An Agent-Based C4ISR Testbed

Joseph A. Giampapa, Katia P. Sycara, Sean R. Owens, Robin T. E. Glinton, Young-Woo Seo, Bin Yu
The Robotics Institute, Carnegie Mellon University
Pittsburgh, PA 15213-3890 U.S.A.
E-Mail: {garof,katia,owens}+@cs.cmu.edu
E-Mail: {rglinton,ywseo,byu}+@cs.cmu.edu
URL: http://www.cs.cmu.edu/~softagents

Charles E. Grindle, Yang Xu, C. Michael Lewis
School of Information Science
University of Pittsburgh
Pittsburgh, PA 15260 U.S.A.
E-Mail: {cgrindle,yxu,ml}+@sis.pitt.edu
URL: http://usl.sis.pitt.edu/ulab/Team.htm

***Abstract*— This paper describes six functional extensions to the OneSAF Testbed Baseline (OTB) modeling and simulation environment: (1) the addition of modules to access OTB's terrain database, (2) a supervised, on-line "batch-mode" interface to OTB for running experiments, (3) the addition of virtual reality (VR) agent proxies for integration with a low-cost, off-the-shelf VR engine, (4) a two-way OTB-agent bridge for the real-time query and control of OTB entities, (5) an unsupervised, off-line "batch-mode" interface for automatically loading, generating, reconfiguring, and executing experiments, and (6) the addition of new sensor model entities. These extensions enable the integration with OTB of C2 and multi-level information fusion algorithms that would not normally integrate with its event-based modeling and simulation engine. This paper describes the system architecture of the testbed and some of the research applications that demonstrate its use.**

***Index Terms*— C4ISR testbed, OneSAF, agent, RETSINA**

I. Introduction

In order to research and design the automation of real world intelligence gathering, analysis and fusion systems, it is necessary to have a test system that models uncertainty of information, behavior, and environment. It is very difficult and expensive, however, in terms of time, cost and labor, to acquire such uncertainty models, let alone to develop a model and simulation system for them, and there is always the risk that the models that researchers create are biased towards their own algorithms and approaches. To address the need for such models, we have adopted the use of the *OneSAF Testbed Baseline (OTB)* v1.0 [1] as a modeling and simulation environment. Many other military simulators exist, such as the Objective OneSAF System (OOS) and other SAFs, which emphasize different entity behaviors in diverse environments, yet, based on what we know of the design of such systems, we believe that many of the same extension techniques described in this paper can be applied to those systems, as well.

OTB models common military vehicles, aircraft, sensors and munitions, and simulates: (a) unpredictabilities of action, (b) conditions that serve as force multipliers, such as improved hit and survivability rates if tanks fire in an echelon form or from behind tree lines, and (c) information uncertainty from the sensors that it models. It was written to be extensible in three ways: (1) by compiling new entities, entity behaviors, and functionalities into its code base of nearly one million lines of C code and over 500 software libraries [2], (2) by adding other simulators that can communicate with it via multicast-based *Distributed Interactive Simulation (DIS) Protocol Data Units (PDUs)* [3], and (3) through interoperability with HLA (high-level architecture)–compliant systems. This has some drawbacks, however, particularly in the use of OTB as a testbed for new algorithms for *Command, Control, Communications, Computers, Intelligence, Surveillance, Reconnaissance (C4ISR)* applications, such as: the automated performance of multi-level Information Fusion (described in Section II), the automated development and analysis of Courses of Action (COAs), the automation of the Intelligence Preparation of the Battlefield (IPB) [4] process, and the automated development of the Modified, Combined Obstacle Overlay (MCOO) [5] artifact. Namely, the integration of external software entities, either directly or through HLA, requires that they be modified to be invoked through OTB's data- and event-driven software architecture, and many C4ISR algorithms do not lend themselves easily to such conversions. Communication by DIS PDUs does not effect interoperability with such algorithms, either, since DIS PDUs are bit-encoded words that represent a hierarchy of OTB system control, communication, and entity state information. A Command and Control (C2) algorithm for the automatic role assessment and assignment of two autonomous entities, for example, requires the exchange of messages following a different protocol or knowledge representation scheme that cannot map to PDUs. There are additional problems derived from the fact that DIS packets are transmitted via multicast, which is a stateless protocol that is prone to high rates of packet loss and suppression by network routers. Not only would distributed C4ISR algorithms need to be modified to handle such transmission unreliability, but they would also need to communicate significantly more state information in order to be effective. Such requirements would actually be counter to the proposed environments in which such algorithms would be used.

The algorithms that we use for gathering, analyzing, and fusing the information derived from OTB are written and maintained in a non-OTB, native format. That is, the designs of algorithms, data structures, and communication protocols are made with consideration of the problems that they address, not the implementation of the specific testing environment

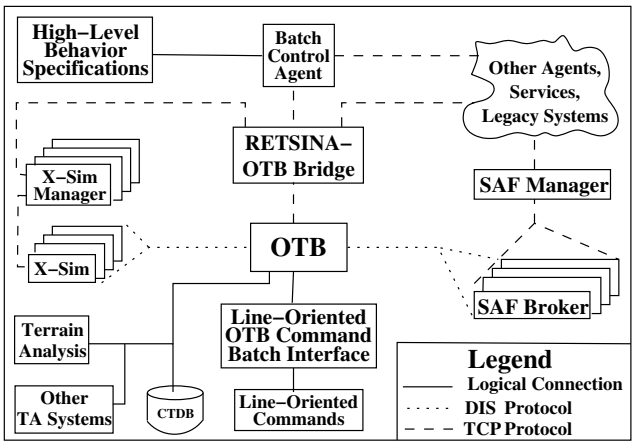


Fig. 1. Agent-Based C4ISR Architecture

in which they are evaluated. This has been accomplished by building an architecture that extends functional and logical components of the OTB system in six ways.

Fig. 1 illustrates our extensions to OTB, implemented in the RETSINA [6] multi-agent system, which is briefly described in Section II, below. RETSINA was chosen for its lightweight libraries that can be deployed on multiple platforms and quickly adapted to other legacy systems. The first extension (lower left corner) is the use of the OTB CTDB (compact terrain database) for purposes other than OTB's internal modeling and simulation. Information flow, indicated by the solid black line, "Logical Connection", is unidirectional from the CTDB to the components that use CTDB terrain data. The entities of Fig. 1 that receive CTDB data do not necessarily communicate directly with each other. The module labeled *Terrain Analysis* identifies work that is described in Section III. The module labeled, *Other TA Systems* (Other Terrain Analysis Systems) refers to other uses of CTDBs, such as for agent-based route planning, or for its inclusion in a virtual reality simulation system [7].

The second extension to OTB is indicated by the boxes of Fig. 1 that are labeled, *Line-Oriented OTB Command Batch Interface*, and *Line-Oriented Commands*, respectively. They are explained in Section IV, below.

The third extension to OTB is the addition of the *SAF Broker* and *SAF Manager* agents, which are described in Section V. As can be seen from Fig. 1, the SAF Broker agents listen to DIS PDUs, and then transmit them to a SAF Manager agent, which collects and organizes the information that they contain about any one entity for any other agent or system that subscribes to its information updates.

The *RETSINA-OTB Bridge*, described in Section VI, was a significant fourth extension to the OTB simulation environment. It enables the direct creation, addressability, and tasking of any SAF entity. It also allows for the custom specification of OTB tasks, and for entities' partially-executed tasks to be interrupted or modified.

The fifth component of the agent- and OTB-based C4ISR testbed is the addition of a TCP-based *Batch Control Agent* that can configure and execute experiments in OTB that are expressed in a *High-Level Behavior Specification*. Thus, it is possible to specify the creation of SAF entities, their task execution orders, and to specify the termination conditions of their task (e.g. "fight for X minutes," or "fight until either side sustains more than 80% losses," etc.). We have used the Batch Control Agent to run hundreds of unsupervised batch experiment iterations in which random tank configurations (e.g. vee, echelon right, wedge, etc.) were evaluated to determine their effects as force multipliers.

The sixth and final component shown in Fig. 1, the *X-Sim Manager Agents* and the *X-Sim Agents*, illustrates how completely novel sensor types can be added to OTB, mounted on SAF entities, and integrated in a C2 application. The sensors are described in Section VII.

A recurring motif of this paper is that many of the integrations and extensions to OTB are with agent-based systems. The reasons for this are discussed in Section II along with some of the background of this work. We conclude in Section VIII.

II. Motivations and Background

Our particular motivation for having a C4ISR testbed is to have an environment in which algorithms for multiple levels of information fusion can be developed and tested. Information fusion is described in terms of functional levels by the Data and Information Fusion Group within the Deputy Director of Research and Engineering's Information Systems Technology Panel at the U.S. Department of Defense [8]. The lowest levels, 0 and 1, are concerned with the identification of individual entities (e.g. US M1A1 tank, Krasnovian T-80 tank, etc.) from the fusion of often low-confidence data from multiple types of sensors. Level 2 fusion attempts to aggregate the individual entities into larger organizational structures such as force echelons in order to perform reasoning at the third level of information fusion, on the expected behavior, intent, or threat that those organizational structures may pose. Level 4 fusion is concerned with the information acquisition process that was used throughout the lower levels, and on performing meta-level reasoning about how that process may be adjusted to be more accurate or use resources more efficiently in the gathering of that intelligence.

An example of a military process that exercises all four levels of information fusion is the Intelligence Preparation of the Battlefield (IPB) [4], an intelligence gathering process that begins with terrain analysis as its foundation. One of the procedures for performing terrain analysis is a create-and-revise process which results in the *Modified, Combined Obstacle Overlay (MCOO)* [5]: annotations of terrain, known obstacle and force deployments, and the identification of likely avenues of approach, engagment areas, and named areas of interest. *Avenues of approach (AA)* are paths of relative least resistance that a military force can take to reach an objective. Military planners will usually identify a primary AA and alternative, secondary AAs, when planning their missions. *Engagement areas (EAs)* are usually open areas of terrain

where two opposing military forces are likely to meet and fight. *Named areas of interest (NAIs)* are tactically significant entities such as, for example, key terrain, bridges, or buildings, the control of which would offer superior or decisive advantage in a battle. Generating the MCOO is often a level 3 fusion process, as it is based on general knowledge of opposing force (OPFOR) capabilities and tactics, and on specific — though typically incomplete — knowledge of OPFOR forces in a commander's sector. Once the MCOO artifacts have been generated, military staff officers then generate worst-case, most probable, and rarely, best-case scenarios, as time permits, called *Courses of Action (COAs)*, that they then war game, or simulate, to imagine how the COAs might evolve. Through this human, mental simulation exercise, military staff can determine the consistency of the information that they gather. If information that can be critical to a scenario is missing, the staff may request that a commander task assets to attempt to acquire the missing information — a level 4 process.

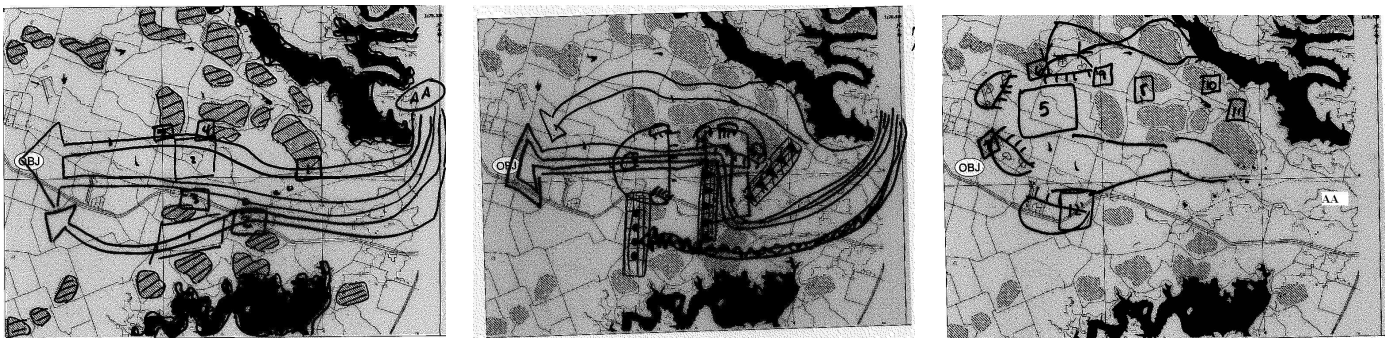
The modeling and automation of these types of information-fusion processes, in particular those of a goal-directed and dynamic nature, lend themselves to solutions based on a robust multi-agent system (MAS) such as RETSINA [6]. In MAS research there are investigations of many properties and autonomous behaviors of agents, but system-level interoperability and autonomy are the behaviors of direct relevance to the system described in this paper. *System-level interoperability* is the ability to integrate multiple software systems that were not designed to interact with each other without the need for a human to provide the "runtime integration". Systems that are incapable of such automatic interoperability and that require a human to provide the runtime integration are often called, "stove-piped systems". *System-level autonomy* is when an agent demonstrates behaviors of seeking and attempting to semantically interoperate with other autonomous agents so as to achieve its goals in light of changing environmental conditions, such as the loss or introduction of an information source, a change of subtasks, or to take advantage of new services by integrating them and their outputs into a new, meaningful sequence. *Semantic interoperation* refers to the ability of agents to collaboratively perform a task (e.g. solve a problem, or produce a service) based on the exchange of meaningful information, and not based on the choreographed timing of their collective program executions.

Another reason for using MAS technology in the context of this work is that multi-agent systems presume a common abstract architecture of functional services that can be implemented in heterogeneous ways. This facilitates the integration of a myriad of disparate software systems and components. These abstract architectures also guide decisions about how components within the architecture will interface with each other. Article [6] provides a complete explanation of these architectures, other features, and justifications for developing applications as multi-agent systems.

III. Extensions for Terrain Analysis

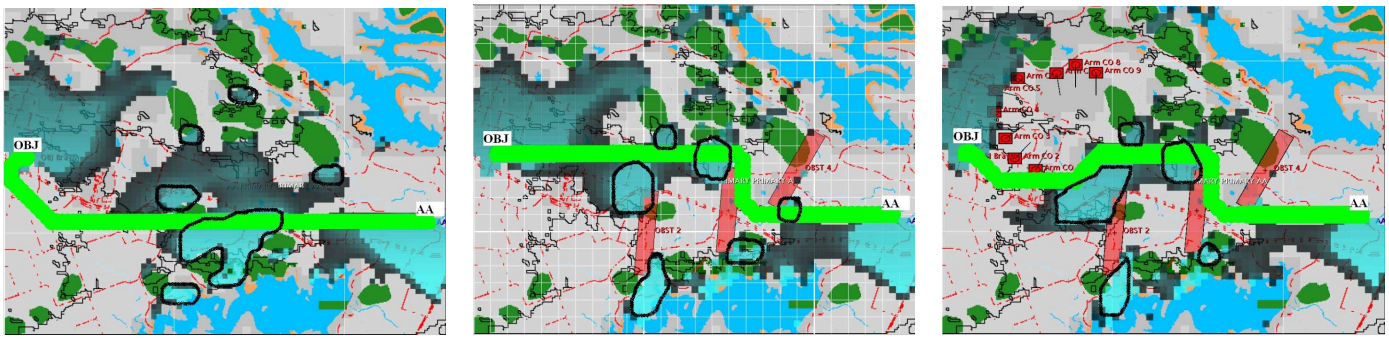
The first extension to OTB was to develop tools for accessing data stored in the compact terrain database (CTDB). Terrain information adds significant context for reasoning at information fusion levels 2, 3 and 4. CTDB terrain data represents: elevation, slopes, vegetation, soil type, surface drainage, soil characteristics due to weather conditions, bodies of water, and natural trenches. The algorithms to access CTDB data typically involve some form of sampling of the terrain data, relative prioritization of which terrain feature of a sampled cell will be the dominant feature of the sample terrain "pixel", and re-estimates of the continuity of the terrain represented by the sampled pixels. CTDB data consists of a rasterized height map that is based on a sampling of every N meters of terrain from a reference map, a triangular irregular network, or polygon mesh, adds significant elevation points at a finer resolution than the rasterized sampling, and linear features for roads, rail roads, rivers, tree lines, and forested areas. Accessing such terrain data involves not only reading the GIS data in raster format, but also accessing the linear feature vector data that is interpreted by the OTB modeling and simulation engine. Since CTDB terrain can be rendered at arbitrary resolutions, we first extract the target map area at the desired resolution using a combination of our own and OTB libraries, and then perform the sampling on that extracted map. If a sampled terrain cell contains multiple features, such as a river, road, and bridge, heuristic algorithms determine the final sample terrain label. For example, if a river is crossed by a road with a bridge, then the terrain type of "bridge", or "road" will be used to represent the sample pixel. If there are two roads and a river in the sample cell, but no bridge to connect the roads, then the dominant terrain feature will be the river. The final process of the terrain sampling is to unite sampled linear terrain pixels, such as roads or rivers, into continuous terrain features so that they may be considered by terrain reasoning modules for the "conduit" or "barrier" properties that they may have.

For the purposes of information fusion applications, the ability to automatically read and process CTDB data is essential to providing context for information fusion level 2 force aggregation and level 3 threat inferencing, and for determining where to task assets to look for OPFOR forces (level 4). Fig. 2 shows the results of a subject matter expert (SME) performing the MCOO process on OTB CTDB terrain maps. Fig. 3 illustrates an application of our extensions to OTB for terrain analysis on the same maps: the automatic generation of the MCOO. In both sets of maps (16km X 10km), the analysis is for a friendly force to depart from an assembly area (labeled **AA**) in the eastern portion of the map and to take the objective (labeled **OBJ**) in the western portion of the map, with the assumption that the opposing force (OPFOR) will try to halt the movement and lay down obstacles to try to shape the friendly's movements. Both friendly and OPFOR forces are presumed to be battalion-sized for these experiments, although the automatic terrain analysis is capable of reasoning at platoon, company and brigade levels, as well. The automatic



(a) No known obstacles or OPFOR battle positions. (b) Known obstacles, supposed OPFOR positions. (c) Known obstacle and OPFOR battle positions.

Fig. 2. An SME's terrain analysis. Key: double arrow — primary avenue of approach, single arrow — secondary avenue of approach, large boxes — engagement areas, small boxes — named areas of interest, "AA" — friendly assembly area. The SME did not draw the avenues of approach for Fig. 2(c) because they were the same as in Fig. 2(b).



(a) No known obstacles or OPFOR battle positions. (b) Known obstacles, unknown OPFOR positions. (c) Known obstacle and OPFOR battle positions.

Fig. 3. Output of automatic terrain analysis. Key: bright line — primary avenue of approach, outlined areas — engagement areas, "AA" — friendly assembly area, "OBJ" — objective for friendly force.

terrain analysis algorithms worked directly with data from the CTDB, creating each overlay in approximately nine minutes on a dual-2.4GHz Intel P4 processor with 2.0 GB RAM. The particular three-map overlay shown in Fig. 2 was produced in a little more than 60 minutes with the SME following speak aloud protocols, where he explained his reasoning as he created his MCOO, an activity that parallels the actual collaborative MCOO development process.

Not shown in Fig. 3 are the named areas of interest (NAIs) that the algorithms can calculate. NAIs are typically chosen to be in locations that command the best line of sight view of the engagement areas, and that are also within range of OPFOR direct and/or indirect fires. Considering that the automatic terrain analysis will assign a higher priority to NAIs adjacent to engagement areas along the primary avenue of approach, we can effectively achieve a degree of level 4 fusion through the reprioritization of NAIs as intelligence data enters the system. Such intelligence data can be HUMINT (human intelligence) from scouts or observers, or the fused output of multiple sensors.

IV. Line-Oriented OTB Command Batch Interface

The "unextended" version of OneSAF has two interfaces that can be used to place, query, and control entities. One interface is the *Command Editor*, a GUI that allows a user to place SAF entities on its rendering of a CTDB map. The other interface is a text-based command-line parser. This interface allows a user to create, place, and query entities in OTB through textual commands. This latter interface, also referred to as the OTB debug interface, is highly interactive, processing one human-entered command line at a time. While faster than navigating the GUI (for an expert user), it quickly becomes evident how tedious this interface is for effecting any complex and non-trivial operations in OTB.

We modified the library that manages the processing of commands through this interface to also read and write files that contain such commands. Such files are logically indicated in the box labeled, "Line-Oriented Commands," in Fig. 1. The extension is designed to poll a directory for the existence of a file containing line-oriented commands. If the process detects such a file, it renames the file so that it will not be detected again, opens it for reading, and begins to execute the commands that it contains, one line at a time. The executions are blocking, meaning that no command line or batch file will execute before its predecessor has completed. If one of the commands is to query the status of an entity, then the LOOCBI (line-oriented OTB command batch interface) will write the status information to a file.

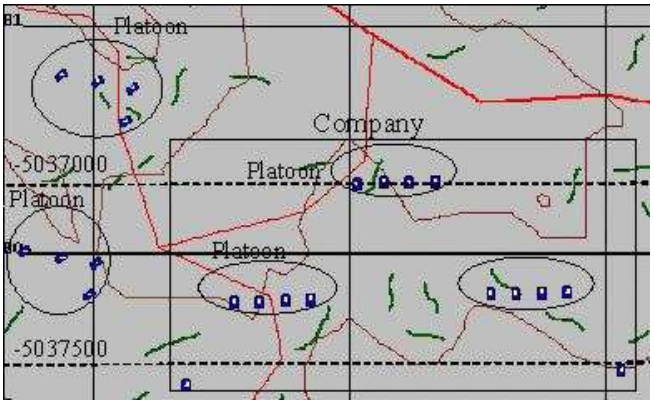


Fig. 4. Level 2 information fusion: recognizing echelon types and behaviors.

Since files can be created, renamed, or accessed by humans, agents, or web services, this interface can be used to perform rapid rudimentary batch mode experiments. While this type of extension is fairly quick to implement and easy to learn to use, the drawbacks of this method are that it: (1) requires meticulous manual preparation and editing of the command files, (2) requires the meticulous tracking of SAF entity addition/deletion requests in order for a person or program to know the OTB identification number of an entity, and (3) only offers coarse-grain query and control capabilities. That is, a command file must first finish executing before OTB will execute another command file.

This extension has been successfully used to test the coordination of three M1 tank platoons by autonomous agents in a dynamic environment [9]. In more recent experiments, this interface proved useful in the rapid development and testing of algorithms for level 2 fusion, as illustrated by Fig. 4, such as the recognition of tank platoons, companies, and their behaviors (e.g. bounding overwatch movement) [10].

V. SAF Broker and SAF Manager Agents

SAF Broker agents listen to DIS PDUs that are in the same multicast group as the OTB that transmits them. A SAF Broker can listen to as many multiple OTB simulations as are on a multicast channel, but cannot listen to multiple multicast channels. Other agents, such as a SAF Manager agent, can subscribe to SAF Broker services, and request that the Brokers filter only PDUs that originate from a certain OTB simulation image, or that pertain to a specific entity. If multiple simulators produce PDUs about the same SAF entity, a SAF Manager agent will accumulate such updates, add them to its internal database, and forward only those updates that have been requested by a subscribing program or agent.

Applications that use this agent system (ex. [7]) should follow OTB expectations of performing their own *dead reckoning*, which is an extrapolated estimation of an entity's state until the next PDU to update its state is received. As unsequenced, stateless UDP packets that are sent to ethernet addresses within the same multicast group, DIS PDUs may be lost or dropped without consequence. At the typical rate of 30 DIS PDUs per second, even if tens of PDU packets never reach their destination, the next packet that does will contain all of the current state information of the simulation environment. In a network with a high loss rate due to high volumes of message traffic and congestion, it is expected that the entity reading the DIS packets will perform its own dead reckoning.

Although it was recognized that converting DIS PDU messages into TCP messages could dramatically increase network traffic, our use of this system did not cause any perceptible degradation of the quality of the OTB updates. We believe that this has been because: (1) clients to the SAF Broker typically only need to read state information for visualization effects, and the overhead of parsing TCP messages for such state information is enough to handle multiple messages per millisecond, (2) since the TCP messages are generated from transport-unreliable UDP packets, if the client needs to have high-fidelity knowledge of entity state, it must implement dead reckoning, anyway, and (3) because of this, the use of dead reckoning obviates the need to improve the data communications model of the SAF Brokers and SAF Manager.

VI. The RETSINA-OTB Bridge

The purpose of the RETSINA-OTB Bridge is to allow for the finer-grained access and control of OTB entities and the simulation system, itself. It was implemented by adding a reduced (optimized for speed) C version of the RETSINA Communicator [6] program library to OTB, and building lightweight message processing routines to translate Communicator messages to and from OTB events and callback registrations. This internal library is called *libretsina*. The libretsina module receives specially-formatted TCP messages, and depending on the content, dispatches the content to the appropriate OTB event handling routine. If the message contains a query, then a RETSINA callback is registered with the OTB event processor. If the message contains a command or a task, then the corresponding OTB function is registered for execution.

The RETSINA-OTB Bridge, proper, resides outside of OTB so that it can optimize the streaming of messages to and from libretsina in OTB. Since OTB executes as a single-threaded process, any backups due to incoming message queue overflows will cause a degradation of system performance. As an external process, the Bridge can manage the message pacing into OTB without adversely affecting its performance. Messages leaving OTB have less of an impact on the system, but can still reduce the accuracy of simulation of OTB if queried too frequently.

Table I illustrates the impact of polling once and five times per second on OTB. Zero agent updates per second (cf. Table I) indicates that libretsina has not been registered with the OTB event processor. These results were produced by using the native OTB benchmark program to determine how many entities OTB can simulate in parallel at real time speed without OTB reporting that it is not able to "keep up" with internal entity state updates. On a dual-2.4 GHz processor Intel XEON computer with 2.0 GB of RAM, running

TABLE I

THE IMPACT OF THE AGENT LIBRARIES ON OTB'S PERFORMANCE.

Number of Platoons	Number of Entities	Threading	OTB Optimized	Agent Updates per Second
69	276	single	optimized	0
68	272	single	optimized	1
66	264	single	optimized	5
65	260	single	no	0
65	260	single	no	1
64	256	multi-	no	0
63	252	multi-	no	5
61	244	single	no	5
60	240	multi-	no	5

a multi-threaded RedHat Linux 7.1, kernel 2.4.20, connected to the 100mbs campus ethernet network, that limit has been around 272 M1A1 tanks, with the variations due roughly to the complexity of the terrain and the degree of interaction among the simulated entities. The reader should note that OTB was designed as a single-threaded architecture, hence its abysmal performance when multi-threading was enabled. Many parameters can be tweaked in an attempt to tune the performance of an OTB system, and many of these parameters depend on the nature of the operating environment and what is being simulated. Since most of our simulation exercises have been at the platoon and company levels, we have typically run the simulator with $50 - 75$ internal, SAF-native entities plus another $30 - 50$ external SAF entities, such as the SARSim, EOSim, GMTISim, etc. (explained in Section VII), associated with some of the SAF-native entities, all in the same image.

Communications based on the RETSINA-OTB Bridge need more careful considerations of the implications of transport reliability, since both the Bridge and the TCP communication end-point are considered to be transport reliable. While the rate at which a TCP message can be generated and transmitted is multiple messages per millisecond via the RETSINA-OTB Bridge, the number of messages that the client program can effectively process depends greatly on the type of message processing that must be performed by the Bridge client application. Some applications, such as one that visualizes and animates point-to-point communications between SAF entities, are not able to keep pace with the RETSINA-OTB Bridge updates, and so would block the reception of further Bridge updates until it could empty its incoming network message queues. This blocking of the transmissions would cause the Bridge, in turn, to block the iteration of SAF simulation cycles, and thus OTB would appear to freeze at times. This problem was resolved by enabling a RETSINA Communicator option whereby input messages can be discarded if the client application could afford to drop messages and its input message queues were full. Other areas where performance tuning can be effected are in adjusting the size of the message queues, and adjusting the size of the messages. We noticed increases in performance as: smaller messages were joined into larger messages, their content was streamlined into a flat structure for rapid extraction, and dead reckoning was employed by client applications, if appropriate.

VII. SIMULATED MOUNTED SENSORS

One of our significant contributions to OTB is to add three simulated mounted sensors, *SARSim* (synthetic aperture radar simulator), *EOSim* (electro-optical simulator), and *GMTISim* (ground moving target indicator simulator), to the simulation environment, thereby increasing the types of surveillance and reconnaissance that can be performed, and augmenting the type of level 1 fusion data that can be used for the development of our fusion algorithms. High fidelity simulations of such sensor systems exist, but they are either classified or prohibitively expensive, and ultimately, inaccessible to a research group that is interested in developing and testing command and control algorithms that manage the scheduling and tasking of such simulated platforms as a C-130, F-16, UAV, or WASM (wide-area search munition), on which the sensors can be mounted. Considering that multiple sensors may be mounted on the same platform, or that some platforms may double as a munition as well as a sensor, the command and control of these platforms with such wide-ranging and diverse capabilities is a non-trivial task, and using OTB for the simulation of their real world dynamics and behaviors would allow us to investigate these problems. The three sensor models were integrated with OTB as external modules for better modularity, expediency and convenience. With a code base of nearly one million lines of code, and over 500 software libraries [2] in which entity behavior is determined by inherited behaviors from multiple classes, it was easier, quicker, safer, and just as effective and scalable to integrate these sensors as external entities instead of as internal, compiled entities. Approximate characteristics of the sensors are described in Section VII-A, while the integration of the three models with OTB is described in Section VII-B, below.

A. Sensor Characteristics and Algorithms

The three sensor simulators have many characteristics in common, so we will first describe the SARSim so as to establish a baseline understanding. Real SAR sensors have the capability of maintaining the same spatial resolution (i.e. feet, meters, etc.) independent of range, although the farther a sensor is from a target in some modes of operation, the more prone they are to error due to atmospheric conditions. Standoff ranges for publicly documented SAR sensors range from 10km to 100km, and have resolutions ranging from 1 meter to less than 1 foot. For example, the SAR mounted on a GlobalHawk UAV flying at 650 km/hr at an altitude of 65000 feet is capable of imaging an area in spotlight mode that is 100km away at a resolution of 1 foot. *Integration time*, or the time that it takes the SAR to acquire an image, is roughly a function of parameters such as: arclength, velocity, subtended angle, angle from broadside, radar wavelength, and desired resolution. For a platform flying at 560 km/hr, looking straight on at its target, imaging in 1 foot resolution spotlight mode with a 3 cm (10 GHz) radar wavelength, the time to acquire an image is roughly 7.7 seconds. The SARSim model allows for the setting of all of these parameters. For the purposes of our simulations, and given that our terrain maps are typically around 75km X 75km, we have configured the SARSim to sense at a range of 25 km, mounted on an F-16D, flying at around 600 km/hr. Combined with a negligible time for ATR processing, the total integration time for scanning an area is usually completed in 10 seconds, or less. The GMTISim has roughly the same configurable parameters as the SARSim, so for the purposes of our simulations, we also mount the GMTISim on an F-16D, flying at around 600 km/hr, and activating at a standoff range of 25 km to the target. The standoff range for EO sensors is much shorter, so we configure our EOSim to sense at 15 km from the target.

The three simulators interoperate with OTB by receiving ground truth data about the OTB entities and "confusing" it according to a sensor-specific *confusion matrix* model. The sensors begin by subscribing to entity updates that arrive via DIS packets, and maintain an internal table of all such entities and their ground truth status. When the sensor is tasked to scan an area, and the sensor is within sensor range, it reads the entity ID, its orientation on the ground relative to the sensor, and then produces a list of possible target identities with associated levels of confidence according to its confusion matrix model. The confusion matrix does not model the real fidelity of the sensor, as that is classified, but it does provide a series of low-confidence estimates in which the highest-confidence identification is not necessarily the correct classification. Thus, it is possible for a simulated sensor to confuse an M1 tank with a T-80 tank. So as to produce false positives and false negatives, the sensors have random functions that either generate entities where they do not exist or that suppress the reporting of entities where they do exist.

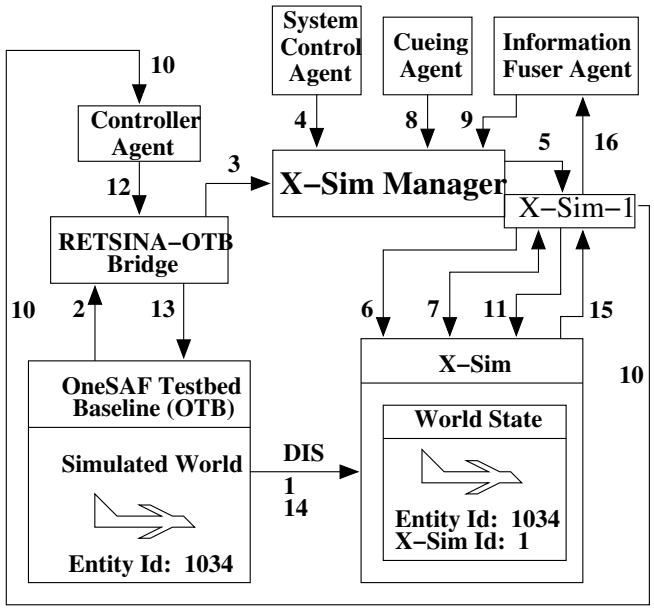


Fig. 5. Integration of OTB, X-Sim, and the Agents

B. OTB, X-Sim and Agent Integration

X-Sim is the abstraction used to describe the architecture that is common to the SARSim, EOSim, and GMTISim simulators. The following is a brief summary of the steps that are represented in Fig. 5.

An entity is created within OTB, and has an identification number assigned to it by OTB. In this example, that number is "1034". **(1)** As soon as an entity is created in OTB, entity state PDUs are multicast via DIS packets to all programs that are part of the same multicast group and exercise identifier as the OTB simulation. The X-Sim creates an entry in its "World State" table for every entity that is created in OTB. **(2)** Entity state information is transmitted to the RETSINA-OTB Bridge via RETSINA messages, which are based on the TCP protocol. **(3)** The RETSINA-OTB Bridge transmits all entity state information messages to whichever agents are subscribed to receive its update notifications. In Fig. 5, this is the X-Sim Manager. **(4)** A System Control Agent contacts the X-Sim Manager and issues the command to associate an X-Sim instance (e.g. "X-Sim-1") with a particular OTB entity (e.g. "1034"). **(5)** The X-Sim Manager spawns a new thread which creates a new RETSINA Communicator proxy with the identity of the newly requested X-Sim instance (e.g. "X-Sim-1"). The X-Sim Manager will route all communications for the X-Sim entity to that proxy. **(6)** The X-Sim-1 proxy sends an "install" command that notifies the X-Sim module that there should be a new instance of a simulation model, it should be associated with a specific OTB entity (e.g."1034"), and that it will base its behavior on the simulated behavior of that specific asset (e.g. entity "1034"). **(7)** Should the "install" command be received before the X-Sim has received the DIS packet announcing the existence of the entity, the X-Sim will reply to the X-Sim-1 proxy that the entity was not found. The X-

Sim-1 proxy will continue to resend its "install" request after a small delay until the X-Sim acknowledges the entity, or until the "install" command is canceled.

(8) A Cueing Agent sends a message request to the X-Sim Manager to scan an area. Requests are queued in the order that they are received and assigned to the proxy of the first available OTB asset (e.g. proxy "X-Sim-1" for entity "1034") without consideration of that asset's proximity to the target area. Requests to cancel a scan may be sent to the X-Sim Manager. **(9)** At any time, a service requesting agent such as the Information Fuser Agent, Belief Display Agent, or an interface agent that displays the X-Sim data, can submit a monitor query request to receive any and all notifications from the X-Sim Manager. Those notifications will be generated by the X-Sim sensor instances that the X-Sim Manager controls. **(10)** The assigned X-Sim proxy (e.g. "X-Sim-1") generates a new OTB destination and path plan for its asset to travel. This "order" is submitted to a Controller Agent which manages the scheduling of multiple requests from other agents in the simulation system.

(11) In parallel with the asset order, the assigned X-Sim proxy "orders" the simulation model instance to survey the area when the asset on which it is mounted is within the sensor's range. **(12)** The Controller Agent sends the next scheduled task for the asset to the RETSINA-OTB Bridge. **(13)** The RETSINA-OTB Bridge checks the received task for syntactic and semantic correctness and if valid, forwards the request to OTB. The RETSINA library within OTB interprets and applies the task to the entity (e.g. "1034"). **(14)** When an entity state PDU indicates that the asset on which the sensor is mounted is within range of the area to scan, the sensor simulator, "X-Sim", "turns on" its simulation instance (e.g. "X-Sim Id: 1") for the duration required by that modeled sensor to perform its task. **(15)** Once the sensor simulator has completed its imaging, it sends the results back to its X-Sim Manager proxy. If the asset on which the sensor was mounted moved out of range before the sensor finished the task, the sensor returns partial results, if that is what it would do in reality, otherwise no results are returned. If the asset on which the sensor is mounted is destroyed, then the X-Sim Manager will delete its proxy and notify the X-Sim to delete the simulator instance that was mounted on that asset. The task that was assigned to the asset and sensor is also lost and will need to be rescheduled. **(16)** The X-Sim Manager proxy sends any and all results from the simulator to all agents that are subscribed to its notification service.

VIII. Conclusions

This paper has demonstrated the need and ability to extend the OneSAF Testbed Baseline modeling and simulation system into a C4ISR testbed for the research and development of algorithms for multiple levels of information fusion, and for the automatic command and control of military assets. We do this via a variety of interfaces and extensions to the native OTB platform, both through the OTB interfaces that were intended for system expansion, and by making small but very effective modifications and additions to the system. Some of these small but effective modifications were the addition of libraries to enable OTB to communicate with agent-based systems. This permits the expansion of OTB into a system that is highly-interoperable with a heterogeneous array of other C4ISR components and test platforms. While the scope of this claim is limited by our ignorance of the full range of military operations, this notion has been bolstered informally by favorable reviews of the work by military personnel and by transitions of this technology to the military. Indeed, a branch of the military has already applied some of the technology described in this paper to a specific logistics planning problem, and certain classified research labs have used these extensions for range tests that integrate versions of our algorithms, control and behavior of the actual hardware, and multiple simulated entities. Future work will continue to refine and enhance these interoperability components as we continue to research and develop high-level information fusion and C2 algorithms and test scenarios.

Acknowledgments

The authors would like to thank past members of the Intelligent Software Agents Lab who contributed to the development, testing and use of our system: Jason Ernst, Brent Langley, Martin Van Velssen and Wei Yang. Many thanks to our partners at Northrop-Grumman, led by Dr. Robert Mitchel, for their assistance in developing some of the external simulators. This research has been sponsored by Air Force Office for Scientific Research (AFOSR) Grant F49620-01-1-0542.

References

[1] "OneSAF Testbed Baseline (OTB)," http://www.onesaf.org, accessed June 2005.

[2] Advanced Distributed Simulation Technology II, "OTB assessment final report," Lockheed Martin, P. O. Box 780217, Orlando, FL 32878, Tech. Rep. ADST-II-CDRL-ONESAF-9800101, 8 May 1998, http://www.onesaf.org/OTBFinalReport.doc, accessed June 2005.

[3] J. J. Brann, "Enumeration and bit encoded values for use with protocols for distributed interactive simulation applications," Institute for Simulation and Training (IST), 3280 Progress Drive, Orlando, Florida 32826, Tech. Rep. IST-CF-02-01, 16 April 2002.

[4] *Intelligence Preparation of the Battlefield*, Headquarters, Department Army, Washington, D.C., 8 July 1994, FM 34-130.

[5] *Terrain Analysis*, Headquarters, Department Army, Washington, D.C., July 1990, FM 5-33.

[6] K. Sycara, J. A. Giampapa, B. K. Langley, and M. Paolucci, "The RETSINA MAS, a case study," in *SELMAS*, A. Garcia, C. Lucena, F. Zambonelli, A. Omici, and J. Castro, Eds. Springer-Verlag, 2003, vol. LNCS 2603, pp. 232–250, invited paper.

[7] J. Manojlovich, P. Prasithsangaree, S. Hughes, J. Chen, and M. Lewis, "UTSAF: A multi-agent based simulation bridge for distributed military simulation interoperability," *Simulation*, vol. 80, no. 12, pp. 647–657, December 2005, ISSN 0037–5497, http://sim.sagepub.com.

[8] D. L. Hall and J. Llinas, Eds., *Handbook of Multisensor Data Fusion*. CRC Press LLC, 2001, ISBN 0-8493-2379-7.

[9] J. A. Giampapa and K. Sycara, "Team-oriented agent coordination in the RETSINA multi-agent system," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213-3890, Tech. Rep. CMU-RI-TR-02-34, December 2002, presented at AAMAS 2002 Workshop on Teamwork and Coalition Formation.

[10] J. Ernst, J. A. Giampapa, and K. P. Sycara, "Learning to recognize coordinated multi-entity actions," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213-3890, Tech. Rep. CMU-RI-TR-03-53, 23 December 2003.