The data system environment simulator (DASYS)*

by LAURENCE E. DeCUIR and ROBERT W. GARRETT

System Development Corporation
Santa Monica, California

INTRODUCTION

For the past nine years, SDC has been Integration Contractor to one of the largest satellite tracking, commanding and control networks in the nation. During this period the software portion of the total system has increased in both dollars and development time. Software is now a major element in the over-all system cost.

One of the prime factors complicating this situation is the typical requirement to implement such software on a hardware system that is in a parallel state of development. A second is the use of actual mission supporting system hardware for development and integration of new software capabilities. The nature of this problem differs from that of initial over-all system development since the hardware already exists in developed form. However, the interference between normal operations of support hardware and software development can cause major disruptions to both efforts.

Past attempts to alleviate this problem have included a variety of approaches utilizing software only, hardware only, or a combination of the two to simulate system functions in a non-operational environment. Although the results of these efforts have been good, they have also been piecemeal, and the time required in the operational environment has still been painfully large.

The support role and configuration

The mission of the organization for which SDC is Integration Contractor is to acquire, track, command, monitor and recover spacecraft in a multiple satellite environment.

The organization utilizes a very large, extremely complex computer-based general purpose command and control system. The system consists of a central computer complex with associated command functions and a number of tracking stations located throughout the world. An elaborate communications network connects the stations with the central computer complex.

The general purpose portion of the system provides acquisition data to the various tracking stations, acquires the satellite as it passes over, collects telemetry data, transmits commands to the satellite, and provides tracking data used to update the ephemeris for future acquisitions.

The system provides the eyes, ears, nervous system, and muscle to the over-all process.

Each tracking station has one operational computer. This computer handles the tracking, commanding, and telemetry data. It is tied to a buffer computer in the central computer complex, which can be automatically switched from station to station as it follows the orbiting satellite. Orbit ephemeris computations, acquisition predictions, command generation, and other associated functions are performed on off-line computers using data from the buffers. Acquisition and command data are transmitted through the buffers to the tracking stations.

Because of the size and complexity of the computer programs that make up the system, the organization employs a number of separate firms for software development. SDC has contracted to integrate the resulting computer program system.

SDC's integration role

A software system is a host of computer programs: each performs one aspect of a complex job, each must be coordinated and compatible with the others. A system must be complete and must effectively fulfill the requirements for which it was designed and built. To say that a system is complete and effective means that all of its parts fit together, that they cover all of the tasks

* © Copyright System Development Corporation 1970.
necessary to do the total job, that the parts and the system have been tested and proved reliable, and that personnel have been trained to operate the system and keep it running. The process of making a system complete and assuring its quality is called integration.

To comprehend how sizable the integration effort can be, one need only consider the following:

- There are currently five system program models in various stages of work, and there have been as many as seven. Two are operational, one is in the final phase of validation testing prior to installation, one is being coded, and one is in design.
- Each program model consists of approximately 1,600,000 instructions. New models are introduced based on changing requirements.
- Change control (design changes and program changes resulting from detected errors) must be maintained for each model. A typical program model has dozens of design changes and hundreds of discrepancy correcting changes made during its dynamic operational life.
- The system uses three different computers. Conversion to new configurations requires retraining of personnel.
- Interface requirements must be specified for each model. New or modified simulation techniques and tools, so vital in the validation testing activity, must be developed continually.
- Training of operations personnel and on-going support must be provided for continuity and feedback.

The scope of SDC's integration effort varies slightly depending on the schedule for each program model, but in order to provide continuous quality control and assurance, a level of effort concept is clearly mandatory. A discussion of problems encountered in software development in the live environment follows.

OPERATIONAL PROBLEMS

The software developer has a great deal of confidence in his ability to cope with problems which occur in a closed computer and computer peripheral environment. When this environment is extended to include complex, real-time asynchronous acting and geographically remote system elements, his confidence and ability are significantly reduced.

In any network-wide test condition which is set up to exercise software much of the system technical performance is invisible to, and beyond the control of, the software developer or integrator. He will be uncertain to the actual system element status and will generally rely on a telephone network for some of the information transfer and control. The setup of his test condition will require scheduling which impacts on normal system operations. Test personnel availability and travel to scattered locations are also some significant problem areas. This type of operation may also require extended time at remote locations to run tests, analyze results, and factor system changes back into the software under development.

The use of live vehicles in a system test environment is designed to provide conditions as close to operational as possible; however, even the best of these tests are often analyzed on a statistical basis in an attempt to get the most information out of a very expensive set of test conditions, but here again compromises are made to hold the cost and time involved to reasonable values.

In general it can be said that a command and control system involving real-time operational software must be tested or demonstrated in an environment as close to operational as possible prior to final acceptance. However, these tests or demonstrations are very difficult and costly to design, set up, staff, and control. These problems make it very desirable to reduce as much as possible the system time required for these activities in the operational environment.

After careful evaluation of the cost and utility of providing a complete data environment functional simulation system (test bed) to minimize the problems, SDC built the Station Ground Environment Simulator (STAGES) for the Air Force. The system consisted of interfacing hardware, a simulation computer and the simulation system software. An updated version of STAGES is being installed now.

The Data System Environment Simulator (DASYS) represents a refinement and expansion of the STAGES system on a custom basis to accommodate any simulation requirements.

ALTERNATIVES

The ground rules established for the design and development of the software test bed are:

1. That the total support environment be available (includes telemetry plus tracking and command),
2. That a direct operator interface be provided through the Operator Console,
3. That the operational computer process be as close to real-time as possible,
4. That software being tested be in the exact deliverable condition (no octals required to accommodate test bed peculiarities),
5. That dynamic control of every bit of each word passing through the computer interface be available,
6. That recording capability of the operations be available for analysis,
7. That environment modification require a minimum of software changes.

Two alternatives for a software test bed were considered by the Air Force before selection of SDC's DASYS approach.

1. That the functional support environment be provided to the real-time computer by means of digital simulation (the DASYS approach),
2. That the configuration consist of all support components of the actual tracking station.

The advantages of Software Checkout by simulating the functional support environment are:

1. Complete real-time environment is provided for testing, using dynamic inputs,
2. Implementation and operation costs are significantly less than in a facility using real hardware,
3. The user controls the test environment; there is no requirement for a larger complement of support personnel or extensive communication systems,
4. Versatility is provided by simulation rather than actual components. This allows:
   (a) Practically unlimited computer provided data input parameter control
   (b) Experimentation with alternatives
   (c) Automatic or manual control of environment
   (d) Operation and control by user with minimal support,
5. Computer time, calendar time, and cost for program checkout are reduced,
6. There is ability to repeat specific tests for each program model and automate comparison of results,
7. The system can be readily modified to reflect new interface characteristics,
8. Non-interference between hardware and software subsystem checkout processes is guaranteed through final acceptance,
9. Real-time dynamic and post-test analysis of all computer/hardware interface is available,
10. Real-time program checkout is conducted independent of operational system availability,
11. Simulation of hardware anomalies and future hardware capabilities is available prior to equipment availability,
12. Software checkout and integration support are the sole roles,
13. Additional applications areas are available for:
   (a) Mission rehearsal
   (b) Operator training under nominal and anomalous real-time conditions
   (c) Data system test and exercise.

The only disadvantages of simulating the support environment are:

1. Inability to investigate hardware-specific problems,
2. No checkout of hardware diagnostic types of programs.

The advantages inherent in the actual tracking station configuration were:

1. The system could be used to accept, check out, and integrate hardware to be installed in the tracking stations,
2. Hardware modifications could be tried and tested in a non-operational environment,
3. Hardware diagnostics and other supporting programs could be checked out on a non-interference basis,
4. The equipment components could be used as a training aid for both hardware operators and maintenance personnel,
5. The system would provide a limited operational software checkout capability on operational computers.

The fact that only a limited checkout capability existed in the duplicated tracking station test bed (alternative 2) plus the disadvantages listed below were the reasons for not taking this approach. The disadvantages are:

1. The configuration resembles a tracking station to such an extent that many of the disadvantages of site use for software checkout also apply to the test bed.
   (a) Full-scale preventive maintenance is required for configuration components because real, rather than simulated, hardware is used.
   (b) All engineering changes apply to the components, the same as they would to the tracking stations. This increases system costs and interferes with software checkout.
   (c) Extra manning is required for monitoring and operating the components that have no real bearing on the software subsystem test.
2. The primary role of the test bed would be hardware support. The amount of software checkout time required would all but eliminate this role.

3. Parameter control is severely limited.

4. No capability exists to:
   (a) Checkout completely controlled error conditions for any vehicle command functions
   (b) Run with multiple controlled environment
   (c) Repeat tests under manual control.

5. The absence of system-oriented hardware/software technical interface personnel hinders resolution of problems.

The first alternative was chosen as being the best and most economical solution for developing the software test bed. The entire data system functional environment would be provided by a relatively small simulation computer.

THE DATA SYSTEM ENVIRONMENT SIMULATOR (DASYS)

There were two approaches available for the design of DASYS. First, the simulation computer could be large enough and fast enough to provide information to the operational computer in real-time or near real-time. This would require the simulation computer to be several times larger than the operational computer and the test programs would have to be essentially real-time programs. This approach—although used by several government agencies—did not offer any savings over the conventional method of program testing and evaluation; in fact, the cost of the simulation computer and the cost of writing the real-time programs for this system would cause it to be more expensive than using the actual tracking station.

The second approach was to use a relatively small simulation computer, operating in nonreal-time, and to modify the operational computer to make it operate as if it were in real-time. This is accomplished by stopping the clock in the operational computer while the simulation computer processes the next input stream. Thus, the operational computer is operating in segmented real-time.

General description

The Data System Environment Simulator consists of a small simulation computer with associated peripherals plus simulation software and interfacing hardware (Figure 1). In this system, user operational controls and displays are included in the interfacing hardware.

The interfacing hardware in the system provides all of the externally generated signals and data to the operational computer including user controls in the same manner that the live environment does when interacting with a live moving vehicle.

Since the simulation computer must complete its functions without compromising the timing integrity of the operational computer, the operational computer clock may be stopped before the next input from the simulation computer. This permits operational computer transactions to be performed as if the operational computer were operating in a continuous time mode, and allows the simulation computer to run in nonreal-time. Thus, the environment simulation is not time constrained by the simulation computer.

One part of the simulation system "drives" the environment by functionally simulating data, equipment responses, and operator interactions to whatever level is required for the purpose at hand. Operations in real-time, delayed time, and accelerated time are all possible. During software tests, data rates and complexity may be varied over ranges that are much broader than those in actual operations.

Another part of the simulation system records the results of each run, so that the software's interaction with its environment may be analyzed in detail after the simulation run. The required analytic tools may be run on the simulation computer.

The user can operate the system in any of three ways: there is manual control of the system using the hardware alone, automatic control using previously generated tapes, and semi-automatic control using tapes and manual intervention with the hardware.

The capability to simulate a data system environment is defined generically as data system environment simulation. The parameters of an operational system which must be considered are operational computer I/O channels, word size, and environment functional responses at the computer I/O channels.
Interfacing hardware

The interfacing hardware consists of electronic equipment that connects the simulation computer to the operational computer. The hardware complements the simulation computer by providing functions not feasibly provided by software either in cycle time or economy. It also provides monitoring functions of the data exchange occurring at the operational computer interface by providing all intelligence with time tagging in microseconds for recording on magnetic tape. No modification has to be made to the operational computer data channels due to the matched interface logic provided by the hardware.

General functions provided are: master timing, operational clock control, data recording and time tagging, simulation computer interface, controls and indicators, and environmental functions which can be hardwired.

Operational controls and displays are provided to the user through the interfacing hardware to allow recording of user-generated inputs or inputs from the simulation computer.

Simulation computer

The simulation computer configuration includes appropriate peripherals to provide the programmed functional environment and control signals through the interfacing hardware to the operational computer. The simulation computer is smaller than the operational computer but will transfer data in and out of the interfacing hardware at a rate fast enough to cause the simulated environment to respond to the operational computer stimulus in near real-time. The ratio of operating time to real-time will normally be of the order of 1.5 : 1. The simulation computer configuration can perform utility and off-line functions for the operational computer when the required peripherals are included in the system.

Simulation software

The software operating in the simulation computer will provide nominal environmental inputs and responses to the operational computer unless a specific anomalous condition is programmed into the system. This computer will generally input varying environmental parameters from exercise-specific, system parameter tapes containing data such as track, telemetry and operator inputs, command responses, and communication data. These data will be output to the operational computer with timing integrity. Anomalies can be entered manually from a card reader or keyboard, or automatically from a tape.

The software can cause the printout of selected time-tagged operational computer interface data (during an exercise or after an exercise) from the recording tape which has been written by the simulation computer.

Summary

Using DASYS, a programmer can checkout operational programs in a real-time environment without disrupting system operations or having to involve others. There is considerable versatility in the way he can use both hardware and software. DASYS also permits experimentation with alternative programming approaches and determination of the exact time-tagged operational implication of each. There can be manual control capability of the system, using the simulator hardware alone, or the user can operate from a previously generated environment tape that provides automatic control. This real-time simulation is also practical for mission rehearsals, system tests, and operator training.

DASYS can handle functional simulation requirements for nearly all system elements—sensors, command equipment, telemetry receivers, radars, system operator consoles, timing systems, alarms, and displays. Functional simulation of hardware being developed and objects being or to be supported, sensed, or directed by the software system being tested can also be provided. It makes little difference whether these objects are satellites, missiles, aircraft, railroads, or items being run through processing plants.

COST EFFECTIVENESS

The process of developing, testing, and integrating large computer programs includes detecting and correcting program system errors. This points to a very significant way of measuring the effectiveness of any means to accomplish these tasks.

The parameters involved are numbers of errors, time between locating and correcting errors, and cost to spend system time in locating and correcting errors. Additional considerations include errors remaining in the system, level of confidence, system performance parameters control, and availability of test environment.

The measurement of progress in development of computer programs has been the subject of much study and analysis. The detection and correction of an error constitutes a step in this process. Measures of program status can be related to the rate at which new errors
come to light. The lower the error detection rate, the closer the system is to the error-free asymptote.

Figure 2 portrays the method in which software is normally tested and accepted. The vertical axis represents the level of confidence represented as percentage. The top horizontal line represents the limit or 100 percent level of confidence while the horizontal dashed line represents an acceptance level of confidence. When the acceptance level is reached the program or system is usually declared operational and turned over to the maintenance programmers. The horizontal axis represents time from the start of system testing.

The slope of Curves A, B, C, D, and E represents the error detection possibilities. Therefore, any curve with zero slope has zero error detection possibilities. Curve A represents the assembly and testing cycle. The initial error rate is steep since the program is assembled and cycled for the first time but levels off very fast as the limitation of this procedure is reached.

Curve B starts (point 1) when Curve A approaches its limitation and represents the system where the environment is simulated by hardware, function generators, and special purpose simulators. The error detection rate will immediately go up since the capability of simulating various subsystems of the live environment has been introduced. Again the cumulative number of errors detected and corrected cannot exceed the capability of the facility and it is necessary to go to the next facility. This system is limited primarily by the inability to test the total system and the limitations of special purpose hardware, generally analog, which cannot be controlled to the desired precision.

Curve C represents final testing in the live environment. It is a step curve since testing is limited by hardware status, scheduled support of operations, and the inability to repeat the exact set of conditions to test corrections. The length of the horizontal segments of this curve will vary depending on the above factors. The system is accepted when it reaches the acceptance line of the chart (t1).

Curve D represents the error detection possibilities of using a simulation computer operating in nonreal-time. Due to the fact that it can exercise the entire system and repeat the exact conditions as many times as necessary, it has steep error detection capability up through the acceptance level. Realizing that no one would accept a system that had not operated satisfactorily in the live environment, the acceptance test on the operational system could be accomplished at time t1.

Curve E represents a simulation computer operating in real-time. The only reason it does not have as high a confidence level as Curve D is that at some time in the testing, the capability of the computer to continue to provide real-time input to the operational computer would be exceeded. Larger and larger systems could be installed to eliminate this problem, but the hardware and software costs would be prohibitive.

There have been no statistics gathered as to how great a savings in time and money can be realized through use of the Data System Environment Simulator. It is estimated that the time savings (t2−t1) could be as much as 70 percent and that the dollar savings could easily be 50 percent or more.