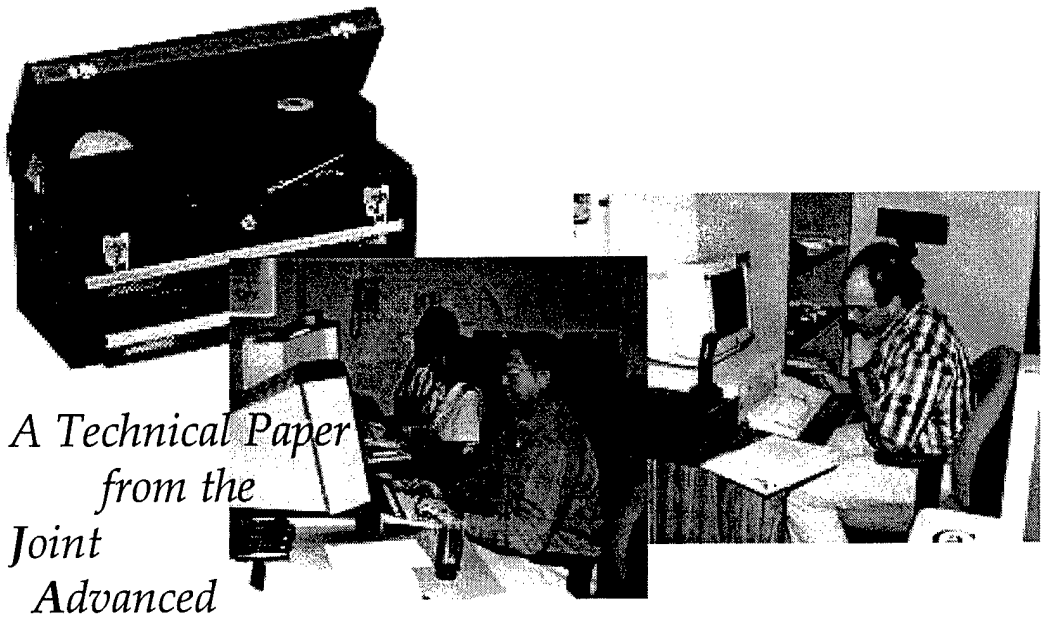
The JADS Analysis Toolbox A Tool for Analysis of Distributed Simulations



A Technical Paper from the Joint Advanced Distributed Simulation Joint Test Force

DISTRIBUTION STATEMENT A
Approved for Public Release
Distribution Unlimited

Mr. Dean Gonzalez, Science Applications International Corporation
Mr. Jerry Black, Science Applications International Corporation

Presented at the Simulation Interoperability Workshop
March 15-19, 1999

JADS JTF
http://www.jads.abq.com
11104 Menaul NE
Albuquerque, NM 87112-2454
(505) 846-1291
FAX (505) 846-0603

The JADS Analysis Toolbox
(A Tool for Analysis of Distributed Simulations)

Dean G. Gonzalez
Jerry Black
SAIC
11104 Menaul Boulevard, NE
Albuquerque, NM 87112
505-846-0527, 505-846-0603
gonzalez@jads.kirtland.af.mil, black@jads.kirtland.af.mil

Keywords: DIS, Analysis, Software

Abstract: *Science Applications International Corporation (SAIC) has developed a software product for the Joint Advanced Distributed Simulation (JADS) called the JADS Analysis Toolbox that has been very helpful in the troubleshooting, analysis, and visualization of distributed interactive simulation (DIS) data. The toolbox comprises a set of C++ routines integrated into a single user interface. It allows users to view tabulations and plots of protocol data unit (PDU) data in near real time, to play and/or get selected data post-test in text-readable format from JADS log files, and to obtain various plots and tabulations of PDU statistics for post-test analyses. This paper discusses modifications to the toolbox from its use in tests of short duration with only three entities to its use in tests lasting all day with 10,000 entities. The paper also discusses SAIC work in progress concerning the development of a similar toolbox for high level architecture (HLA) applications.*

Background

The Joint Advanced Distributed Simulation (JADS) Joint Test Force (JTF) is investigating the utility of advanced distributed simulation (ADS) technology for test and evaluation (T&E). To obtain data for its evaluation, JADS is executing three tests representative of portions of the T&E testing spectrum. These tests have linked live test assets, constructive models, and virtual simulations at multiple test facilities and test ranges across the country.

The first of these tests, the System Integration Test (SIT), was an air-to-air missile test in which participants were linked together via ADS technology. The test involved a shooter aircraft, a target aircraft, and a missile. In the early phases of the test, the aircraft and target were simulators. In the final phases, they were live participants. In all phases, the missile was represented by a hardware-in-the-loop simulation.

The second of these tests, the End-To-End (ETE) Test, was a Joint Surveillance Target Attack Radar System (Joint STARS) test. For this test, thousands of battlefield entities were created via Janus[1]. ADS technology was used to feed Janus data to the Joint STARS aircraft so that the Joint STARS displays would portray the battlefield below populated with Janus entities. The initial phases of system integration have been completed.

JADS discovered early on that the test controller and analysts need displays for troubleshooting, analysis, and

---

[1] Janus is an Army constructive, entity-level model located at White Sands Missile Range that represents up to 10,000 ground entities.

visualization. To meet these needs, Science Applications International Corporation (SAIC) developed a software product for JADS called the JADS Analysis Toolbox. The toolbox has been invaluable in JADS analyses. It comprises a set of C++ routines integrated into a single user interface. It allows users to

- view protocol data unit (PDU) data in near real time,
- replay PDUs from a log file, post-test, and
- obtain various PDU data/statistics, post-test.

The JADS Analysis Toolbox, developed for the SIT, worked very well for the SIT. However, without modifications, it was totally inadequate for the ETE Test. The point-and-click, snappy toolbox that worked so well for the SIT became a point-and-click, wait-for-a-long-time toolbox for the ETE Test.

The major cause of toolbox delays was the large size of the ETE Test log files. In addition, the number and method of creating entities for the ETE Test were significantly different from the method used for the SIT. This required changes in the way analysis routines handled data from the ETE Test files. The following review of the SIT and ETE Test illustrates their major differences.

The SIT had the following attributes:

- 3 entities
- 3,000 total PDUs per test
- 10 PDUs per second per entity
- 20 2-minute tests per day

The ETE Test had the following attributes:

- 10,000 entities
- 100,000 total PDUs per test
- periods with hundreds of PDUs per second and periods without PDUs
- 1 8-hour test per day.

This paper discusses the changes made to the toolbox routines to accommodate the ETE Test analyses due to the following issues:

- Entity List Issues
- Entity Identification Issue
- File Access Time Issue
- Memory Issues
- Busy Cursor Issue

This paper also provides a brief review of the JADS High Level Architecture (HLA) Analysis Toolbox currently under development.

JADS Analysis Toolbox

The following paragraphs provide an overview of the toolbox.

Toolbox Real Time Analysis/Monitoring Routines

The real-time analysis/monitoring routines provided by the toolbox display received PDU statistics and/or PDU latencies.

The PDU statistics include

- the number of PDUs received,

- the number of each type of PDU received, and
- the rate at which PDUs are being received.

The PDU latency display is a plot of latency (time in transit) for all entity state PDUs as they are received. PDU latency is the difference between the time when the PDU was received and the time the PDU was created, i.e., the PDU timestamp (see Appendix A for the method of handling the conversion of the timestamp to current time).

Toolbox Post-Test Analysis Routines

The toolbox has a number of post-test analysis routines. These routines are grouped as follows:

- routines for predefined analyses,
- routines that get ASCII data from the log files,
- a routine for replay of log files, and
- miscellaneous routines.

Discussion of these routines is beyond the scope of this document. However, the routines are well documented in **The JADS Analysis Toolbox User's Manual** (this document may be downloaded from the JADS website: http://www.jads.abq.com/html/jads/techpprs.htm in pdf format).

All of the post-test analysis routines presume that the JADS logger was used to record the PDUs.

Entity List Issues

Several issues arose that are related to the number of entities. These issues will be discussed in the following paragraphs.

Test Differences

For the SIT, one item of interest was data dropouts. Since the test was configured so that each entity issued 10 PDUs per second, a plot of the total number of PDUs received each second showed, at a glance, whether or not there were data dropouts. For example, if there were no data dropouts and the missile was not active, there were 20 PDUs per second; if the missile was active and there were no data dropouts, there were 30 PDUs per second.

For the ETE Test, data dropouts could not be established in the same simple way because the ETE Test entities had dead reckoning algorithms that did not provide for a constant PDU rate. However, a similar item of interest for the ETE Test was which entities had issued PDUs.

For the ETE Test, Janus was set up to handle 9,999 battlefield entities. Entities were generated based on the analyst's inputs to the JANDIS (the Janus distributed interactive simulation [DIS] interface) and Janus' own rules for creating and maintaining battlefield entities given a scenario. Because of bandwidth constraints, the generation of entities was set up to occur over a period of about one hour, using the DIS heartbeat. After one hour the heartbeat had issued about 3 PDUs for each entity and the heartbeat was turned off. Dead reckoning algorithms were set so that entity state PDUs were not issued unless entity velocities (magnitude or direction) changed significantly. In this manner, the ETE Test kept the PDU rate low enough to pump required data through a space satellite communications link to the live Joint STARS aircraft.

Thus, the analysts needed a list of entities that had issued PDUs, i.e., the list of entities for which entity state PDUs had been sent and received. For the SIT, this list was a simple list of three integers, and was easily displayed on the screen. However, for the ETE Test, a simple list was totally inadequate, as the following example illustrates.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41
42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79

80 81 83 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

The example above shows a list of 100 entities. For 10,000 entities, 100 more list boxes such as this would be needed. Further, it is not obvious from a quick review of the example that entities 82 and 84 had not issued PDUs. Thus, a method of displaying the active entities using straightforward lists was impractical and fairly useless.

Using a List Notation

It was decided to use a dash-comma list notation to depict which entities had issued PDUs. Using this notation, the desired information is readily apparent to the user and requires a minimum of screen space. As the following example shows, it can be easily seen that all entities from 1 to 10,000 are present except for entities 82 and 84.

1-81, 83, 85-10000

Using a List Class

In terms of classic C++ software development, a class was needed that would handle the nonsequential lists and transitions between the various manifestations of the nonsequential lists.

Integers Array	Nonsequential List	Nonsequential List Sets
1	1-3,i-n,N	1-3
2		i-n
3		N
. . .		
N		

One of the lists is just the array of integers. Another is the non-sequential-list, as discussed. Yet another is the set of non-sequential list sets required for displaying the lists in selection boxes. Once this class was established, it was relatively easy to manipulate entity lists to obtain the desired results.

Entity Identification Issue

Test Differences

The DIS standard permits an entity number at one site-host to be identical to an entity number at another site-host. Unique entity identification requires all entity identification fields, i.e., site identification (ID), the host ID, and the entity number to be filled in.

For the SIT, care was taken to ensure that entity numbers were unique. Thus, there was no need for the site and host IDs in the entity identification of the three simple players. However, this care was not taken in the ETE Test. The set of entities from 2 through 32 occurred at both the White Sands Missile Range site-host, as well as at one of the site-hosts from Fort Sill. Thus, site-host information was necessary to uniquely identify an entity.

Whereas a 1-character field sufficed for entity identification for the SIT, the ETE Test required a 17-character field.

Site ID	5 characters

dash	1 character
host ID	5 characters
dash	1 character
entity number	5 characters

Each of the entity identification fields (site ID, host ID, and entity number) is a number between 1 and 65,536. Most of the JADS analysis tools used the 17-column layout needed for general entity identification. However, some of the tools were set up so that the entities from Fort Sill (one node of the test) were assigned the numbers from 9969 to 10,000, while entities from White Sands Missile Range (another node) were assigned the numbers 1 to 9968). Thus the fields could be kept small, and the nonsequential list notation could be used.

A Lesson Learned

A lesson learned from this is that care should be taken to assign blocks of entity numbers to each site-host so that duplicate entity numbers do not exist. A Janus-type host may need a block of 10,000 numbers, but that still leaves 55,000 numbers to assign. If each site-host is given a block of 1000, that's 55 site-hosts, or more than enough for any envisioned tests.

There is a significant payoff in terms of software handling and in the general day-to-day work dialog when the source of a given entity number is unambiguous.

File Access Time Issue

Another major issue associated with the ETE Test, not present in the SIT, is file access time.

Test Differences

The time to read through a file increases directly with the size of the file. For SIT, log files were less than 400,000 bytes, and took less than two seconds to read. Analysis routines that read the file to determine start and stop times or entities present could take the time to do that task with barely perceptible delays to the user. However, the file examinations to get these data for the ETE Test took up to 90 seconds – much too long for a point-and-click response. The following paragraphs discuss the details and methods used to overcome at least some of these delays.

Reducing File Access Times

Almost all analysis routines needed to know the start and stop times once a user selected a log file and which entities were present for that test. Thus, these data, obtained via one complete pass through the log file, were saved to a hidden file[2] by the first routine using the log file. These files were hidden so that users would not tamper with them. The hidden file data are shown in the following example.

[2] In UNIX, files that begin with a period have the hidden attribute and do not display themselves with the normal file listing command.

PDU Start Time	00:00:00.332	(00000332 msecs)
PDU Stop Time	07:00:05.360	(25205360 msecs)
Log Start Time	13:41:04.597	(49264597 msecs)
Log Stop Time	21:40:23.278	(78023278 msecs)
List of Entities	1-9781, 9969-9999	
Number of Entities	9812	
Total Log Time	07:59:18	

Thus, the user was subject to delays associated with obtaining the start-stop times and the entities only *once* in the analysis of a given log file. Subsequent analysis routines had the hidden data available without having to read through the log file. This simple means of saving frequently needed data saved considerable time for the analysis routines.

Memory Issues

Another difference between system integration-type tests and ETE-type tests is the memory required for data calculations.

Test Differences

The basic data obtained from the binary log files for each entity state PDU and used in the analyses are shown in the following example.

Entity ID	int	4 bytes
Log Time	int	4 bytes
PDU Time	int	4 bytes
Latitude	double	8 bytes
Longitude	double	8 bytes
Altitude	double	8 bytes
TOTAL		36 bytes

These data were the same for both the SIT and ETE Test.

If an array of PDU data is maintained for each entity, then the expected size of the array is equal to the expected number of entities times the expected number of PDUs per entity. These numbers are shown in the box below.

Item	SIT	ETE
PDU data memory required	36 bytes	36 bytes
Number of entities	3	10,000
# PDUs per entity	2000	20
Total memory required	216,000	7,200,000

Thus, if an array of basic entity state data is kept in memory for calculations, 0.22 megabytes (MB) are needed for a system integration test, but 7.20 MB are needed for an ETE test -- that's before calculated data elements are included and assuming the user only works with one log file at a time.

Further, if software is set up so that it's independent of a particular test program, i.e., the maximum number of entities and the maximum number of PDUs are constants, then the total memory storage becomes

(10,000)*(2000)*(36) = 720 MB.

This led to a search for a method which did not require so much memory and which was independent of the number of

entities or PDUs in a log file.

A review of the toolbox analysis routine calculations showed that none of them "required" that all the PDUs be in memory, i.e., they could perform their calculations using only the current and previous PDU. (The standard UNIX function of DIFF and SORT were used in some routines to compare and sort files. These routines require, or at least appear to require, all of the data from the files in memory to do their work, but their memory allocation and deallocation are transparent to the user).

Thus, the routines were written so that, instead of placing all of the PDU data into memory and then doing the calculations, the calculations were done as the PDUs were read from a file. Thus the results were written into a results file during the reading and calculations. This method of handling the PDU calculations made the routines require a minimum of memory (less than 1 kilobyte). Also, the memory requirements became independent of the number of entities or PDUs – a very desirable attribute.

Busy Cursor Issue

The SIT did not require the use of busy cursors. When the user used the point and click dialogs, actions took less than two seconds, and results appeared nearly instantaneously. However, this was not true for the ETE Test.

As mentioned previously, because of the size of the ETE Test log files, a simple read through a log file required more than a minute. Without feedback to the user, it would appear that nothing was happening after the user clicked on menu items that triggered routines that read through a log file. Thus, a busy cursor was added to routines at these points. After clicking, the cursor immediately changed to an hourglass, providing feedback to the user that something was indeed happening, and that patience was needed.

Initial efforts included a busy dialog box in addition to a busy cursor. The busy dialog box informed the user of the cause of the busy status and provided the option of canceling the action. However, the dialog box with the option to cancel was dropped because it increased the processing time too much (the system had to keep checking the status of the dialog box to see if the user had canceled). More than likely, the option could have been fine tuned to reduce the time the system spent checking the cancel action. However this was not pursued, as the busy cursor seemed to do the job.

The JADS Analysis Toolbox for HLA

The JADS Electronic Warfare (EW)(HLA) Test

The JADS EW Test is using high level architecture (HLA) federations to represent all elements of an open air range test environment. The EW system under test for this ADS test is the ALQ-131 Block II self-protection jammer. The JADS EW Test federation links six federates on six host computers exchanging attributes and interactions representing EW systems operating in a live test environment.

Test Differences

Data Exchange Protocol: HLA has no predefined application-level data exchange protocol (such as DIS PDUs). Federations define the structure, format, and update rates of data exchanged among federates. Data consist of object attributes and interactions communicated from one federate to another through the Runtime Infrastructure (RTI). While this provides a flexible environment for linking simulations for distributed tests, it is this same flexibility that complicates data collection and analysis.

Data Logger: Since there is no standard protocol for data exchange, a "stealth" or omniscient logger can no longer recognize and record all simulation traffic. Thus, the prior JADS logger that worked so well for the SIT and ETE Test could not be used for the EW Test.

Analysis Requirements: In addition to the logger differences, the EW Test analysis requirements were different from those of the SIT and ETE Test. Whereas the SIT and ETE Test were mainly concerned with the effects of ADS on entity performance (position, velocity, etc.), the EW Test is concerned with the effects of ADS on interactions between entities (between the threats and the jammer). For example, an item of high interest is "Did the jammer responded in a timely manner to a mode change from a threat?"

HLA Tools Development Status

JADS RTI Interface Logger

The JADS tool initially developed for the JADS EW Test was the JADS RTI interface logger. This tool is the subject of another paper presented at this conference (see the references at the end of this paper). The JADS RTI interface logger captures all traffic in and out of a federate, i.e., between the federate and the RTI. As with the JADS DIS logger, the JADS RTI interface logger creates binary log files.

HLA Analysis Tools

JADS has had two analysis tools developed by two organizations external to the JTF. One of these tools, Automatic Data Reduction Software (ADRS), was developed by an organization with prior EW test analysis experience and tools, e.g., Georgia Tech Research Institute (GTRI). The other tool, an analysis federate, was developed by an organization with prior HLA analysis tool development experience, e.g., U. S. Army Training and Doctrine Command Analysis Center. These tools are described in the references at the end of this paper.

GTRI is the primary developer of tools for the EW Test. However, the JADS software development group has been active in developing tools to assist JADS analysis and network personnel in troubleshooting EW Test simulations, HLA, and network architecture during the test preparation phase. It is expected that these tools will be integrated in a JADS HLA Analysis Toolbox. These tools are the following:

logfile_summary: The logfile_summary program creates a report containing summary statistics for a log file. The report contains the name of the federate and the start time from the log file. Then for each attribute and interaction (data type) sent or received by the federate, the report lists (by source) the number of updates, the minimum, maximum, and mean latency for the data type.

display_time: For every attribute and interaction sent or received by the federate, the display_time program displays all of the time information associated with the update. The raw log time (seconds since 1970 and microseconds), the log time converted to milliseconds (ms) since midnight, the header time (in milliseconds since midnight), and the latency for each update are displayed.

mode_changes: The mode_changes program is used to show all of the threat mode changes and jammer responses that occurred during the test. The mode and code value are displayed along with the time generated and the time logged. The *federation* latency is the time from when a mode is generated by a threat until the time it is received by the jammer, plus the time from when a jammer response is generated by the digital system model of the jammer to the time it is received by the threat. The output from the mode_changes program is used to determine these values.

create_adrs_file: The ADRS, created by GTRI, is an important analysis tool used by JADS. It primarily displays threat and aircraft position data real time during a test. However, it is also an important post-test tool. For post-test analysis, it accepts comma-delimited files containing the attribute and interaction data. These data files are normally generated from the raw range data using a script generator. The create_adrs_file is a JADS-developed program that extracts attribute and interaction data from a log file and creates a comma-delimited text file that can be read into ADRS.

calculate_rate: The calculate_rate program calculates the update (output), receive (input), and total rate for attributes and interactions every second. The output file records contain the time (ms since midnight) along with all three rate

values for every second of log time. The maximum values for each rate value are printed to the screen for each log file.

read_jads_log: The read_jads_log program reads a log file and writes all RTI application program interface methods that were logged to an output file. Some of the method parameters are also displayed with the method name. This program gives a good picture of the sequence of events that occurred in the federate.

Much of the HLA analysis tools software reused software developed for the JADS Analysis Toolbox.

Summary

Early on, it was understood that much of the data management software could be used in more than one of the JADS tests. Design and coding decisions were made so that the software could be reused. User interfaces were standardized as C++ classes. Directory structures were standardized to organize the log files and analysis results by date. General purpose functions (e.g., time conversion and file manipulation) and classes were defined so they could be reused. Log file naming conventions were similar for all tests.

These decisions have proven to be wise. Modifications made to the toolbox to accommodate the large numbers of entities and the lengthy files of the ETE test did not require an extensive software rewrite; and, when completed, expanded the utility of the toolbox. Much of the new software developed for the EW Test analyses used existing DIS toolbox software. It is hoped that these tools will become useful as a JADS legacy product that can be improved and expanded to support future ADS test efforts.

References

Gonzalez, Dean G., and Jerry Black. "The JADS Analysis Toolbox, A Tool for Analysis of Distributed Simulations," December 1998. Prepared by SAIC for the JADS Joint Test Force.

Gonzalez, Dean G. "The JADS Analysis Toolbox User's Manual," December 1998. Prepared by SAIC for the JADS Joint Test Force.

Black, Jerry. "Data Collection in an HLA Federation," December 1998. Prepared by SAIC for the JADS Joint Test Force.

Murphy, Major William S. Jr. "Analysis Federate." A paper presented to the 1999 Spring SIW, U. S. Army TRADOC Analysis Center.

"Automated Data Reduction Software (ADRS) User's Guide," Georgia Tech Research Institute, 20 May 1995.

Author Biography

Dean Gonzalez is a senior software engineer for SAIC in Albuquerque, New Mexico. He has a B.S. in Aerospace engineering from the University of Texas in Austin and a M.S. in Engineering Management from the University of Missouri at Rolla. Mr. Gonzalez has more than 32 years experience in developmental and operational testing as an analyst and as a software developer. He has been supporting the JADS Joint Test Force as a software analyst responsible for real-time and post-test analyses tools since 1995.

Jerry Black is a senior software engineer for SAIC in Albuquerque, New Mexico. He has a B. S. in Computer Science from the University of New Mexico. Mr. Black has more than 17 years of experience developing software applications for DoD flight software testing and real-time data collection. He has been supporting the JADS Joint Test Force as a software analyst responsible for time synchronization, data collection, and analysis since 1995.

Appendix A: Calculation of PDU Time

The PDU time that was used by the SIT was the relative time specified by the DIS standard and amounts to the division of an hour (since DIS time recycles each hour) into what can be stored in 31 bits (the 32nd bit indicates that relative time is being used).

For the ETE Test, PDUs from the Janus site had simulation (or game) time, and PDUs from the Fort Sill site had standard DIS time. The game time was encoded in the DIS PDU as one millisecond for each of the 32 bits. The standard DIS time was encoded as relative time as prescribed by the DIS standard.

The log time used by the JADS logger is the UNIX time which consists of two words (one is the number of seconds past January 1, 1980, or so, and the other word is the number of microseconds past the second). Machines participating as loggers were set up so that the system times were kept in the Greenwich Mean Time (GMT) frame of reference using XNTP and a global positioning system time reference. In this manner, machines at sites in different time zones would record the same time, i.e., GMT, if an event occurred at the same time at both machines.

The method of converting the PDU time from time elapsed since the hour, since it was necessary to compare the log time to the PDU time, was to add the hour from the first log time in the file to the PDU time elapsed since the hour. Also, logic was included to increment the hour each time the seconds added up to one hour, and logic was included so that time would not roll over because of late PDUs arriving just after the hour had changed.