and graceful recovery to deal with both these types of problems.

Data Lags & Irregularities During Operational Events

The final type of data problem we discuss results from unexpected data activity when data is expected to change because of an operational event such as a state transition or commanding. Data that is expected to change may "flicker" or "lag" before reaching a new stable state. Given a set of data that is expected to change at transition time T, subsets may flicker or lag, causing the data transition to occur over some delta time t. Typically delta t is 1-3 seconds. The following transition graph depicts five examples of data transitions from low to high values, including a normal data transition and four anomalous cases. The delta time for this data set is 2 seconds because it is the time it took for every piece of data in the set to change to its new expected state.

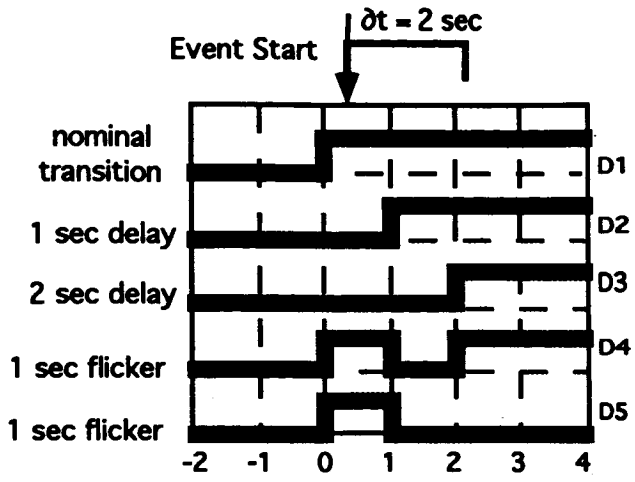

Figure 2. Examples of Data Lags and Irregularities

Data lags and irregularities during events and state transitions may occur because of noise in the telemetry or may be due to the physical properties of the hardware being monitored. In either case there is no smooth transition. Failing to include this behavior in the system's monitoring rules will often lead to erroneous conclusions. Examples from early DESSY work include (1) incorrect sensor failure conclusions at Event Start T due to data lags - D2 and D3, (2) state value fluctuation from T to T+2 from flickering data - D4, and (3) incorrect command conclusion at time T due to a temporary active value in command telemetry - D5. These problems were corrected using our data handling techniques.

Methods Used to Handle Data Problems

Rule Disabling

The most straightforward method of dealing with data of uncertain quality is to ignore it by not responding to changes in that data. In any system there will be times when data quality is so low that the data should not be used at all. The expert system should therefore have the capability to ignore data when it is unreliable by a method such as disabling of rule sets. This immediately highlights the need for a manner of determining faulty data, such as the OI-Quality measure provided to DESSY. When the value of OI-Quality is any number other than 100, certain diagnostic and state transition rule sets are disabled in DESSY. When quality returns the rules are again enabled.

Although the above tactic seems simple enough, it has the problem of the quality number and corresponding data being in the same time frame, making immediate filtering or rule disabling impossible. Also as the shuttle enters or leaves the Zone of Exclusion, data deteriorates, making OI-Quality itself unreliable at that time. However, even though erroneous data may have already entered the system, it is still desirable to disable rules to minimize the number of faulty conclusions. Event rules such as command and state transition, and diagnostic rules should be disabled, but corrective rules should not be disabled. Corrective rules will be discussed in the section on graceful recovery.

Disabling rules is the simplest of the techniques we use in dealing with real-time data problems. Unfortunately, it effectively eliminates the usefulness of the expert system during the time the rules are disabled, retaining only its function as a raw data monitoring source. For this reason the technique is only used when absolutely necessary - when it is certain that quality is low and data is unreliable, such as in a complete loss of signal.

To supplement this automatic rule disabling, the DESSY user may also "turn off" the expert system portion of DESSY at any time, leaving DESSY to act as only a data monitor. Actually this turning off merely grays out the parts of the DESSY display that present expert system conclusions, allowing DESSY to continue to work in the background. Even if DESSY has made incorrect conclusions and the user has grayed out the expert system part of the DESSY screen, the built in corrective rules should eventually lead to a graceful recovery. The intent is that even if DESSY has been turned off due to erroneous expert system conclusions, it will recover by itself and the user will once again be able to use the expert system part of DESSY. Nonetheless, this feature gives the user the opportunity to override the expert system at the level of the user interface.

Context-Sensitive Bounded Pattern Recognition

The second technique we have implemented to assure DESSY's robustness deals with characteristics of the sets of data that DESSY's rules use to detect events and identify failures. Because of problems with data lags and irregularities, the expert system often has insufficient or even erroneous evidence from the data set upon which it can determine the occurrence of an event. Also because of the nature of space operations, there is often an insufficient amount of sensor data available, making event determination even more difficult. For example, if the event is that of an *MPM stow*, there are only two sensors to indicate the stowed state once the MPM has reached its new stowed position. This data set is insufficient to determine with certainty the stowed state because of a requirement we have that DESSY must continue to monitor events, given a single data failure in any set of data

we consider. If one of these two pieces of data was active and the other inactive, as is the case of a single failure of either of these pieces of data, the state would be inconclusive. The data must therefore be supplemented with additional information.

In addition to considering data set size, when appropriate we include context such as the system state or the detection of a prior event. When context is considered, we say that we are using a context variable (CV) in the rule. Use of CV's lowers the requirements of the data set the rule must consider, and some rules use only CV's. Table II gives an example of a DESSY rule using two pieces of data and two context variables.

We now discuss the general guidelines developed in determining necessary data sets and context variables for DESSY rules. The lower bound or minimum requirements needed to make reliable conclusions is addressed first, followed by a brief discussion on the upper bound or maximum data set recommended. Establishment of a lower bound is necessary because enough data must be observed to correctly monitor an event, given that some predetermined amount or percentage of the data set being considered is bad. An upper bound is established because of the impact the data set size has on computer hardware performance. An example from DESSY is provided.

DESSY Rule: state transition*	Data/CV set
if the state of any MPM is *stow-in-transit*	CV
and (the sys1-stow-microswitch = *1*	data
or the sys2-stow-microswitch =*1*)	data
and the command of MPM-SYSTEM is *stow*	CV
then conclude that the state of the MPM is *stowed*	conclusion

Table II. Rule Example with 2 Pieces of Data and 2 Context Variables

Establishing a Lower Bound

Given an operational event there exists a set of data S that the expert system directly monitors to determine when the event occurs. In addition, there exists a second usually larger set S', a superset of S, that makes up the context in which the event will occur. The set S' indicates state, status and any other operational context of the system being monitored and includes both data and context variables.

The CV's from S' in the example of Table II are the current MPM state and the current MPM command. Thus we have the following sets.

S_{stowed} =
{sys1-stow-microswitch, sys2-stow-microswitch}

S'_{stowed} =
{sys1-stow-microswitch, sys2-stow-microswitch,
current state, command}

If the set S were the only data we could consider, we would have to impose the constraint that both data elements be active to eliminate the ambiguity and ensure that we were in a stowed state; however, this does not meet DESSY's requirement to continue to monitor given a single failure. Because in the extended set S' context is considered, we can relax our constraints to require only one of the two microswitches to be active in determining the stowed state. We have $|S_{stowed}| = 2$ and $|S'_{stowed}| = 4$ where $|S|$ denotes the number of elements in the set S. Because the set size of 2 is insufficient for determining the event given our requirements, the data set S must be expanded to S' providing a more plausible set.

Even in the case where 3 pieces of telemetry are available, the data is probably insufficient. Although in actual operations, a double hardware failure must occur to lose 2 of the 3 sensors, in a situation where noisy data is occurring, the possibility that 2 of these 3 may erroneously become "turned on" is fairly high. In this situation a larger data set including context variables is desirable. CV's such as state and command in the example above, can be used to obtain the minimum information set by imposing physical constraints or providing the current system configuration. CV's limit the scope of a rule which in turn limits the chance that it will fire incorrectly given it has received bad data.

Although we use CV's to impose physical constraints in DESSY, procedural constraints are not used in DESSY rules, since humans are too likely to break procedural rules. We learned this while testing an earlier DESSY version when a flight rule was broken during a training simulation. The software failed to monitor the operations due to a procedural constraint embedded in DESSY's monitoring rules. Therefore, as a policy, we implement no procedural constraints in DESSY.

Establishing an Upper Bound

In real-time operations, every piece of data observed has an associated cost in CPU time. Thus we wish

* MPM Stowing is the rolling in of the RMS to put it away. Deploying is rolling out the RMS.

to impose limits on the amount of data DESSY is allowed to inspect in a given rule. The best way to implement this is through context variables which hold summaries of data values and are usually already stored in DESSY for other purposes. Use of CV's allows us to consider performance constraints to limit the size of the data sets we inspect. In some cases a CV can take the place of all but one piece of data. In other higher level rule firings, such as for summary level information, conclusions are made based only on context variables.

We have established the upper bound of a rule as 4 pieces of telemetry data or 2-3 pieces of data given a CV. The real-time expert system should *usually* not be overloaded with a larger set, although there will probably be exceptions in safety critical or unreliable areas. In conclusion, context-sensitive pattern recognition with appropriate set size, including both data and context variables, reduces the chances of incorrect expert system behavior when bad data is present.

Graceful Recovery Through System Correction

Although many systems attempt graceful degradation in the face of trouble, DESSY has extended this concept to one of graceful recovery. If the system makes faulty conclusions because of bad data, a set of corrective rules will "kick in" once good data returns, and restore the expert system to the proper configuration. This includes both state correction and status correction. The system does not have to be restarted by the user because the corrections automatically restore offending parts of the knowledge base.

Corrective rules are similar to other system monitoring rules, except that they do not allow for any inconsistencies in the data sets they observe when making conclusions, i.e., every piece of data in the set for which the rule applies must be exactly correct. Additionally, corrective rules are written only for the cases that it can be determined with certainty that the system is in a particular configuration. These rules, therefore, can be thought of as the axioms of the expert system. They can be as simple as determining that a single piece of data is reliable again because it returned to a legitimate value after it had previously been deemed as unreliable. A more sophisticated example is the re-evaluation of system state when all microswitch data for a new state becomes active.

Examples of corrective rules are provided in Figures 3 and 4. The rule in Figure 3 shows the re-evaluation of the status of a single microswitch. If the microswitch status is *questionable-on* because the microswitch is inappropriately active, but the microswitch value returns to inactive (0 or off), the status is reset to *functional*. Figure 4 is an example of the re-evaluation of the state of MPM's. If the state is not stowed, but both stow microswitch indicators become active (1 or on), then the MPM state is reset to stowed.

If the status of sys1-stow-microswitch is questionable-on
and the sys1-stow-microswitch = 0
then conclude that the status of sys1-stow-microswitch is functional

Figure 3. Corrective Rule: Microswitch Status

if the state of any MPM is not stowed
and the sys1-stow-microswitch = 1
and the sys2-stow-microswitch = 1
then conclude that the state of the MPM is stowed

Figure 4. Corrective Rule: MPM State

The example of Figure 3 may occur when an MPM stow microswitch (normally indicative of the stowed state) becomes active while the MPM is in the deployed state. This event could be due to a microswitch failure, but is more likely to be caused by bad telemetry. When this event occurs, several conclusions are made. First the microswitch is determined to be *questionable-on*. Assigning this value is a strategy to allow the user to realize DESSY is at this point making only a tentative conclusion about a failure of this microswitch. If the microswitch remains active, it will cause a motor to be inhibited, preventing that motor from driving if a stow command is given. (However, there are two redundant motors for stowing, and the other motor will still operate.) Secondly, because the MPM is deployed and a stow microswitch is *questionable-on*, the status of the MPM will be updated to *Expect-Single-Motor-Stow*. If before stow operations occur, the microswitch returns to its inactive state, *questionable-on* is removed by the rule in Figure 3 and the MPM status is restored to operational.

If stowing operations proceed and the motor is inhibited by the microswitch, procedure monitoring rules will notice this fact. The MPM status will be updated to *Single-Motor-Stow* once the stow is complete and it has been confirmed that only one motor was operational. However, if the motor is not inhibited, but the *questionable-on* microswitch was due to bad telemetry, procedural monitoring rules note that the motor performed nominally and the MPM status is adjusted to *Nominal-Stow*. The microswitch status, however, will remain as *questionable-on* until the microswitch value changes to inactive.

A benefit of self correction is demonstrated in a scenario observed during STS-49. Because of hardware problems, a significant amount of data frames were being lost, so that DESSY did not receive data for seconds at a time. Unfortunately, those were crucial seconds in which an MRL state transition was taking place. DESSY was unable to monitor the transition procedure because it never received the data. Fortunately when DESSY did again receive a data frame, although the procedure was over, it was able to evaluate the current data and reflect the new MRL state. Another example of why graceful recovery is crucial for real-time expert systems occurred during MRL release operations during STS-46. Seconds after the release began, an LOS occurred and the telemetry downlink was lost. This LOS lasted for approximately seven minutes, and when data returned the release was complete. At this time DESSY evaluated the new data and reconfigured itself to reflect the new MRL state. In both these cases, it was unfortunate that we were not able to monitor the operations, but the fact that DESSY was able to keep up once data returned prevented a situation where the wrong state would be reflected, certainly causing a loss of user confidence.

Additional examples of how the corrective rules were crucial to system success were demonstrated near other times of LOS. Because poor data quality is not always detected soon enough, unfiltered erroneous data enters the system and leads to faulty conclusions. Even if it were possible to filter out the bad data as soon as it was detected, the use of unfiltered data is desirable because it gives the flight controller a better understanding of the downlinked telemetry. The expert system interpreting the data, however, is unlikely to be correct and should provide an assessment that its interpretation is suspect. Regardless, once quality is nominal again, the expert system should re-evaluate the scenario and adjust any faulty conclusions it made. State should be re-evaluated and any status discrepancies that are no longer accurate should be removed.

There is a final area in which corrective rules are utilized. This case takes place when a real failure (or even a single data or sensor failure) occurs and is later followed by recovery. The corrective rules are used to restore the status to nominal in these situations. In these cases when a failure actually has occurred, the human would like to know the failure history even if the failure is corrected. (This is currently a future DESSY version enhancement.) A history is kept in the form of a message list at present.

We have found these correction features not only to be very useful, but to have good side effects. Complementing the correction rules (and often actually the same rule), initialization rules allow DESSY to be started with any stable MPM/MRL configuration and to be initialized to the proper states and statuses. The rule in Figure 4 would initialize DESSY's MPM state to *stowed* if DESSY were started with the MPM in the stowed position. If DESSY is started during transition periods, once the transition is complete and the system is again static, DESSY will at that point initialize itself. This has been an important and even necessary feature of our real-time expert system.

In summary, it is necessary to keep the system from making lasting erroneous conclusions based on bad data. When working with real-time data, we must expect the unexpected in data problems. An attitude must be maintained that in spite of filtering and other techniques to keep bad data out of the system, some will always get through. When it does the only choice is to design for recovery when things return to normal.

IV. CONCLUSIONS IN MAKING DESSY ROBUST & USEFUL

In building real-time expert systems, it is not only important to have good system design, but also crucial to create a system that is robust enough to be useful in situations that are less than ideal. The techniques developed through the DESSY project were created out of the necessity of building a system that could withstand bad data, a common occurrence in the environment in which DESSY runs. These methods have been successfully used in DESSY's operation and have repeatedly demonstrated DESSY's robustness.

In conclusion, real-time data monitoring systems that reside in high-risk environments such as NASA's Mission Control Center must be built to be robust and useful given bad and missing data. Although the specifics of these techniques will differ from one project to another, we believe they are general enough to be applied to any real-time monitoring expert system, and that the guidelines presented in this paper provide a good set of ground rules from which a robust and useful system can be built.

BIBLIOGRAPHY

Collins, David, "Payload Deployment and Retrieval System Overview Workbook," PDRS OV 2102, Mission Operations Directorate, Johnson Space Center, Houston, Texas, February, 1988.

Dvorak, Daniel, "Expert Systems for Monitoring and Control," AI 87-55, The University of Texas at Austin, Austin, Texas, May, 1987.

Malin, Jane, Schreckenghost, Debra, and Thronesbery, Carroll, "Design for Interaction Between Humans and Intelligent Systems During Real-Time Fault Management," Proceedings of Fifth Annual Space Operations, Applications, and Research Symposium, NASA Johnson Space Center, Houston, Texas, July 9-11, 1991.

Malin, Jane, Schreckenghost, Debra, et.al., "Making Intelligent Systems Team Players: Case Studies and Design Issues," NASA Technical Memorandum 104738, NASA Johnson Space Center, Houston, Texas, September 1991, pp. 530 - 538.

Moore, Robert, and Kramer, Mark, "Expert Systems in On-Line Process Control," Lisp Machines, Inc., Los Angeles, California.