Proceedings of the 1997 Winter Simulation Conference
ed. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson

A PERFORMANCE MONITORING APPLICATION FOR DISTRIBUTED INTERACTIVE SIMULATIONS (DIS)

David B. Cavitt
C. Michael Overstreet
Kurt J. Maly

Department of Computer Science
Old Dominion University
Norfolk, Virginia 23529-0162. U.S.A.

ABSTRACT

This paper discusses the design, development, and use of a performance monitoring tool for Distributed Interactive Simulations (DIS). A typical DIS environment consists of hundreds of simulations distributed on both local and wide area networks. The adoption of the High Level Architecture (HLA) by the U.S. Department of Defense (DoD) creates even larger and more complex simulation environments and since DIS continues to interoperate within HLA, the resource demands continue to cause performance bottlenecks that limit its effectiveness.

The monitoring tool described in this paper can provide meaningful performance information used as a guide in making decisions about the configuration and control of the available hardware and software resources used in DIS. This information can also be used to support DIS modeling requirements and design.

Results are presented from tests comparing alternative implementations of the performance monitor; the goal is to achieve a balance between the amount and types of performance data collected and the intrusiveness of the instrumentation code. Testing, analysis, and use of the monitoring system during DIS exercises has demonstrated the utility of a tool that provides high-level performance information to support persons making inferences about dynamic system and simulation behavior.

1 INTRODUCTION

Distributed Interactive Simulation or DIS is a simulation technology developed for the DoD, and successfully used to create synthetic battlefields suitable for military operational training, doctrine development, test and evaluation, and systems acquisition. DIS technology is suitable for other application domains as well, and its design is significant enough that it has been proposed as an IEEE Standard (IEEE Standards Board 1993). The DIS standard provides a high-level of interoperability among applications participating in the simulation environment which can be geographically distributed throughout LANs and WANs (DIS Modeling and Simulation Resource Repository, 1996).

The evolutionary development of DIS applications has coincided with their successful use by the DoD in Synthetic Training Environments (STE). A cost-effective technique for creating these environments is to integrate DIS applications with virtual environments and live participants. STE can be used to teach people to function within complex systems without the real-world limitations of safety, cost, training areas, or personnel. A discussion of STE and the role of DIS applications in STE is found in Cavitt, Overstreet, and Maly (1996a). An emerging standard for DoD STE is the High Level Architecture (HLA). The goal of HLA is to provide a common technical framework to facilitate the interoperability and reuse of models and simulations (DoD High Level Architecture, 1997). DIS standards contribute to its use within the HLA. Because of the large infrastructure of existing applications (and other legacy systems), DIS will continue to play a significant role in HLA and STE. As such, additional techniques are needed to provide better performing, more realistic DIS.

Achieving more realistic STE necessitates populating the virtual environment with many computer-generated entities. This workload often reduces the performance of the workstations currently used as simulation engines to an unacceptable level. To achieve the desired level of performance many workstations and simulation engines must often be integrated into the local-area and wide-area networks. Making appropriate decisions regarding the hardware and software requirements for STE requires relevant performance data and provides the rational for the design and development of more effective performance monitoring systems.

One successful DIS application is ModSAF (Modular Semi-Automated Forces), the original variant built by Loral Advanced Distributed Simulation. It is a DIS used for military training and combat doctrine development. ModSAF simulates the hierarchy of military units and their associated behaviors, combat vehicles, and weapons systems (ModSAF Software Architecture Design and Overview Document, 1995). ModSAF's design and implementation provides a useful case study to examine key issues in performance monitoring of DIS applications.

The remainder of this paper discusses the design and implementation of a software-based DIS performance monitoring system called PerfMETRICS. The primary goals in developing PerfMETRICS are to 1) evaluate metrics that can be used during the DIS exercise life-cycle including monitoring simulation run-time performance, planning and configuring training exercises, and supporting the design and implementation of DIS applications, 2) provide a flexible monitoring system to evaluate and validate alternative techniques to measure performance of DIS or other distributed simulation architectures, and 3) provide a mechanism to understand costs associated with DIS performance monitoring and make assessments about acceptable levels of intrusiveness relative to the benefits of providing high-level performance information to persons making decisions about DIS exercise management.

This paper discusses performance monitoring requirements in the context of the PerfMETRICS architecture, provides an overview of the components that make up the system, and presents results from a preliminary analysis of the costs associated with performance monitoring ModSAF, the DIS application used as a case-study for this research. It also presents a summary of the results and conclusions drawn from using PerfMETRICS during test phases of the Synthetic Theater of War (STOW) Advanced Concept Technology Demonstration (ACTD) sponsored by the Defense Advanced Research Projects Agency (DARPA). The culmination of this effort will be a week long DIS exercise, STOW '97, to include hundreds of workstations simulating up to 50,000 entities, virtual manned-simulators, and live participants. The exercise will take place at facilities geographically distributed throughout the U.S. and the United Kingdom.

Through experience, the DIS community is realizing the necessity of meaningful and relevant simulation and system performance information to help manage large-scale DIS exercises. Other design and development efforts include tools to provide feedback of simulation "game board" information and limited amounts of performance data suitable for post-exercise analysis (Schow et al. 1997, Meliza and Brown 1997, Meliza and Paz 1997). Our objectives are to present a more detailed and comprehensive view of distributed simulation performance. We are interested in issues relevant to the costs and benefits of various performance metrics: what data are useful, how the data can be collected, and what the costs are. This study uses ModSAF as an environment to ensure that we are addressing real issues in efficient data collection and are providing useful information to users.

2 PERFORMANCE ABSTRACTION

The notion of performance is multifaceted and means different things to different people. For persons managing DIS exercises, one significant need is to understand how large a DIS exercise can be with any given resource set. Alternatively, they might need to assess the impact of changes made in the configuration or control of a DIS exercise.

In the context of simulation run-time characteristics, the domain of performance improvements includes faster response times when executing simulation events and operations, executing events and operations at a greater rate (i.e, better throughput), greater utilization of simulation resources and services, correct and reliable simulation behavior, and greater availability of the resources and services that are required for DIS. Assessing performance from a hardware and systems viewpoint involves information related to system resources and services such as swap space requirements, paging activity, and disk and network contention.

The nature of the current uses of DIS always pushes performance limits of CPUs and networks. Many tools exist to assist software designers and developers in development of efficient systems. We contend that few tools are available to guide those responsible for planning and managing exercises, particularly related to exploiting the capabilities of the hardware and modeling resources available for a given exercise. If a tool is to assist in these decisions, it must present information in terms of choices available to the decision maker. This type of information often requires aggregation or filtering of lower level detailed data. We refer to this type of data composition as high-level performance information.

The motivation for the development of PerfMETRICS is to provide high-level performance information for people making decisions about simulation exercises including exercise goals, system configuration planning and resource requirements, run-time feedback and performance assessment, and afteraction-reviews and assessments about the success or failure of the exercise. Cavitt, Overstreet, and Maly (1996b) propose a framework for identifying and deriving meaningful and relevant performance information for the high-level description of distributed simulation performance. Justification for the acquisition of high-level performance information for the purpose of exercise management can be found in IEEE Standards Board (1995) and in Butler (1996). Brig (1996) provides a useful example of correlating communications costs with aggregated DIS entity information. Additional rationales can be found in Swauger (1996), Stender (1996), and Sudnikovich et al. (1996).

3 MONITOR ARCHITECTURE

The performance monitor presented in this paper collects and presents DIS performance information. Obtaining the raw data to derive that information requires a monitoring scheme to be developed for use in dynamic and static analysis. A purely software driven monitoring method is implemented to provide flexibility when defining what performance data are to be detected, collected, and analyzed. The bases for the design and implementation of PerfMETRICS was taken from experiences gained during the development of other software-based distributed monitoring systems (Chodrow 1991, Joyce et al. 1987, and Dodd and Ravishankar 1992). A complete discussion on monitoring distributed real-time systems is found in Tsai and Yang (1995). Figure 1 shows a system model for the PerfMETRICS monitoring and presentation system. The monitoring system has three principal aspects that affect its design and development: performance event identification and detection, data collection, and data reduction and analysis. The model depicts multiple workstations collecting performance data from a multi-site DIS application. A separate process (perfcollectd) performs data collection and

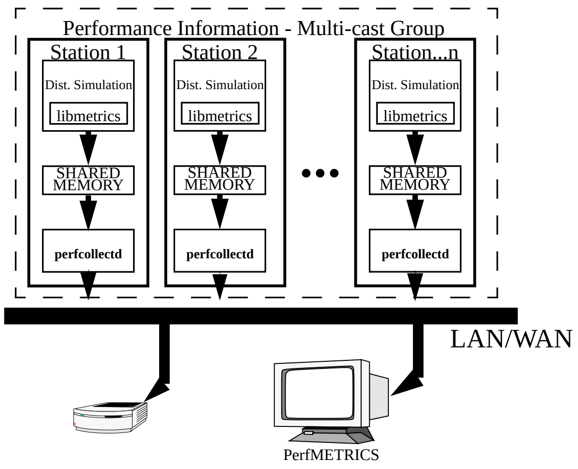


Figure 1. - PerfMETRICS System Model

reduction on each workstation and communicates with the simulation on that workstation through a shared memory interface. The performance data is collated and stored for either static analysis or for run-time presentation by a designated instance of the collection process, located on a machine not running a DIS. The run-time display of the performance information is done by a separate analysis and display process called PerfMETRICS.

The remainder of this section discuss the fundamental concept and requirements for the individual components of the monitoring system: the DIS instrumentation, the *collection daemon*, and PerfMETRICS. Existing metrics used to assess DIS performance are based on what are considered the limiting factors in DIS applications, the number of entities which a workstation can simulate, and communication overheads associated with replicating entity state information throughout the DIS environment (Vrablik and Richardson 1994, ModSAF 1.4 Reverse Engineering Report 1995, and Smith et al. 1996). The low-level and static analysis reported in the literature, however, is not sufficient to fully articulate the factors limiting performance during actual DIS exercises. Additional factors relevant to understanding performance include processing requirements for entity kinematics, behavioral modeling, graphics and visualization, and the overhead associated with simulation infrastructure. Characterizing the performance of these factors can help clarify the causal relationships that exist between performance problems and what specifically caused them.

The initial implementation of PerfMETRICS is used for monitoring performance of different versions of ModSAF participating in DARPA's STOW ACTD effort described in Section 1. Performance metrics are categorized into three classes: entity-based, simulation-based, and system-based. Table 1 lists some categories of performance. Note that some aspects of performance are specific to a particular DIS. This is evident from the metrics in Table 1 which list entity performance data relevant to ModSAF entities (such as weapons modeling for military vehicles). This fact however does not preclude the overall system architecture of PerfMETRICS from being independent of a specific DIS domain.

Run-time measurements collect the raw performance data required to derive high-level performance metrics and information specified by some performance abstraction as described in Section 2. Identification of performance events is assisted by using a framework to describe distributed simulation performance such as that proposed in Cavitt, Overstreet, and Maly (1996b). The performance metrics either directly correlate with specific logical and physical components of a distributed

Table 1: Metrics Categories (Performance Data for each DIS)

Entity	Simulation	System
Kinematic Models (Movement)	Schedular Slack Time	Network Traffic
Communications - Outgoing & Incoming PDUs	Process Times	CPU Utilizations
Sensor Models (e.g., Radar, Visual)	Update Performance	IPC Primitives
Weapons Models (e.g., Guns, Missiles)	Schedular Wait Time	Memory Utilizations
Behavioral Models (e.g., Activities, Tasks)		
Graphics (Visual Display Processing)		

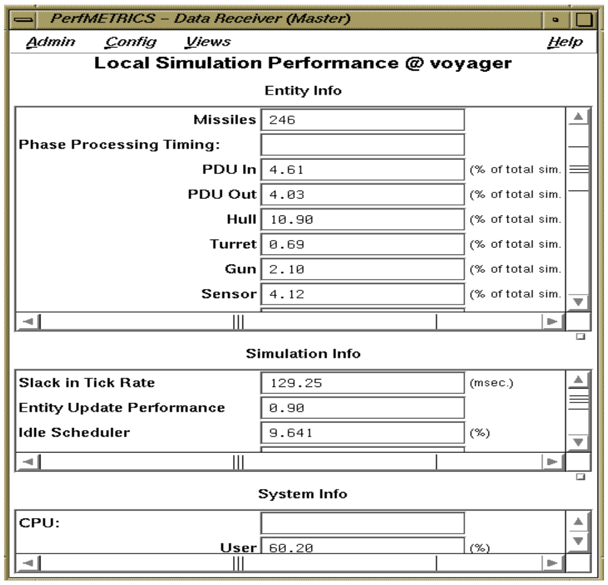
simulation (e.g. how much load can be attributed to simulation of a particular F-14 aircraft) or are derived.

Performance data is detected and collected at run-time by the execution of hand-instrumented code at the appropriate simulation or system interface. Identifying the proper interfaces is application specific and requires a programmer's understanding of the simulation's software architecture. Efficient techniques must be used to minimize the time spent executing instrumentation code since it directly effects the performance of the simulation and can skew the results of any performance analysis. The current version of PerfMETRICS only requires instrumentation in two libraries (of over 300 in ModSAF). This makes the manual instrumentation of the code much easier to maintain and understand. A small library of monitoring support routines is linked into the simulation executable. These routines are called from within the instrumentation points.

In the PerfMETRICS architecture, the DIS communicates with the collection daemon through a shared memory IPC mechanism. Access to this critical region is controlled with semaphores. After proper initialization of the IPC mechanisms, the simulation continues executing until either a run-time performance event occurs or periodic processing associated with performance data collection occurs. The data collection, reduction, and analysis components of the monitoring system are a separate process from the DIS (event detection component) for the following reasons. Doing so reduces the level of direct intrusion on the execution of simulation code, a significant factor since the monitoring system is software-based. Decoupling these components from the simulation also provides for a more flexible and extendable architecture and reduces the level of effort to integrate and maintain monitoring code in the highly dynamic software development environment necessitated by the ACTD effort.

The collection daemon is responsible for transmitting performance data packets over the network for the run-time feedback component of the PerfMETRICS system as well as managing control information for the monitoring process (e.g., collection interval, multi-cast group). Instances of all performance data packets are included in a single structure that also contains the control information for the monitoring system. At a specified interval, the collection daemon reads the performance data into a buffer and multi-casts a copy of the buffer over the network to members of the specified performance reporting group. A uniquely designated collection daemon receives the packets from all other collection daemons interfacing with a simulation and if requested to do so, stores the data to disk. The same daemon provides an interface to the run-time feedback component of PerfMETRICS. The daemon also performs any required data reduction to reduce storage requirements.

The GUI-based component (called PerfMETRICS) handles the run-time analysis and display of DIS performance data. It is the only completely non-intrusive component of the PerfMETRICS system. As mentioned, this component interfaces with the collection daemon that receives packets of performance data from all participating DIS and provides both local and global views of DIS performance. Figure 2 shows an illustration of the local view of the PerfMETRICS GUI. The local view of DIS performance shows relevant performance



Local Simulation Performance @ voyager

Entity Info

Missiles	246	
Phase Processing Timing:		
PDU In	4.61	(% of total sim.
PDU Out	4.03	(% of total sim.
Hull	10.90	(% of total sim.
Turret	0.69	(% of total sim.
Gun	2.10	(% of total sim.
Sensor	4.12	(% of total sim.

Simulation Info

Slack in Tick Rate	129.25	(msec.)
Entity Update Performance	0.90	
Idle Scheduler	9.641	(%)

System Info

CPU:		
User	60.20	(%)

Figure 2. - PerfMETRICS Graphical User Interface

information for each workstation participating in the simulation exercise. The figure illustrates the three different types of performance information as previously described: entity, simulation, and system. A global view presents performance information that is relevant to an entire simulation exercise, much like a global snapshot. The global metrics describe the aggregate performance of all workstations reporting performance data during the DIS exercise. An example of a global metric is a concurrency index measuring the variance in workload on each workstation. The index always lies between 0 and 1. If the machines are being utilized at nearly equal levels, the index will be close to 1.

4 INSTRUMENTATION COSTS

If performance monitoring is required for other than the development and testing phases of the DIS life-cycle (i.e., for experimentation or production use), a software monitoring architecture could be too intrusive on system performance. Therefore a compromise must be made on run-time measurements necessary to obtain performance data, the objective being to establish a balance between the adequacy of measured data (for the purpose of analysis) and the intrusiveness of the monitoring system (and its perturbation of the performance analysis). A relevant discussion of the issues surrounding timing errors introduced by instrumentation is found in Malony, Reed, and Wijshoff (1992).

This section presents results of a timing analysis done on ModSAF source code instrumented for performance monitoring with PerfMETRICS. Two alternative implementations are compared to a non-instrumented baseline of ModSAF. The results illustrate the effect of certain implementation decisions on the costs associated with performance monitoring. The scope of the timing analysis is restricted to assessing the direct costs associated with performance monitoring and relate those costs in terms of execution times and workload capacities. The principal component of direct costs is the execution of instrumentation code. Indirect costs such as potential reordering of simulation events were not considered.

The principal differences between the two alternative implementations are the locking granularity for the shared memory segment, and the acquisition of elapsed execution times associated with simulation events. Intuitively, the Version 1.0 implementation is more intrusive than Version 1.1 and this is quantified in the testing and results presented below. Two different test types were performed. The first test was designed to measure the increase in execution times caused by the instrumentation code. The test scenarios were characterized by a high level of interaction among simulation entities (vehicles) resulting in significant CPU and memory utilizations. The second test was designed to understand the effect of the monitoring process on the overall workload capacity of the simulation; that is how many vehicles can be simulated on the workstation with and without instrumentation code. The capacity threshold is based on ModSAF's intrinsic requirement to update each entity's state at a 2 Hz rate. An *entity state update violation* occurs if this requirement is not met. This provides certain assurances about the fidelity and consequent validity of the models.

Figure 3 shows results from the first set of tests (the timing analysis). Version 1.0 exhibits at worst a 12% increase in the relative execution time of the instrumented code. By reducing the locking granularity and utilizing the simulation's real-time clock, Version 1.1 is able to reduce the instrumentation cost to a 4.5% increase in total execution time for the routine. Our experience in using PerfMETRICS during DIS exercises, indicates that a 4.5% increase seems to be a reasonable penalty to pay when considering the benefits of reporting performance information.

The second set of tests revealed the effects of increased execution times; decreased simulation capacity in terms of the number of vehicles that can be simulated. These effects reflect design and implementation trade-offs and can be explained by examining Figure 4, a response surface for the three versions of ModSAF used in the capacity tests. This figure compares the percentage of entity state update violations for a given number of simulated vehicles. This relationship is plotted for each test version reflecting progressively increasing levels of instrumentation costs; that is the non-instrumented baseline of ModSAF, Version 1.1, and the most intrusive implementation, Version 1.0. Since the test terminates once the previously described capacity threshold is exceeded (10%), the simulation capacity in terms of the number of vehicles is at the apex of the curve shown for each test version.

The significant interpretation of this surface reveals the non-linear relationship that exists between the increasing costs of instrumentation and simulation capacity. The non-instrumented version of ModSAF simulated 31 vehicles before reaching the critical capacity threshold. The initial version of PerfMETRICS, Version 1.0, reached the capacity threshold with 21 vehicles, providing only 68% of the capacity of the non-instrumented baseline. Version 1.1 simulated 28 vehicles before reaching the threshold, providing 90% of the capacity of the non-instrumented baseline. Version 1.1 provides identical performance information in terms of utility by decision-makers when compared to Version 1.0. However, more efficient data collection decreases the intrusiveness of performance monitoring and increases

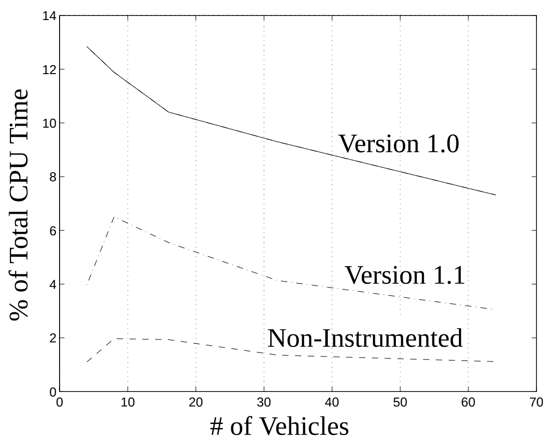


Figure 3. - *Tick Routine* Cost

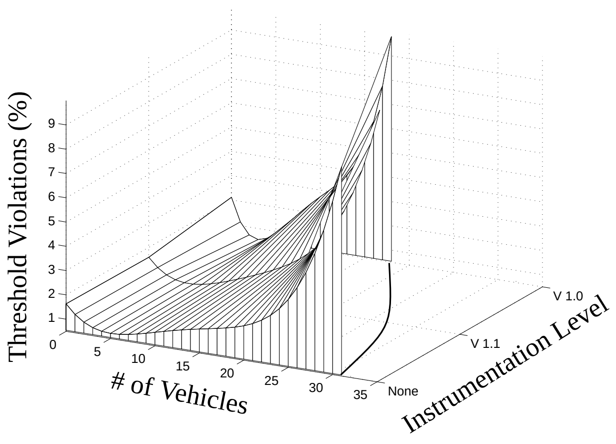


Figure 4. - *Instrumentation* Cost

Version 1.1's ability to simulate more vehicles. This representation of instrumentation costs can be used as a guide when making decisions about the level of instrumentation that can be tolerated relative to the performance goals of a simulation exercise.

From these results it is clear that alternative implementations can significantly reduce the instrumentation costs associated with performance monitoring and provide an equivalent amount of performance information. On-going development and analysis of PerfMETRICS will continue to focus on reducing monitoring costs; the objective is to use more efficient data collection methods that reduce overheads and achieve a balance in volume of performance data and monitoring intrusiveness.

5 RESULTS

The PerfMETRICS monitoring system has been tested and successfully used during functional tests. The response from the simulation community has been favorable. The performance information provided to exercise managers, planners, analyst, and other decision-makers, now allows them to make timely inferences about system behavior as it relates to specific simulation events. The minor performance impact of the current implementation (Version 1.1) relative to its usefulness brings to light the fact that it is possible for DIS performance to degrade to some level, yet the effectiveness of the overall training environment is not compromised because the trainees are unable to perceive the effects of decreased performance. This is just one of many factors that must be considered when making qualitative assessments about performance information requirements. More discussion about Human Factors (HF) issues associated with STE can be found in Glazewski and Rolek (1997).

Even in its very initial form, the use of PerfMETRICS elucidated many facts surrounding the design and implementation of this DIS performance monitor. First, the performance data associated with each instance of a DIS are collected on the machine executing the simulation. This allows the DIS exercise to scale without effecting the ability to monitor and collect performance data. Secondly, an important aspect of performance analysis is the ability to correlate the performance information with specific time periods or simulation activity. Specific performance behaviors can be correlated with simulation activity by using the timestamps associated with the DIS trace data and the performance data saved by PerfMETRICS.

Additionally, the timing analysis discussed in Section 4 and the use of PerfMETRICS during DIS exercises allows us to make several conclusions about the feasibility of DIS performance monitoring. The ability to monitor run-time performance of DIS applications is useful and required for some domains, so there must exist some qualitative assessment on what is an acceptable level of intrusiveness of the monitor. Decision-makers must make this assessment based on the utility of the performance information relative to the monitoring costs. And finally, specific design and implementation decisions can greatly effect the level of intrusiveness so alternative schemes and implementations should be considered when designing an effective DIS performance monitor.

6 FUTURE WORK & RESEARCH ISSUES

The design and implementation of PerfMETRICS is only one approach to monitoring DIS. It does provide a baseline for measuring the effects of alternative DIS performance monitoring techniques and strategies. Future modifications and enhancements to PerfMETRICS will continue to assess the costs of certain design and implementation decisions in terms of their impact (intrusiveness) on DIS. Research issues include:

- the interoperability of performance monitors with different STE applications.
- performance modeling in an HLA environment.
- visualization and other analysis techniques.

Another significant area for DIS research surrounds issues of building and predicting the performance of more scalable simulation systems. Developing effective performance monitoring systems to obtain realistic DIS workload characterizations is a prerequisite for accurate predictive models (Advanced Simulation Technology Thrust, 1996).

7 SUMMARY

This paper has reported on the design, implementation, and successful use of a DIS performance monitoring tool. An analysis of the PerfMETRICS system has showed that the cost associated with monitoring DIS applications can be negligible while providing persons with adequate information for making decisions about DIS performance. Results of using PerfMETRICS during STOW ACTD testing has provided initial assessments of DIS performance monitoring requirements and techniques. Its use has also helped identify additional requirements and issues open for future work and research in DIS performance monitoring.

ACKNOWLEDGMENTS

Support for this research was funded by BMH Associates, Inc., Norfolk, VA, and DARPA under grant number BMH-01N61339-95-C-0030. Appreciation is also expressed to ODU Research Assistants, Brendan C. Barsness and Mary C. McCorristin for providing preliminary test data.

REFERENCES

Advanced Simulation Technology Thrust, BAA 96-39. 1996. http://www.jsims.mil/adv_tech/adv_tech.html

Brig, M.P. 1996. DIS and SIMNET Network Log File Analysis. In *Proceeding of the 14th DIS Workshop on Standards for the Interoperability of Distributed Simulations*, March 1996, 227-238. Institute for Simulation & Training, Orlando, Florida.

Butler, B. 1996. Changes To The Standards Document: Recommended Practice For Distributed Interactive Simulation - Exercise Management And Feedback. In *Proceeding of the 14th DIS Workshop on Standards for the Interoperability of Distributed Simulations,* March 1996, 523-534. Institute for Simulation & Training, Orlando, Florida.

Cavitt, D.B., C.M. Overstreet, and K.J. Maly. 1996a. Modeling and Distributed Simulation Techniques for Synthetic Training Environments. In *Modeling and Simulation of Advanced Computer Systems: Applications and Systems*, ed. Kallol Bagchi, 237-260. Gordon and Breach Publishers Inc. Philadelphia, Pennsylvania.

Cavitt, D.B., C.M. Overstreet, and K.J. Maly. 1996b. A Performance Analysis Model For Distributed Simulations. In *Proc. 1996 Winter Simulation Conference*, December 1996, 629-636. Association for Computing Machinery, New York, NY.

Chodrow, S.E., F. Jahanian, and M. Donner. 1991. Run-Time Monitoring of Real-Time Systems. In *Proc. Real-Time Systems Symp.*, 74-83. IEEE CS Press, Los Alamitos, California.

DIS Modeling and Simulation Resource Repository. 1996. http://msrr.sc.ist.ucf.edu/.

DoD High Level Architecture. 1997. http://www.dmso.mil/projects/hla/.

Dodd, P.S., and C.V. Ravishankar. 1992. Monitoring and Debugging Distributed Real-Time Programs. *Software-Practice and Experience*, 22:863-877.

Glazewski, S.R., and E.P. Rolek. 1997. An Investigation on Gathering Human Factors Data Within Advanced Distributed Simulations. In *Proc. 1997 Simulation Interoperability Workshop*, March 1997, 97S-SIW-039. DMSTTIAC Service Center, Orlando, Florida.

IEEE Standards Board. 1993. IEEE Standard for Information Technology-Protocols for Distributed Interactive Simulation Applications, Technical Report IEEE-Std-1278-1993. Institute for Electrical and Electronics Engineers, Washington, D.C.

IEEE Standards Board. 1995. IEEE Recommended Practice for Distributed Interactive Simulation - Exercise Management and Feedback (Working Draft Proposal) IEEE-Std-1278.3-1995. Institute for Electrical and Electronics Engineers, Washington, D.C.

Joyce, J., G. Lomow, K. Slind, and B. Unger. 1987. Monitoring Distributed Systems. *ACM Trans. Computer Systems*, 5:121-150.

Malony, A.D., D.A. Reed, and H.A.G. Wijshoff. 1992. Performance Measurement Intrusion and Perturbation Analysis. IEEE Trans. Parallel and Distributed Systems, 3:433-450.

Meliza, L.L., and B. Paz. 1997. Integrating Exercise Control And Feedback Systems In DIS. In *Proc. 1997 Simulation Interoperability Workshop*, March 1997, 97S-SIW-032. DMSTTIAC Service Center, Orlando, Florida.

Meliza, L.L., and B. Brown. 1997. Increasing Speed And Flexibility Of Feedback Systems For DIS Exercises. In *Proc. 1997 Simulation Interoperability Workshop*, March 1997, 97S-SIW-033. DMSTTIAC Service Center, Orlando, Florida.

ModSAF Software Architecture Design and Overview Document (SADOD) 1995. LADS Document Number 94070v1.0. Loral Advanced Distributed Simulation, Cambridge, Massachusetts.

ModSAF 1.4 Reverse Engineering Report. 1995. Technical Report. Applied Research Laboratories, University of Texas at Austin.

Schow, G., R. Freeman, A. Lashley, L. Pfalz, J. Swauger, and B. Lander. 1997. Distributed Exercise Management (DEM) for HLA Simulations. In *Proc. 1997 Simulation Interoperability Workshop*, March 1997, 97S-SIW-021. DMSTTIAC Service Center, Orlando, Florida.

Smith, J.E., L.C. Schuette, K.L. Russo, and D. Crepeau. 1996. Rational Characterization Of The Performance Of Distributed Synthetic Forces. In *Proceeding of the 14th DIS Workshop on Standards for the Interoperability of Distributed Simulations*, March 1996, 665-673. Institute for Simulation & Training, Orlando, Florida.

Stender, J.F. 1996. Simulation Management for Large Exercises. In *Proceeding of the 14th DIS Workshop on Standards for the Interoperability of Distributed Simulations*, March 1996, 893-896. Institute for Simulation & Training, Orlando, Florida.

Sudnikovich, W.P., A. Huynh, J. Pasirstein, R. Wood, and W. Walker. 1996. A Proposal For Simulation Performance PDUS. In *Proceeding of the 14th DIS Workshop on Standards for the Interoperability of Distributed Simulations*, March 1996, 951-959. Institute for Simulation & Training, Orlando, Florida.

Swauger, J. 1996. Desired Capabilities of DIS Exercise Support And Feedback Tools. In *Proceeding of the 14th DIS Workshop on Standards for the Interoperability of Distributed Simulations*, March 1996, 639-645. Institute for Simulation & Training, Orlando, Florida.

Tsai, J.J.P., and S.J.H. Yang. 1995. Monitoring and Debugging of Distributed Real-Time Systems. California: IEEE Computer Society Press.

Vrablik, R., and W. Richardson. 1994. Benchmarking and Optimization of ModSAF. In *Modular Semi-Automated Forces: Recent and Historical Publications*, LADS Document No. 94007 v.1.0, 141-147. Loral Advanced Distributed Simulation, Cambridge, Massachusetts.

AUTHOR BIOGRAPHIES

DAVID B. CAVITT is a doctoral student at Old Dominion University, Norfolk, Virginia. He received a BS degree in Computer Science at Old Dominion University. He has 10 years of experience in the use and development of simulations for military and engineering applications. His research interests include modeling and simulation, performance analysis, and distributed systems. Mr. Cavitt is a member of ACM, and IEEE CS.

C. MICHAEL OVERSTREET is an Associate Professor of Computer Science at Old Dominion University. He is past chair of the Special Interest Group in Simulation (SIGSIM) of the ACM. He received his B.S. from the University of Tennessee in 1966, an M.S. from Idaho State University in 1968, and an M.S. and Ph.D. from Virginia Polytechnic Institute and State University in 1975 and 1982 respectively. He has been a visiting research faculty member at the Kyushu Institute of Technology in Japan. His current research interests include model specification and analysis, distributed simulation, high performance networking, support for interactive instruction, and static code analysis in support of software maintenance tasks. He is currently a principal investigator in tasks funded by DARPA, ICASE at NASA Langley, and the National Science Foundation. Dr. Overstreet is a member of ACM, and IEEE CS.

KURT J. MALY received the Dipl. Ing. degree from the Technical University of Vienna, Austria, and the M.S. and Ph.D. Degrees from the Courant Institute of Mathematical Sciences, New York University, New York, NY. He is Kaufman Professor and Chair of Computer Science at Old Dominion University, Norfolk, VA. Before that, he was at the University of Minnesota, both as faculty member and Chair. He also is Visiting Professor at Chengdu University of Science and Technology, People's Republic of China and is Honorary Professor at Hefei University of Technology, PRC. He was a member of Board of the Microelectronic and Information Sciences Center, Minneapolis. His research interests include modeling and simulation, very high-performance networks protocols, reliability, interactive multimedia remote instruction, Internet resource access, and software maintenance. His research has been supported by DARPA, NSF, NASA, CIT, and the U.S. Navy among others. Dr. Maly is a member of IEEE CS, and ACM.