RAND

SEMINT: Seamless Model Integration

Jed Marti, Phyllis Kantar,
William Sollfrey, Keith Brendley

National Defense Research Institute
Arroyo Center

The research described in this report was sponsored by the Office of the Secretary of Defense under RAND's National Defense Research Institute, a federally funded research and development center supported by the Office of the Secretary of Defense, the Joint Staff, and the defense agencies, Contract No. MDA903-90-C-0004 and by the United States Army under Contract No. MDA903-91-C-0006.

Library of Congress Cataloging in Publication Data

SEMINT : seamless model integration / Jed Marti ... [et al].
p. cm.
"Sponsored by the Deputy Under Secretary of Defense for Acquisition and Technology and the Office of the Assistant Secretary of the Army for Research, Development, and Acquisition."
"MR-403-OSD/A."
Includes bibliographical references.
ISBN 0-8330-1567-2
1. Computer simulation. 2. SEMINT (Computer file).
3. Computer network protocols. I. Marti, Jed.
QA76.9.C65S45 1994
003'.3—dc20 94-27715
CIP

Published 1994 by RAND
1700 Main Street, P.O. Box 2138, Santa Monica, CA 90407-2138
To order RAND documents or to obtain additional information, contact Distribution Services: Telephone: (310) 451-7002; Fax: (310) 451-6915; Internet: order@rand.org.

RAND

SEMINT: Seamless Model Integration

Jed Marti, Phyllis Kantar, William Sollfrey, Keith Brendley

Prepared for the Office of the Secretary of Defense and the United States Army

National Defense Research Institute
Arroyo Center

Preface

This report describes a RAND software connection system for linking disparate simulation systems. The SEMINT program, for Seamless Model Integration, connects both time- and event-driven models. These are written in different programming languages and operate on different computers. We argue that SEMINT is a more cost-effective approach to constructing a large model from existing components than building a completely new model in a monolithic programming environment. Also it is more flexible, in that it can run model components in a stand-alone manner. Likewise, building model-specific network protocols, in the short run, is less expensive than adhering to emerging simulation protocol standards.

This report will be of general interest to users and builders of combat simulations and network protocol implementors. We assume no detailed knowledge of combat simulations, computer networks, or computer programming.

This research was conducted by the Rapid Force Projection Technologies (RFPT) project. The RFPT project is jointly sponsored by the Deputy Under Secretary of Defense for Acquisition and Technology and the Office of the Assistant Secretary of the Army for Research, Development and Acquisition. The RFPT project is a joint effort carried out in the Acquisition and Technology Policy Center of the National Defense Research Institute (NDRI) and the Force Development and Technology Program of the Arroyo Center (AC), both federally funded research and development centers within RAND. NDRI supports the Office of the Secretary of Defense, the Joint Staff and defense agencies; the AC supports the U.S. Army.

Contents

Preface iii

Figures vii

Summary ix

Acronyms xi

1 INTRODUCTION 1

2 BACKGROUND 3

3 APPROACH 4
- 3.1 The Control Program 5
  - 3.1.1 Symbol Table 5
  - 3.1.2 The Parser 5
  - 3.1.3 Object/Type Equivalence 5
- 3.2 Object Storage 6
- 3.3 Timing 7
- 3.4 Simulation Loop and Interfaces 8
- 3.5 Error Control 9
- 3.6 Deficiencies 11

4 APPLICATION OF SEMINT TO COMBAT MODELLING 12
- 4.1 JANUS-A 12
- 4.2 RTAM 15
- 4.3 CHAMP 16
- 4.4 RJARS 17

5 CONCLUSION 19

References 20

List of Figures

1 SEMINT Interface Organization 4
2 Typical SEMINT Object Type Equivalence Conversions . . . 6
3 Common Object Name Space 8
4 Standard Start-Up Handshaking 9
5 Main SEMINT Control Loop 10
6 JANUS Map Display 13
7 CHAMP Display 16
8 RJARS Display 17

Summary

This report details the design of the Seamless Model Integration program (SEMINT), a software connection system that connects engineering and simulation models without significant reprogramming. We used SEMINT to augment the JANUS combat simulation with improved target acquisition models, surface-to-air combat models, and flight planners. The approach is less expensive than combining these models into a single, monolithic entity. Since recoding is not particularly intrusive, the models can still run in their original stand-alone form.

The advantage of our approach is the reuse of existing models, simultaneous use of multiple processors, use of otherwise incompatible programming languages and systems, and ease of implementation and maintenance. Its drawbacks are limited scale-up potential and a central failure point.

We used SEMINT to complete a number of analyses that require special intervisibility computations of low-observable vehicles and high-resolution, ground-air combat involving helicopters.

This report discusses the approach, the models integrated to date, and the lessons learned during implementation.

Acronyms

AC	Arroyo Center
ALSP	Aggregate Level Simulation Protocol
CAGIS	Cartographic and Geographic Information System
DIS	Distributed Interactive Simulation
DMA	Defense Mapping Agency
DTED	Digital Terrain Elevation Data
GVT	Global Virtual Time
LANDSAT	Earth Resources Satellite
MRT/MRC	Minimum Resolvable Temperature / Minimum Resolvable Contrast
NDRI	National Defense Research Institute
NVEOL	Night Vision Electro-Optics Laboratory
RFPT	Rapid Force Projection Technologies
RJARS	RAND's Jamming Aircraft and Radar Simulation
RTAM	RAND Target Acquisition Model
SAM	Surface-to-Air Missile
SEMINT	Seamless Model Integration
TCP/IP	Transmission Control Protocol/Internet Protocol
UTM	Universal Transverse Mercator
VAX-VMS	Digital Equipment Corporation proprietary operating system

1 INTRODUCTION

Analysts develop computer models to understand the behavior and performance of machines and people. *Engineering models* anticipate system performance, while *aggregate models* simulate group performance. Typically, aggregate model design is based on inadequate assumptions of system performance. Likewise, engineering models do not predict the behaviors of groups.

Ever increasing computer power enables us to build composite models of aggregate systems at the engineering level [1]. Typically these are monolithic entities: they are written in one programming language, require a specific operating system, and exploit characteristics of particular computer hardware. Such an approach can, with difficulty, use special-purpose hardware for graphics and computation. However, generally it cannot take advantage of past expenditures: it requires significant investment to reimplement and revalidate engineering models.

The SEMINT (Seamless Model Integration) approach exploits existing code through a simple network interface. We build a composite model by connecting existing engineering and aggregate models through a central control program, allowing the disparate models to communicate through custom interfaces. Our approach has many advantages over the monolithic approach:

1. We accommodate different programming languages and styles. Lisp, C, and Fortran simulations can be intermixed freely. Object-oriented programming is not required.

2. Specialized machine architectures are connected through the standard Transmission Control Protocol/Internet Protocol (TCP/IP). Simulations can exploit both graphics workstations and compute engines from different manufacturers.

3. Simulations can be spread over different operating systems. We successfully ran simulations mixing incompatible versions of UNIX. The sole requirement is TCP/IP support.

4. The system supports large granularity multiprocessing at the model level rather than at the object or algorithm level. Algorithm design and debugging are simplified.

5. Model life span is increased — a single model can be used in multiple environments.

6. Component models operate in both stand-alone and multiprocessing modes.

7. Combinations of time- and event-driven simulations can be built; a global virtual time mechanism enforces a uniform advance rate.

SEMINT exacts a run-time penalty, although some potential parallelism can be exploited for increased performance.

2 BACKGROUND

Several existing mechanisms have varying degrees of complexity for connecting simulations. However, these either did not provide object-level synchronization[1] or the interactions that our models require.

For example, the Aggregate Level Simulation Protocol (ALSP) [2] provides many facilities similar to SEMINT's. However, we felt that ALSP's general-purpose nature required significant additions to support the special communication needs of the models interfaced by SEMINT. Likewise, we felt the need to experiment with load balancing and multiple instantiations of participating models, neither of which fits easily into the ALSP system.

The *Backtalk* system [3] supports distributed artificial intelligence and provides a centralized router for messages between communicating expert systems. However, it is not designed for maintaining the consistent object database performed by SEMINT. Likewise, the *Open Systems* approach to simulation systems [4] requires a standard interface but provides no implied support for object maintenance.

The Distributed Interactive Simulation (DIS) standard [5] is an attractive candidate for communications protocol; however, its insistence on real-time response precludes its use by simulations that run slower than real time. For such systems to participate, a large real-time program must be added, and the slowest simulation must run faster than real time.

SEMINT objects are simple abstractions of entities most frequently encountered in our simulations: air objects, ground objects, and sensors. Unlike approaches with a large declarative component [6], all objects are expected to lie within these generic classes. Likewise, system interaction is imperative rather than declarative. We decided against a declarative approach to (1) avoid run-time cost of dynamic representation, (2) escape additional implementation cost, and (3) avoid increased implementation complexity. We balanced these points against the advantages of generality and decided that the implementation cost could not be justified.

[1] Our parallelism appears when objects run on different processors as opposed to particular parallel algorithms.

3 APPROACH

SEMINT consists of a control program and several system-specific interfaces to its participating models. The control program starts the various simulations as directed by a SEMINT configuration file, directs initialization, and synchronizes virtual time advance. As depicted in Figure 1, system interfaces exist on both the SEMINT and simulation sides. The simulation side interface converts nonobject representations into objects and controls the advance of virtual time. Generally, the simulation side interfaces require very close coupling to their hosts. To date, they have been written by the simulation programmer in the C programming language, based on a template provided by a SEMINT test interface.

The SEMINT side interfaces convert simulation object representations into SEMINT objects and pass messages between programs when appropriate.

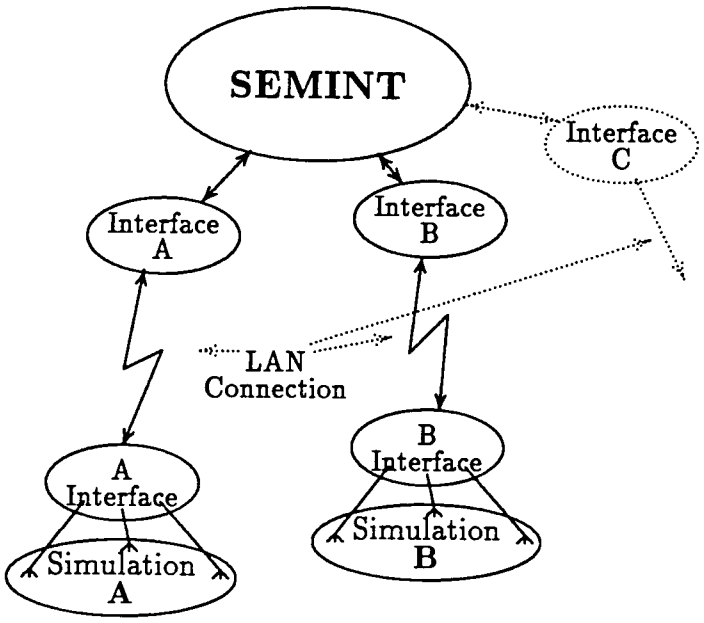

Figure 1: SEMINT Interface Organization

3.1 The Control Program

To accommodate many different multiprocessor configurations, SEMINT is directed by a configuration file. Per instructions in this file, the control program directs jobs to different processors, makes network connections and controls the initialization sequence. Additional commands provide interactive access to a system symbol table, object storage, debugging messages, and system status.

3.1.1 Symbol Table

The system symbol table contains variables to control SEMINT's operation. Symbols can be used anywhere in the configuration file and have their numeric or string values substituted where they occur. A simple infix expression parser permits arithmetic and string operations where appropriate. When SEMINT is operating under interactive control, the user can query the state of these variables.

3.1.2 The Parser

The SEMINT parser reads and accepts statements either from control files or direct interaction with the user via a terminal. A straightforward context-free, parsing scheme reads and interprets the statements. However, the system expands UNIX environment variables when encountered as well as special variables known to the system. The user has control over whether the expansion is local to SEMINT or remote. For debugging purposes, users can inspect object state. This feature has proved less useful than expected since most errors occur long before they can be diagnosed at this level.

3.1.3 Object/Type Equivalence

All simulations must agree on the objects that they are working with. What appears to be an M1A1 tank to one simulation must not accidentally be an F-16 fighter in another (although the consequences might be amusing).

SEMINT adopts a common naming convention for all objects: it is up to the subordinate simulation to provide SEMINT with a translation between its object types and SEMINT's. Our standard is to use JANUS object type names and provide equivalences determined off line. A more comprehensive solution will only be possible when a common naming convention is established.

The equivalence tables are established at simulation start-up time. All simulations must, as their first task, send the equivalence between their names and the standard names. SEMINT builds two tables: one translating the remote object into the standard name and the other doing the reverse. Figure 2 demonstrates this conversion.

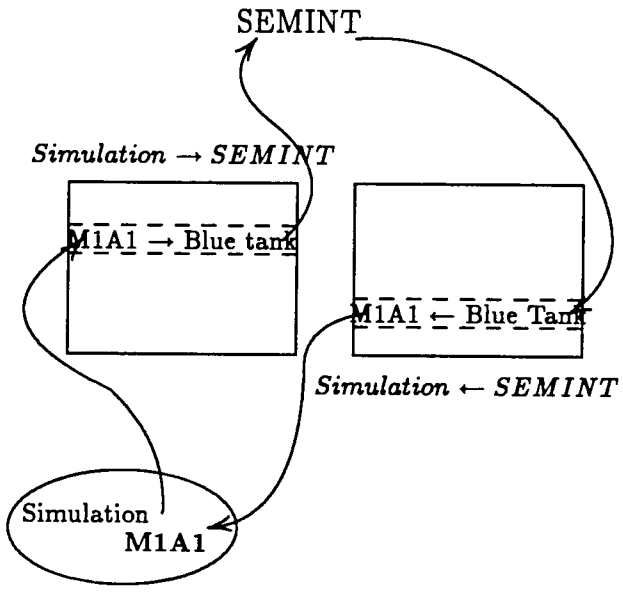

Figure 2: Typical SEMINT Object Type Equivalence Conversions

All communication across the interface is by the remote name or type. However, all SEMINT communication is through the standard name set.

3.2 Object Storage

SEMINT stores *mirror* copies of all public simulation objects and their public data. Currently, this includes three object types and their associated data:

- **Ground Objects** have a location, heading, velocity, status, and list of sensor objects.
- **Air Objects** have a location, pitch, yaw, roll, heading, velocity, rotor angle, and status.

- **Sensors** have a status, category, Minimum Resolvable Temperature / Minimum Resolvable Contrast (MRT/MRC) curve number, magnification, height above containing object, minimum range, and horizontal field of view. Most importantly, a sensor has a pointer to the SEMINT object of which it is a part.

This "common object" representation has many advantages over an approach with object types copied from host simulations. It

- Simplifies implementation
- Allows connection of nonobject-oriented simulations
- Provides a simple object standard for all connected simulations.

Each SEMINT interface establishes the relationship between the common object storage and the object names for the target simulation. More specific object information is carried in the text object name rather than structure or class information. For example, this mechanism establishes the following relationship for an M1A2 tank between two simulations, A and B:

- Object 5 in simulation A is a "Blue-M1A2-Tank."
- SEMINT's representation of A's object 5 is stored in object 3.
- Simulation B's object is number 253, side 1.

The connected simulation sees only the object representation it is familiar with, SEMINT managing the conversion between systems (depicted in Figure 3). Note that the link between the equivalence tables and the global object table is bidirectional.

3.3 Timing

We designed the synchronization scheme to minimize implementation risk and to avoid deadlock. Some of the simulations are time driven, while others are event driven. SEMINT permits both but with some loss of timing indeterminacy for some connections (discussed with each simulation). Unlike the optimistic and conservative simulation systems [7, 8], this strategy avoids much multiprocessing overhead and does not require significant reprogramming of the host simulations.

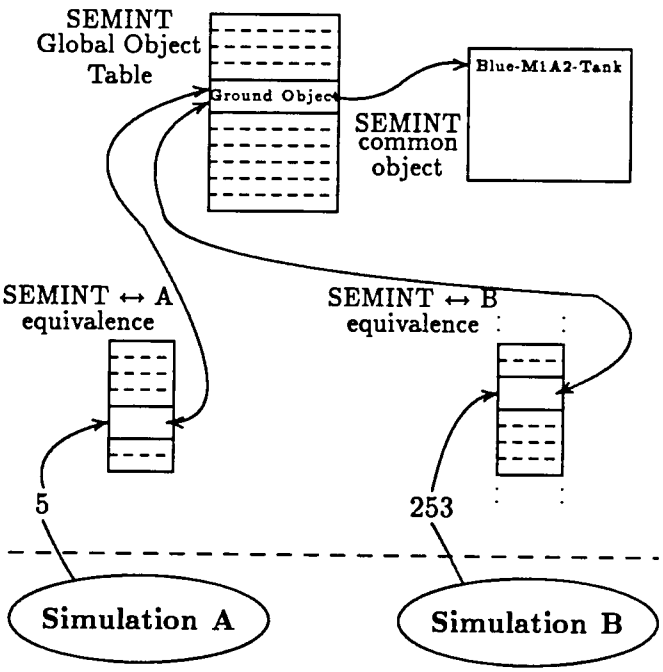

Figure 3: Common Object Name Space

The simulation setup requires four distinct phases (see Figure 4):

- Connection handshake is made.
- Simulation transmits object equivalences.
- Simulation transmits its own objects.
- Simulation receives remote objects.

3.4 Simulation Loop and Interfaces

System interfaces are constructed on both the SEMINT and simulation sides. Perhaps the greatest difference between ALSP and SEMINT is this lack of standardization of interface protocols. The disadvantage of this approach is the need for additional SEMINT programming for each new system, but its advantage is implementation simplicity. However, every system we have

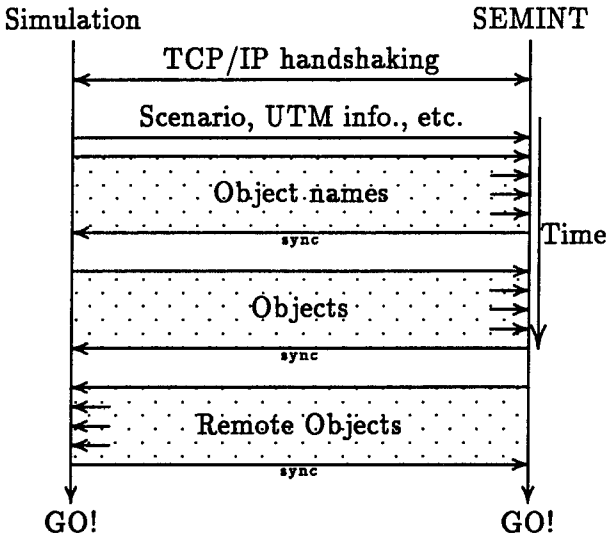

Figure 4: Standard Start-Up Handshaking

worked with has interfaced well with the C programming language — the interfaces have not been difficult to construct.

The basic SEMINT control loop provides periodic updates of position and status for all known objects, with two caveats:

- No simulation gets a status of its own objects.
- A system can block all but certain object classes to avoid useless communication.

SEMINT advances according to *Global Virtual Time* (GVT), the virtual time of the simulation farthest behind. The entire simulation advances only at the rate of its slowest connected component. An advance in global virtual time causes object status to be sent to all connected simulations as above.

The SEMINT control loop asynchronously accepts updates from each connected simulation. Each update terminates with a time stamp. A simulation is not allowed to advance beyond GVT under control of synchronization messages. The main control loop activity is depicted in Figure 5.

3.5 Error Control

Any number of errors must be caught by SEMINT and dealt with correctly. Typically, multiple runs of a suite of SEMINT-connected models are sched-

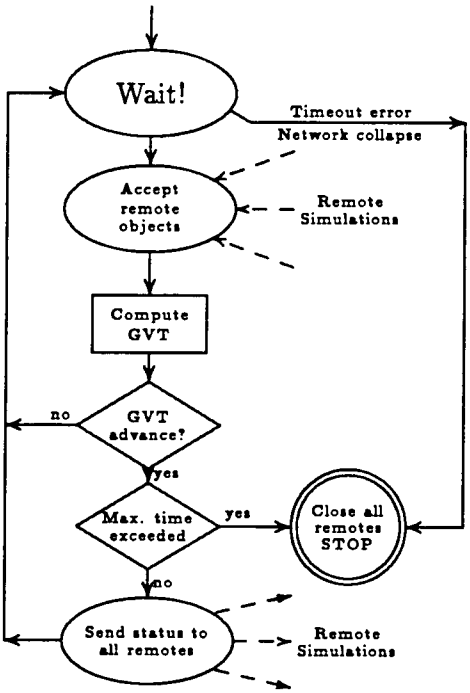

Figure 5: Main SEMINT Control Loop

uled for overnight completion. We have identified several errors that can occur during these runs:

- **Handshake Failure** — The host machines are too loaded, and the processes cannot start correctly. The user can specify how long SEMINT will wait for the first connection. After a time-out, any connections already made will be canceled and the SEMINT execution terminated with an error status.
- **Connection Failure** — This failure is usually caused by a remote process terminating prematurely. SEMINT responds by closing all network connections and terminating.
- **Connection Time-out** — Remote jobs performing very poorly or in infinite loops cause this error. SEMINT counts the number of such time-outs and after a very long one or repeated offenses, cancels all connections and terminates with an error. If no network activity has

occurred for a specified amount of time, a time-out error is signaled also.

3.6 Deficiencies

During a year's SEMINT usage, we have identified a number of implementation deficiencies (design deficiencies appear in the conclusions):

- SEMINT makes only one simulation run at a time. Permitting multiple runs in a loop requires freeing the dynamic storage allocated for objects during the run and cleanly terminating any orphaned processes. The use of multiple runs would enable us to minimize simulation initialization time (quite significant for JANUS and other simulations requiring large terrain files).
- SEMINT does not currently allow multiple instantiations of compute bound jobs. For example, SEMINT running multiple RTAM (RAND's Target Acquisition Model program) copies could batch requests among them, collect and synchronize the results for JANUS, and maximize system efficiency through distributed computation.
- Control of JANUS objects by other simulations was difficult. We should extend JANUS to create objects at run time and include a flag indicating external control.[2]
- SEMINT does not guarantee the consistency between systems of either object representations or databases. Rather than expend considerable programming energy on this activity, we decided that the simulation users must manually perform this consistency checking.

[2]Extensions to JANUS are constrained by its large user community. RAND can implement improvements but cannot allow JANUS development to diverge too far from the community model.

4 APPLICATION OF SEMINT TO COMBAT MODELLING

We connected several models through SEMINT to exploit the best components of each. The current operational configuration connects the JANUS ground combat model, the RTAM target acquisition model, the RAND Jamming Aircraft and Radar Simulation (RJARS) ground-to-air model, the BLUE MAX (fixed wing) and CHAMP (rotary) flight planners and RAND's Cartographic and Geographic Information System (CAGIS) [9]. JANUS conducts the ground battle using RTAM as an adjunct when it requires a more accurate calculation of detection probability. When air operations are incorporated into the conflict, the flight planners are used to create and fly aircraft missions. Defenses against aircraft are conducted by RJARS, which performs detection, tracker assignment, tracking, jamming, and surface-to-air missile operations. CAGIS provides geographic information to all the simulations, although, in the case of JANUS, this information is converted in a preprocessing step.

4.1 JANUS-A

JANUS-A [10] is a high-resolution, stochastic, interactive, ground combat simulation, used for Army studies, analyses, and training. JANUS-A is a moderately large (more than 100,000 lines) FORTRAN program written for VAX-VMS (a proprietary operating system that runs only on Digital Equipment Corporation machines) and ported to the UNIX environment. A typical JANUS visual display is shown in Figure 6.

JANUS defines each system with the attributes of size, speed, sensor, armament, armor protection, thermal/optical contrast, and flyer-type for helicopters and fixed-wing aircraft. System vulnerability is characterized by probability-of-hit and probability-of-kill data sets individually associated with weapon-versus-system pairs. Presently, up to 250 types of systems and 250 weapons may be defined, per side. Engagements with 1200 entities on each side are permitted. Each system may search and detect with two different sensors, both wide and narrow field of view, employing the Night Vision Electro-Optics Laboratory (NVEOL) detection model.

JANUS detection and acquisition has two aspects: the detectability of a target to an observer (a sensor system) and the observer's response to that detectability. The target's detectability is measured in *cycles* resolved by the observer on the target. Cycles depend on the contrast between the image of

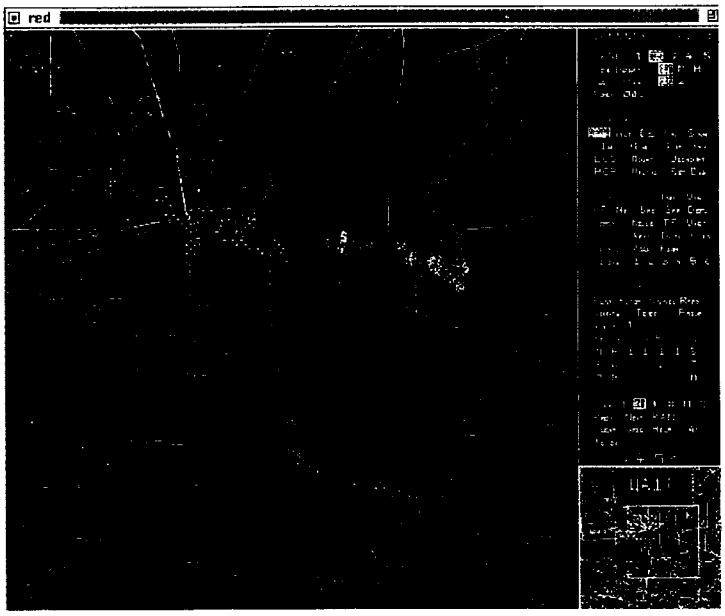

Figure 6: JANUS Map Display

a target, its range, and its background, and the target's minimum presented dimension. During each of the 20 detection periods (each detection period is 3 seconds long), a list of potential target units is formed for each observer unit. To be included in this list, the target must be within visibility range, the observer must have line of sight to the target, and the target unit must be in the observer's sector of search. Finally, the observer must resolve a sufficient number of cycles on the target – compared with a threshold cycle for the pair, determined at initialization time by a random draw. Only the 10 targets with the most cycles are put on the list. During each detection period (the value can be specified as input), an observer attempts to acquire the units in its target list. To determine if the target is acquired, a random draw is made and compared to a calculated probability of detection for that target.

RAND diverges from the traditional JANUS detection algorithm in that search is conducted using wide field-of-view sensors. If the observer unit is cued via movement or firing by the target, the observer switches to a narrow field of view of its best sensor, centered on the target. For special low-observable vehicles, JANUS employs RTAM's higher-fidelity capability to calculate both cycles and probability of detection.

JANUS is not object oriented in the classical sense; communication with SEMINT requires constructing the objects before sending messages.

First, we augmented JANUS-A's sensor model to improve the reliability of the algorithm used to detect low-observable vehicles. This task requires significant amounts of special data derived from LANDSAT infrared satellite imagery not normally part of JANUS-A. We accomplished this task by communicating with the RTAM model described in a subsequent section. Second, we augmented JANUS-A's air model with helicopter and fixed wing operations simulated by RJARS, CHAMP, and BLUE MAX. CHAMP and BLUE MAX are components of the larger CAGIS geographic information system.

We treat JANUS-A as a "gray" box, trying to minimize the number of modifications to the program code. The types of messages passed between JANUS-A and SEMINT are outlined below:

- **Location and Scenario Data** — JANUS-A operates in a modified Universal Transverse Mercator (UTM) coordinate system while most of the other models operate in a Longitude - Latitude system. This message enables SEMINT to perform the appropriate coordinate transformations.

- **Object Name Equivalence** — This message establishes the equivalence between each JANUS-A system name (a character string) and internal JANUS-A number. The character name is the standard by which other systems identify their objects.

- **Object Instantiation** — This message causes SEMINT to instantiate a new object and establish an equivalence between its SEMINT identifier and JANUS-A's identifier.

- **Object Status and Position** — Each message provides an object's status and position. All live JANUS-A objects report this information once every major clock cycle (set for a particular scenario, normally every 2 virtual seconds).

- **Time Synchronization** — This message indicates that JANUS-A has completed processing for a virtual time and that it will accept updates for objects it does not control. The 2-second position reports allow only a location inaccuracy of some 30 meters for a 60 kph vehicle to remote simulations.

- **Detection Data, Requests, and Responses** — Replacing JANUS-A's target acquisition algorithm requires sending sensor data to the acquisition model, querying it for computation, and receiving a response.

4.2 RTAM

We developed RTAM [11] to improve the simulation of target acquisition beyond that available in JANUS-A. In stand-alone operation, the system provided analysis of low-observable vehicles in various terrain settings through the use of RAND's CAGIS system that provides utilities for processing LANDSAT and Defense Mapping Agency Digital Terrain Elevation Data (DMA DTED) images.

The current JANUS-A terrain database includes only terrain elevation (on 100-meter posts), some culture data, and roads and rivers. Calculating observability requires both elevation and spectral background data. The RTAM module uses DMA DTED and LANDSAT images at five optical and infrared wavelengths to calculate visibility. The shape of the object and its reflection or emission characteristics are used to establish the signal available to the observer. The duration of observation or of scanning limit the actual signal, which is then processed through a contrast detector or thermal detector and compared to the background through a suitable algorithm to determine the probability of detection. This probability is returned to JANUS via SEMINT. RTAM also sends detectability data to JANUS to establish the target acquisition list.

To minimize the chances that RTAM would use the wrong response curves for different sensors, JANUS passes the appropriate MRT/MRC curves to RTAM through SEMINT. Since RTAM accepts positions of target vehicles and their scanning sensors from JANUS via SEMINT, the *part-of* relationship must be implemented inside SEMINT.

RTAM can also operate in a *requirements* mode, in which the probability of detection is expressed in terms of range, or in a *3D* mode, in which the probability is calculated from object characteristics and background. The choice is specified by the SEMINT control file.

RTAM is not really a simulation: it maintains no event list or virtual time. It responds to requests from any other SEMINT task to compute the probability of detection between a target and sensor. Currently, the requesting task blocks until an RTAM response is received.

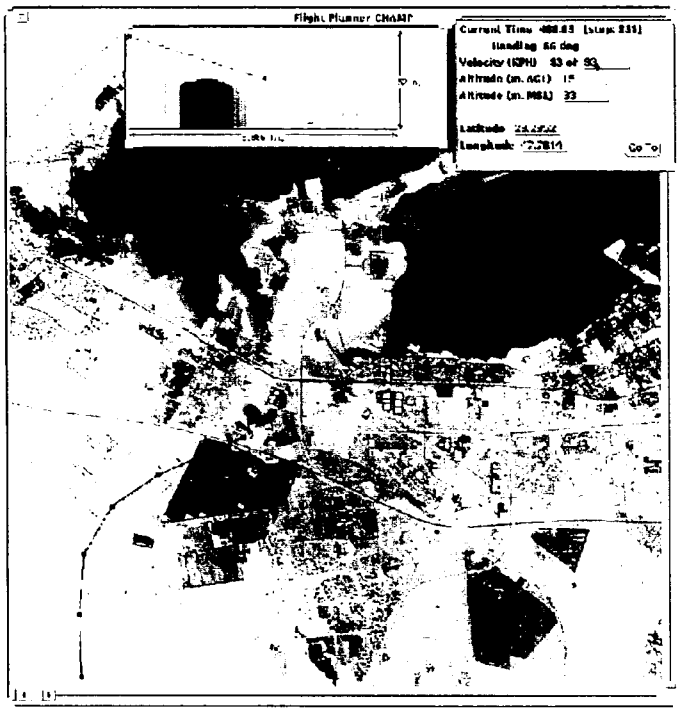

Figure 7: CHAMP Display

4.3 CHAMP

CHAMP is a RAND-developed flight path simulator for helicopters with realistic flight dynamics. At RAND, CHAMP is usually operated by experienced helicopter pilots, but the model can be used by others with appropriate training. Figure 7 shows a typical CHAMP flight plan being generated over Kuwait City (LANDSAT visual display).

In operation, a mission is selected for the helicopter, which may involve hovering, nap-of-the-earth flight, or high-speed travel. The pilot uses a terrain map generated by CAGIS. A set of waypoints is selected, with arbitrary spacing, that covers the path required to perform the mission. The pilot selects the initial pair of points, clicks on them, and the model flies the connecting path. Here it takes into account the maneuvering capabilities of the helicopter and plans the intermediate path, so that it may pass either above or around an obstacle. The next waypoint is used to continue the path and so forth. The pilot observes the terrain features and may be given some information about potential defenses.

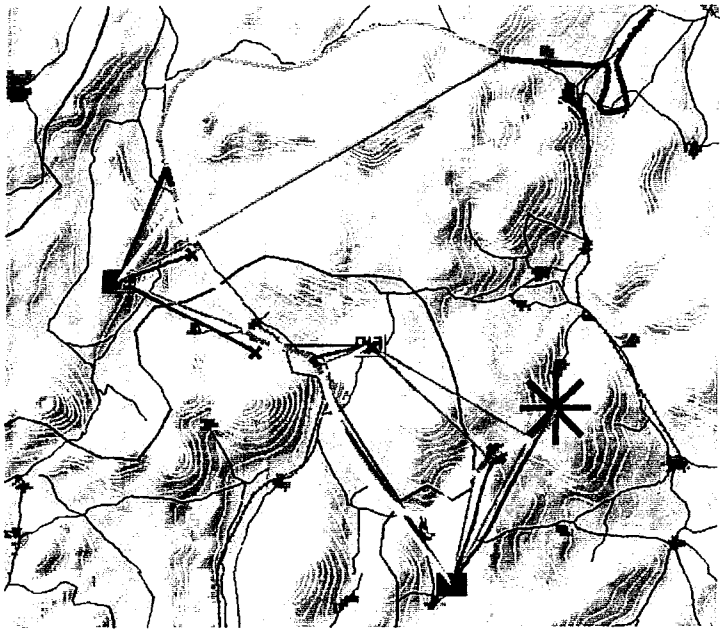

Figure 8: RJARS Display

CHAMP is most often operated as a preprocessor. The path is flown, and then SEMINT is used to read it concurrently with the operation of JANUS and RJARS. We intend to develop interactive procedures in which information from RJARS about threats to the helicopter may be fed back to CHAMP, which could then alter the path. This method would permit direct contact among all the models.

4.4 RJARS

RAND's Jamming Aircraft and Radar Simulation (RJARS) [12] has been interfaced with the SEMINT environment. RJARS, a ground-to-air combat model, was originally developed at Johns Hopkins but was improved significantly at RAND to investigate tasks such as reconnaissance and close air support. This time-stepped simulation of about 25,000 lines of C code dynamically allocates memory space to match the size of the scenario being studied. For example, Figure 8 shows the result of flying an aircraft against three defenders, displaying the flight path, radar sight lines, surface-to-air missile (SAM) flight paths, misses, and the kill.

RJARS establishes a terrain field, then places on it a defensive system including search radars, height finders, acquisition radars, tracking radars, infrared and optical systems, antiaircraft artillery, and a command and control system. Into these defenses come aircraft flying prescribed paths, carrying radar warning receivers, jammers, antiradiation missiles, and air-launched cruise missiles. The defense and offense then conduct the battle simulation using intermediate level algorithms (for example, radars calculate the received signal level using waveform and scan characteristics but do not perform pulse-to-pulse processing). All elements of RJARS (aircraft, radars, etc.) are treated as individuals rather than aggregates.

When operating with SEMINT, RJARS reads elevation and multispectral imagery warped and registered by RAND's CAGIS system. The ground threat is established by preprocessed data supplied by the JANUS analyst who developed the scenario. The aircraft are flown by the flight planners (BLUE MAX for fixed-wing aircraft and CHAMP for helicopters) along paths chosen by the scenario developer to perform their desired attack missions. SEMINT sends data from JANUS to RJARS that add additional defenses (such as mobile missile launchers).

The JANUS and RJARS simulations run in parallel, with JANUS updating and perhaps killing the ground equipments and RJARS updating and perhaps killing the airborne elements. JANUS may employ the aircraft to attack enemy ground units. This division of labor has permitted more realistic simulations and the possibility of treating scenarios that could not be investigated otherwise. Location indeterminacy during unsynchronized simulation periods is minimized by having RJARS perform its acquisition and kill computations at 1/2 second intervals with dead reckoning between the 2-second updates.

The timing control to match the event-driven JANUS and time-stepped RJARS is provided by SEMINT. RTAM has been integrated as well, so all the models may run at once. Since JANUS and RJARS run in parallel during time intervals, the response time may be expected to be determined by the slower of the two, plus overhead time from the network interface. Which model is slower depends on the particular scenario and the machine environment.

5 CONCLUSION

The SEMINT approach mediates the connection of simulations written in various styles and programming languages. Its key contributions are

1. Rapid prototyping of simulation components in a large-scale context
2. A uniform, transportable set of objects
3. Distribution of processes among hardware and software platforms
4. Minimizing the impact of large processes on relatively small workstations
5. A convenient error control and disaster recovery point
6. Its simplicity, reliability, and inexpensive maintenance.

The SEMINT approach is not amenable to general-purpose use because of the following drawbacks:

1. Its general-purpose objects encapsulate only those of combat simulations
2. Its star network arrangement is a single failure point and potential network bottleneck
3. Its performance is limited by blocking events requiring an extra network hop through SEMINT.

Despite these drawbacks, we believe that the additional cost of generality does not balance the added utility. It would be cheaper to implement a new SEMINT for different purposes than to extend the current version.

References

[1] K. W. Brendley and J. Marti, "A distributed network approach to variable resolution modeling," in *Proceedings of Conference on Variable-Resolution Modeling*, pp. 248–255, May 1992.

[2] R. Weatherly, D. Seidel, and J. Weissman, "Aggregate level simulation protocol," in *Proceedings of the Summer Computer Simulation Conference*, pp. 953–958, Society for Computer Simulation, July 1991.

[3] K. Sayre and M. A. Gray, "Backtalk: A generalized dynamic communication system for DAI," *Software–Practice and Experience*, vol. 23, September 1993.

[4] H. A. Barker, M. Chen, P. W. Grant, C. P. Jobling, and P. Townsend, "Modern environments for dynamic system modeling," *International Journal of Modelling & Simulation*, vol. 13, no. 2, pp. 67–71, 1992.

[5] PM TRADE, *Military Standard: Protocol Data Units for Entity Information and Entity Interaction in a Distributed Interactive Simulation.* No. IST-PD-91-1, Orlando, Florida: Institute for Simulation and Training, October 1991.

[6] D. D. Cowan, T. M. Stepien, R. Ierusalimschy, and C. J. P. Lucena, "Application integration: Constructing composite applications from interactive components," *Software-Practice and Experience*, vol. 23, pp. 255–275, March 1993.

[7] D. Jefferson, "Virtual Time," *ACM Transactions on Programming Languages and Systems*, vol. 7, July 1985.

[8] K. M. Chandy and J. Misra, "Asynchronous distributed simulation via a sequence of parallel computations," *IEEE Transactions on Software Engineering*, vol. SE-8, pp. 401–412, July 1982.

[9] A. L. Zobrist, L. J. Marcelino, and G. Daniels, *CAGIS: A Guide to System Use*, N-3172-RC, RAND, Santa Monica, California, 1991.

[10] TITAN, *JANUS(A) 2.0 model,* DABT 60-90-D-0002, US Army, Leavenworth, Kansas.

[11] K. Brendley and J. Grossman, "War-game simulations challenged by advanced sensors," *Photonics Spectra*, June 1992.

[12] W. Sollfrey, *RJARS: RAND's Version of the Jamming Aircraft and Radar Simulation,* N-2727-1/1-AF/PA&E/OSD, RAND, Santa Monica, California, 1991.

Produced Monday, January 2, 1995 at 9:20 AM

088: RAND/MR-403-OSD/A
245: 00 SEMINT :|bseamless model integration /|cJed Marti ... [et al.].
260: Santa Monica, CA :|bRAND,|c1994.
300: xi, 21 p. :|bill. ;|c28 cm.
506: 1 UNCLASSIFIED
520: This report describes a RAND software connection system for linking disparate simulation systems. The SEMINT program, for Seamless Model Integration, connects both time- and event-driven models. These are written in different programming languages and operate on different computers. The authors argue that SEMINT is a more cost-effective approach to constructing a large model from existing components than building a completely new model in a monolithic programming environment. Also it is more flexible, in that it can run model components in a stand-alone manner. Likewise, building model-specific network protocols, in the short run, is less expensive than adhering to emerging simulation protocol standards.
695: 1 Computerized simulation.
695: 1 Mathematical models.
695: 1 Computer networks.
695: 1 Computer communications.
700: 10 Marti, Jed.|w
700: 10 Kantar, Phyllis.
700: 10 Sollfrey, W.|w
700: 10 Brendley, Keith W.|w
982: 3