N90-24988

# Some Issues Related to Simulation of the Tracking and Communications Computer Network

Final Report

NASA/ASEE Summer Faculty Fellowship Program - 1989

Johnson Space Center

Prepared by: Robert C. Lacovara, Ph. D.

Academic Rank: Assistant Professor

University and Department: Stevens Institute of Technology
Dept. of Electrical Engineering
and Computer Science
Hoboken, NJ 07030

NASA/JSC

Directorate: Engineering

Division: Tracking And Communications

Branch: Communications Performance
and Integration

JSC Colleague: J. C. Dallas

Date Submitted: August 11, 1989

Contract Number: 44-001-800

## Abstract

The Communications Performance and Integration branch of the Tracking and Communications Division has an ongoing involvement in the simulation of its flight hardware for Space Station Freedom. Specifically, the communication process between central processor(s) and orbital replaceable units (ORU's) is simulated with varying degrees of fidelity.

This report presents the results of investigations into three aspects of this simulation effort. The most general area involves the use of computer assisted software engineering (CASE) tools for this particular simulation. The second area of interest is simulation methods for systems of mixed hardware and software. The final area investigated is the application of simulation methods to one of the proposed computer network protocols for space station, specifically IEEE 802.4.

## INTRODUCTION

Simulation methods represent a broad area of knowledge and technique. The investigations herein are oriented towards a specific simulation, namely that of the space to ground subsystem of the communications and tracking system for the space station.

This particular simulation consists of a set of programs written chiefly in ADA which a) emulate the state of many ORU's and b) provide fault detection, diagnosis and recovery based on the status of the various ORU's. The simulation was written by a team of about six people, and is consequently sufficiently complex to warrant the application of CASE tools. The result of application of CASE tools to the existing simulation yielded some insight into the value of CASE tools and into the simulation itself. This is deemed useful in that the present simulation (called 3B) is a forerunner to more realistic simulations to be written in the near future.

During the study of the use of CASE tools, it became apparent that the techniques used for the simulation were highly specific. For example, some actual hardware (network hardware) will be available for use during the simulation sessions, and will therefore be incorporated. Predominantly synchronous techniques were used, and little "instrumentation" (other than journal files) of the actual simulation was written into the code. This study indicates that there is some reason to believe that some commercially available asynchronous simulation tools would be of value in future efforts.

A specific area of interest to the simulation group was the performance of certain computer networks specified for use on the space station. Some effort was made towards obtaining simulations of these networks, but this work remains to be completed.

This report now presents three sections which describe the study and results of the three areas described above: CASE Tools; Simulation Tools; and Network Simulation.

## CASE TOOLS

Computer Assisted Software Engineering (CASE) tools are intended to assist the development of large and complex software systems, particularly those which involve multiple programmers and extensive physical systems. At the present

time, there are several commercial CASE systems which are similar to a great extent.

These systems accept a specification of a software system as their input data. This input specification includes descriptions of the physical inputs to the system (human or machine generated information), descriptions of the processes needed within the system, descriptions of internal storage for the system, and specifications of the output product of the system (reports and forms.) The actual form of the specification is a graphic of data flows within the software system, and graphical representation of specific resources used in the system. The overall effect is to form a picture of a complex system which shows the flow of data through processes which modify the data.

From a description of a software system, the CASE tool makes checks for consistency of data flows from process to process, and compiles information on those flows. For example, redundant data flows, or data flows which are partially specified are automatically identified. Complete descriptions of the objects used in a system may be extracted by the CASE tool. Finally, some CASE tools can build significant portions of the final code from the description. Even simple CASE tools can produce data declarations in several programming languages. (Unfortunately, ADA does not seem to be a common choice.)

Figure 1 is the representation of the simulation of the Space to Ground subsystem of the Communications and Tracking system. External sources and sinks of data (command and status) are shown as square boxes. Data flows are shown as directed lines. Processes (programs and algorithms) which modify, use or generate data are shown as rectangles with rounded corners. In this specific case, the processes shown correspond to the efforts of single programmers. Figure 2 shows an "explosion" of a single process. The explosion has the same inputs and outputs as the corresponding process of Figure 1, but shows the internal processes in greater detail.

Associated with the data flows of Figure 1 and Figure 2 are specific record structures for the data passed. These records are not described further in this report, but by examination of the records some inconsistencies were noted in the system, and corrections were facilitated thereby.

The description of the simulation system shown in Figures 1 and 2 was obtained after most of the code was written. CASE tools, however, are intended to be used before the generation of code. Nevertheless, several interesting

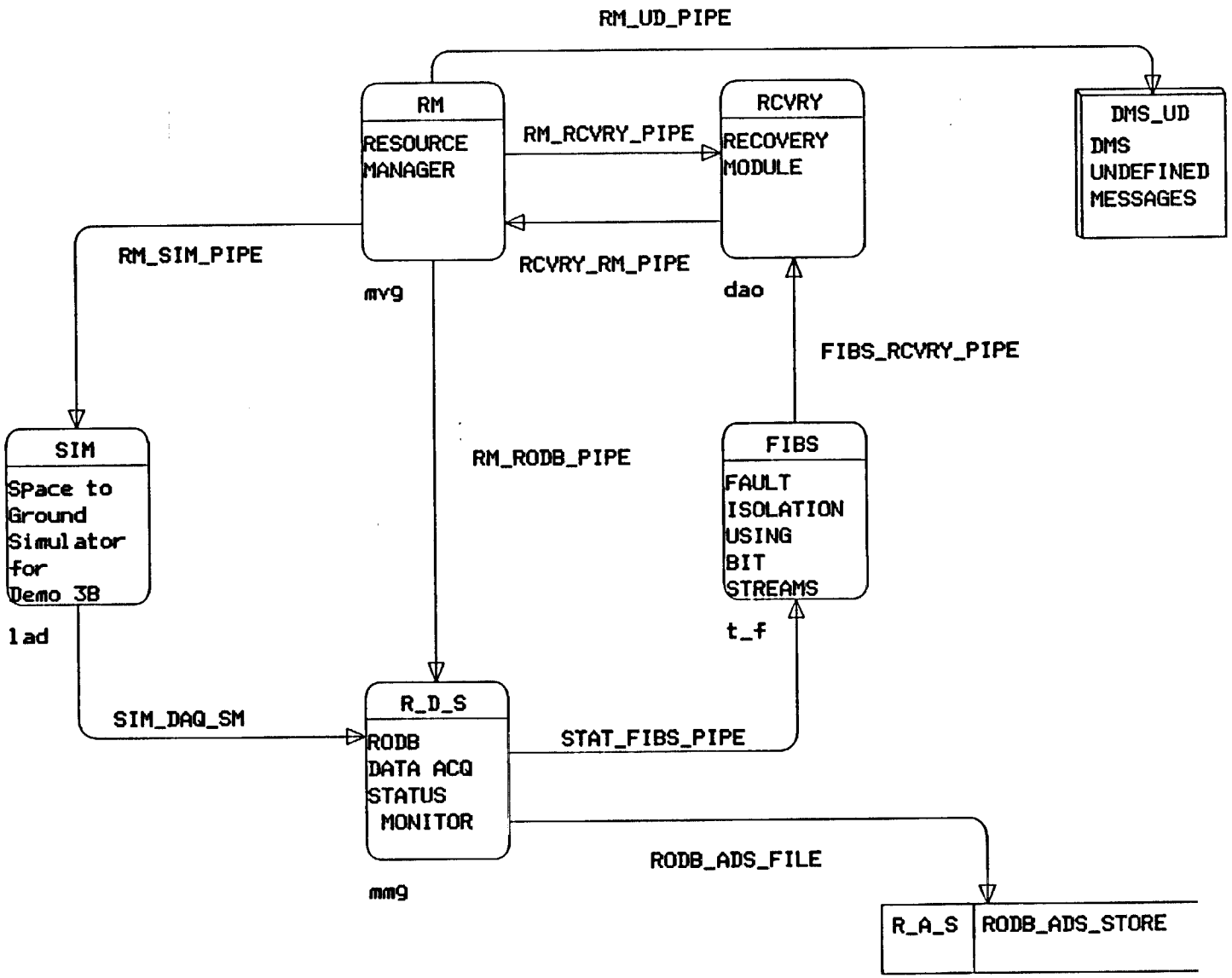


FIGURE 1: SPACE TO GROUND SIMULATION

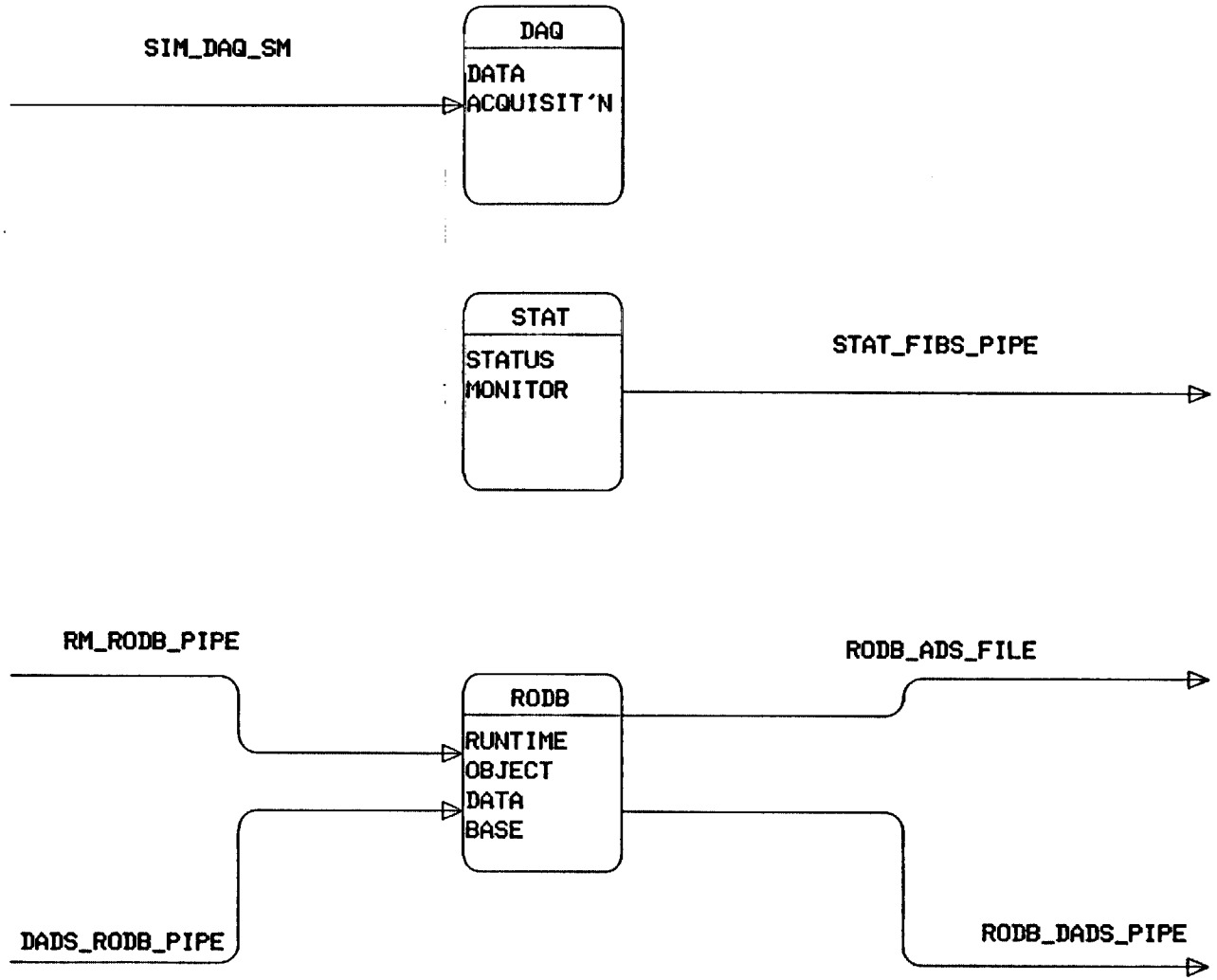


**FIGURE 2: EXPLOSION OF A PROCESS**

observations were made concerning the simulation system.

After the initial description of the simulation was distributed to programmers, several inconsistencies were noted in names of structures, and some inconsistencies in the uses of the structures were found. These faults would have surfaced readily enough at run time, but it is not clear that they would have been diagnosed quickly. The value of the CASE tool is that some types of problems which may be difficult to diagnose during debugging runs are actually explicitly visible from the output of the CASE tool. In that case, the potential error is corrected before "trial and error" runs are made.

A secondary effect of the CASE tool effort was a certain amount of interaction on the part of the programmers in an effort to standardize the use of some data structures. These changes were probably not important to the run time performance of the code. The changes tended to make different interfaces in the system look alike. Most likely, the overall clarity of the system improved by a small amount once all of the interfaces between programmers became explicit. (Prior to the use of the CASE tool, interfaces between programs were negotiated by the programmers pair-wise.)

Finally, it appeared that the existence of the CASE tool printouts provided a convenient "look and point" tool during some phases of the integration and testing of the simulation.

It seems reasonable that the use of the CASE tool even after most of the simulation was designed and written was of some concrete value. The total effort involved in the use of the CASE tool was not excessive: about 36 hours to learn to use the tool, and perhaps 8 hours to enter the description of the simulation system.

More importantly, it seems that much of the advantage of using a CASE tool will accrue from its use from the beginning of an effort. Programmers may then attend to the details of program operation from the foundation of a firm and unambiguous specification. This should speed the overall design, elaboration, and debugging process associated with a complex software system. It should also reduce the amount of time spent in "pair-wise" negotiation by the programmers.

## SIMULATION TOOLS

In the studied simulation, only one of the processes actually "simulated" anything at all. This is the module marked SIM in Figure 1. This process ran

multiple tasks which were numerical simulations of the behavior of ORU's. Other processes actually performed software tasks which will be performed in the future by flight software. Consequently, the other processes in the simulation are actually limited-scope study models of software yet to be written for flight.

The SIM module simulates some number of ORU's (about 40). Various parameters of the ORU's are simulated. However, the actual values are generated by keyboard input at a "command console." At the time of this writing, time-varying behavior of the simulation is being considered. To that end, several methods of simulation programming are considered.

There are several types of ORU in the simulation. Briefly, these may be considered only by the behavior desired of their various outputs. The simplest type of ORU has nearly static outputs and must only change for power up/down, or for gross failure. A second type of ORU reports the changes in some external event (auto-track antenna pointing, for example) and except for gross failure, may be characterized by deterministic functions of time. A third type of ORU has outputs which may be characterized best by some probability distribution function.

Individually, any of these three behaviors could be simulated by asynchronous or synchronous techniques. However, all three are needed and there are at least twenty if not forty systems which require simulation simultaneously. A reasonable method seems to be the use of a commercial simulation language such as SIMSCRIPT or GPSS, at least for the non-deterministic portions of the system. These languages hide the simulation mechanism (queues, lists, timers, etc.) from the programmer and provide access to asynchronous techniques. They are also well documented, and these two languages are both mature (more than 10 years old), and available on almost any computational platform (MS-DOS machines through supercomputers.) There is also no reason that the existing simulation (written in ADA) could not call tasks written in the formal simulation languages.

The chief benefit of the use of these languages is the ease of documentation and maintenance of the simulation. It is usually easy to change the underlying probability distribution functions in any simulation. However, changing the behavior of special purpose programs for simulation may require extensive re-writes. It appears that SIMSCRIPT or GPSS -type simulations are relatively

easy to change. This will be important as the fidelity requirement of the simulation increases.

## TOKEN BUS 802.4

At the time of writing, and prior to the August 1989 scrub exercise, the 802.4 token bus standard was to be used in flight hardware for low rate local area networks (LAN's) on Space Station. The simulations discussed above have not at the time of writing progressed to simulations of the LAN's, but this is anticipated. Accordingly, a study was initiated with the intent of identification of a means of simulation of LAN operation. As 802.4 had been specified, the simulation of this LAN was of some interest.

Interestingly, a great deal has been published on LAN simulation, and 802.4 in particular. The references which follow pertain to LAN simulation and performance. The 802.4 standard was studied extensively in the early 80's by the National Bureau of Standards (NBS) by a working group composed of both industrial concerns and the NBS.

This group did a vast amount of work in 802.4. The NBS team wrote several simulations of the standard. Further, it built an instrumented hardware implementation of a network using four or six nodes, and verified its simulation programs against the hardware systems. Results of these efforts are published in several workshops and conferences. Refer to the references.

At the present time, little commercial interest remains for 802.4 token bus. The chief reason appears to be its performance in comparison to 802.5 token ring systems and other systems of higher data rate media. In the papers surveyed and listed in the references, there appears to be no performance regime in which 802.4 is superior to 802.5 [Stuck 83]. The 802.4 standard does seem to outperform Ethernet, but this is hardly surprising, in view of the fact that Ethernet uses a collision resolution algorithm to resolve contention (specifically, CSMA-CD.) Furthermore, LAN's which employ higher rate media than 802.4, such as FDDI (at 100 Mbs as compared to 10 Mbps) seem to have leap-frogged the 802 standards for many applications. These higher rate systems provide multiples of 802 performance data rates at far under multiples of cost.

In the process of investigation, a public-domain simulation of 802.4 was discovered at the NBS. This is written in SIMSCRIPT, and has been obtained for any value it may present to simulation authors in the future. This simulation

may be useful when the details of LAN operation are desired in the Space to Ground Simulation. Further work remains in order to use the simulations.

## CONCLUSIONS

During this summer residency, the author divided his efforts along the three lines described above. The simulation efforts were particularly interesting since they have led to a consideration of mixed simulation systems. These systems will consist of hardware (probably actual LAN hardware), software written specifically to imitate flight functions, and software written in formal simulation languages.

## REFERENCES

(Alphabetically by first author)


ANSI/IEEE Standard; Draft International Standard, "802.4 Token-Passing Bus Access Method," ANSI/IEEE Std 802.4-1985; ISO/DIS 8802/4.

ANSI/IEEE Standard; ISO Draft Proposal, "802.5 Token Ring Access Method," ANSI/IEEE Std 802.5-1985; ISO/DP 8802/5.

Jean-Luc Archambault, "An IEEE 802.4 Token Bus Network Simulation," U. S. Dept. of Commerce, National Bureau of Standards, NBSIR 84-2966.

Werner Bux, "Local Area Subnetworks: A Performance Comparison," IEEE Trans. on Communications, Vol. Com-29 No. 10, Oct 1981.

Imrich Chlamtac and Raj Jain, "A methodology for building a simulation model for efficient design and performance analysis of local area networks," Simulation, Feb. 1984.

M. Alex Colvin and Alfred C. Weaver, "Performance of Single Access Classes on the IEEE 802.4 Token Bus," IEEE Trans. on Communications, Vol. Com-34 No. 12, Dec. 1986.

Rhonda Alexis Dirvin and Arthur R. Miller, "The MC68824 Token Bus Controller," IEEE Micro, June 1988.

Larry Press, "Benchmarks for LAN Performance Evaluation," Comm. of the ACM, Vol. 31 No. 8, Aug. 1988.

William Stallings, "Local Network Performance," IEEE Communications Magazine, Vol. 22 No. 3, Feb. 1984.

Bart W. Stuck, "Calculating the Maximum Mean Data Rate in Local Area Networks," IEEE Computer, May 1983.