On automatic testing of on line real time systems

by JON S. GOULD

Computer Consultants Corporation
Denville, New Jersey

INTRODUCTION

One of the major problems confronting the development of on-line, real-time information systems is overall system debugging—the final testing of all the integrated pieces working in concert under load conditions. An indicative analogy can be drawn from a look at similar project schedules (Figure 1), for systems of equal magnitude when one system is a conventional development and the other is on-line, real-time system (OLRT), our analysis will reveal the following:

1. Both systems require essentially the same amount of time and effort in the first three (3) phases.
2. During unit testing they are also quite similar with OLRT leading slightly.
3. In the final phases of integration, acceptance and cutover testing, the OLRT system extends several lengths over a conventional system.

The problem is quite simple. Although many would argue that OLRT systems are more intricate and difficult, I think most experts would agree that the larger non-OLRT systems are every bit as intricate and difficult and in many cases, perhaps more so. The problem, reduced to its simplest terms, is:

THERE ARE ESSENTIALLY NO COMMERCIALY AVAILABLE OLRT DEBUGGING TOOLS!

Therefore, the OLRT system developer is left to fend for himself, to develop whatever aids can be done within the scope of available resources. This effort, if it is undertaken at all, generally amounts to some weak attempts to build simple trap and trace subroutines, and perhaps some appropriately placed 'core dump' routines. This is integrated into the actual OLRT system and tailored specifically to this task and its associated programs, to the extent of rendering the routines useless to any other similar subsequent systems development effort. Therefore, when this project is completed, this 'test' code will be discarded or inactivated and probably never reused. Since test code is almost always required and never planned, it contributes greatly to project over-runs and in some cases system failure.

Let us look at the two types of projects (conventional and OLRT) from a testing and integrating viewpoint. A large scale conventional 'batch' processing system will involve several data files spread over disks and tapes, and will probably involve a variety of SORTS and MERGE runs, sprinkled liberally with processing programs. To test this system we must make up several dummy data files, or perhaps live data if it is available, layout a set of transactions which will test the sundry error conditions, and then run the system program-by-program, printing and analyzing the intermediate data and structures file as we go. Everything has been done neatly and locked-in to the processing unit in the computer room. Each phase or sub-task of the test is under the direct control of the programmer or operator.

Contrast this with the OLRT system: To test a system, which may look like those in Figure 2, we must now arrange to have operations personnel standing by terminals in various parts of the country to enter certain specified data, in a specified sequence, at our command (We have sent them the instructions several days ago and after many phone calls they now understand what they have to do.) We must establish communications, have the data entered, and after the test have the corresponding results from the remote terminals sent in
for analysis. Obviously, we have very little control over this type of testing. As the size of the system increases, the debugging problem grows exponentially. In summary, OLRT systems development is generally constrained and may fail because of:

1. A lack of sophisticated debugging aids.
2. System load testing to an over-capacity state is generally not attempted until cutover.
3. The test tools that are developed are custom tailored to an application and virtually useless for subsequent reuse.
4. Time compression capabilities to create real-time situations are not usually available.

It is desirable to have a method of testing, debugging, and analyzing an OLRT system that may be used over and over without extensive modification. A system should contain the following attributes:

1. It must be an easy-to-use debugging aid.
2. It must be reusable, which means it must be generally transparent to the application undergoing the test.
3. It must be able to subject the OLRT to data loads which are at or near its capacity.
4. It must be readily modified to insert special system functions.
5. It must be capable of compressing time—to test now, those events scheduled several months or years away.
6. It must be useful for debugging modifications and new applications after cutover, as well as during the initial development effort.
7. It must be available at the right time in the development schedule.
8. It should provide a useful training aid for maintenance programmers, operators, etc.
9. It must be ‘external’ to the OLRT system so that its influence or overhead is not additive to system occupancy.
10. OLRT should not require any special coding whatsoever to use the test vehicle.

This is a tall order, to say the least. However, the application description which follows provides these
things and more. This is primarily because the development of this system addressed itself to this problem head-on, it is not a by-product of an on-line system.

The technique itself is straightforward and simple; we assume that to provide meaningful tests of an OLRT system, especially during the integration phases of the system, the test vehicle should not reside in the same hardware system as the OLRT system. The OLRT system must be exercised in precisely the same manner as the 'real' network, including interrupt loads, etc. The primary differences between the technique described here and most others is:

1. THE TEST VEHICLE DOES NOT RESIDE IN THE ON-LINE SYSTEM.
2. THE TEST VEHICLE DOES NOT REQUIRE ANY MODIFICATIONS TO THE OLRT SYSTEM.

EQUIPMENT

The test vehicle, hereafter referred to as ATOLS, resides in a completely independent computer complex, which may be the back-up hardware for the OLRT system, or it may be contained in one of the popular mini-computers, which could be used to provide this function solely.

Regardless of the method selected, since the systems are independent, there is no requirement for any special engineering to effect the interface hook-up. Some extra common carrier supplied hardware, such as patch panels for communications lines and line 'battery' supplies may be required. However, this is standard off-the-shelf equipment of which most communications systems have an abundance. The application is described here, using the back-up computer system, since it was a completely redundant installation.

The communications carrier provides the equipment installation of phone lines and terminal board hardware which is commonly referred to as their DEMARKATION POINT or DEMARKATION STRIP. The common carrier terminates his equipment on the strip and provides a corresponding termination point for the attachment of 'foreign' equipment, such as the computer system. This is the termination point for the carrier's maintenance responsibility. The computer hardware vendor wires to the corresponding termination and the lines are connected and ready for on-line operations.

ATOLS is also terminated on the DEMARKATION STRIP in a similar manner as the on-line connections. The effect of this connection is to have both hardware systems 'split-wired' to the telephone lines. Figure 3 illustrates a typical connection at the DEMARKATION STRIP. It is important to note that the installation of the common carrier's equipment is not necessary to provide a simple ATOLS-to-OLRTS interface. If there is a delay in the common carriers installation, the machines can be temporarily wired together on a limited number of lines.

After these line connections, we are ready for testing. ATOLS is a fully implemented OLRT system with some special features added for the test environment. It has both on-line and off-line components for test data generation and data analysis, as well as the on-line operation.

THE ATOLS OFF-LINE SYSTEM COMPONENT

The off-line system provides for the generation of test data to provide the inputs for a test run, and the analysis of data that has been produced as outputs from the test run, after transmission through the OLRT system.

TEST DATA PREPARATION

One of the obvious problems of working with one computer driving another is to generate enough test data in the test vehicle to be able to sustain a volume level to be transmitted to the on-line system over a period of
time. Running at full speed, a system with several hundred lines would process several thousands of messages in an hour. Therefore, the system must have the capability to generate massive amounts of input information.

This is accomplished by a set of data preparation programs which accept punched card images and generates magnetic tape with the finished message formats and their destinations coded into the tape records. Data preparation is essentially free-form with the exception of a few control parameters, such as the number of messages or copies of the message to be generated and the hardware addressing characters, if any. Several fields are permitted to be imbedded within the data for the data generator program to update, such as sequence numbers, account numbers, and items of this sort that would be incremented or updated on message-by-message basis.

As the tape is generated, a copy of each message can be printed on the high-speed printer for verification of the final data format. The data generator function runs in the background of the on-line system and is able to produce test data while the switching system is in operation. However, it is generally done off-line so that the data can be verified before transmission to the OLRT system.

The recommended method of data preparation is off-line. The analyst determines the type of tests to be run in advance, codes the data, and it is keypunched and subsequently generated by the system. There are certain types of inputs that you would like to enter into the system while testing is in progress, such as special message types and console orders. For these inputs, a standard input device is available (the computer console, teletype, etc.) to enter traffic directly into the on-line system for transmission immediate action. Therefore, the data need not be completely prepared off-line, and may be entered at will from a supervisory station.

Figure 4 is an illustration of several input message formats that the data preparation programs would accept, expand, and generate to a magnetic tape file.

After the tape has been generated and the data content has been verified on the high-speed printer, ATOLS is ready to bring this tape into the system to queue the messages to the appropriate destinations. To initiate this process, the computer operator enters several commands through the operators console to 'close' the file and to 'load' it in for data transmission. At this point, the on-line data preparation programs take over, read the tape in, and queue the information to a disk or drum in preparation for transmission. For descriptive purposes, the generation and transmission have been separated; actually, these functions may be going on simultaneously.

When the data is loaded and queued to the lines, the system notifies the operator of the number of messages queued, queue reports to appear, etc. The complex is ready for on-line testing of the real-time system. At this point, the OLRT system is loaded in the on-line computer complex, and is awaiting input data or initiating some network action.

DATA ANALYSIS

The current implementation has several basic functions. First, the inputs to the data analysis package are the ATOLS journal tapes from the on-line run and the initial input data tape that contains the initial test data. These tapes are processed, merged, and sorted together to produce a single tape file containing the chronological events of the test. This file is input to a series of COBOL and FORTRAN data processing programs to reduce the data and to check its content. Some of the items that are checked are:

1. Message routing in a switching application is checked. The message sent by the test vehicle is matched against the message received to verify that it was transmitted on the correct line, and the time differential to transit the on-line system.
2. A check is made character-by-character of the message sent against the message received to see if any characters have been dropped or in some way modified from the original message. Characters are checked within boundary limits that are coded in the input data. This permits processing programs on the on-line system to modify the messages and still have some sections verified.
3. ATOLS contains the difference in the time the message was sent and the time it was received, the line it was received on, and the sequence it
was received. This is used to verify the sequence of priority messages in a system where several levels of priority transmissions exist.

The off-line system produces these reports on the high-speed line printer for further analysis. The significance of this is that a test of several hours under a full load, where thousands of messages are transmitted, is a virtually impossible clerical function to analyze for minute errors. ATOLS reduces this to a meaningful task, since it reports only on errors. The analyst need not only concern himself with the error traffic, rather than the total.

Other off-line functions, which are very useful are a series of utility programs to generate the communications network. The network to be generated for a given test is specified symbolically on punched cards. It is quite simple to run several completely different tests just by changing the network, and switching terminals. This facility allows us to compress time in an implementation schedule when remote installations are scheduled to cutover during several months or year periods. By generating a network now that will look like the network two years from now, for instance, we can test the system and debug it under that load and be assured today that when that 200th terminal is cutover some 18 or 20 months from now, the system is not going to collapse when the first message is sent. By providing network flexibility, these functions become available quite readily.

THE ATOLS ON-LINE SYSTEM COMPONENT

The on-line system component that provides the actual test vehicle is quite similar to most message switching systems. It is in fact a complete message switching system with all of the bells and whistles, plus a few subtle differences. The first being that only a minimal amount of message processing is done in the switching system because of the flexibility required. Second, error conditions are handled somewhat differently than they would be in a conventional on-line system in that ATOLS is more concerned with identifying time stamping and saving error conditions than it is in correcting them. Therefore, the test vehicle will time stamp and collect on file any data and or error condition that is detected on the lines or in the response to its stimuli. Aside from this, it has all of the components of a store-and-forward message switching system.

The software structure contains five main line program subsystems, which are:

1. The Base Level Executive.
2. Communications Control.
4. Input/Output Control System (Non-communications).
5. The Overlay Controller.

These are the mainline resident programs. There are other sub-systems for checkpoints and recovery, operator commands, and several utility and report programs to complete the system software. The latter are mentioned briefly to clarify the text as required; however, they are not worthy of a great deal of detailed explanation at this time, since they are not concerned with the main concept to be presented.

Figure 5 is a diagram of the software structure of the five mainline programs.

The Base Level Executive is the main control program of the test vehicle. Its functions are to determine the priority of the other major elements of the test vehicle and to pass control to these elements as required. Priority of execution of each element is established by its relative position in a monitor control table which is set up at system assembly time. All programs release control to the Base Level Executive, which maintains the status of this table and allocates resources accordingly.

When the Base Level Executive Program has serviced all the work it has to do in its control table and nothing is outstanding, the Executive loops in an idle timing loop, waiting for another activity to require action. In this state, it keeps track of this idle time, which is provided as one of the statistics in the system.

The Communications Control Programs are logically divided into input communications and output communications. The responsibilities of these are essentially
that of adaptive teleprocessing line control. The Communications Control Program has the following responsibilities:

1. To determine for each line in the system whether to send or receive on that line.
2. To address it or poll the line.
3. Determining if it must be dialed-up or it is private wire.
4. Receiving and logging any data other than the normal line addressing characters.
5. Notifying the logging and journaling programs that data has been received.
6. Providing all of the error testing and reporting for the transmission control units, lines, and the data.

The Line Control Programs are responsible for providing the characteristics of each line discipline that the system is testing. Also, it provides for the other functions of lines and terminals, such as to set lines up and down if excessive errors occur. Another function is to keep the system operator informed of network status, lines that are in trouble, and the characteristics that are on those lines. It also provides the basic timing functions for open lines, stuck transmitters, inter-character time-outs, no message time-outs—to protect the system from lines that may hang up for one reason or another. Again these occurrences are posted to the system operator.

The Message Processing Program is simple and straightforward. Since the test vehicle is essentially data transparent, the only requirement of the Message Processing Program is to properly route a message to its output queue. This is done from information supplied by the analyst on the data cards, indicating the line and terminal this information is to be queued to. The imbedded text of the message is not modified or analyzed in any way. On incoming messages that are generated by the on-line system, the Message Processing Program time stamps, sequence numbers, and writes the message to the journal tape for further processing.

The Input/Output Control System (IOCS) in the test vehicle schedules and executes all I/O operations for programs under the control of the Base Level Executive. With the exception of the operator's console and those communications line operations previously mentioned, IOCS contains all of the I/O channel and hardware device dependent error recovery procedures, addressing, and is initiated by a request from one of the Base Level Programs, after which it is interrupt-driven. That is to say that IOCS is dormant until a request is issued by some program under the Base Level Executive. It then schedules the requested operation and starts the physical I/O process. It is dormant again until further processing is triggered by input/output interrupts from the I/O channel. Inter-program communication between IOCS and the application program is accomplished through data control block linkages and the data per se. The programs currently support the disk drives, magnetic tapes, card reader punches, and high-speed printers. The IOCS program is queue driven and has several levels of priority so that it can be made sensitive to a particular type of data if that is desirable for the test.

The Overlay Controller Program and its buffer area is used in ATOLS to reduce the main core requirements for the system. Generally, Overlay Programs are used for supervisory controls, special processing routines, error procedures, and seldom used routines—such as retrieval, statistics programs where there is no restriction on time and/or space required. Any number of Overlay Programs may be included in the system since they are resident on external storage. Any Overlay Programs that are executed in the system under the direct control of the Overlay controller are initiated by the Base Level Executive Program. The Overlay controller is queue driven, such that any routine wishing to request an Overlay Program places the appropriate request information in a common queue for the Overlay Controller Program. When the request reaches the top of the queue, the requested program if it is possible at the time will be pulled in off the external storage device and executed.

This method of using the Overlay Program is under the direct control of the controller module, which permits some specially tailored routines for a particular on-line system to be included into the overall system without a great deal of effort required for programming. Since all of the mechanics of loading, executing, and scheduling are there, the only thing that need be provided is the actual processing program. This method allows for a certain amount of personalization or custom tailoring of software to a particular system to be tested.

The test vehicle is equipped with a command console to enter various orders to the system to re-configure networks, start up and shut down, recovery, retrievals, and things of this nature. In general, the types of commands which can be entered are to two categories. First there are network orders and second, general orders. In the area of network orders, we have line command orders and terminal command orders. These orders provide for dynamic network modifications, such as lines up, lines down, queue status reports, terminals up, terminals down, terminal status reports, and those functions associated with network parameters. The general orders group is a catch-all for everything...
else—set the time, set the date, start the system up, shut the system down, etc. These functions are provided so that the test director or the programmer debugging the on-line system has the ability to direct the test vehicle to perform in various degrees, depending on what options he is debugging at the time. He may want to run just one line, or he may want to run a hundred lines; he may want to stagger them, or produce instantaneous peaks. The programmer debugging the on-line system needs a great deal of control over the test vehicle to make sure that it provides the external inputs that he wants, when he wants them, and in the manner that he wants them.

Now, with all in readiness, the on-line system to be tested waiting for an external stimuli, the test vehicle primed with the test data to be transmitted, the programmer then brings up the test vehicle in a fashion that is suitable for the test—those lines and terminals that he requires. At this point, the test vehicle takes over and begins driving the on-line system in the way the ’real’ network will when cutover takes place. The programmer then sitting across the room on the on-line system can then proceed to debug various features including load capacity, graceful degradation, etc. As the test system is driving, the programmer has at his command the ability to increase the load, run in an ambient state, or decrease the load by entering commands into the test vehicle as the test proceeds. While the testing is in progress, the test vehicle is logging the activity that is going on in the network to magnetic tape. This magnetic tape will become one of the inputs to the off-line programming system for the analysis of the run, if required. In addition, as the system is running, a constant stream of information is printing on the operator’s console to inform the programmer of how the network is reacting, responding, what the queue status is, that the alarms are going off, and what errors have been detected.

The network and system constraints of ATOLS are primarily the amount of core memory available, and the bandwidth capability of the processor. The network specifications for this implementation provide for:

1. 256 communications lines.
2. Several thousand terminals.
3. Supporting most of the popular line disciplines, on a line-by-line basis (adaptive).

OTHER ATOLS USES

One of the extra benefits of the system has been to monitor and, indeed, substantiate modifications made to the on-line system several months after cutover. These modifications have been completely tested before being incorporated into the on-line system. We were able with a reasonable amount of insurance to go on-line with ‘new’ programs. In addition, the process of training operations and maintenance personnel has been greatly enhanced since we can provide real situations for analysis; we can overload the system, have the operators go through the fall back and recovery procedures (this is especially useful when an OLRT system stabilizes—the operations personnel tend to get rusty over a period of several months). ATOLS provides for continuous training throughout the life of the system, as required.

One benefit derived from the ATOLS system that was not anticipated at the outset of the project was monitoring of the on-line system. The test vehicle was modified to ‘ride’ the lines of the on-line system and to capture and time stamp the activities on those lines—sort of a ‘big-brother’ concept. We were time stamping data transmitted on the lines and monitoring the on-line system performance without having to introduce a large amount of test or monitor equipment. The test vehicle itself deals with the collection, and the off-line package provided the results.

BENEFITS

I believe at this point, many of the benefits are obvious. However, to mention a few:

1. The throughput and communication line error rate can be measured.
2. All of the error paths of the on-line system can be exercised randomly and over a long period of time.
3. Special features that may be built into the on-line system that may not occur very frequently may be tested.
4. Peak load and duration analysis can be made on the system.
5. Behavior patterns can be developed.
6. Routing information can be checked.
7. The entire problem of fall back, recovery switch-over under various load conditions can be completely tested.
8. The so-called degradation factors can be checked and the entire system configuration of the on-line hardware can be put to the test even though the lines and/or terminals may be several months from installation.
9. Time compression—The entire system for many months and many years can be run continuously;
those things that are planned in the future can be checked out now while the system and the software is fresh in the development terms.

SUMMARY AND CONCLUSIONS

The implementation of an on-line system can be a traumatic experience. The failure rate is high, largely due to the lack of tools available for testing and debugging large, complicated, time-dependent systems. It is my opinion that the only real way a large on-line system can be satisfactorily debugged is by another computer; as one computer to another supplying the inputs as they are required. The test vehicle described here is a giant step in providing such a tool. At the same time, it falls far short of being a panacea. However, experience to date has proved it to be a very liable solution to an extremely difficult problem. OLRT systems are going to be around for a while and there will be evolutions of systems even larger and more complicated. The cost of developing a test vehicle such as this becomes trivial and can be written off over several systems to be developed by the company over a period of time. It has proven to be more than simply a good, sophisticated debugging tool to test out a system before it goes on the air. It has also proven to be an invaluable training aid on a continuous basis to train maintenance programmers joining the organization to get some experience without upsetting the on-line environment, to train rusty operators, and to test other applications that are not necessarily involved with the on-line environment but run in the background of the on-line system.

Considerably more work should be done in this area to reduce the mortality rate of OLRT systems. The costs are tremendous; the probability of success must be increased. This is the challenge for the real-time systems community.