U.S. Gov't

MISSION OPERATIONS DIRECTORATE

RECONFIGURATION MANAGEMENT DIVISION

REAL TIME DATA SYSTEM (RTDS)

MAY 18, 1991

DP/JOHN F. MURATORE

ND185000

REAL TIME DATA SYSTEM (RTDS)

LESSONS LEARNED FROM OPERATIONAL REAL-TIME EXPERT SYSTEMS

- WHAT IS RTDS
- BASIC SYSTEM ARCHITECTURE
- LESSONS LEARNED FROM SUCCESSES AND FAILURES

REAL TIME DATA SYSTEM (RTDS)

WHAT IS AN EXPERT SYSTEM?

- ANY SOFTWARE SYSTEM THAT PERFORMS TASKS TO A STANDARD THAT WOULD NORMALLY REQUIRE A HUMAN EXPERT
    - LOOSELY ADOPTED FROM FEIGENBAUM

- IS NASTRAN (NASA STRESS ANALYSIS SYSTEM) AN EXPERT SYSTEM?

- EXPERT SYSTEM IMPLIES KNOWLEDGE CONTAINED IN DATA RATHER THAN CODE
- EXPERT SYSTEM IMPLIES USE OF HEURISTICS AS WELL AS ALGORITHMS

MOST IMPORTANT BOTTOM LINE - DOES NOT MATTER THE TECHNIQUES WE APPLY - IMPORTANT QUESTION IS WHETHER OR NOT COMPUTERS ARE DOING TASKS IMPORTANT TO NASA'S MISSION THAT PREVIOUSLY WERE ONLY DONE BY PEOPLE.

REAL TIME DATA SYSTEM (RTDS)

- STARTED IN 1987 AS RTOP FROM OFFICE OF AERONAUTICS, EXPLORATION AND TECHNOLOGY TO DEMONSTRATE READINESS OF EXPERT SYSTEM TECHNOLOGY TO PERFORM IN REAL OPERATIONAL ENVIRONMENTS
- FIRST CONCENTRATION AREA - INTEGRATED COMMUNICATION OFFICER (INCO) INTELLIGENT ASSOCIATE
  - FUNDING ARRIVED MAY 1987
  - FIRST USE IN CONTROL CENTER APRIL 1988 USED IN STS-26 SIMULATIONS AND FLIGHT
  - COMBINATION OF TASK AUTOMATION AND RULE-BASED EXPERT SYSTEMS
- EXPANDED TO BOOSTER (SPACE SHUTTLE MAIN ENGINES) PRIOR TO STS-26
  - CRITICAL FAULT MODES IDENTIFIED DURING STAND-DOWN AFTER CHALLENGER
  - DETECTION LOGIC COULD NOT BE PUT IN MISSION CONTROL MAINFRAME IN TIME FOR STS-26
  - SIMPLE TASK AUTOMATION
  - STARTED IN MAY 1988 - OPERATIONALLY USED DURING STS-26 SEPT 88
- EMERGENCY MISSION CONTROL CENTER DEMONSTRATION (DEC 1987)

REAL TIME DATA SYSTEM (RTDS)

- FEB 89 - STS-29 RTDS EXPANDED TO INCLUDE:
  - TIRE PRESSURE AUTOMATED MONITORING
    - PREVIOUSLY REQUIRED FULL TIME PERSON TO ACQUIRE DATA, COMPENSATE FOR TEMPERATURE, CONVERT TO STANDARD PRESSURE AND PLOT (TASK AUTOMATION)
  - VISUALIZATION OF FLIGHT INSTRUMENTS (TASK AUTOMATION)
  - ASCENT GNC MONITORING - (TASK AUTOMATION)
  - INSTALLED MONITORS IN SOME CONSOLES REPLACING MAINFRAME DISPLAY UNITS
  - NETWORK INSTALLED FOR DISTRIBUTING SOFTWARE AND REAL TIME DATA

REAL TIME DATA SYSTEM (RTDS)

- LATE 89 - STS-34
  - REMOTE MANIPULATOR SYSTEM VISUALIZATION (TASK AUTOMATION)
  - REMOTE MANIPULATOR SYSTEM UNATTENDED MONITORING (TASK AUTOMATION)
  - ON-ORBIT ATTITUDE CONTROL/DIGITAL AUTOPILOT MONITORING (RULE-BASED EXPERT SYSTEM)
  - TAPE RECORDER MONITORING (RULE-BASED EXPERT SYSTEM)
  - ADDED DATA RECORDING IN WORKSTATIONS AND PLAYBACK
  - TRAINING - SOME CERTIFICATION REQUIREMENTS FOR FLIGHT CONTROLLERS MET BY REVIEWING PLAYBACKS IN RTDS WORKSTATION INSTEAD OF INTEGRATED SIMULATIONS

HAS NOW SUPPORTED LDEF, HUBBLE, GRO, AND IBSS

REAL TIME DATA SYSTEM (RTDS)

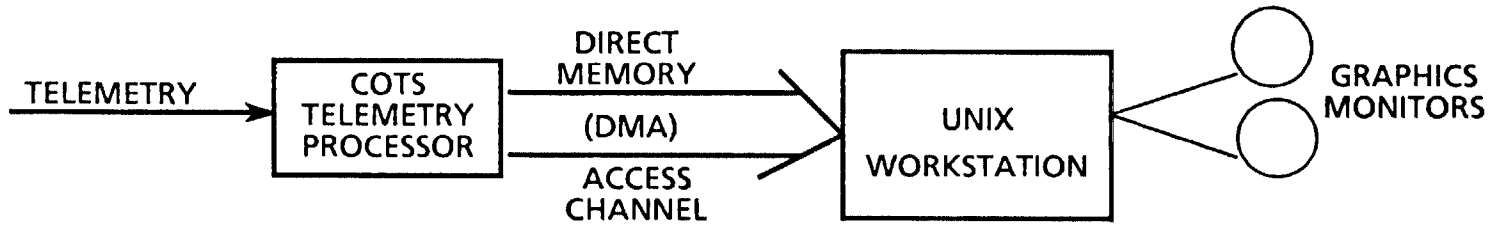
1989 - EXPANSION CONTINUES TO AERONAUTICS

- **DRYDEN FLIGHT RESEARCH CENTER USES RTDS AS BASIS FOR X-29 AUTOMATED MONITORING SYSTEM**

- **AIR FORCE FLIGHT TEST CENTER USES RTDS AS BASED FOR F-15 STOL MANEUVERING TECHNOLOGY DEMONSTRATION (STMD) RULE BASED EXPERT SYSTEM**

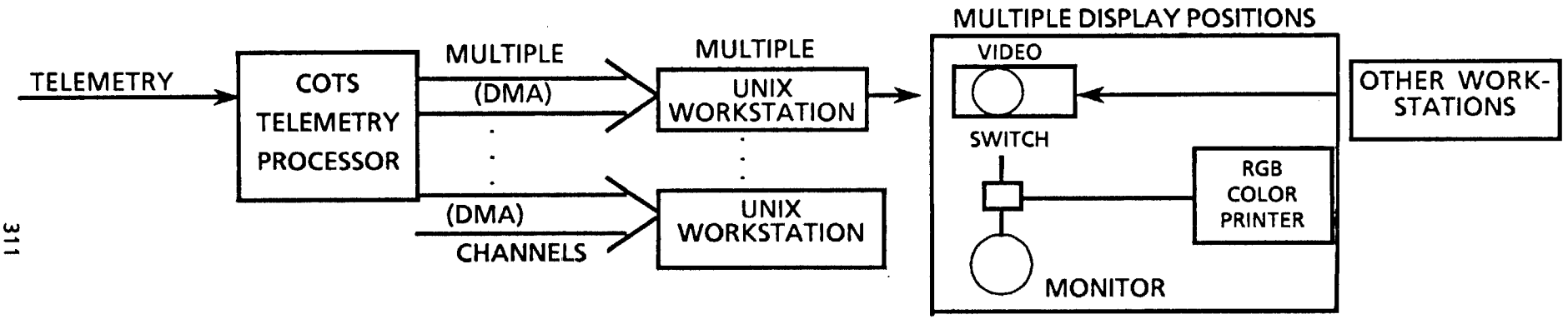
REAL TIME DATA SYSTEM (RTDS) ARCHITECTURE

PHASE 1



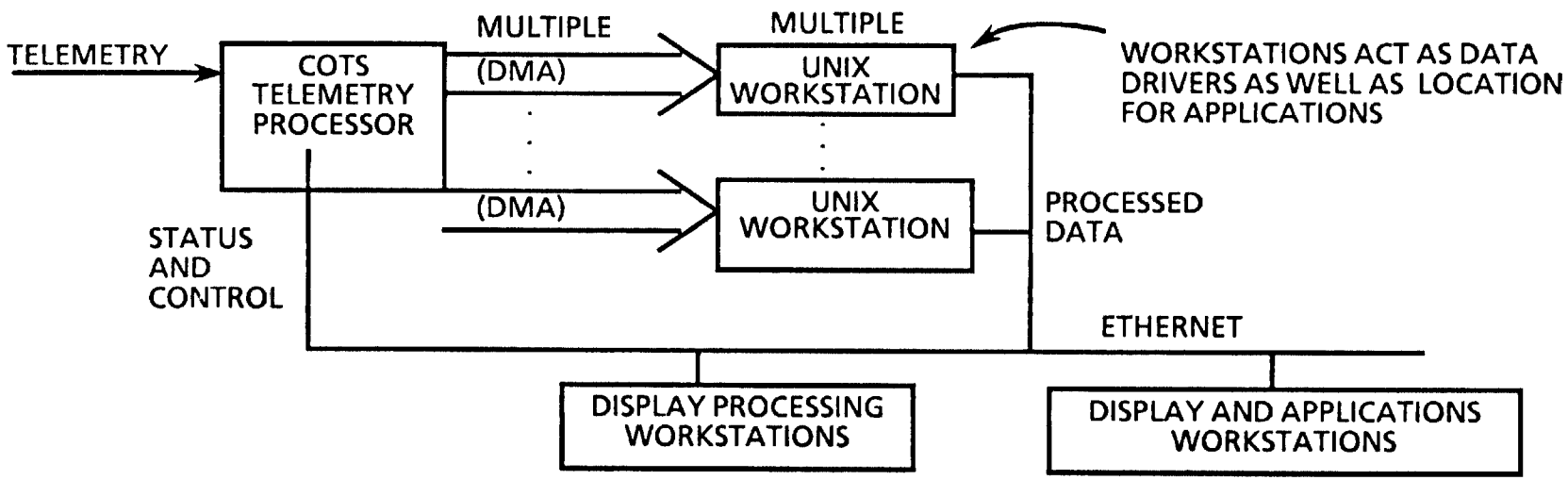
REAL TIME DATA SYSTEM (RTDS) ARCHITECTURE

PHASE 2



REAL TIME DATA SYSTEM (RTDS) ARCHITECTURE

PHASE 3



PHASE 4

TELEMETRY DECOMMUTATION DONE IN DATA DRIVER WORKSTATIONS, COTS PROCESSOR PERFORMS FRAME SYNC ONLY

REAL TIME DATA SYSTEM (RTDS)

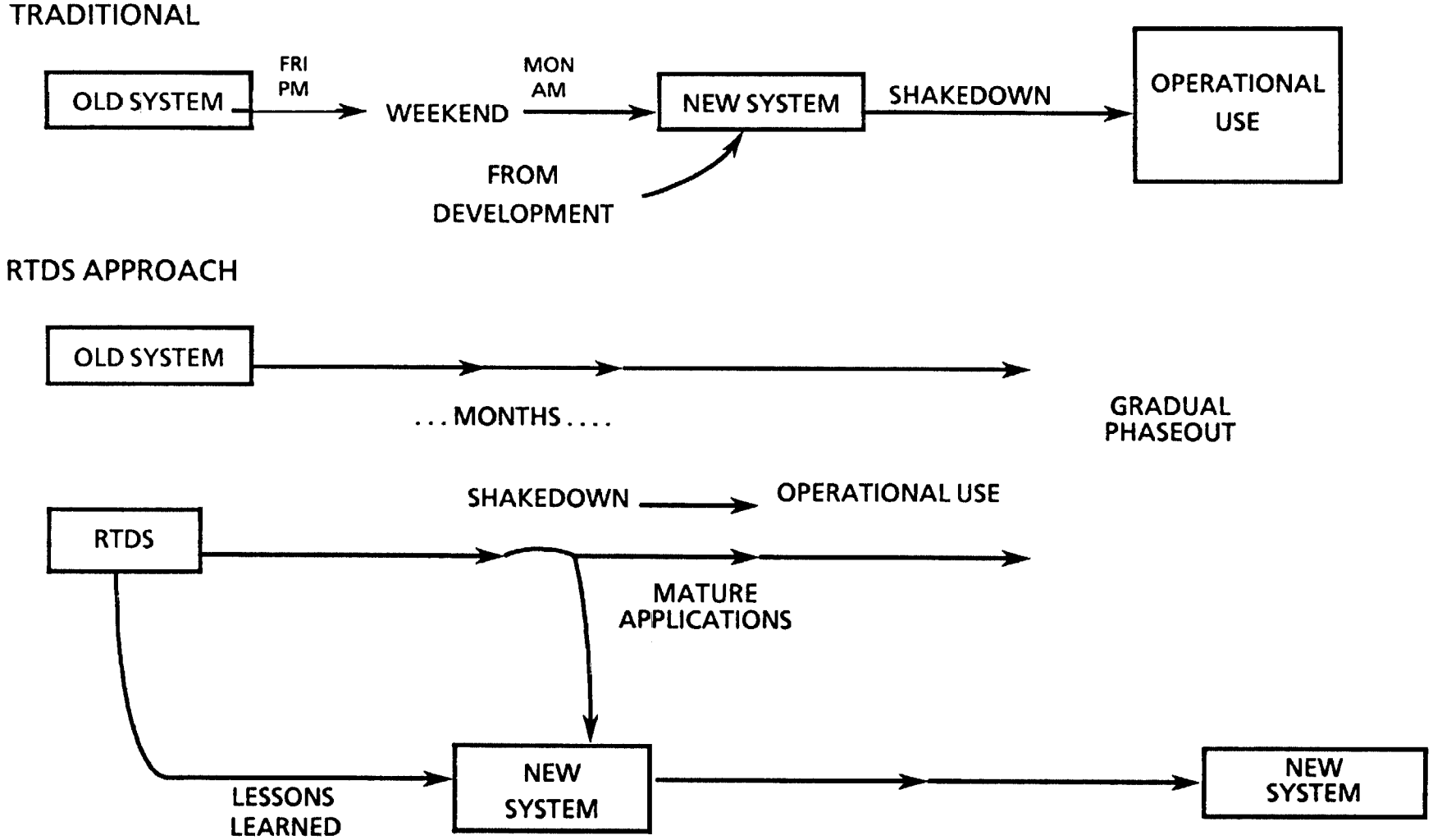
CRITICAL ARCHITECTURE ELEMENT IS ISOLATION

- RTDS IS A SHADOW CONTROL CENTER

  SO THAT DEVELOPMENT ITEMS CAN RUN

  IN PARALLEL WITH OPERATIONS

REAL TIME DATA SYSTEM (RTDS) UPGRADE PARADIGMS



REAL TIME DATA SYSTEM (RTDS)

RTDS TOP LESSONS LEARNED

U.S. Gov't

#1 - FIND A CUSTOMER WHO REALLY WANTS AND NEEDS THE TECHNOLOGY

- ANY SUCCESSFUL PRODUCT STARTS WITH A CUSTOMER AND A PRICE THE CUSTOMER IS WILLING TO PAY
  - NEW TECHNOLOGY ALWAYS HAS PRICE TO BE PAID BY THE CUSTOMER
- RTDS WORKED BECAUSE IT WAS CUSTOMER DRIVEN
- PREVIOUS EFFORTS FAILED BECAUSE DRIVEN FROM OUTSIDE CUSTOMER COMMUNITY

315

#2 - DATA ACQUISITION IS CRITICAL TO SUCCESS OF EXPERT SYSTEMS - MUST BE ABLE TO ELECTRONICALLY ACQUIRE ALL NECESSARY DATA

- EXPERT SYSTEM APPLICATIONS FLOPPED (HYD, PROP, SOME INCO) WHEN COULD NOT GATHER ALL THE REQUIRED DATA
- SUCCESSFUL APPLICATIONS MATCHED CAPABILITIES OF RTDS DATA ACQUISITION WITH APPLICATIONS (BOOSTER, RMS, WINDS, INCO, DPS, GNC)

REAL TIME DATA SYSTEM (RTDS)

RTDS TOP LESSONS LEARNED

#3 - DATA ACQUISITION DEVELOPMENT WILL REQUIRE THE MAJORITY OF EFFORT DURING INITIAL DEVELOPMENT

- KNOWLEDGE BASE CAPABILITIES INITIALLY AFFECTED MORE BY ABILITY TO GATHER/CONVERT DATA THAN BY SPEED OF RULE BASE

- BE INNOVATIVE - RTDS AT JSC USED COMMERCIAL-OFF-THE-SHELF TELEMETRY PROCESSOR TO LOWER WORKLOAD. DFRF TAPPED INTO EXISTING 1553 BUS. SHARP (JPL PROJECT) TAPPED INTO LINE PRINTER PORT

REAL TIME DATA SYSTEM (RTDS)

RTDS TOP 15 LESSONS LEARNED

#4 - SUCCESSFUL APPLICATIONS GENERALLY WERE ONLY ACTIVE DURING A SINGLE FLIGHT PHASE (ASCENT/ORBIT/ENTRY) OR WHERE ACTIVITY WAS SAME ACROSS ALL FLIGHT PHASES (E.G., TAPE RECORDER MANAGEMENT)

- SOMETIMES DUE TO LIMITATIONS OF DATA ACQUISITION

  - EARLY RTDS HAD LONG SETUP TIMES AT PHASE TRANSITION DUE TO TELEMETRY FORMAT SWITCHING

- DETECTING AND STEERING LOGIC IN DIFFERENT PHASES GOT VERY CUMBERSOME IN TASK AUTOMATION

REAL TIME DATA SYSTEM (RTDS)

RTDS TOP 15 LESSONS LEARNED

#5 - MUST DO A COMPLETE SUBFUNCTION
- BETTER TO DO DEPTH IN A SINGLE AREA THAN BREADTH
- DATA AVAILABILITY LIMITING IN RTDS EXPERIENCE

#6 - GET INTO OPS LOCATION AS SOON AS PRACTICABLE (8 MOS - 1 YEAR AFTER START)
- AVOID " LAB QUEENS"
- COMMENTS/EXPERIENCE FROM OPERATIONAL USE ARE THE MOST IMPORTANT INPUTS

#7 - FIND WAYS TO PRESENT FUNCTIONS GRAPHICALLY
- PEOPLE RELATE TO PICTURES
- EASIER TO DEMONSTRATE AND SELL
- DO NOT BE AFRAID OF COMPLEXITY IN DISPLAYS
  - OPERATORS USUALLY COMFORTABLE WITH DENSITY
- BIG TRAINING VALUE - WHENEVER OPERATOR USES SCHEMATIC TYPE DISPLAY THEY ARE BEING TRAINED ON SYSTEM

REAL TIME DATA SYSTEM (RTDS)

RTDS TOP 15 LESSONS LEARNED

#8 - SUCCESS IS NOT HAMPERED BY MISSION CRITICALITY OF APPLICATIONS - IN FACT, IT MAY BE ENHANCED

- HAD ORIGINALLY PLANNED TO START IN LOW CRITICALITY ORBIT PHASES ONLY
- NEEDS OF PROBLEM DROVE US INTO ASCENT HI CRITICALITY

- USERS WERE HIGHLY MOTIVATED AND THE PRESSURE OF HI CRITICALITY PROBABLY FOCUSED AND MATURED THE PRODUCT SIGNIFICANTLY

  - "THIS IS THE GAME WE CAME TO PLAY" - HI RISK/HI GAIN

REAL TIME DATA SYSTEM (RTDS)

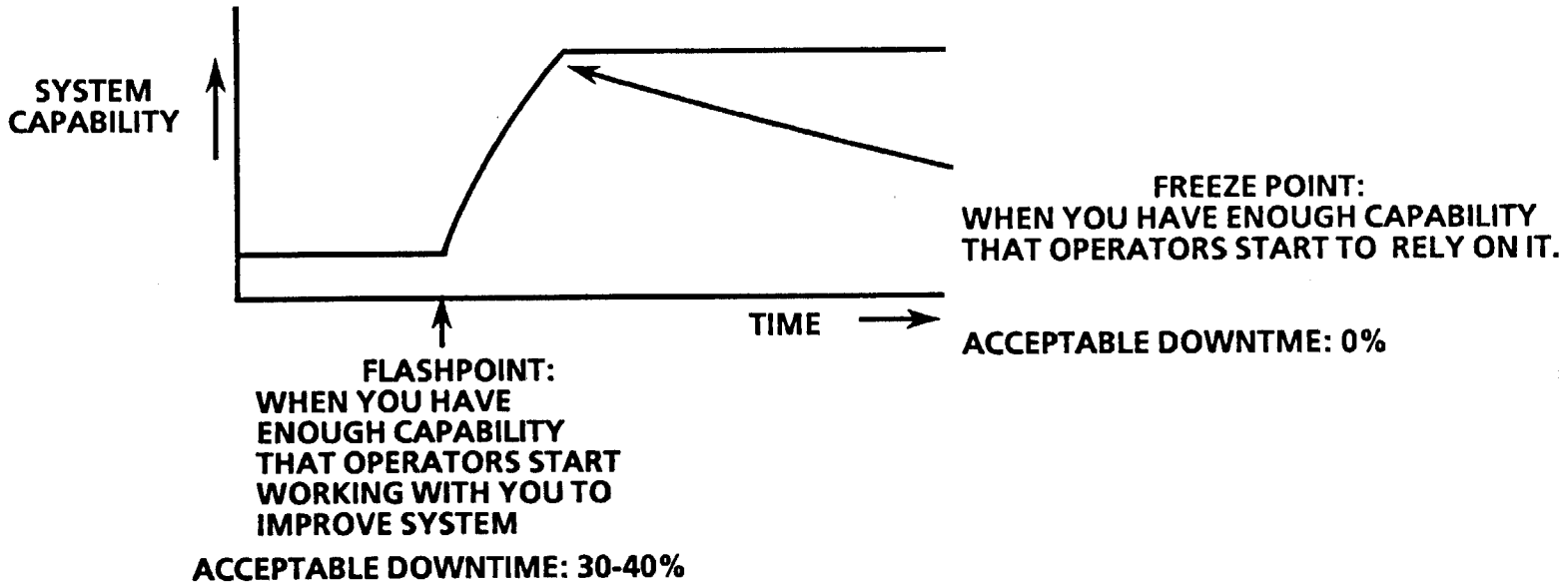
RTDS TOP LESSONS LEARNED

#9 - THE KEY TO SUCCESS IS THE ABILITY TO RAPIDLY IMPLEMENT CHANGES THROUGH ARCHITECTURE THAT ISOLATES DEVELOPMENT AND OPERATIONS

- EXISTING SYSTEMS ARE VERY CAPABLE AND HAVE CULTURAL SUPPORT DUE TO YEARS OF CUSTOMIZING AND TESTING
- EXISTING SYSTEMS ACHILLES HEEL IS INABILITY TO RAPIDLY CHANGE
  - SIGNIFICANT BACKLOG OF UNIMPLEMENTED REQUIREMENTS (, $1M) WAS IMPLEMENTED BY RTDS AT SMALL PERCENTAGE OF ORIGINAL COSTS
- NEW TECHNOLOGIES MUST:
  - PROVIDE RAPID CHANGE AS AN ADVANTAGE
  - UTILIZE ARCHITECTURAL ISOLATION SO THAT DEVELOPMENT AND OPERATIONS CAN RUN IN PARALLEL
  - ALLOW RAPID CUSTOMIZATION IN ORDER TO COMPETE WITH HIGHLY CUSTOMIZED SYSTEMS

REAL TIME DATA SYSTEM (RTDS)
RTDS TOP LESSONS LEARNED

#10 - ONCE IN OPERATIONS, MUST PLAN AROUND THE CLOCK SUPPORT IN THE OPERATIONS LOCATION



- EXISTING SYSTEMS ARE USUALLY HIGHLY RELIABLE
- USERS WILL EXPECT NEW SYSTEMS TO BE AS RELIABLE
  - PROBABLY NOT POSSIBLE INITIALLY BUT MUST MAKE HERCULEAN EFFORTS TO MAINTAIN USER CONFIDENCE

REAL TIME DATA SYSTEM (RTDS)
RTDS TOP LESSONS LEARNED

#11 - NEED A PLAN FOR SUCCESS

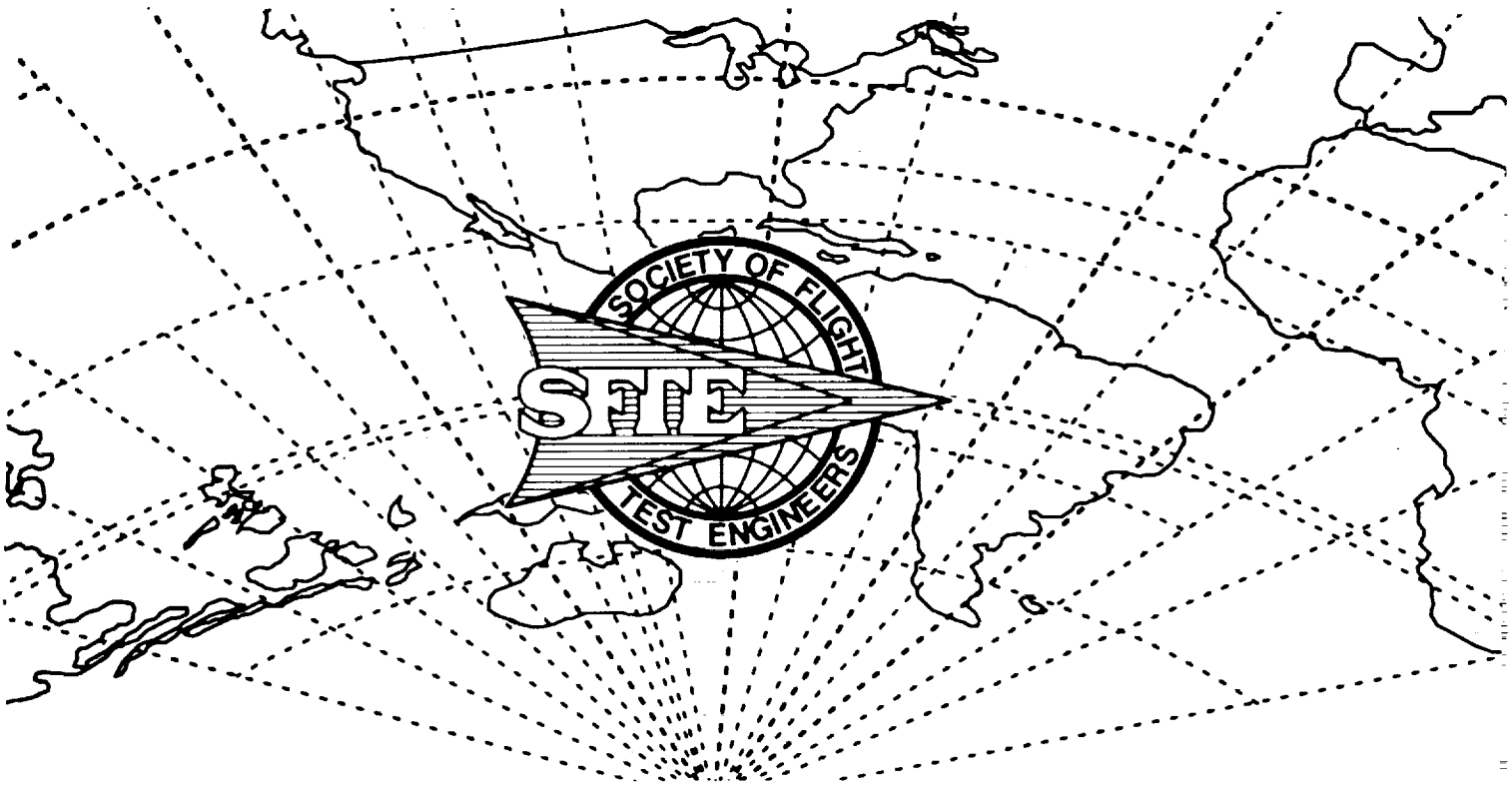
- RTDS GOT CAUGHT "BEHIND THE POWER CURVE" TWICE
- RIGHT AFTER INITIAL INSTALLATION WE DID NOT HAVE A PLAN OR HOW TO MEET THE DEMAND FOR OTHER NEW PROJECTS
- INSTALLATION PLANNING/SUPPORT WAS/IS A MAJOR HEADACHE
- AFTER THE INSTITUTION ACCEPTED IT, WE DID NOT HAVE A PLAN TO TURN IT OVER TO THE INSTITUTION FOR LONG-TERM SUPPORT
- ABSENCE OF PLANS LOST US OPPORTUNITIES

REAL TIME DATA SYSTEM (RTDS)

CONCLUSION

FINAL THOUGHTS

- ALL TECHNOLOGY TRANSFER SITUATIONS ARE UNIQUE - SOME OF THIS ADVICE MAY NOT APPLY TO YOUR SITUATION
- IT IS POSSIBLE TO IMPLEMENT ADVANCED INFORMATION SYSTEMS TECHNOLOGY IN NASA IN WAYS THAT SIGNIFICANTLY IMPROVE MISSION CAPABILITIES
- TECHNOLOGY TRANSFER IS A BODY CONTACT SPORT
  - ROUGH AND TUMBLE BETWEEN OPERATORS AND DEVELOPERS
  - DO NOT TRY IT IF YOU ARE NOT PREPARED TO GET BRUISED (EGO OR PHYSICALLY)
- WHEN YOU SEE PEOPLE RELYING ON YOUR SYSTEMS TO MANAGE NASA MISSIONS, IT IS ALL WORTH IT!!



21st ANNUAL SYMPOSIUM PROCEEDINGS

"Flight Test XXI . . .
The Sky's No Longer the Limit"

6-10 August 1990
Garden Grove, California

Sponsored by the Los Angeles Chapter

Real Time Data Acquisition For Expert Systems in Unix Workstations at Space Shuttle Mission Control

John F. Muratore, Troy A. Heindel and Terri B. Murphy
National Aeronautics and Space Administration

Arthur N. Rasmussen and Mark Gnabasik
MITRE Corporation

Robert Z. McFarland
Unisys Corporation

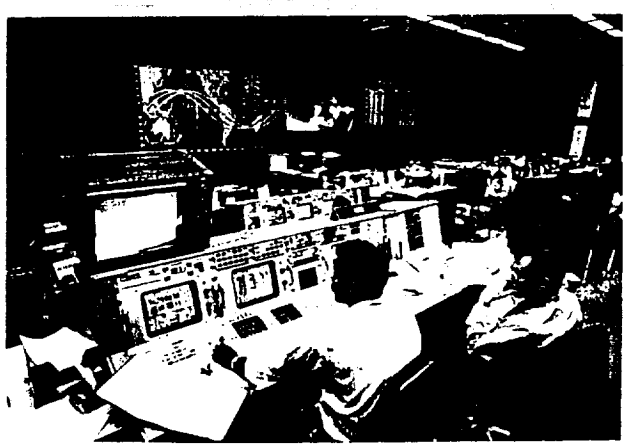
Samuel A. Bailey
Dual and Associates

Perhaps one of the most powerful symbols of the United States' technological prowess is the Mission Control Center (MCC) at the Lyndon B. Johnson Space Center in Houston, Texas. The rooms at Mission Control have been witness to major milestones in the history of American technology such as the first lunar landing, the rescue of Skylab, and the first launch of the Space Shuttle. When Mission Control was first activated in the early 1960's, it was truly a technological marvel. This facility however has received only modest upgrades since the Apollo program. Until recently it maintained a single mainframe based architecture that displayed data and left the job of data analysis to human beings. The display technology utilized in this system was monochrome and primarily displayed text information only with limited graphics (picture 1). An example display of 250 communication parameters is shown in picture 2.

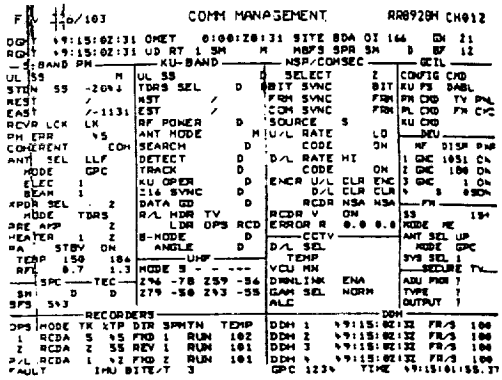
The system processed incoming data and displayed it to the flight controllers, however it performed few functions to turn raw data into information. The job of turning data into information upon which flight decisions could be made was performed by the flight controllers. In some cases, where additional computational support was required, small offline personal computers were added to the complex. Flight controllers visually copied data off the console display screens, and manually entered the data into the small personal computers where offline analysis could be performed.

Although, this system was technologically outdated, it contained years of customizing efforts and served NASA well through the early Space Shuttle Program. Several factors are now driving NASA to change the architecture of Mission Control to accommodate advanced automation. First is the requirement to support an increased flight rate without major growth in the number of personnel assigned to flight control duties. We are attempting to fly more missions with the same staff to control operational costs.

NASA is using automation to expand the capabilities of individuals so they can accomplish more work. This concept of "more work for the same dollar" is very different from trying to automate a factory where the desire is to replace humans with robotics and "do the same work for less dollars." In Mission Control, the goal is to support the human operator, and not to eliminate the human.



Picture 1 - Space Shuttle Mission Control



Picture 2 - Typical Mainframe Display

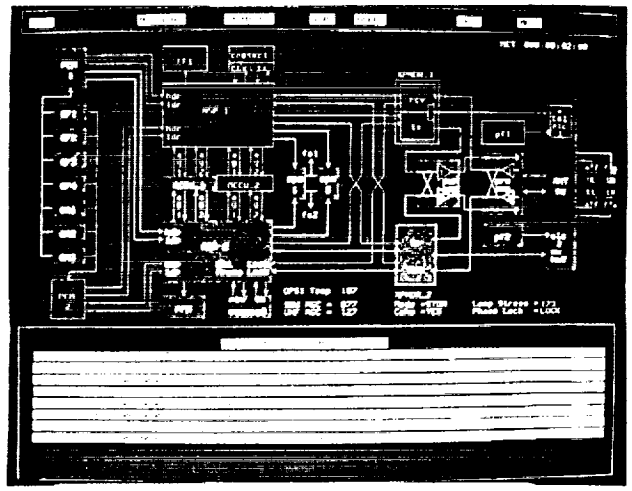
A second major concern is loss of corporate knowledge due to the unique bimodal age distribution of NASA. Hiring freezes between the Apollo and Shuttle programs have resulted in two primary groups in NASA. Approximately, half of NASA consists of Apollo veterans within 5 years of retirement. The other half consist of personnel under 35 with Shuttle-only experience. NASA considers it highly desirable to capture the corporate knowledge of the Apollo veterans in knowledge based systems before they retire. Because the mainframe complex is primarily oriented to data display, it is a poor environment for capturing and utilizing knowledge.

These factors have resulted in aggressive efforts by NASA's Mission Operations Directorate to utilize a distributed system of Unix (TM) engineering-class workstations to run a mix of online real time expert systems and traditional automation to allow flight controllers to perform more tasks and to capture the corporate knowledge of senior personnel. Starting with the first flight of the Space Shuttle after the Challenger accident, this effort, named the Real Time Data System (RTDS), has played an increasingly significant role in the flight-critical decision making process.

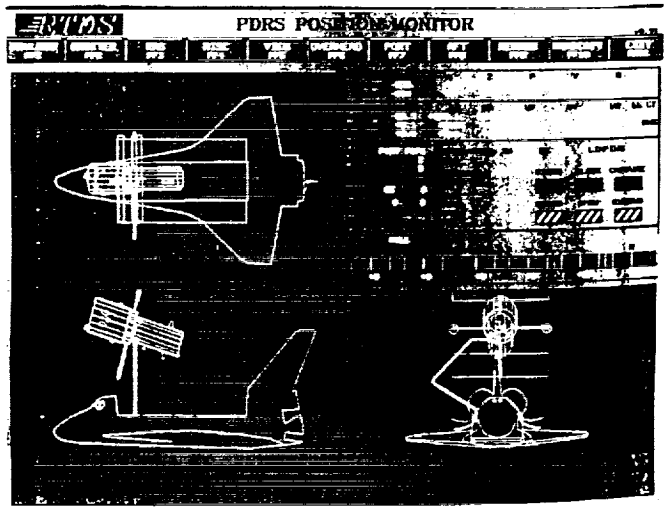
APPLICATIONS EXAMPLES

The application of these techniques has resulted in a new "look and feel" to Mission Control. Picture 3 shows an telemetry-animated schematic of the Shuttle's communication and tracking system. This display contains all of the information contained on the traditional monochrome text display shown in picture 2. The display utilizes color graphics to organize the information into a schematic. It also contains rules which draw inferences about the systems performance and operation from the telemetry. Previously, a major part of an operator's training was to learn how to look at complex displays of digital data and build a mental model of the system. Only after this training was complete, could an operator be trained to evaluate the situation and make recommendations. Utilizing the RTDS approach allows the operator to utilize the expertise of senior operators captured in the display program to build a mental model of the system and jump to learning how to evaluate the system.

This effort has also resulted in dramatic new and unexpected capabilities. For example, flight controllers who monitor the Shuttle's Remote Manipulator System (RMS) traditionally determined the position of the "robot arm" by observing digital readouts of the angles of each of the arms joints. A combination of offline tools and mental gymnastics allowed operators to determine the arm's position and advise astronauts on operation. Picture 4 shows an RTDS application which acquires real time telemetry of the arm's angles and animates a view of the Shuttle showing the arm's position. This application not only lowers the flight controller's workload, but also allows the controller to visually monitor for potential collisions of the Shuttle and payloads.

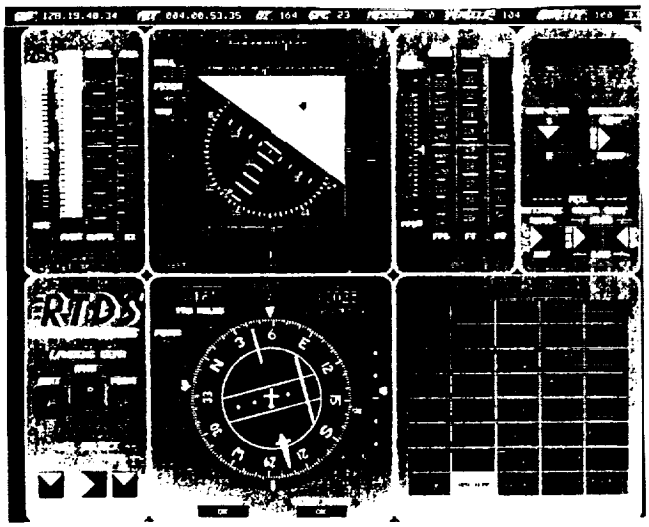


Picture 3 - Telemetry-Animated Communications Schematic On Workstation



Picture 4 - Remote Manipulator System Workstation Display

In another example, flight controllers typically monitored the performance of the Shuttle's flight instruments by watching digital displays of instrument telemetry describing the Shuttle's roll, pitch and yaw attitude angles or the bearing to the landing site. Picture 5 shows an RTDS display which acquires the real time telemetry on the flight instruments and displays an emulation of the flight instruments on a workstation screen.



Picture 5 - Workstation Emulation of Shuttle Flight Instruments

RTDS has also been applied to real time data sources other than telemetry. One of the most significant applications has been in the evaluation of weather data. The Flight Director is the NASA individual in Mission Control with final authority on all flight decisions. One of the most difficult tasks for the Flight Director during landing is the selection of runways based on winds. Because the Shuttle during landing is essentially a very large glider, it has very critical crosswind limits. Traditionally, during landings the Weather Officer would read out wind values from the landing site and the Flight Director was required to determine if the winds were within limits by consulting a paper crosswind graph. This could get quite hectic as there are several runways and the winds at Edwards Air Force Base are notoriously variable. In RTDS we built an application which receives the wind reports electronically from the landing sites, computes crosswind components, applies flight rules to determine if winds are within limits, and displays the results graphically to the Flight Director in real time. This application dramatically lowers the Flight Director's workload and enhances the safety of flight.

In developing RTDS, NASA met several challenges in the use of Unix (TM) workstations to operate in real time and to provide real time expert systems and color graphics in mission-critical environments. All of these applications required access to real time data. The remainder of this paper explains the techniques developed to acquire real time data under Unix and supply it to expert systems. The techniques described in this paper have not only been used on Space Shuttle but also on aeronautics applications. Systems have been developed using RTDS for aircraft flight test operations by NASA's Dryden Flight Research Facility (reference 1) for monitoring the X-29 and by the Air Force Flight Test Center (reference 2) for monitoring the F-15 short takeoff and landing demonstrator aircraft.

HARD AND SOFT REAL TIME CONSTRAINTS

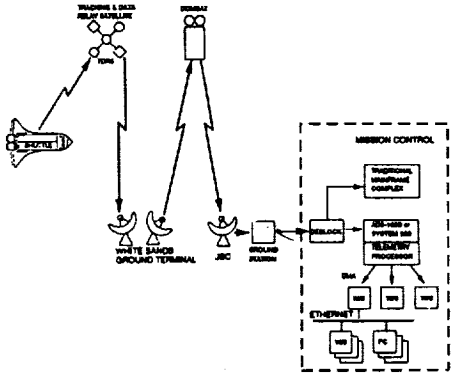
The most important item to understand in developing real time Unix applications is the difference between hard and soft real time constraints. As defined by Stankovic (Reference 3) real time systems are those where the correctness of a computation is a function of both the result of the computation and the time at which the computation is delivered. Hard real time systems are those where the function is completely failed if the computation does not always meet the time constraint. Soft real time systems are those where system performance is degraded if the time constraint is not always met, but can be fulfilled if the conditions are met with a response distribution. A typical hard real time system is an aircraft flight control system where loss of control occurs if the system does not meet time constraints. An example of a soft real time system would be an airline reservation system where slow response results in degraded operations but not system failure.

In RTDS we had to meet both hard and soft real time constraints. The hard real time constraint occurred in the acquisition of real time data into workstations. The soft real time constraints were in the performance of fault detection algorithms, fault detection rule based expert systems and data displays. Understanding the difference in these types of constraints is the key to successful implementation.

Unix Data Acquisition is A Hard Real Time Constraint

In RTDS we tapped into Space Shuttle telemetry as soon as the data reached the Mission Control (Diagram 1). Telemetry is a uniquely structured data

NASA DIAGRAM 1 REAL TIME DATA SYSTEM OVERVIEW



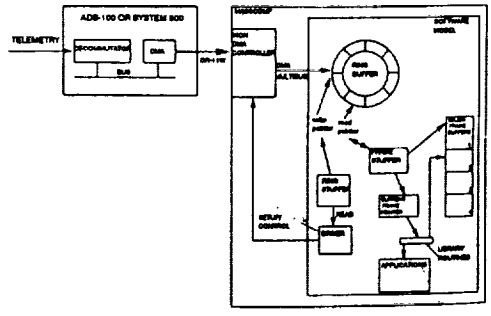
stream. In order to minimize hard real time processing in the Unix workstations the basic telemetry processing roles of bit, frame, and subframe synchronization, and parameter extraction (decommutation) were performed in a dedicated telemetry processor (Loral Instrumentation ADS-100 or System 500). The extracted parameters (approximately 5,000 16-bit words per second) were communicated to the workstations (Concurrent 6600) via a Direct Memory Access (DMA) interface. The DMA is based on the Digital Equipment Corporation (DEC) DR-11W standard using an Ikon Corporation interface board installed in the Masscomp. The telemetry processor buffers the incoming telemetry data in a 1,000 word First In First Out (FIFO) buffer. The buffer unloads over the interface when polled by the workstation. The hard real time constraint was that the FIFO buffer would lose data (overflow) and require a reset if it was not polled before it filled. The Unix workstation had to poll the system sufficiently to drain the buffer before it overflowed.

Unix has some well-known problems that hinder its use in hard real time applications. Unix normally does not provide the capability to assign hard priorities to tasks. This makes it impossible to require that a task execute at a given rate, such as polling a telemetry processor at four times a second. This is complicated by the fact that in most Unix implementations the kernel cannot be pre-empted. Thus, even if an external device, such as a telemetry processor, needs service, the kernel may block its servicing. Additionally, most Unix schedulers, reduce a task's switching priority in proportion to the processor time used.

This is a problem when data acquisition is to be performed continuously over a several day mission. Most Unix implementations perform task swapping; in addition, virtual memory versions also perform on-demand paging. If a task has critical code or data paged to secondary memory or if the entire task is swapped out, then the time to recover critical code and data may violate real time constraints. Unix also typically buffers all disk transactions in buffer cache in primary memory. This introduces an uncontrolled factor in critical disk I/O tasks such as data logging. Concurrent Corporation's Real Time Unix (RTU - TM Reference 4) has specific features to allow the user to deal with these constraints. In specific RTU allows tasks to lock a memory segment, and to circumvent the normal changes in switching priority by specifying a fixed real time switching priority. RTU provides contiguously allocated disk files that allow the user to bypass the disk buffer cache mechanism and perform direct I/O to disk. RTU provides kernel pre-emption within specified time constraints.

Even with these capabilities, data acquisition systems and real time applications under Unix must be carefully structured. Specifically, the tasks performing data acquisition must be isolated from applications processing so that increasing the applications load does not prevent the data acquisition from meeting the hard real time constraints. In order to deal with this problem, a technique was developed to take advantage of multiprocessing capabilities (Diagram 2).

NASA DIAGRAM 2 DATA ACQUISITION SOFTWARE



In this technique a single "stripped-down" task manages the DMA controller, instructing it to fill different segments of a ring buffer in sequence. This "Ring Stuffer" was identified as the highest priority real time task in the system. A second task, called the "Buffer Stuffer", reads the ring and performs decoding steps to place data into time homogeneous buffers in shared memory that can be accessed in parallel by many applications.

The two stuffers interact in several ways. The buffer stuffer is set to a real time priority so that its switching priority does not degrade with accumulated CPU time, but at a lower priority than the ring stuffer to prevent interfering with the hard real time performance of the ring stuffer. The Ring and Buffer stuffers communicate their position in the ring through shared memory. If a buffer being accessed by the Buffer Stuffer is about to be overwritten by the Ring Stuffer, then the Ring Stuffer still performs DMA to meet the real time constraint and prevent overflow of the telemetry processor FIFO. The data is transferred into a "bit bucket", a spare buffer dedicated for this purpose. Although this mechanism can lose data, acquired data is not contaminated. Whenever data is lost, tasks are notified by flags in shared memory. When two processors are available in a workstation, the two stuffers are run in parallel on separate CPU's to prevent contention.

This dual-stuffer technique was also implemented in a 80386 Personal Computer running the Lynx (TM) real time operating system to acquire weather data for the Flight Director Winds application. The weather data was acquired over an asynchronous serial line and placed in a ring by one task, and organized for evaluation in time homogeneous buffers by another task.

Shared memory is vital for interprocess communication in this approach. The Unix AT&T System V shared memory implementation is very good for real time data acquisition and distribution. It is important to perform the data acquisition task as a service and make acquired data available to all applications by shared memory. If shared memory is not available, then each application would have to perform significant data acquisition tasks and this would severely limit the number of simultaneous applications. Shared memory was also an important troubleshooting tool for the data acquisition software. Normal debugging techniques typically involve messages written into files or to the terminal. This assumes that the debugging techniques will not impact the proper operation of the process. However, in a real time environment the use of such techniques significantly affects the timing characteristics of the system and cannot be used. Instead, RTDS logs information about the data acquisition (such as pointers, number of bytes transferred, overflow flags) to shared memory. This allowed us to build graphic monitors which can be observed in real time to troubleshoot data acquisition problems.

Several other RTU features are critical to the performance of RTDS. These included the ability of an application to delay for time periods less than 1 second. This function was required because high priority tasks such as the buffer stuffer had to delay for incoming data and the standard Unix minimum sleep of 1 second was too long to meet rate constraints. RTU provides a mechanism allowing delays as short as 1 millisecond. RTU also provided a time of day clock (for rate calculations) and signals for trapping floating point errors. Floating point error traps are critical because noise in data can cause floating point errors and applications must trap and handle these errors. Real time tasks that perform continuous cyclic display also need the capability to poll for user input without holding. Conventional Unix terminal drivers provide the O_NDELAY option to allow a single application to read the keyboard without delay. This however does not provide a mechanism for controlling which applications should receive the keyboard input. An X Windows approach would be the modern solution to this problem and is being used by RTDS in its newest applications. At the time we started RTDS however, X Windows was not available on our equipment, so we used mouse button signals with signal handlers to provide inputs to applications. The mouse signals did not require polling, so cyclic displays would not delay for user input. All of the applications received any mouse input so that the handlers were required to determine from the cursor location if the inputs were intended for their application.

This technique has been highly successful in processing real time data. During a recent mission, a workstation running a moderate applications load ran for over 3 days without an overflow occurring. With a heavily loaded workstation we experience an overflow every 10-12 hours. This vulnerability occurs because the ring stuffer runs as a Unix application. It context switches from application state to kernel state

whenever it executes the DMA Controller driver. If another application requests a kernel service while the ring stuffer is in applications state, the context switch of the ring stuffer can be delayed. To minimize the effects of this case we modified the telemetry processor board to automatically reset itself when an overflow occurs. This is not completely satisfactory as data loss occurs during the reset and we are currently developing a version of the DMA driver that performs all of the ring stuffer activity in the kernel.

RTDS is an interesting statement on the power of current engineering workstations. The Apollo Mission Control Center used for the lunar landings in 1969 processed less than 1,000 parameters a second in the large mainframes of the time. RTDS processes 4,000 parameters a second in a single workstation.

Data Display and Expert Systems are Soft Real Time Tasks

When we started RTDS we thought of data display and computation along traditional mainframe-based terms. Specifically, we expected all data to be displayed and all computations to run on every data sample. After some early experimentation it became clear that it would not be possible to display data and run rule based expert systems in a hard constrained fashion. The nature of rule based expert systems makes it difficult to guarantee hard real-time performance. In rule based expert systems, computational load varies based on the number of rules fired and this varies with circumstances being monitored.
Examining the actual monitoring tasks performed by flight controllers reveals that although data is displayed at once per second, it is not monitored at that rate. Flight controllers are themselves multitasking and monitor several screens, event lights, and voice loops as well as using other materials such as procedures and schematics. The human monitor is a "soft" real time implementation.

We also found that the tasks could be structured so that only key detection logic was being evaluated every second. Supporting logic was only activated when primary logic detected a problem. This significantly improved our real time performance. In specific, a rule-based expert system monitoring the Shuttle's communications system utilized approximately 500 rules. These were initially implemented in CLIPS, a rule based expert system tool developed by the Mission Planning and Analysis Division at Johnson Space Center. This tool does not have any special real-time support. By structuring the rules into phases and enforcing certain precedence, we found that approximately 100 rules were required to capture the key detection logic. In the Unix environment, CLIPS was able to fire these 100 rules approximately 2 to 3 times a second. When the key logic rules detected problems, additional rules fired, which slowed down the system and caused only momentary violation of real time constraints.

It is important to realize that the Shuttle systems do not normally operate with continuous failures being introduced. One of the goals of the expert systems is to provide expert evaluation when failures occur. If the system slows down when the failure occurs, but is still able to provide the expertise, then it is meeting its desired function.

When we performed tests on the performance of the fault detection, we found that a large percentage of the processing was meeting the once-a-second constraint and all of the processing was being performed on a 2 to 3 second cycle. This was not detectable by flight controllers looking at the RTDS displays.

We also found that RTDS displays telemetry 3 to 4 seconds ahead of the mainframe complex. This is because the RTDS architecture minimizes the number of data transfers between processors. Because the mainframe performs such a large number of computations, it requires extensive minicomputer preprocessing to meet real time constraints. This introduces significant delay in the mainframe system.

On several occasions during actual Shuttle flight, RTDS has detected problems and brought them to the attention of flight controllers before they noticed the problems on the conventional displays. In several cases, the mainframe displays have been completely removed from the control center and the controllers rely entirely on the workstation based displays.

SUPPORT TECHNIQUES FOR REAL TIME EXPERT SYSTEMS

In developing the real time data acquisition support for expert systems we utilized several critical techniques. The most important technique is that of the time homogeneous buffers. If task automation or a rule based expert system is to combine several different pieces of sampled information and make a decision on them, then the time relationship of these samples must be known. Typically we try to only combine

data from the same data acquisition sampling cycle or major frame. A major frame is the time period in a sampled data system when all measurements are sampled at least once. On Space Shuttle the major frame is sampled and transmitted once per second. All measurements from a given major frame represent a time homogeneous dataset bounded by the sampling rate.

An example of the importance of this type of relationship is detecting a failed reaction control system jet thruster on the Space Shuttle. There are two conditions that we want to detect. A jet that does not fire in response to command is considered failed off. A jet that does not turn off when the command is removed is failed on. So we have two rules :

a. If jet command is on and jet chamber pressure is low,then jet is not firing and failed off.

b. If jet command is off and jet chamber pressure is high,then jet is firing and failed on.

If the jet command telemetry parameter is not from the same sampling period as the jet chamber pressure command, two things can happen which cause a normal jet firing to be misdiagnosed as a failure. If the command measurement leads the response measurement, then the first rule will be satisfied indicating that the jet is not responding to commands. If the command lags the response, then the second rule will be satisfied indicating that the jet is firing without a command. It is only when the command and response are from the same frame that a normal firing will be properly evaluated.

In RTDS we placed data into time homogeneous buffers in shared memory on major frame boundaries. We use four buffers on a round robin basis. The Buffer Stuffer places telemetry parameters in the round robin buffers in named locations where they can be extracted by applications using standard library routines. Whenever a parameter is placed in the buffer it is marked as valid for that major frame. The stuffer also searches for the frame markers in the telemetry stream. When a major frame marker is detected, the buffer stuffer closes the buffer being updated and makes it available to applications for reading. Flags are set in shared memory to indicate the most recently updated buffer. After releasing the completed buffer to applications, the Buffer Stuffer then opens the next round robin buffer. Before starting to fill this new buffer, data from the last major frame period is copied into the new buffer. All data is marked as invalid for the new major frame when it is copied forward. As each parameter is processed in the new major frame, the parameter status is updated to valid. By copying forward the most recently received data, we ensure that applications always have access to the most recently received data (with appropriately marked validity), even if the data is not received in a given major frame due to errors in transmission. By switching buffers and making them available to applications at major frame boundaries, we maintain the time homogeneity of the original sampled data stream.

In order to maintain major frame time-homogeneity in the applications, it was necessary to ensure that once the data acquisition library starts to pull data from a buffer, that all of the parameters are pulled from only that one buffer (major frame). This must happen while Unix is switching applications, paging and swapping. It make take more than 1 second for an application to complete its computation cycle between data acquisitions. The four round-robin buffer design gives an application three major frame times to complete an acquisition before the buffer is overwritten.

This major frame buffer technique maintains the data time characteristics for automated monitoring and expert systems. Alternative approaches have been used in other telemetry computer systems which lose this critical time relationship information. An alternative technique that has been implemented in at least one major NASA system and two new commercial off-the-shelf telemetry monitoring systems is called the Current Value Table (CVT). In CVT, telemetry data is acquired and the most current values of parameters are placed in a single table without regard to the major frame. When an application requests data, the CVT ships out the most current value received for the requested parameters. Because the requests occur asynchronously with the data acquisition, it is almost certain that data from multiple major frames are in the same data request. This technique may be acceptable for low rate data displays and limited automated monitoring but is not acceptable for advanced automation using rule-based systems.

Major Frame Buffers For Logging and Distribution

The major frame buffers are also a powerful structure for logging data. In RTDS, the buffer stuffer logs all parameters to disk in contiguous files. RTDS has an "instant replay" mode where real time data acquisition is stopped and a "replay stuffer" is used to stuff data into the major frame buffers.

In this way all of the real time applications can be used in playbacks.

The replay stuffer has a control panel which is fashioned after a conventional Videocassette Recorder control panel. We chose this interface because it was familiar to almost everyone and it has all of the functionality needed. With the VCR control panel users can playback data, view in fast forward, "rewind," or Shuttle between set points. Speed of the playback can be adjusted for slow-motion analysis.

This capability has turned out to be essential for three reasons. First, the capability was essential for debugging the automation applications. Data signatures captured during actual flight or simulations can be replayed time and time again to work out bugs in automation and expert systems as well as to perform regression testing. Second, as a real time tool this capability has enabled operators to significantly cut the time required to view playback data. This capability was used dramatically after the pad engine shutdown and countdown abort on the first STS-34 launch attempt. RTDS enabled engineers and managers to replay the shutdown within minutes to troubleshoot the cause. This is a big improvement over the current playback systems which can take from 30 minutes to 5 hours to retrieve playback data. In fact, the director of Mission Operation has stated that RTDS paid for its entire development in those few minutes. Third, there has been an unexpected training benefit from the playback capability. Flight controllers can record simulations and then play them back at their convenience for training. Several training objectives in flight controller certification are now met by this technique. This saves the large costs associated with meeting training objectives by a full-up simulations with the entire simulator, control center, and flight team in place.

The major frame buffer is also natural format for distributing data over local area networks. In RTDS we distribute major frame buffers over Ethernet (TM).

This enables RTDS to provide remote telemetry monitoring and software checkout, even on computers which could not normally support real time data acquisition. The User Datagram Protocol (UDP) subset of TCP/IP was used to provide a connectionless unacknowledged data transfer. This allowed the transmitting workstation to be unaffected by the receiver workstation if the receiver was unable to keep up with the transmitter. This capability has allowed us to conduct operational demonstrations where flight controllers monitored data out of their offices. We sent the data to the experts, rather than sending the experts to the control center. This technique will become more important as NASA pursues long term missions such as Space Station where it becomes less feasible to tie experts down to a central location.

Data Quality of Frames and Individual Parameters

In order for expert systems and task automation to use real time data, it is necessary to determine the quality of the data. There are two measures of data quality, the quality of a major frameand the validity status of individual parameters.

In order to determine the validity of a frame of data, we utilized the fact that each major frame of telemetry is is divided into a number of smaller frames, called minor frames. Each minor frame contains an identifying counter. In Shuttle there are 100 minor frames per major frame and one major frame per second. Parameters are spread across the minor frames. In telemetry systems, data can be interrupted due to radio-frequency noise on the space-to-ground link. Typically noise is of short duration and will only affect one or two minor frames.

In order to inform applications when noise was present, the minor frame counter is transferred via DMA to the workstation. The Buffer Stuffer

examines each frame counter to ensure that all 100 frames are received in sequence for a given major frame. This Quality parameter is expressed as a number from 0 to 100 indicating a rough percentage of the quality of the data. The Quality parameter is placed in shared memory, with one quality estimate for each major frame buffer. Applications receive the buffer Quality value whenever they request data from a buffer. In this way applications can chose to display or discard data based on overall data quality. In many data display tasks in RTDS we chose to display all of the data even in high noise (low Quality) conditions. In critical task automation, we discard all data that does not have a 100 percent quality to prevent erroneous results.

There are approximately 32,000 parameters in the Space Shuttle system but only approximately half of them are being downlinked at any one time. This is due to the restriction of the downlink bandwidth of the telemetry system and recognizes the fact that certain data is not required during all flight phases. For example, engine data is only needed during launch. If during a major frame, data is not in the specific telemetry format, then RTDS marks the individual parameter status as invalid. When an

application attempts to acquire the parameter, the application receives both the value and the status of the parameter from the buffer.

Individual parameter validity status is an important technique that has been overlooked in several NASA and commercial systems and caused serious architectural problems when retrofitted into these systems. This function must be provided by the data acquisition subsystem. Without a system solution, each application must contain sufficient format definition information to independently assess validity. This is a severe software maintenance and leaves the possibility that applications might function improperly due to stale data.

Calibration and Conversion

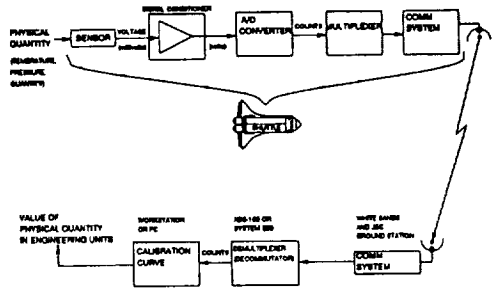
In typical flight vehicle telemetry systems data originates from one of two major sources. Some data is acquired directly from sensors and other data is the result of computations in the onboard computer. Calibration is the function of calculating engineering unit values (temperature, pressure, etc..) from sensor telemetry. Conversion is the process of transforming values from the word formats of the flight vehicle computer into the word formats of the ground computer. Both of these functions must be performed if an expert system is to use telemetry.

Typical sensors convert some physical quantity (e.g. temperature, pressure) into an analog value proportional over a specified range (Diagram 4). Sensor output voltages are normally amplified and then converted to a digital value. These values are usually called "counts." The sensor value in counts (8, 10, 11,12 and 16 bit are popular sizes) is placed in a serial stream for transmission to the ground. On the ground, the computer system converts these to a number representing a physical quantity. This is because it is much easier for humans and expert systems to reason about physical quantities than "counts." This is done with a calibration curve of the form:

$$y = A(0) + A(1)*X + A(2)*X^2 + A(3)*X^3 + \ldots$$

where the counts are supplied as the X values and the A(N) values are the coefficients. In RTDS we use the Shuttle program standard fifth order polynomials. The coefficients for this polynomial are stored in shared memory so they can be viewed and altered and to assure they are common to all applications. Acquired data is placed in the shared memory in "counts" and when applications request data, the shared memory is examined to obtain and apply the calibration curve.

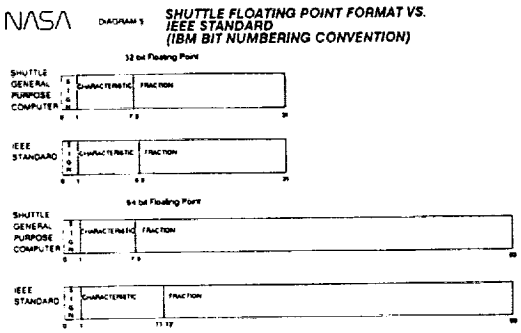
NASA DIAGRAM 4 CALIBRATION CONCEPTS



Flight vehicle computers historically have unique architectures due to the demands on weight, size, power consumption and reliability in hostile environments. In fact,the Space Station program is the first NASA manned program to specify a hardened standard commercial-off-the-shelf architecture for a flight computer (80386). Because flight computers tend to be unique, they usually have unique floating point formats (Diagram 5). Unix workstations usually use IEEE standard floating point formats or manufacturers variants. The data acquisition subsystem must be able to convert between these different number representation subsystems if the display user or expert system is to interpret the data. In RTDS we keep the parameter conversion information in shared memory together with the calibration curve. Data is kept in the flight vehicle form in the shared memory. When an application requests data, the shared memory is used to select and apply the appropriate conversions. Twelve different types of conversions are required on Space Shuttle. Override modes for both calibration and conversion are provided in the library calls so that user applications can acquire raw data directly as it was downlinked from the vehicle.

Noise Filtering

Even when the quality and individual parameter validity mechanisms are used, there still is the possibility of getting incorrect data into a real time expert system. There are two basic sources of error. First, communications noise may be of very short duration so as to only affect a small number of bits. If the noise doesn't affect a frame counter or frame synchronization marker, then it is difficult to determine (from a data communications standpoint) that an error has occurred. Some telemetry systems use parity bits on individual parameters and frames or a forward-error-correction technique but these are not available on Shuttle. The second



source of noise is the sensor itself. Sensors are nonideal devices and can be noisy. In RTDS we applied "noise-filtering" techniques to minimize the effects of these errors on applications.

The first technique was applicable to discrete (binary 1 or 0) values, such as switch settings and valve positions. Whenever one of these items changed, it would not be provided to the expert system unless the change was present for a specified number of seconds (N). This N-count noise filter is a technique which has been used successfully in the onboard automation of the Space Shuttle. The problem with this filter is that a "chattering" sensor is not detected if it changes state faster or at the same rate as the noise filter. Operationally, the N-count algorithm worked well for our applications.

The second technique was applicable to numbers. A numeric value may take many values. In a slowly changing situation the same N Count algorithm used for discretes can be used. But where values change rapidly, comparing updates to the last value can result in no updates being made available to the expert system because the last value never stabilizes. For fast changing numeric values we used reasonableness tests. The expert systems and automation would reject values that were not in a specified reasonable range for that parameter.

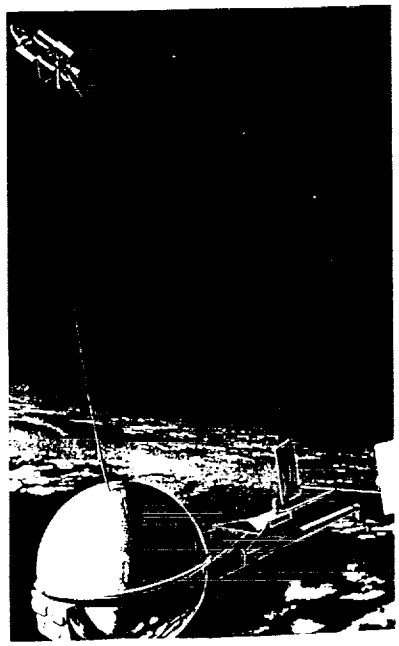
In some cases, these checks were performed within the expert system or automation. But in many cases the same noise-filtered value was required by several applications (automated fault detection, displays etc...). In these cases, we wanted a single authoritative copy for all applications. We used another shared memory buffer to which the noise filtering routines could write. This "signal buffer" was accessible by all applications by requesting parameter names through a library routine. This "signal buffer" did not represent a major frame time-homogeneous buffer.

Because noise filters are set at different values for each parameter, there was no way to maintain time homogeneity. The mechanism however worked very well as both a repository for "best-estimate" values for low dynamics tasks and a general purpose mechanism for communicating results between applications. Many applications used this mechanism.

FLIGHT TEST - THE SKY'S NO LONGER THE LIMIT

It is appropriate that this conference's title includes the statement that the sky is no longer the limit for flight test. RTDS and similar systems will be used in the next few years by NASA to perform the most extensive set of space flight technology tests since the early 1960's. NASA's manifest for the next few years contains several missions which will flight test new technologies such as tethers, aerobrakes, and free-flying telerobots in near-earth orbit. All of these missions require advanced telemetry monitoring and visualization techniques similar to those developed in the RTDS project.

The Tethersat will be flown on STS-46, currently scheduled for late 1991 (picture 6). This will be the first attempt to ever use a large scale tether between two orbiting objects. This flight will explore the motion and electrodynamics of tethers in orbit.

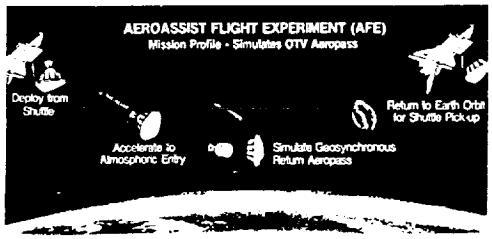


Picture 6 - Tethersat

Also in late 1991, STS-49 will test elements of the Flight Telerobotic Servicer (FTS picture 7). The FTS is being developed by Goddard Space Flight Center as a tool to assist in the assembly of the Space Station. It can operate in free-flying mode or attached on the end of the RMS. The STS-49 flight wil be used to conduct flight experiments on prototype hardware elements. The first actual flight of the complete FTS will be performed on STS-72 in late 1993. This will qualify the equipment for use in Space Station assembly in 1995.

Picture 7 - Flight Tele-Robotic Servicer

In late 1994 the Aero-Assist Flight Experiment (AAFE) will be flown on STS-82. This flight will deploy an unmanned free-flying vehicle which will fly a aerobraking profile (picture 8) to demonstrate the feasibility of using atmospheric aerodynamic braking to alter orbits. This technology is considered crucial for capturing vehicles returning to earth from the moon in future lunar exploration programs.



An Aeroassist Flight Experiment (AFE) mission scenario calls for the AFE to be deployed from the cargo bay of the Space Shuttle. A solid rocket motor (SRM) will accelerate the vehicle to 33,800 feet per second simulating the speed at which a spacecraft travels in geosynchronous orbit. After the burn, the SRM is jettisoned. A dip through the Earth's upper atmosphere is expected to slow the AFE so it can rendezvous with and be retrieved by the Shuttle. Back in the cargo bay, the AFE will be returned to Earth for in-depth analysis.

Picture 8 - Aero-Assist Flight Experiment

CONCLUSIONS

The advances in workstations and real time Unix have enabled small programming teams to implement real time telemetry systems that have made major improvements in NASA space and aeronautics mission operations. Several real time adjustments must be made to Unix and applications properly structured to meet ard and soft real time demands. Many monitoring problems are actually soft real time problems and thus can be implemented using current workstations and expert system technology. New techniques in data acquisition have been developed to ensure that the correctness of the expert system recommendations is not affected by the data acquisition process. These mechanisms are general and can be applied to any real time expert system. Although, these techniques are more traditionally associated with real time systems or process control rather than expert systems, they must be applied for real time expert systems to provide useful information in mission critical environments.

References

1. Mackall, McBride, and Cohen, "Overview of the NASA Ames-Dryden Integrated Test Facility", NASA TM-101720, 1990

2. Jones, Madison, and Flanders, "Real Time Discrete Monitoring", AIAA Fifth Biannual Flight Test Conference, 1990, AIAA-90-1311

3. Stankovic and Ramamritham, "Hard Real Time Systems - A Tutorial", Computer Society Press (IEEE), Washington DC, 1988

4. "Understanding Real Time Unix", Concurrent Computer Corporation 1989

Acknowledgements

The development of RTDS at Space Shuttle Mission Control has been a team effort. Flight controllers Michael Dingler, Jon Redding, Ronald Montgomery and Joe Hughes each developed specific expert systems in their discipline areas. Cheryl Whittaker, Deborah Horton, and Erick Kindred developed significant applications. Brian Kowalski provided invaluable assistance with the first version of the DMA driver. The development of the X-29 system was performed by Dale Mackall, Dorothea Cohen, and Dick Simon from NASA's Ames-Dryden Flight Research Facility. The F-15 STOL system was

developed by a team headed by Robin Madison of the Air Force Flight Test Center. Funding for RTDS was provided by NASA's Office of Aeronautics and Exploration Technology, The Space Shuttle Advanced Development Effort and the Space Station Freedom Advanced Development Program.