RIACS

Research Institute for Advanced Computer Science
NASA Ames Research Center

Evaluating The Operations Capability Of Freedom's Data Management System

Henry A. Sowizral

October 1990

Research Institute for Advanced Computer Science
NASA Ames Research Center
Moffett Field, CA

RIACS Technical Report 90.48

Evaluating The Operations Capability Of Freedom's Data Management System

Henry A. Sowizral

October 1990

Research Institute for Advanced Computer Science
NASA Ames Research Center
Moffett Field, CA

RIACS Technical Report 90.48

Evaluating The Operations Capability Of Freedom's Data Management System

Henry A. Sowizral

October 1990

Research Institute for Advanced Computer Science
NASA Ames Research Center
Moffett Field, CA

RIACS Technical Report 90.48

The Research Institute of Advanced Computer Science is operated by Universities Space Research Association, The American City Building, Suite 311, Columbia, MD 2104, (301)730-2656.

The work reported on herein was supported by Cooperative Agreement Number NCC2-387 between the National Aeronautics and Space Administration and the Universities Space Research Association.

Evaluating the Operations Capability of Freedom's Data Management System

Henry A. Sowizral

Research Institute for Advanced Computer Science
NASA Ames Research Center
Moffett Field, CA

RIACS Technical Report 90.48

ABSTRACT

As NASA and its contractors develop more elements of Space Station Freedom's Data Management System (DMS), we can begin testing whether the DMS, as a whole, can perform its operational functions. This report examines three areas of DMS performance, raw processor speed, the subjective speed of the Lynx OS X-Window system, and the operational capacity of the Runtime Object Database (RODB). (The RODB is the central component in DMS services.) The report concludes that the proposed processor will operate at its specified rate of speed and that the X-window system operates within users' subjective needs. Unfortunately, it also concludes that the RODB cannot provide the required level of service, even with a two-order of magnitude (100 fold) improvement in speed.

SUMMARY

This report evaluates three aspects of Space Station Freedom's (SSF) Data Management System (DMS), as currently envisioned, and it examines whether or not the DMS will be able to perform its operations function.

We approached this study by obtaining access to existing DMS hardware and software in prototype or pre-prototype form; examining the hardware and software for conformity to the DMS plan; quantifying the hardware and software performance; and, analyzing the results in terms of operational needs.

The report examined three areas of performance, the raw speed of the Intel 80386DX processor chip, the subjective speed of the X-window display system, and the operational capacity (the speed) of the Runtime Object Data Base (RODB) system—the central component of the DMS.

We found that the Intel 80386DX processor ran a nominal 3-4 MIPS rate depending on the application. Though the 80386 did run at its nominal processing rate, according to a recent trade study [2], that rate is inadequate. That study, using nominal latencies and speeds for required operations, concluded that two of the SDPs (80386 processors) were 100% oversubscribed. It observed that a processor with an 8 MIPS processing rate would still struggle to keep up with the processing demand.

We found that the Lynx OS X-window implementation would serve adequately as a window server—it performed on a par with other X-window environments such as Sun Microsystem's 3/50.

We found that the RODB performance and architecture were woefully inadequate to meet the demands of current DMS requirements. Assuming an RODB that ran 100 times faster than it currently runs on the preliminary prototype DMS kit, we computed that each of the 10 SDPs in the DMS network would be overload anywhere from 343% to 1096% just mirroring RODB objects on the ground—ensuring that ground-based systems had the same data as onboard systems.

MOTIVATION

The last decade has produced a plethora of computer hardware. Every few years another, more powerful, processing element arrived on the market place. Software designers and programmers, writing machine specific software, could not keep up with the rapid rate of hardware evolution. Hardware no longer dominated the cost of a computer system, software development became the major cost factor. In an effort

to decouple themselves from specific hardware platforms, software designers began pursuing processor independent and display independent development methodologies. However, such methodologies incorporate inherent compromises that necessarily sacrifice execution speed for ease in porting. Despite this possible drawback, such methodologies have been adopted for use in developing SSF's DMS.

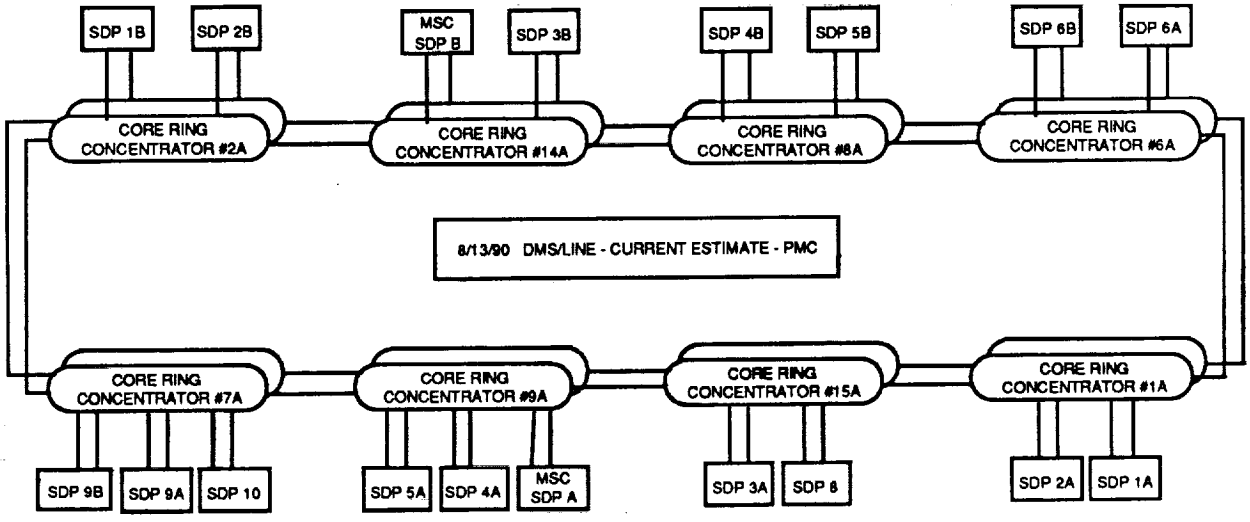
The functionality of Space Station Freedom's Data Management System (DMS) has been well defined, but very little analysis has been done to determine if the DMS can perform its tasks in the time allotted. Many hardware and software decisions have been made without considering the overall performance requirements of the DMS. Some choices might severely impact SSF operations, yet no systematic study quantifying the impact of design decisions on DMS performance can be found.

THE BASELINE DMS

The Data Management System (DMS) is the SSF subsystem that integrates onboard information into a cooperative whole. It provides integrated data processing and communications for both the core and payload functions, access to operational information by the crew through the Multi-Purpose Applications Consoles (MPAC), and interfaces to virtually all other onboard subsystems including the Communications & Tracking System (CTS) which links all the digital data to the ground.

Hardware Environment

The baseline hardware design for the Preliminary Manned Configuration (PMC) consists of 17 Standard Data Processors (SDP's), 2 Mass Storage Units (MSU's), 4 Multi-Purpose Applications Consoles (MPAC's), 63 Multiplexor-Demultiplexors (MDM's), and various associated communication links: the DMS Optical Network,



two types of local buses, and High Rate Links (for higher bandwidth payload telemetry).

Each of the hardware components is built into one of a small number of chassis called Orbital Replaceable Units (ORU's). All ORUs plug into a common backplane consisting of standard power connections, thermal sink connections, back-plane buses (Multibus II and MicroChannel), and appropriate communications networks. The SDPs, MSUs, and MPACs each contain an Intel 80386DX processor chip as their central processing element. The communication links conform to popular standards appropriate to their rates: 1553B (1 Mbps), 802.4 (10 Mbps), FDDI (100 Mbps), and passive fiber-optic cables and patch panel for HRL's.

The DMS hardware also includes the Emergency Monitor and Distribution System (EMADS) which is purposely independent of the DMS network, and the Time Generation Unit and Time Distribution Bus. The hardware design is specified in great detail in the Architecture Control Document (ACD) and the CEI Specification.

Software Environment

The development language chosen for SSF is ADA. The underlying real-time operating system is Lynx OS, a real-time operating system based on Unix. Lastly, the display environment is based on the X-window display system.

The DMS consists of eight Computer Software Configuration Items (CSCI):

- Network Operating System (NOS),
- OS/ADA Run Time Environment (OS/ADA RTE),
- Standard Services (STSV),
- Data Storage and Retrieval (DSAR),
- DMS User Support Environment (USE),
- System Management (SM),
- Master Object Database Manager (MODB),
- Operations Management Application (OMA).

The requirements and preliminary design of these CSCI can be found in IBM's Critical Design Review (CDR) documents. In overview, the NOS provides applications with the functionality needed to perform network transparent operations and POSIX compliant real-time operations; the OS/ADA RTE provides a paired down operating environment (though no decision has been made as of yet, it appears that the NOS will run on SDPs while the OS/ADA RTE will run on the MDMs); the STSV provide access to DMS services, most importantly access to the RODB; the DSAR provides access to the MSUs; the USE controls an application's access to an

MPAC's display; the SM controls the initialization, re-initialization, and recovery of the DMS system at initial start up, warm, and cold boot; the MODB permits the creation and initialization of new objects in the RODB; and finally, the OMA has oversight functions over all SSF onboard operational activities.

Of the eight CSCIs, the Operations Management Application acts more globally than the other seven CSCI. It, in fact, acts much like other DMS applications: it uses the functionality provided by the other seven CSCIs to perform its intended tasks.

The DMS presently contains nine applications:

- Communications and Tracking (C&T),
- Crew Health Care Systems (CHeCS),
- Electric Power System (EPS, SEPS),
- Environmental Control and Life Support System (ECLSS),
- Fluid Management System (FMS),
- Guidance, Navigation, and Control (GN&C),
- Operations Management Application (OMA),
- Thermal Control System (TCS).

The nine applications perform various functions that ensure SSF's safe operation. They monitor and control all aspect of day-to-day and minute-to-minute SSF activities. Their names provide a rough idea of their functional responsibilities. Of the nine applications, only the OMA plays a duel role. It is part of the baseline DMS as well as a distinct DMS applications. The OMA orchestrates the activities of the other applications and, when necessary, arbitrates among the applications when conflicts occur. Unfortunately, none of these applications is available for testing.

Because the OMA is of such importance, we will keep its functional needs firmly in mind during our capability analysis.

The Operations Management Application

The OMA is the onboard portion of the Operations Management System (OMS). The OMS, as a whole, is tasked with the coordinated operation of Station. It is divided into two portions the Operations Management Ground Application (OMGA) and the Operations Management Application (OMA). The OMGA is located on the ground and performs longer term planning and scheduling functions. The OMA is located on Station and performs shorter term replanning activities and provides oversight functions. The OMGA and the OMA coordinate with one another via Short Term Plans (STP).

The Short Term Plan contains all the information necessary to perform station ac-

tivities over a time period lasting up to several days. Activities that the STPs can represent and control include: crew, system, element, and payload timelines; operational constraints and environmental requirements; resource availability, requirements, and allocation; system and payload input parameters; and, the automatic execution of command sequences.

The OMA must interact with the OMGA, Station crew members, and Station elements and systems. To perform these activities, the OMA uses most of the seven DMS CSCIs. It uses the NOS to communicate with the OMGA, the OS/ADA RTE to control real-time applications and inter-application communication, the STSV to access sensor values and control actuators stored in the RODB, and the USE to allow interactive communication with crew members.

DMS DESIGN DECISIONS

NASA had decided on a number of hardware and software issues as they relate to SSF. Some decisions were made on the basis of availability, other decisions seem to have less substantial rationales.

Hardware

NASA chose to simplify its hardware design by standardizing on the Intel 80386DX chip. The 80386 processor is quite popular, widely available, and well on its way to being space qualified. Intel also intends on continuing to produce a family of processor chips upwardly compatible with the 80386. This evolutionary path may be the only hope for the DMS since the 80386DX already appears to be underpowered. According to a recent trade study [2], SDP 5 and 6 show the 80386 as being more than 100% over utilized:

> "In fact, the amount of processing power required by the application EDP cannot be realized unless its processing rate... is at least 8 MIPS, however, this still results in relatively large queue sizes and a [still]-high utilization." (pp. 5-23).

In another vein, four years have passed since the selection of the 3-4 MIPS 80386DX. By assembly complete, the 80386 chipset will be 13 years old and the 80486 will be nearing 10 years of age. Past experience has shown that chipsets older than 5-7 years tend to be obsolete and thus less available in the marketplace.

It appears that the onboard DMS may suffer severely from a lack of computational capability. Insufficient onboard computational capability could require that ground-based computers or human operators assume the responsibility for minute-

to-minute operational decision making. Shifting operational control to the ground may have a tremendously negative impact on operational costs.

Software

NASA chose two mechanism for simplifying the complexity of DMS software. First it chose a restricted set of software development tools. Second, it adopted specific interaction protocols for use in the DMS's distributed environment.

Three software tools occupy a central place in NASA's movement toward vendor-independent software development methodologies: ADA, Unix, and X-Windows. ADA provides a hardware independent programming environment. ADA is the DoD standard embedded system development language. It has a broad base of support and thus an assured future. UNIX provides a hardware independent operating system. It is by far the favorite "new" operating system because of its flexibility, its portability, and its wide acceptance. X windows, provides a vendor independent windowing and graphics environment. X-windows has achieved broad industry acceptance and provides a display technology that decouples the application's display commands from the ugly details of device specific display commands.

All three tools decouple the programmer from hardware specifics by providing the programmer with a uniform environment containing well defined operations that get translated into the appropriate hardware-specific operations. ADA provides a programmer with a uniform processor. UNIX provides a uniform set of system services, and X provides a uniform bit-mapped display device. The price for simplifying the programmer's work is paid in potentially higher memory requirements and slower computation speeds.

In the area of interactions protocols, the DMS Critical Design Review (CDR) [1] lists 11 restriction that "must be followed if an application is to correctly operate in a DMS environment." Three of the restrictions specify how applications may interact with one another. The three restrictions are:

3) Displays are independent of applications. All data to be displayed is written to the Runtime Object Data Base (RODB). Display generators can read the data from the RODB at their discretion.

4) All data shared by programs is shared through the RODB.

7) All program-to-program communication is provided through RODB actions and attributes. This is true for both intra- and

inter-node communications.

These three rules make the RODB the repository for all non-local SSF information. By using the RODB in this manner, the DMS can decouple applications, one from the other. An application need not know which application or applications are responsible for generating its inputs, nor need it know which application or applications use the output it generates. All an application needs to know is the name of the object or objects in the RODB that contain its input and the name of the object or objects that will contain its output.

This decoupling between applications allows system designers and system users to view an application as a plug compatible software "card" that can be extracted or inserted into the "RODB data bus." Decoupling applications in this manner simplifies application design considerably and as such is conceptually quite useful. However, it creates a tremendous bottleneck: the RODB becomes a central data structure in almost all computations: all access of sensor data, all actuator changes, all inter-application interaction, all displays of information, and all operational changes. All information flows through the RODB. If the RODB cannot handle this traffic, if its bandwidth is too low, the DMS cannot function. "Centralizing" information, even in the partially distributed style of the DMS's RODB, seriously bends a cardinal rule of distributed systems: use no central data structures.

The choice of hardware, software tools, and interaction protocols may seem reasonable considering the projected 30-year life-span of Space Station Freedom. After all, it might makes sense to sacrifice a little efficiency in the short term for a greater degree of flexibility over the life of the project. However, by making these individual decisions independently of one another, we may have sacrificed the viability of Space Station Freedom.

THE STUDY

Duplicating the computational environment onboard Space Station Freedom is not currently feasible. Neither the exact hardware nor software is available; however, a good approximation to SSF's computational environment can be assembled. We did just this in two separate testbeds.

We examined three aspects of DMS performance. On the first testbed, we studied the raw processing power of the Intel 80386 processor and the ability of Lynx OS to create X-windows. On the second testbed, we studied the RODB's ability to retrieve objects.

We did not include an ADA compiler in our study for two reasons. First, the Lynx OS at present does not support virtual memory—without paging, none of the available ADA compilers can execute in 4 megabytes of memory. And second, Space Station Freedom has not settled on a specific ADA compiler: at present DDCI and Alsys compiler are both under consideration.

The First Testbed Architecture

Our first two studies were performed on a subset of the Ames Research Center (ARC) testbed. The testbed was configured to match the computational environment of Space Station Freedom as closely as possible.

The hardware involved in our testing consisted of an IBM PS/2 Model 70-A21 computer configures with 4 Megabytes of memory. The processor also contained a high density (1.44MB) diskette, a math co-processor, a 3Com Etherlink ethernet card, a 16" Color monitor, and a mouse, though we did not use either the diskette or the math co-processor in our evaluation. We also had available an IBM PS/2 486/25 Power Platform that allowed us to use the computer as if it contained an 80486 processor chip.

The software configuration consisted of the Lynx OS/386-PS real-time operating system, version 1.0. It included X-window support (the X386PS-F product) and TCP/IP support (the TCP/386PS product.) The testbed also includes the GNU-based development products including the gnu-C compiler, gnu-C debugger, and gnu-Emacs editor.

Raw Speed. We tested the PS/2 on a number of program using the built in performance features that Lynx OS provides. We confirmed that the processor performed at its nominal 3-4 MIPS rate.

X-Windows. Two possible ways of testing the X-windows facility present themselves. The first was a program that opens and closes a window inside a loop. Such an approach seems an inappropriate measure of X-window speed since it ignores program loading time. The quantitative amount of time necessary to open a window is of less importance than the subjective time necessary to open a window.

The second approach requires a program that opens a window, prints out ten phrases in different areas of the window, and then pauses. When the user clicks the mouse in that window, the program deletes the window and exits. This program let us simulate the time it takes to load a program and then open a window. Using a stop watch to perform timings our program was able to load and establish a window in

under one second—comparable in speed to the Sun 3/50.

We found that an 80386DX processor running Lynx OS and the Lynx X-window system served as an adequate window server when a single application was requesting window services.

The Second Testbed Architecture

JSC has constructed a DMS testbed. That testbed presently consists of IBM's preliminary prototype DMS kit. The kit contains of an SDP, an MPAC, and an third IBM processor that simulates the functions of an MDM and its attached sensors and actuators. The MPAC is connected to the SDP via a fiber interconnect. The SDP is connected to the IBM processor that simulates the MDM via a 1553 data bus.

The testbed does not contain the full suite of DMS software. Of the software that is available, not all of it conforms to the current baseline specifications. As an example, the preliminary prototype DMS kit uses IBM's AIX operating system version 1.1 as its underlying operating system rather than the baseline specified Lynx OS. The testbed software does include some DMS services, most importantly, it includes code to access the RODB. The testbed also includes a simulated DMS application (ECLSS) and as mentioned earlier an MDM simulator that include an emulation of sensors and actuators.

In the testbed the ECLSS simulation resides on the MPAC, the RODB resides on the SDP, and the simulated MDM resides on yet a third IBM processor.

Experience With The Preliminary Prototype DMS Kit

JSC contractors wrote a very simple test program to time RODB access. The program repetitively reads an 80-byte object 10 times and averages those ten reads over 100 executions. The program performs 1000 reads in total. The test program resided and executed on the SDP, the same processor that contained the RODB.

Four test runs were performed. In two of the tests the data object was available in RODB's local memory. In the other two tests the data object was not available in local memory and the RODB needed to retrieve the object from the MDM simulator via the 1553B data bus. The other factor that varied in these tests was the ECLSS application. In half the tests the MPAC was executing ECLSS. In the other half, ECLSS was not running. The following table summarizes the RODB access times in the four configurations. The values represent the average time it took to perform 10 RODB reads.

	ECLSS not running	ECLSS running
data available locally	4.29 sec	8.39 sec
data retrieved from MDM	13.59 sec	29.49 sec

The table is not a misprint. In the best case, with the information available locally and no application load, 10 RODB reads took 4.29 seconds or on average 4 1/3 seconds to access a single data object.

These number are significantly slower than what would be required if application actually used the DMS in the manner specified in the DMS CDR. Admittedly, the number are taken from a preliminary prototype DMS kit and thus the raw speeds are quite suspect; however, the numbers do point out a potential problem area.

Assuming that the RODB programmers work very hard in optimizing their code, they may be able to improve throughput by 2 orders of magnitude or 100-fold. In such a scenario, our best case access time would drop to 4 milliseconds. The remote RODB access times would require 14 milliseconds. Translated to bandwidths, the RODB could handle somewhere between 73 and 233 read requests per second.

If the RODB were to serve as central a function as that described in the DMS CDR, a 100-fold increase speed increase would be insufficient.

An Simple Scenario—Displaying Astronaut Information

A simple scenario illustrates the need for more speed in the RODB. The OMS must access the RODB when an astronaut requests information. The astronaut might float over to an MPAC and select a particular display. The associated display program would then retrieve the necessary information from the RODB, format it appropriately, and then issue the X-window commands needed to display the data on the MPAC.

We can envision two ways of performing this operation. If we assume that the display requires 40 values, we compute two different elapsed retrieval times.

Using the first technique, the application program accesses each object individually by sending a request to the RODB via the FDDI network. An RODB request message would have a network latency of 50ms, the access would require 4 ms, and the return message would require an additional 50 ms, a total of 104ms. Forty such request cycles would require more than 4 seconds. Another second is required to format, create, and display the information. Overall this technique would require over

5 seconds—an unacceptable amount of time.

Using the second technique, the application program optimizes RODB access by predefining an RODB display object that has indirect pointers to the remaining 40 parameters.This technique requires 100ms for the round trip message latency and 164ms to retrieve the display object and its associated 40 objects—a total of 264 milli-seconds to access and retrieve the necessary information. Another second is required to format, create, and display the information. This technique would require 1.3 seconds. Written in this manner, a request to display astronaut information could be performed in a reasonable amount of time—assuming no other operations were accessing the RODB at the same time.

These computed numbers are minimal elapsed times. Other factors such as process scheduling delays and RODB loading caused by other applications would impact elapsed times significantly. In fact, just running one other application (ECLSS) would double the time needed to retrieve the display parameters. A DMS loaded down with 9 applications could easily extend our computed times into the multi-second range--unacceptable from a human factors perspective.

A Second Scenario

The previous scenario is a simplistic one. It does not take into account much of the complexity found in the architecture of the DMS applications, though it does begin to show some of the problems SSF will face given a poorly performing RODB implementation.

In this scenario we examine the impact of another DMS design decisions: the mirroring of all RODB objects on the ground. We once again assume that a team of programmers can achieve a two order of magnitude improvement in RODB speed, that is that SSF's RODB will execute 100 times faster than the preliminary prototype DMS kit's RODB.

The latest scrub activities resulted in a trade study [3] that envisions SSF with only 8,000 sensor and 2,000 effectors, down from a previous 40,000 sensor/effectors. The study details a DMS with 500 scan lists each containing 16 sensors. If we assume an uniform distribution of sensor/effectors across MDMs and SDPs, and we assume a DMS that contains 10 usable SDPs each with an RODB, we would end up having 800 sensor values residing in 80 composite objects in each and every RODB. Given SSF's nominal sensor polling rate of 100 millisecond, each RODB would update its sensor values 8,000 times per second. This would give us a sensor object up-

OMIT — DUPLICATE

SUMMARY

This report evaluates three aspects of Space Station Freedom's (SSF) Data Management System (DMS), as currently envisioned, and it examines whether or not the DMS will be able to perform its operations function.

We approached this study by obtaining access to existing DMS hardware and software in prototype or pre-prototype form; examining the hardware and software for conformity to the DMS plan; quantifying the hardware and software performance; and, analyzing the results in terms of operational needs.

The report examined three areas of performance, the raw speed of the Intel 80386DX processor chip, the subjective speed of the X-window display system, and the operational capacity (the speed) of the Runtime Object Data Base (RODB) system—the central component of the DMS.

We found that the Intel 80386DX processor ran a nominal 3-4 MIPS rate depending on the application. Though the 80386 did run at its nominal processing rate, according to a recent trade study [2], that rate is inadequate. That study, using nominal latencies and speeds for required operations, concluded that two of the SDPs (80386-processors) were 100% oversubscribed. It observed that a processor with an 8 MIPS processing rate would still struggle to keep up with the processing demand.

We found that the Lynx OS X-window implementation would serve adequately as a window server—it performed on a par with other X-window environments such as Sun Microsystem's 3/50.

We found that the RODB performance and architecture were woefully inadequate to meet the demands of current DMS requirements. Assuming an RODB that ran 100 times faster than it currently runs on the preliminary prototype DMS kit, we computed that each of the 10 SDPs in the DMS network would be overload anywhere from 343% to 1096% just mirroring RODB objects on the ground—ensuring that ground-based systems had the same data as onboard systems.

MOTIVATION

The last decade has produced a plethora of computer hardware. Every few years another, more powerful, processing element arrived on the market place. Software designers and programmers, writing machine specific software, could not keep up with the rapid rate of hardware evolution. Hardware no longer dominated the cost of a computer system, software development became the major cost factor. In an effort

date rate of 800 times per second. To mirror all these values on the ground, one of the DMS applications would need to access each RODB 800 times per second. As a result each RODB would experience an overload of between 343% and 1096%. This overload would happen before any other DMS application accessed the RODB.

REFERENCES

[1] McDonnell Douglas Space Systems Company, Space Station Division, Software User's Guide (Data Management System) (DR SY-40.1), MDC H4542 Resubmittal 2, September 1990.

[2] System Engineering and Integration Trade Study: DMS End-to-End Simulation Analysis Report, MDC H4875, April 1990.

[3] System Engineering and Integration Trade Study: Runtime Object Database (RODB) Sizing White Paper, DR SY-01.3I, July 31, 1990.