

The Integration of the Joint Warfare Simulation (JWARS) and Global Command and Control System (GCCS) with Intelligent Agents through Web-Services

Ranjeev Mittu

U.S. Naval Research Laboratory
4555 Overlook Avenue
Washington, DC 20375
(202) 404-8716
mittu@ait.nrl.navy.mil

Virginia T. Dobey

SAIC/SETA Support to DMSO
1901 North Beauregard Street
Alexandria, VA 22311
(703) 824-3411
Virginia.Dobey.CTR@dmsomil

Gerald DePasquale

CACI
1600 Wilson Boulevard
Arlington, VA 22209
(703) 558-0283
gdepasquale@caci.com

Keywords: *Multi-Agent Systems, Intelligent Agents, Simulation, C4I, Plan Monitoring, JWARS, GCCS, ITEM*

ABSTRACT: For years, simulations have been used by analysis and planning staffs to develop and rehearse operation plans, analyze results, and develop doctrine. Typically, combat simulations are used most heavily during the planning stages of an operation, prior to battlefield action. However, simulations are increasingly being used during operations to perform course of action analyses (COAA) and develop real-time forecasts of future conditions on the battlefield. Recent efforts by the Defense Modeling and Simulation Office (DMSO) to improve the interoperability of C4I systems with simulations has provided a powerful means for rapid simulation initialization and analysis during exercises, and made simulations more useful and responsive as the exercises are executed. These DMSO efforts involve technology development to support the integration of operational C4I systems, such as those in the Global Command and Control System (GCCS), with simulations such as the Integrated Theatre Engagement Model (ITEM) and the Joint Warfare Simulation (JWARS).

This paper will describe the FY03 *GCCS-ITEM-Intelligent Agent Federation project* and its support for agent-based plan monitoring, discuss project results, and present its conclusions. It will also describe phase I (FY03) of the *JWARS-GCCS Integration project*, presenting initial results as well as preliminary conclusions. Finally, it will describe the FY04 *Simulation-to-C4I Connectivity project*, which is integrating the results of these two projects, leveraging the agents developed for the GCCS/ITEM federation, and extending them with additional monitoring capabilities to support the use of JWARS as an embedded tool for the C4I operator. In this integrated project, interfaces between operational C4I systems, simulations, and intelligent agents are designed to exploit web-service technologies and the standardized

information exchanges described by the NATO and Multilateral Interoperability Programme Command and Control Information Exchange Data Model (C2IEDM).

1. Introduction

During FY03, DMSO sponsored two significant projects in the area of Simulation-to-C4I technology. The first project addressed the integration of GCCS, ITEM, and intelligent agents to support execution monitoring. The idea behind this approach was to develop a plan to be executed (accomplished using ITEM), use that same plan in ITEM to establish the baseline expectations for the battlefield course of action, and compare those expectations to the actual battlefield course of action (as represented in GCCS). The focus was on the use of intelligent agents to identify deviations in actual operations from what was expected by the plan and alert the GCCS operator when the deviation exceeded a predefined threshold.

Connectivity in the integration involved GCCS-ITEM federation communication and exchange of track data through the High Level Architecture (HLA) Run-Time Infrastructure (RTI). The Critical Mission Data over RTI (CMDR) toolkit was used to bridge the RTI-based federation interface to an agent federation residing on the Control of Agent Based System (CoABS) Grid. Agents on the grid were responsible for detecting deviations between the actual execution picture represented in GCCS and simulated tracks in ITEM, and initiating alerts when such deviations exceeded a predefined threshold. This proof of concept was successfully demonstrated in February, 2004.

Simultaneously, the second project addressed the ability of GCCS and JWARS to share real-time track data, compare them algorithmically, alert the GCCS operator when results of the comparison exceeded a predefined threshold, and allow the GCCS operator to use JWARS for real-time CoAA. This project differed from the other

in its focus on simulation-to-C4I data sharing. While the first project relied entirely on simulation data within ITEM to supply both the planned and actual scenarios, the second used real world (and proved the potential for real-time) track data to initialize JWARS and feed GCCS. As a result, while the first project determined the viability of incorporating intelligent agent technology in problems of this nature, this second project instead used a simple algorithmic comparison. Another difference is that the second project “embedded” JWARS operationally, emphasizing the use of JWARS to establish the baseline expectations for battlefield activity as well as to develop multiple options for follow-on action when the operator determined that an alert required a change in the predetermined operational plan. This proof of principle was demonstrated in October, 2003.

As a result of these successes, DMSO funded a continuation of these projects, integrating them to bring these technologies together and adding a new connectivity focus. Built upon leveraging the value of intelligent agents and simulation technology to empower the battlefield commander, the current (FY04) project incorporates Web interfaces and provides a proof-of-principle for addressing the technical challenges of the Global Information Grid.

2. Project 1: GCCS-ITEM-Intelligent Agent Federation (FY03)

In establishing the Simulation-to-C4I program for FY03, DMSO identified the need to further examine the maturing research in intelligent agents and evaluate its potential for use in simulations. Intelligent agents have the potential to monitor extremely large amounts of data, sift through them to extract (pre-identified) information, and present that information as existing knowledge to the operator. The strength in this technology is its ability to monitor data; the challenge is to determine the criteria and thresholds that separate “significant” data from all other. The goal for this initial project was to prove that such technology could be integrated into a simulation federation and provide additional value to the simulation operator. The challenge was to develop the necessary connectivity to ensure that the agents were able to fully perform in the federation environment.

This initial project consisted of three principal components: (1) the Track Data Base Manager (TDBM) from the GCCS, (2) the ITEM simulation, and (3) the CoABS grid, which provided the intelligent agents to the federation. Two key connectivity components were also necessary: (1) the GCCS HLA Ambassador (shortened to Ambassador), which acts as a GCCS segment in extracting TDBM data and posting it to the HLA RTI simulation backbone for transfer to other federates, and (2) the CMDR bridge, which resided on the CoABS grid and connected it to the HLA RTI.

The system architecture is shown in Figure 1. Further details of system components can be found in [2,3].

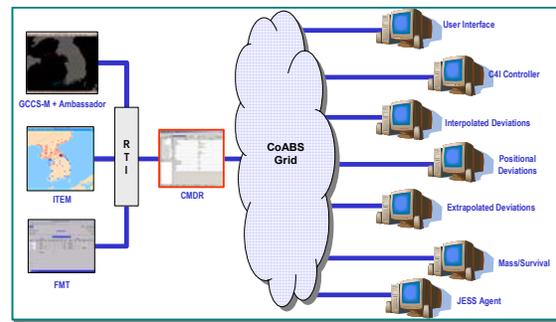


Figure 1: GCCS-ITEM-Intelligent Agent Federation System Architecture

In operation, the process began when the Ambassador extracted “real world” tracks from the TDBM and provided them to ITEM. Both systems were simultaneously initialized, after which the GCCS TDBM published updated track information and ITEM forecast track movement based on the set of initial TDBM tracks. The CMDR subscribed to both sets of track data and transferred them to intelligent agents on the CoABS grid for analysis. As the intelligent agents analyzed the data, alerts or retractions of alerts were generated according to predefined thresholds. Alerts were generated when the deviation of an actual track from a forecast track exceeded a threshold; alert retractions were generated when that deviation, which had previously exceeded a threshold, now fell below threshold limits.

In order to better understand the performance and potential of this technology, it is helpful to provide a detailed explanation of the project execution sequence. It is important to note the interaction of the three major components with

the connectivity components, and in particular how those connectivity components managed the details of the various data exchanges across the simulation-to-C4I interfaces.

Step 1: Develop a set of scenarios using ITEM

Several combat scenarios were developed using ITEM. These scenarios were developed by a Subject Matter Expert (SME) supporting U.S. Forces Korea (USFK). Three separate scenarios were used, the largest one consisting of 490 hostile and friendly ground units and 1054 hostile and friendly ships. Each scenario consisted of a thirteen hour segment of the simulation script, which was recorded for subsequent use.

Step 2: Using a selected scenario from Step 1, modify one copy of that scenario such that some of the tracks deviate from those in the other copy of that originally- identical scenario.

This step is an artifact of the experiment environment. Because we would not be running this experiment in real time, we would need to set up the operational system *as if* it were running in real time. Consequently, we had to develop a scenario script that would cause the operational system to act *as if* it were being used in a battlefield C4I environment.

That same script needed to be used to initialize the simulation, since the simulation would then indicate the forecast activity of the battlefield. Consequently, we developed individual scenario scripts to depict the real time operation of the C4I system and duplicated that script for use in the simulation system. We labeled one as the “real-world plan” and the other as the “simulated plan”. These scripts were developed by ITEM.

The “simulated plan” was used as the baseline to describe the expected progress of battlefield activity. The “real-world plan” was altered to reflect the constantly-changing battlefield conditions and deviations of actual from expected activities. This variant incorporated behaviors of friendly and hostile land units and naval platforms behaviors which differed from the “simulated plan.”

Step 3: Having created a representation of the real world plan as well as of the simulated plan, GCCS published the initial state of the battlefield (then halted), while ITEM subscribed to this initial state from GCCS. This was to support a

correlation step that would be done in the real world to put both systems on the same baseline.

The Ambassador was used to extract tracks of interest from the GCCS TDBM and publish them via the HLA RTI. This artifact represented the actual real-world situation at the beginning of the plan execution. To support our experiments, the Ambassador was modified to be able to publish all tracks and their data to the RTI without the user having to rubber-band and select a group of tracks. This enabled an automatic data flow, guaranteeing that all track information would be made available to the intelligent agents, and allowing the intelligent agents the ability to select tracks of interest.

Very shortly after the Ambassador published all of the tracks to the RTI, and they were received and displayed within ITEM, the scripted “real-world plan” was directed to stop. This established the initial state of each real-world track. Through the HLA RTI’s subscription capability, ITEM received those tracks and was able to ingest them to form the “simulation plan’s” initial state. This synchronization step provided us with a one-to-one mapping between “real-world” and “simulated” tracks. ITEM was modified to assign the same GCCS Local Track Number (LTN)—an internal “house-keeping” variable used by GCCS as a primary key for each track—to its simulated counterpart.

The use of the LTN was sufficient for our initial experimentation; however, we soon discovered that it was not a good choice for our track identifier. The LTN is a number assigned to a track when GCCS loads the information from the TDBM or acquires the track from real world input (ITEM does a similar assignment for its objects). This LTN is unique for the duration of the time that GCCS displays that track. It does not exhibit persistence in the TDBM database, and therefore is not consistent throughout multiple runs. It was soon concluded that a more permanent key would be needed in the future.

We encountered a significant problem when the real-world track did not correlate properly to a track in ITEM. If the GCCS track LTN was not assigned to its ITEM counterpart, an ambiguous track appeared on the GCCS screen. Left unresolved, the agents were not able to match real and simulated track data, thus preventing comparisons and identification of activities of significance to the operator. Consequently, any uncorrelated track (in either system) needed to be

examined and correlated to its appropriate counterpart in the other before the experiment could proceed further.

Step 4: ITEM published all simulation objects to the RTI in order to provide force composition information to GCCS.

Once all simulation tracks were established and correlated to their real-world counterparts, ITEM published the entire scenario, including track objects and health status objects, to the RTI. This necessitated enhancing ITEM to support the publication of objects describing force combat worth in terms of “mass”.

“Mass,” in this problem, is described as the relative value of each entity participating in the simulation as compared to the strength of an M1A1 tank. Establishing the force combat worth of an M1A1 as the baseline (with value 1.0), all other weapon resources (including humans) were evaluated against its strength and assigned a relative value. For example, a soldier with a Rocket Propelled Grenade (RPG) might have a mass of 0.1, signifying that ten human/RPG pairs would equal the combat strength of the M1A1. The objects published by ITEM contained force combat worth (mass) information using these pre-determined values. The combat worth (mass) value of each object was used as a basis of comparison to determine probability of surrender and survival for the object as it remained involved in battlefield activity.

Each published object, with its related “mass,” became accessible via the RTI to the *mass monitoring agents* located within ITEM. These intelligent agents monitored the potential for surrender of battlefield units, while others evaluated the potential for survival of the platforms engaged in battlefield activities. The intelligent agents used the following constraints to trigger alerts:

(Current mass)/(initial mass) < Surrender threshold (Probability of survival) < Survival threshold

Thresholds were defined in the baseline script (maintained as the “simulation plan”).

We noted that a potentially valuable contribution of these *mass monitoring agents* is the ability to analyze an opposing force’s combat worth prior to monitoring the actual execution of the plan. This may prove to be useful in refining the initial plan. Future work could involve combining

Mass Monitoring with the use of Unit Order of Battle information as a means of monitoring force composition and battle readiness.

Step 5: Once ITEM finished publishing the scenario to the RTI and the agents completed analysis of the surrender/survival potential of units and platforms, GCCS was resumed.

In this next step, the GCCS Ambassador and ITEM published track information, representing the real world execution and simulated execution, respectively, to the RTI. Tracks were fed through the RTI via CMDR to the agents registered on the CoABS grid. Several modifications were made to CMDR to support our experimentation, including the ability to translate data into XML as well as improvements to the record/playback feature.

In order to accomplish our monitoring function, we developed several types of track deviation monitoring agents, including *deviation-by-extrapolation agents* [3] and *deviation-by-interpolation agents* [3]. Furthermore, two additional types of agents were developed, the *C4IController agent* and the *UserInterface agent*.

The Java Expert System Shell (JESS) [1] was chosen as the core engine providing the agent reasoning capability. JESS is a Java-based rule engine and scripting environment. Originally based on the CLIPS expert system shell, it has grown into its own distinct paradigm. There were a number of reasons for choosing JESS, the primary one being that JESS’ implementation of clips-like rules in Java facilitates the integration of intelligent, rule-based agents. It contains a Java API for accessing the shell, and the functionality of the shell itself can be expanded through Java. JESS uses the Rete algorithm, a powerful mechanism for triggering rules in the knowledge base efficiently, and can theoretically scale up as the number of rules remains similar to each other (i.e., the antecedent of the rules are similar).

The overall monitoring task of the progression of the plan, as established by the simulation, would be decomposed into subtasks by the plan-understanding agents (future work) and delegated to the monitoring agents. The metaphor is to place a camera into each room of a building to detect small changes in situation that might influence the overall state of the entire building. Rules can be specified to automatically

detect when certain conditions are present and trigger alerts instead of continuously monitoring the simulation.

Each agent registering to the Grid has its own rule-based engine. Its knowledge base is initialized with simulated tracks and its deviation rules select the relevant temporal tracks to be compared against. Further enhancements will consist of incorporating background knowledge to instantiate possible explanations to those deviations and provide guidance on appropriate thresholds of deviation.

The time-step interval in the simulation indicates the temporally relevant tracks. Two different methods to calculate deviations are used to make up for the unsynchronized nature of real-time events with planned events. The *deviation-by-extrapolation* agent projects a position in the future when the course and speed is known and compares that projection against a simulated event. The *deviation-by-interpolation* agent does a linear interpolation between two temporally consecutive simulated events to estimate the current position. That estimation is then compared with the real-time event. The procedure for the distance calculation between those events is described in Figure 2. This distance serves as the decisive factor for triggering alerts when deviations occur above threshold (and retracting alerts when below threshold).

The latitude and longitude, given in decimal degrees, are converted to radians and the earth radius (6378 kms) is added to the altitude. The angle α between 2 points, p_1 and p_2 , is computed first:

$$\alpha = \arcsin(\sin a_1 * \sin a_2 + \cos(b_1 - b_2) * \cos a_1 * \cos a_2)$$

where a_1 is the latitude of p_1 , a_2 is the latitude of p_2 , b_1 is the longitude of p_1 and b_2 is the longitude of p_2 . The distance is then computed using the cosine law:

$$\sqrt{(r + c_1)^2 + (r + c_2)^2 - 2(r + c_1)(r + c_2) \cos \alpha}$$

where r is the earth radius and c_1 and c_2 the respective altitudes of p_1 and p_2 .

Figure 2: Distance Calculation

The *UserInterface* agent interface is shown in Figure 3. Through this interface, the user is able to spawn *mass monitoring* or *track deviation monitoring* agents. The user may enter whether units or platforms are to be monitored for each type of agent. Furthermore, the interface

provides a mechanism to specify threshold values, that, when exceeded, would warrant alerts (which are also captured in the display as well as sent to GCCS for display within its COP). This agent was developed using the Java Swing GUI and provided the ability to “spawn” the track deviation monitoring agents.

Within our implementation, multiple track monitoring agents can be created to monitor the same types of deviations for different tracks. For example, several *deviation-by-extrapolation* agents can be created that monitor different tracks.

The monitoring agents, once created, register tracks of interest with the *CAIController agent*, which then routes track data to them as this information comes in from CMDR.

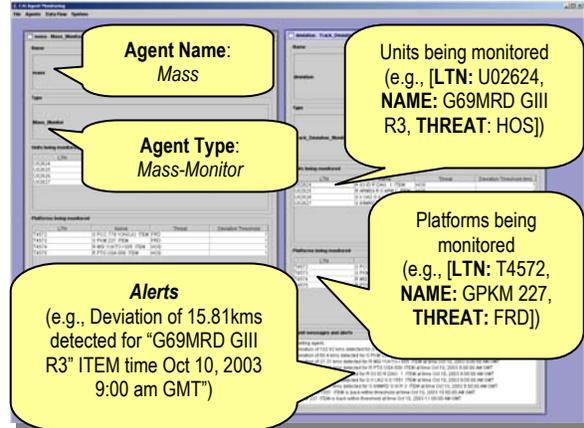


Figure 3: UserInterface agent

Step 6: The agents generated alerts, which were then transferred via the HLA RTI for display on the warfighter's GCCS COP (Figure 4).



Figure 4: Alerts on the GCCS COP

When a deviation of significance was identified, an alert was initiated and transferred to the GCCS for display on the operator's screen. This alert was maintained until the operator acknowledged it, either noting and deleting it or using it as a basis for CoAA with related plan updates.

3. Project 1: Initial Conclusions

The older version of GCCS necessitated the inclusion of several other old component versions, namely the Solaris OS, the RTI, and available platforms. GCCS-M was the latest available version for this integration (which was still quite old – 3.x), however, a cost was paid. With technology moving forward as fast as it is, there is a delicate balance struck when exploring new concepts using old technology. For future work, a newer version of GCCS will be available, and current work can be transitioned forward.

The performance of the overall system was somewhat less than satisfactory. In the largest scenario, thirteen hours of simulated time compressed to 2.5 hours of real time. While this is satisfactory in a laboratory environment, this is not sufficient to meet operational doctrine development and training needs. We theorized that, with newer versions of software and hardware (we used GCCS 3.X), that playback time might be drastically reduced. Despite this, we were able to successfully experiment with the largest scenario without experiencing significant performance (throughput) degradation.

This experiment, by necessity, had to involve a scripted (simulated) operation of the operational GCCS system. It would be very interesting, as a follow-on exercise, to observe the performance and utility of this project when GCCS is being operated in real time.

4. Project 2: The JWARS-GCCS Integration Project (FY03)

Simulations are a potentially powerful tool in the hands of the real-time, real-world C4I system operator. Yet most simulations are stand-alone or designed to operate apart from the day-to-day real-world battlefield operations they are designed to simulate. While this is not necessarily a bad thing, it does limit the value of simulations in real-time operations, when the

operator is most in need of CoAA and other types of tools that simulations can provide.

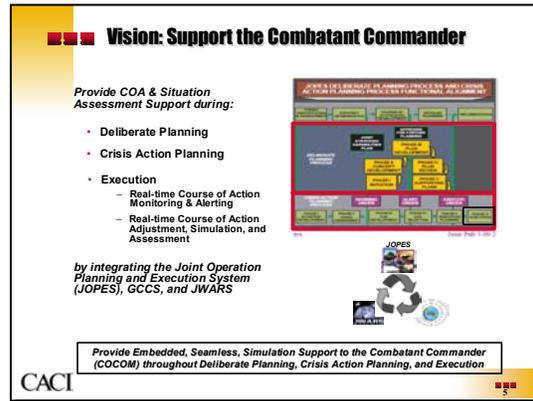


Figure 5: JWARS-GCCS Vision Statement

The ultimate vision of the second DMSO-sponsored Simulation-to-C4I project was to provide the Combatant Commander with COA & Situation Assessment Support during deliberate planning, crisis action planning, and execution by integrating JOPES, GCCS, and JWARS as depicted in Figure 5. The objective was to provide that capability by “embedding” JWARS into GCCS as shown in Figure 6.

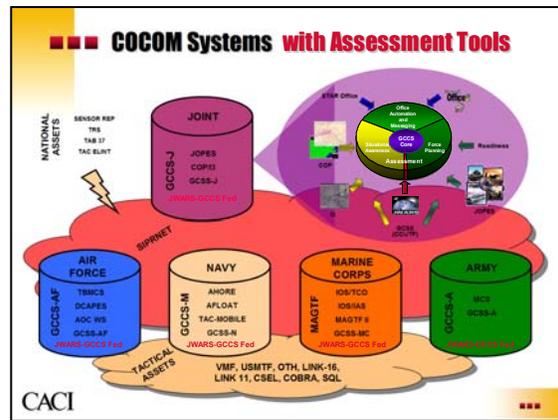


Figure 6: JWARS Embedded in GCCS as a Situation Assessment Tool

The concept of operations from peacetime through military operations is provided in Figure 7. The concept provides the combatant commander with the ability to spawn look-ahead replications when conditions warrant a need to consider alternative courses of action.

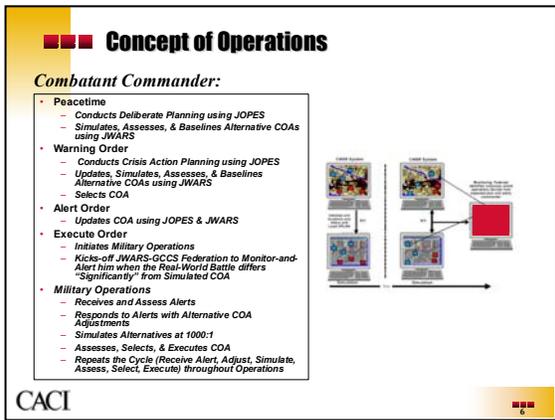


Figure 7: JWARS GCCS Concept of Operations

This project consisted of five principal components as shown in Figure 8. From left to right the components are: (1) the C4I-Gateway, (2) the GCCS TDBM, (3) the GCCS Ambassador, (3) JWARS executing the expected COA, (4) the Alert Handler, and (5) a second JWARS supporting execution of look-ahead replications. Each is now discussed:

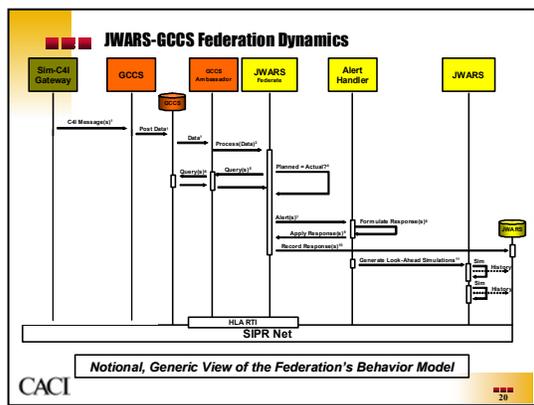


Figure 8: JWARS GCCS Notional Behavior Model

- The C4I-Gateway was used to stimulate the GCCS-M TDBM with operational messages needed to populate it with track information.
- The GCCS Ambassador forwarded that track information to JWARS for comparison to the combatant commander's expected values.
- JWARS compared track information it received from the GCCS Ambassador to the units in play in its simulation. When track information sent to JWARS did not match information in the scenario, it would send an alert to the user to be handled.

- The user formulated a response to each alert and sent instructions for dealing with the alert to JWARS.
- JWARS recorded each user response in a history database for future recall.

The user was afforded several options when alerted. The user could: (1) Ignore and Resume, (2) Synchronize and Resume, (3) Synchronize, Adjust COA, and Resume, (4) Synchronize and Look Ahead, or (5) Synchronize, Adjust COA, and Look Ahead.

To provide the reader with a more complete understanding of the process and proof-of-principle product, a step-by-step description is provided. Similar in nature to the first project, this project also contains artifacts caused by the fact that the operational system activity is simulated and not real time.

Step 1: Establish a common "initial state" for both systems by assuming that the C4I Systems and the Simulation System were using a common data source.

The project presupposed that a common shared initialization data was used by the combatant commander when building his expectations using JWARS and in all of his C4I systems he employs in combat. This assumption eliminated the needed to do extensive mapping of simulation unit identifiers to real-world identifiers.

Step 2: Develop the expected COA.

The user develops the expected course of action using standard JWARS scenario development capabilities.

Step 3: Develop the actual COA.

This step is required to test the system - under actual operations this step would not be required.

The user creates an excursion from the expected COA developed in step 2 using standard JWARS scenario development capabilities. Excursions could be as minor as a slight positional change of one unit or as extreme as the original Operation Iraqi Freedom (OIF) Plan and the actual plan executed.

The procedure in Figure 9 explains how JWARS and the C4I-Gateway were used to produce C4I stimulation data representing the real-world:

- Create C4I scenario using JWARS. Execute C4I scenario with JWARS – collecting GCCS instruments. Export GCCS instruments to ASCII file.
- Execute utility to convert GCCS instruments to SCG replay module. Use Simulation C4I Gateway (SCG) “replay” module to playback GCCS instruments to generate C4I messages via its Standard C4I Message Processor (SCMP).
- Exchange track data (platforms & units) between GCCS-Ambassador and JWARS using the JWARS-GCCS Federation.
- (Optional) Use GCCS RECON segment to record TDBM transactions during SCG track message reporting. This allows for later playback of C4I scenario.

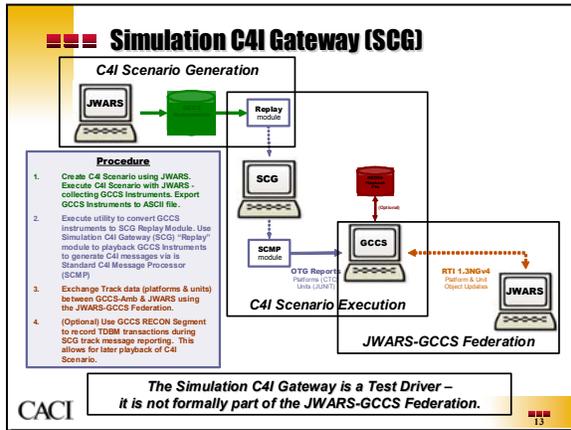


Figure 9: C4I Data Generation for Test

Step 4: Continuously compare “expected state” to “actual state” as the scenario unfolds.

A key activity, this step provided for the comparison of the JWARS “expected state” to the GCCS “actual state” of monitored entities using internal JWARS capabilities. Continuous track data from the TDBM, made possible by the Ambassador federate, was ingested by JWARS. JWARS contains internal components which enabled comparison of the ingested “actual state” track data to the internal “expected state” tracks to determine whether the battle was progressing as expected.

These internal JWARS components performed the comparison tasks much as intelligent agents, of which these components are precursors, would do. Although resident in JWARS and essentially

pre-programmed, the user was able to use them to establish the thresholds that define the significance of a track deviation. During this project, those thresholds were established once, at the beginning of the scenario execution, and maintained as constant throughout.

Step 5: Once JWARS determines that a track deviation is significant, it generates an alert which is transmitted to the GCCS operator.

The GCCS operator is given the option of clearing the alert or acting on it as described above.

Step 6: This step comprises the most valuable contribution that this project is making to the “state of the art.” If the GCCS operator chooses not to clear the alert, JWARS becomes available to him as a CoAA tool.

The JWARS capability for time compression makes it possible for the operator to “see” what would potentially happen should the current deviation continue, as well as test his various options and determine their probable long-term results.

Step 7: Based on the results of Steps 5 and 6, the operator determines the need for issuing an order which alters the expected course of action.

If an order is issued, JWARS must re-execute based on the altered scenario to reestablish the “expected state.” JWARS and GCCS are re-synchronized, and the “actual state” of operations in GCCS resumes.

5. Project 2: Initial Conclusions

The vision of providing a Situation Assessment tool to the combatant commander is technically feasible. Conclusions drawn from this phase of the research include:

- JWARS can be used to compare expected results to actual combat activity.
- JWARS can be used to spawn look-ahead replications in real-time to support what-if analysis.
- Use of JWARS and the C4I-Gateway for test case generation made test case development very simple. Test cases were developed and executed in minutes.
- Comparison of expected to actual values should not be embedded in JWARS. It is a unique process unto itself. It should be a stand-alone service or component.

- An initialization system common to C4I and simulation systems is essential.

6. Simulation-to-C4I Connectivity (FY04)

In fiscal year 2004, DMSO is sponsoring the integration of both FY03 projects, with enhancements to address the challenges of the Global Information Grid (GIG) (Figure 10). A few of the technical challenges to be addressed include the ability to rapidly locate and federate with other components or systems in the GIG (the architecture must support the ability to rapidly form connections between systems and components), overcoming obstacles that impede information interoperability across legacy systems including both Joint and Allied and the

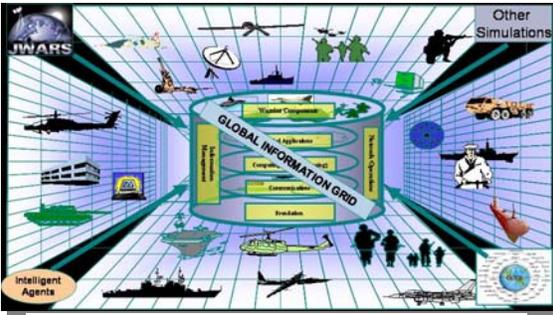


Figure 10: C4I-to-Simulation Interoperability in the GIG

ability of components and systems to automatically (or through a semi-automated fashion) locate and interact with other components. These challenges are further exacerbated by the fact that the U.S. is entering a new era of warfare (e.g., asymmetric) in which responding to crisis action situations will be the norm and speed of execution will be critical for achieving successful military operations.

Several key technologies are being examined to overcome these challenges. Web service technologies are being leveraged to help overcome the challenges associated with rapidly finding and interacting with other systems and components within the GIG. Web services technologies are rapidly maturing, and consequently are becoming a viable option to exploit the underlying backbone of the GIG.

With regard to information interoperability, we are envisioning to use the Command and Control Information Exchange Data Model (C2IEDM) as the common vocabulary for exchange. Systems and components will be required to map their

native information content to the C2IEDM to support the common understanding of concepts.

Having an ability to semi-automatically locate and interact with services will be a key capability, as it will be inefficient to have users in the loop on every web service transaction. Furthermore, systems and components in the GIG currently lack the intelligence to form complex queries for information search and access. We envision intelligent agents to support this functionality through their abilities to semi-autonomously coordinate with each other in support of system requirements for information.

The architecture being proposed to support the integration between JWARS, GCCS and intelligent agents is seen in Figure 11, in which the previously mentioned technologies will be applied to help address the integration and interoperability challenges envisioned in the GIG. The initialization data system (i.e., Army C4I Simulation Initialization System – ACSIS) will initialize both the C4I system (i.e., GCCS-M Track Database Manager) and Theater Battle Management Core System (TBMCS) as well as the simulation system (i.e., JWARS) with current Unit Order of Battle (UOB) such as organization and their relationships, including equipment and facilities. The tactical system (in our case will be an exercise replay through a C4I gateway) will deliver the actual data to the C4I system.

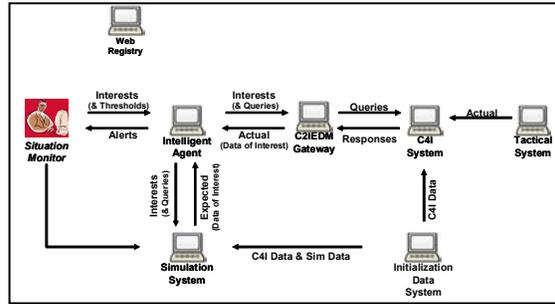


Figure 11: JWARS, GCCS, Intelligent Agent Federation via Web Services

The Situation Monitor is a graphical front end which permits a JWARS user to specify tracks of interest that need to be monitored, and the corresponding thresholds. This information will be communicated to intelligent agents that will make requests to the C4I system and Simulation to obtain the corresponding tracks for subsequent monitoring. These agents will compare both the real and simulated tracks in terms of the thresholds to generate alerts back to the situation

monitor terminal. The alerts may warrant the exploration of alternate CoAA.

The interface between the various components will be accomplished through web service technologies. These include the Universal Description and Discovery Interface (UDDI), Web Services Definition Language (WSDL) and Simple Object Access Protocol (SOAP).

UDDI is a framework that defines XML-based registries in which businesses can upload information about themselves and the services they offer. An XML-based registry contains names of organizations, services provided, and descriptions about service capabilities. XML registries based on the UDDI specification provide common areas through which systems can advertise themselves and their Web Services.

WSDL is an XML vocabulary standard created just for Web Services. It allows developers to describe Web Services and their capabilities, in a standard manner. WSDL helps to expose the Web Services of various businesses for public access.

The SOAP is an XML vocabulary standard to enable programs on separate computers to interact across any network. SOAP is a simple markup language for describing messages between applications. SOAP provides a way for developers to integrate applications and business processes across the Web or an intranet, by providing the platform and programming language independence needed to create the business integration of Web Services.

Each of the components in Figure 11 will register their services with the UDDI registry (e.g., the WSDL specification). Each component that requires information from other components will perform a look-up in the UDDI registry and obtain the WSDL file, from where a determination can be made as to where the service resides and how to invoke it.

The C2IEDM gateway will map information to the C2IEDM vocabulary. The C2IEDM was developed under the auspices of the Multilateral Interoperability Programme (MIP) [8]. The MIP is comprised of volunteers from 27 nations, whose goal is to foster international interoperability between multi-national Command and Control (C2) systems.

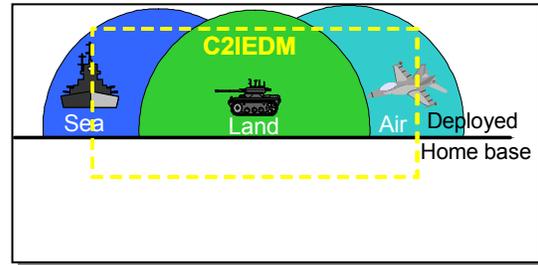


Figure 12: Scope of the C2IEDM

The C2IEDM is a generic model that can be extended as needed to suit evolving military requirements (serving as a “hub”; as such, it was originally named the “Generic Hub”, and evolved to Land C2IEDM and eventually C2IEDM to capture other areas including Air and Surface – Figure 12). The C2IEDM is

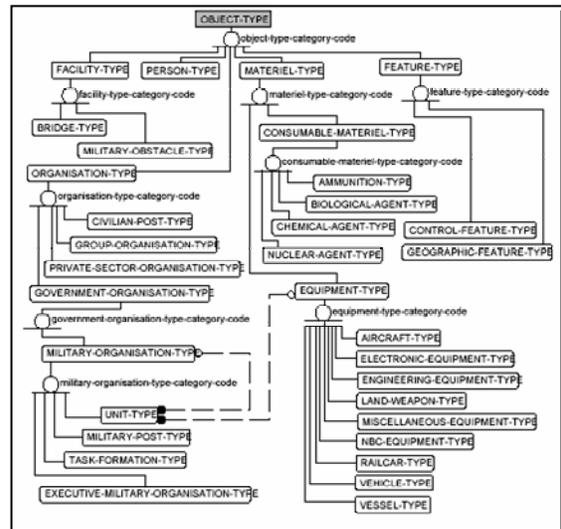


Figure 13: C2IEDM

comprised of a conceptual data model, logical data model and physical data model. The conceptual data model represents generalized concepts, while the logical data model represents further details associated with the conceptual data model. The physical data model defines the physical data storage schema. The main purpose of the C2IEDM is to represent Information Exchange Requirements (IERs) between C2 systems. A very small portion of the data model is seen Figure 13.

The Intelligent Agents that were developed for Project One will be enhanced to support additional monitoring capabilities. The enhancements will include the ability to monitor tracks (both known and unknown) that enter

regions of interest as well as monitoring the changes made to the Air Tasking Orders (ATOs). Initially the agents will compare the changes made to the ATO within the TBMCS, and compare those to ATO in JWARS and alert the user when the plan associated with a specific air track of interest has been modified in some way. Later we plan to monitor the progression of the ATO and provide alerts, again based on user defined thresholds. The ATOs will be represented in eXtensible Battle Management Language (XBML) format and will also be mapped to the C2IEDM.

7. Future Direction and Research Challenges

We have concentrated strictly on the plan monitoring agents. However, there is an opportunity to conduct significant research in the area of plan decomposition agents. In our experimentation, we have placed the burden on the user to select tracks of interest, for which thresholds need to be set, that should be monitored. Through plan decomposition, we would like to be able to identify critical events and relationships, thereby permitting an intelligent agent to monitor the plan in terms of those critical events. We are examining Natural Language Processing (NLP) techniques coupled with sublanguage ontologies to extract semantic relationships from free text documents such as Operational Orders (OPORDs). Another promising area is the use of the Battle Management Language (BML), which provides some of the infrastructure necessary for intelligent agents to reason with OPORDS.

An emerging area for future research is in agent teamwork [4,5]. To realize the full potential of distributed multi-agent systems, the agents will need to cooperate as part of teams to help the operators (acting as their proxies) achieve their goals. For example, teams of distributed software agents with different goals may need to coordinate to decompose and relate multiple plans to determine critical points, which can be passed to a team of agents that are responsible for monitoring.

There are many challenges in realizing such an ambitious effort as a prototype representation of the GIG from an M&S perspective, such as the integration of large legacy systems via the application of new technology (web services) which, although maturing at a fast pace, are still evolving. Service-oriented architectures hold the

promise of enabling such a grand vision as that of the GIG. However, we are still faced with the challenge of integrating large legacy C4I systems and simulations that are fairly stable with new technology that is not quite stable. This, by itself, is a tremendous challenge!

We are also faced with the task of integrating intelligent agent technology with a web-services computing paradigm. Although there has been considerable attention devoted to the field of multi-agent systems such as agent communication languages [12], standards, etc, there has not been significant research into how multi-agent systems will operate in a web-services world, primarily because web services technologies have emerged only recently. One of the key issues associated with deploying multi-agent systems in a web-services environment includes the fact that agents require messaging for communication/coordination; it is unclear whether there will be performance issues between agents that need to communicate via web-service protocols.

Another challenging problem to be encountered in the GIG will include interoperability of systems between Communities of Interest (COI). It is envisioned that heterogeneous agents will operate in the GIG, with different ontological representations. A key challenge that is certain to arise will be the ability of agents (which understand one ontology) to communicate with other agents (having a totally different ontological representation). Moreover, agents may be required to compose services, and therefore, it may be necessary to endow these agents with advanced reasoning capabilities. However, there will be a limit in terms of how much an agent is able to practically reason with, therefore, additional solutions may be adopted, such as human-agent cooperation (mixed-initiative architectures) or the employment of machine learning techniques.

Lastly, will web-service based applications be required to interact with other technologies such as Peer-to-Peer (P2P) architectures [11]? The big question is “*will there be a single technology that provides the infrastructure for the GIG, or will there be several complementary technologies?*” If the latter is true, how to bridge the applications that rely on different technologies?

Our Simulation-to-C4I FY04 connectivity program will afford us the opportunity to begin to investigate a few of these issues, including integration of large scale legacy systems with new technology, multi-agent system operation within a web-services environment and the complex nature of mappings between agent ontologies, web services and the C2IEDM. The other areas will be investigated in later years as DISA charts out the vision for the GIG and as competing/complementary technologies mature.

8. References

- [1] Friedman-Hill E. *JESS in Action, Rule-Based Systems in Java*. Manning, 2003.
- [2] Mittu, R., Furness, Z., Emami, R., "Agent-mediated Interface Between C4I and Simulation". In *Proceedings of 2003 Spring Interoperability Workshop (SIW)*. Orlando, FL, 2003.
- [3] Mittu, R., Walters, J., Abramson, M., "Improving Simulation Analysis through Interfaces to C4I systems and Simulations", In *Proceedings of the 2004 Spring Simulation Interoperability Workshop*, Crystal City, VA.
- [4] Rao A. S. and Georgeff M. P. "BDI agents: from Theory to Practice". In *Proceedings of the First Intl. Conference on Multiagent Systems*, San Francisco, 1995.
- [5] Tambe, M. "Agent Architectures for Flexible, Practical Teamwork". In proceedings of the *National Conference on Artificial Intelligence*, 1997.
- [7] Global Information Grid
[\[http://www.disa.mil/ns/gig.html\]](http://www.disa.mil/ns/gig.html)
- [8] C2IEDM and MIP
<http://www.mip-site.org/>
- [9] UDDI, WSDL, SOAP
<http://www.w3c.org>
- [10] Semantic Web Services
<http://swws.semanticweb.org>
- [11] Peer-to-Peer (P2P) Computing
<http://www.openp2p.com>
- [12] Foundation for Intelligent Physical Agents
<http://www.fipa.org>

9. Acknowledgements

The authors would like to thank the Defense Modeling and Simulation Office (DMSO) for providing the research funds for this effort.

Author Biographies

RANJEEV MITTU is the Head of the Intelligent Decision Support Section at NRL.

Mr. Mittu has fifteen years experience in designing and developing decision support systems. He has worked on programs within the Strategic Defense Initiative (SDI), Anti-Submarine Warfare (ASW) as well as Force Level Planning for the Naval Strike community. He has been involved in numerous DARPA, DMSO and ONR efforts dealing with the development of intelligent multi-agent systems. His research interests are in the areas of artificial intelligence, multi-agent systems and modeling and simulation. He has a B.S degree in EE from the University of Maryland and an M.S. degree in EE from The Johns Hopkins University.

VIRGINIA T. DOBEY is a former Navy Commander (Special Duty—Oceanography) who has been involved in environmental software development, testing, and implementation since 1975. Instrumental in initiating the incorporation of environmental data into the Department of Defense data standardization program, she has been involved in U.S. and international data standardization efforts since 1993. A leader in environmental data standards efforts for both operational and M&S users, she is an expert on environmental data quality and VV&A. A member of the DMSO Environmental Representation Technology Area team since 1998, Ms. DobeY holds a Master of Science in Meteorology and Oceanography degree from the Naval Postgraduate School.

JERRY DEPASQUALE is currently the Technical Lead of the JWARS-GCCS-Agents Project, Technical Lead of Joint Warfare System (JWARS), and the Technical Manager of the CACI Simulation and Modeling Division. He retired from the United States Marine Corps with the rank of Major in 1995. His key Marine Corps experiences include: service in numerous enlisted and commissioned artillery billets, service as the Information Systems Management Officer of the 3d Marine Division - where he developed and deployed a prototype battle center network, service as a liaison officer to a South Korean Infantry Regiment, service as an analyst in the C4I2 Division of Headquarters Marine Corps, service as the software architect of the Architecture and Standards Division of the Marine Corps Combat Development Center. He has a BS in Nuclear Engineering from Purdue University and a MS in Computer Science from the Naval Post Graduate School.