IVSS-2003-MAS-7

CAT/RF Simulation Lessons Learned

Christopher Mocnik
Vetronics Technology Area, RDECOM TARDEC

Tim Lee
DCS Corporation

ABSTRACT

The Vetronics Technology Area of The U.S. Army Tank-Automotive Research Development and Engineering Center (TARDEC) and DCS Corporation developed a re-configurable Unmanned Ground Vehicle (UGV) simulation for the Crew integration and Automation Test bed (CAT) and Robotics Follower (RF) Advanced Technology Demonstration (ATD) experiments. This simulation was developed as a component of the Embedded Simulation System (ESS) of the CAT architecture. The CAT/RF was chosen by the Future Combat System's (FCS) Lead Systems Integrator (LSI) as a surrogate to show the functionality of the Control Vehicle (CV)/Armed Reconnaissance Vehicle (ARV) concept demonstrated in the Unmanned Combat Demonstration (UCD) experiments. The UCD design handled any combination of Javelin-like missile system, Objective Crew Served Weapon (OCSW), and Reconnaissance, Surveillance, and Target Acquisition (RSTA) sensors on the ARVs. The ESS development team faced numerous challenges in simulating multiple vehicles with varying sensor and weapons payloads, controlled by multiple operators. Despite an accelerated schedule, the ESS team was able to support the UCD and ATD experiments conducted between December 2002 and April 2003.

INTRODUCTION

This paper describes the effort undertaken by the Embedded Simulation Team of TARDEC and DCS Corporation to design, develop, integrate, and test the ESS software and hardware to support both the CAT/RF ATDs and the UCD experiments with the LSI. The paper will present brief background information on prior ESS activities in the Vetronics Technology Area of TARDEC and the activities leading up to the current CAT/RF and UCD experiments. It will then discuss specific technical and programmatic challenges faced during CAT/RF/UCD development in key areas, and describe the final implementation used. Potential future enhancements will also be considered where feasible.

BACKGROUND

In the early to mid 90s The Vetronics Technology Area conducted the Crewman's Associate (CA) ATD. The goal of CA was to show that all of the tasks currently performed by the four-crew members of an Abrams main battle tank could be performed with equal or greater timeliness and precision by a reduced crew of two or three. A smaller crew would mean a smaller, lighter, more transportable and supportable vehicle. An improved Soldier-Machine-Interface (SMI) was created and certain levels of automation were assumed. Experiments were conducted with a two-person and three-person man-in-the-loop static simulator that verified this two and three man notion.

The Vetronics Technology Test bed (VTT) was the follow-on effort to the CA ATD. Developed under the Inter-Vehicle Electronics Suite (IVES) Science and Technology Objective (STO), the main goal of the VTT was to demonstrate the capability of one crewmember to perform the functions of both the vehicle Commander and Driver. The advanced crew stations designed under the CA ATD were reconstructed using rugged, real-time hardware and software modules and integrated into an actual ground combat host vehicle. The VTT demonstration took place while operating over militarily significant terrain and while performing a militarily significant mission. In addition, embedded simulation hardware and software was used to create realistic operating and training scenarios for the crewmembers. The ESS provided both the ability to train in-vehicle, and provided virtual targets, weapons control, and sensors during operational exercises. It also provided a battlefield visualization capability enabling the user to move his eye point into and around the virtual battlefield.

The VTT vehicle itself was based on a modified M2A0 Bradley Fighting Vehicle hull that was refitted with drive by wire technology and the two-crew stations. The Bradley turret was removed to give it a lower profile, as was (and is) the vision for most all future armored vehicles. The M2A0 also afforded ample interior volume

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
11 JUN 2003	**Journal Article**	**11-05-2003 to 10-06-2003**

4. TITLE AND SUBTITLE
CAT/RF Simulation Lessons Learned

5a. CONTRACT NUMBER
5b. GRANT NUMBER
5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Christopher Mocnik; Tim Lee

5d. PROJECT NUMBER
5e. TASK NUMBER
5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army TARDEC,6501 East Eleven Mile Rd,Warren,Mi,48397-5000

8. PERFORMING ORGANIZATION REPORT NUMBER
#13857

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army TARDEC, 6501 East Eleven Mile Rd, Warren, Mi, 48397-5000

10. SPONSOR/MONITOR'S ACRONYM(S)
TARDEC

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
#13857

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The Vetronics Technology Area of The U.S. Army Tank-Automotive Research Development and Engineering Center (TARDEC) and DCS Corporation developed a reconfigurable Unmanned Ground Vehicle (UGV) simulation for the Crew integration and Automation Test bed (CAT) and Robotics Follower (RF) Advanced Technology Demonstration (ATD) experiments. This simulation was developed as a component of the Embedded Simulation System (ESS) of the CAT architecture. The CAT/RF was chosen by the Future Combat System?s (FCS) Lead Systems Integrator (LSI) as a surrogate to show the functionality of the Control Vehicle (CV)/Armed Reconnaissance Vehicle (ARV) concept demonstrated in the Unmanned Combat Demonstration (UCD) experiments. The UCD design handled any combination of Javelin-like missile system, Objective Crew Served Weapon (OCSW), and Reconnaissance, Surveillance, and Target Acquisition (RSTA) sensors on the ARVs. The ESS development team faced numerous challenges in simulating multiple vehicles with varying sensor and weapons payloads, controlled by multiple operators. Despite an accelerated schedule, the ESS team was able to support the UCD and ATD experiments conducted between December 2002 and April 2003.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Public Release**	**11**	

to accommodate the crew stations, ESS, radios, and other equipment.

CURRENT PROJECT

The Vetronics Technology Area is currently developing the CAT and RF ATDs. The CAT extends the VTT crew station design to include the FCS fight, carrier, and scout missions. In addition to VTT capabilities, the CAT ATD also supports control of robotic assets both forward deployed and follower, and introduces cognitive and task decision aids as well as an improved ESS. The vehicle selected for the CAT ATD is a Stryker Infantry Carrier Vehicle (ICV). Like the Bradley it also has enough interior space to accommodate crew stations and equipment. Additionally, it weighs less than 20 tons which is more in line with FCS goals for dimensions and weight. The goal of the RF ATD is to advance autonomous mobility technology in a number of areas such as obstacle detection, path following, and road/lane following. The RF ATD also makes use of a Stryker ICV and an eXperimental Unmanned Vehicle (XUV) from the Demo III program. Collectively, the combination of CAT and RF ATDs developed with GDLS is called the Vetronics Technology Integration (VTI) contract.

As with the VTT, the Embedded Simulation Team's goal for the CAT was to provide both a training and operational virtual capability. In a static Systems Integration Lab (SIL), the ESS would provide all visual models, out-the-window views, weapon and sensor functions, as well as mobility for the CAT vehicle. While in the field, the vehicle would be operating under it's own mobility and using it's own indirect vision cameras, but the ESS would still be required to provide virtual weapons, sensors, and targets.

When Boeing was selected as the LSI for FCS they performed an analysis of available resources within the Government that they might be able to leverage. The two FCS class vehicles and the XUV used by Vetronics were seen as valuable assets. Using these, the LSI would be able to demonstrate their evolving unmanned vehicle concepts. The ES Team supported the LSI UCD by providing the simulation systems for both a virtual demonstration that took place in the SIL at TARDEC, and a live maneuver portion that took place in the field at Ft. Bliss Texas. The UCD concept included two ARVs and one CV. One of Vetronics' Strykers and the XUV acted as surrogate ARVs while the other Vetronics Stryker acted as the CV in the LSI's concept. Operators seated at the CAT crew stations controlled the ARVs and their payloads. For the field demos the crew stations were placed in the back of the CV/Stryker. The operators would have the ability to control one or more ARVs as the scenario prescribed. They could manually steer the ARVs (called Tele-oping) or send them mission plans to control their mobility. The modeling and control of the virtual ARVs and their payloads were significant activities for the ES Team.

ENGINEERING DESIGN PROCESS

Adherence to a design process is an important part of any engineering project. The Embedded Simulation team has typically used a modified version of the waterfall model for software engineering shown in Figure 1. The model is often appropriate for rapid engineering as it is easy to understand, implement, and in general, universally understood.

Input Product	Process	Output Product
Communicated Requirements	Requirements Engineering	Requirements Specification Document
Requirements Specification Document	Design	Design Specification Document
Design Specification Document	Programming	Executable Software Modules
Executable Software Modules	Integration	Integrated Software Product
Integrated Software Product	Delivery	Delivered Software Product
Delivered Software Product	Maintenance	Changed Requirements

Figure 1: Waterfall Software Engineering Process

The model was used cyclically within a series of software drops. Five drops were originally planned for software development activities. Each drop would add more functionality over the previous drop, until all requirements for the ESS were met with completion of the last drop. Divergences from this process in several areas lead to numerous problems over the course of the project. The primary problem areas for the ES Team were in communication of requirements, and integration of software.

In any engineering design process the creation of a requirements specification document is probably the single most important step. It not only documents the needs of the customer, but the path forward for the developers during design and programming. Incorrectly specified requirements can lead to rework, functionality that doesn't meet the customer's needs, or software that simply doesn't work. Originally, the CAT program was to be a two-year design effort. This was reduced to one year in order to meet the FCS milestone B decision date. Not only was the development time now cut in half, but a second set of requirements needed to be specified for the UCD. Adding to this was the instability in the FCS Unit of Action (UA) force structure and the LSI's concept objectives, and the uncertainty of physical payload availability to place on the Stryker vehicles and XUV. Because of the reduced development time and instability of the experiment configuration, communicated requirements were largely in flux making requirements

specification very difficult at best. This lead to problems later on as code changes were constantly being made to accommodate evolving requirements.

The ES Team also experienced difficulty during software integration. Typically, ESS software modules were designed and tested in the ES Lab at TARDEC. This code was then taken to General Dynamics Robotic Systems (GDRS) for integration and test on the target hardware. Because of inadequately specified requirements, it was often the case that code developed by GDRS and that developed by the ES Team would not interact cleanly at integration time. This would lead to down time, and functionality intended for that drop being pushed to a later drop. Because progress at integration time was reduced thorough regression testing was not always possible, leaving bugs in the system during demo time. In the future it is recommended that both system requirements and individual drop requirements be base-lined or better defined in advance.

SIMULATION SOFTWARE ARCHITECTURE AND CODE REUSE

If it's feasible to do so, reuse of existing software is of great benefit. Code reuse saves both time and money as it can take significantly less time to port and modify well-understood existing code instead of developing new code from scratch. It increases reliability and reduces risk as previously used components should have already been tested and debugged. If the code is very modular in design, and non-system specific, the task of porting code will be made easier.

Because of the extremely tight schedule and creeping requirements, the Embedded Simulation Team decided to reuse as much of the code from the ESS on the VTT as was possible. Developing all new code from scratch with the resources and time available would have only increased risks. The ES Team successfully re-used the source code from the VTT adding new processes to capture new functionality where needed. The general configuration of the CAT software processes is shown in Figure 2.

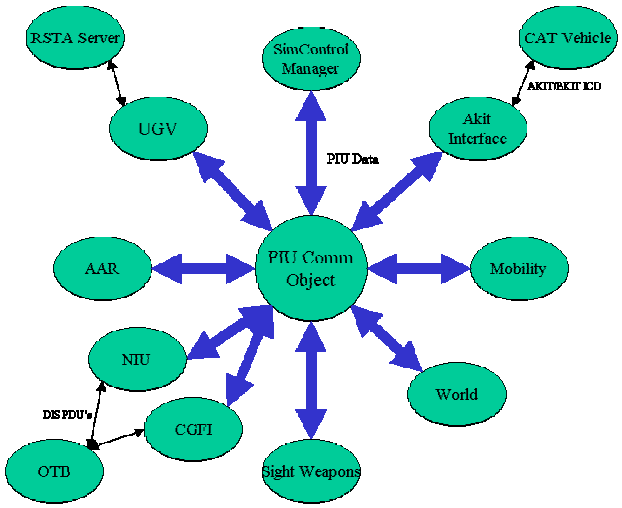

Figure 2: Software Processes Overview

The VTT and CAT (to a lesser extent) utilized a process-oriented design. Specific vehicle functions were isolated and similarly named processes created to implement that functionality. CAT Functionality included own ship vehicle mobility, graphics generation, simulation control, after action review and playback, network interfacing, sight and weapon control, and an interface to the vehicle. All new for UCD was UGV and RSTA control. Data sharing between the processes occurred via PIU Comm object described in a later section of this paper. The CAT/UCD code was the beginning of a departure from the process oriented code of the VTT to an Object Oriented (OO) architecture that used domain named classes.

Although this was a good approach, much of the existing VTT code had to be modified. The Own ship mobility process had to be completely changed as a new vehicle model was being used. This is described in a later section of this document. The A-Kit interface had to be drastically changed as well to account for all the new communication required for the RSTA and UGV processes. Sight/Weapons had to be significantly changed to handle new weapon types and new sensors not previously used in VTT. Some changes were necessary to the visualization manager (World), Network Interface Unit (NIU) and the After Action Review (AAR) data logger. In total the amount of new code developed, and change to existing VTT code ended up being as much as an entirely new development effort.

Another significant challenge was in producing multi-vehicle/crew station software. Though somewhat modular in design, the VTT code was largely system specific. The ESS was tightly coupled to the VTT functional design. CAT implementation was to be free from this constraint. Any function from any station for any vehicle was the desired functionality. See Figure 3.

Given the reuse of the VTT software and the amount of time available to make coding changes for the two programs CAT and UCD, this goal was only partially achieved. The VTT code was expanded upon to allow more functionality than before, but it was not a truly open ended design. Part of the 04 and beyond effort will be to re-architect the ESS code to truly allow a full set of services anywhere anytime.

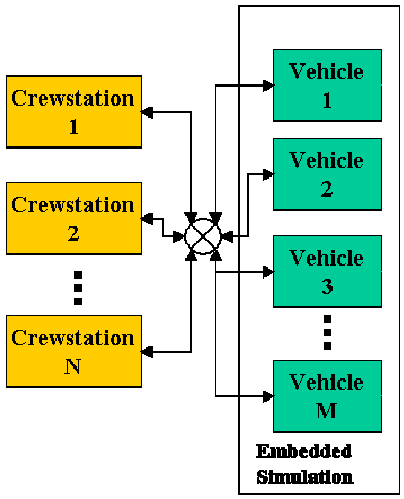

Figure 3: Multi-Vehicle/Crew Station Simulation

UGV SIMULATION

A new ES Team development effort for the CAT and specifically geared for support of UCD was the UGV Process. A high level overview of the UGV process is shown in Figure 4. The main function of the UGV process is to instantiate a UGV object for each UGV present in the simulation, and to pass through data to/from the vehicle via PIU Comm object. The UGV object is the "brains" of the UGV process. The UGV object interprets the mission plan and controls what data is sent to the Platform object and when, essentially controlling the UGV simulation. The Plan object is essentially a mission plan parser that sends mission related data and commands to the UGV object. The UGV Platform object starts a thread that instantiates multiple objects that handle payload and physical features of each UGV such as mobility, RSTA, weapons, etc.

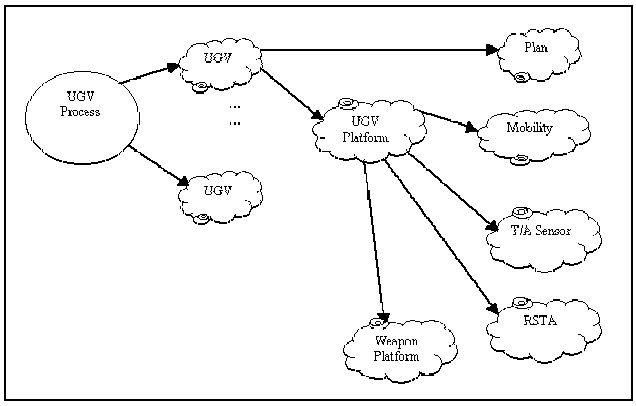

Figure 4: UGV Process Implementation

Successful implementation of the UGV process was a challenge in several ways: communicating with the vehicle software; and executing a live virtual ARV mix. Control of an ARV ultimately is performed from the SMI at the CAT crew station. From here the soldier composes his mission plan for the ARV, performs RSTA scans, or tele-operates the ARV. In any case, button presses and other control input from the SMI are composed into a message set that is sent to the ESS via the A-Kit. Much of this message content was described under the Demo III program from which it was being reused; however, new functionality was being added that didn't exist in Demo III such as missile weapons payloads. Also contributing to the problem was requirement creep. Often times, messages were being defined at software integration drops. An early effort to define message content in advance was made but ultimately rejected as flexibility was preferred given the flux in requirements and potential for quick on the fly changes. In retrospect this was a mistake. Because message content and format was not fixed, both ESS code and GDRS code was developed based on a set of assumptions. At integration time this became an issue as neither party had code fully compatible with the other's assumptions. A possible solution to this problem is to define a common robotics interface in advance, and to design simulation application code around the message content. Future exercises will certainly add control of Unmanned Air Vehicles (UAVs) to the mix forcing a common design philosophy up front.

Another difficulty to overcome during the project was the implementation of a live and virtual UGV mixture. In the SIL environment everything was virtual, both vehicle and payload. UGV position was known as it was being calculated within the ESS UGV process. Every time a RSTA or TA sensor service was requested, the ESS knew exactly where to place the eye point for the requested service. In the field this was not the case. UGV mobility was not engaged as the real vehicle asset was in use and running under it's own mobility. But, the

RSTA and sensor payloads were still virtual. Actual vehicle location data was sent back to the ESS in position updates that occurred roughly every two seconds at the fastest, or on vehicle events such as RSTA scans, tele-op commands etc. As a result, the position from which to perform a RSTA or TA operation had the potential to be lagged by several seconds depending on the service being requested. This was solved by placing certain constraints on the user such as not moving while using a TA sensor, or performing any sensor functions, to account for the potential lag. A possible solution is to use dead reckoning algorithms for determining position when actual position is not available. This would entail calculating the current eye point given the vehicles last reported heading and speed.

OWN SHIP VEHICLE PHILOSOPHY

Another issue in developing the ESS own ship vehicle behavior was the notion of the CV or CAT own ship vehicle being considered an ARV. The ES Team did not consider the CAT vehicle to be an ARV, and did not expect it to be controlled as such. The GDRS team on the other hand, considered the own ship vehicle to be controlled just as any other robot. It made sense for each development team to take this approach, but did cause more integration issues. GDRS did not want to make the status message it sent to the CAT a unique message, and wanted to keep the format and content currently used in the UGV message. It was a difficult task for them to be able to capture all the data needed for the CAT and format it differently. The ES team on the other hand did not view the CAT as just another UGV. The CAT vehicle had functionality that would never exist natively in a UGV such as battlefield visualization, or lethality/sensor visualization control, or data for interfacing with a SAF, etc. Control data for these types of features could not be included in a UGV message. Ultimately, compromises had to be made by both parties to accommodate program needs. In retrospect, a common understanding of what an own ship vehicle is and how it's controlled should have been developed up front. For future efforts, a common design will need to be agreed upon.

RSTA SIMULATION

The VTT vehicle simulation involved a low fidelity simulation of an advanced Automated Target Recognition (ATR) system. As such, the interfaces and operational performance of the simulated ATR was limited from the perspective of the current technology. However, in the CAT ATD, a current state of technology RSTA sensor suite needed to be modeled with as much fidelity as could be implemented. Further, the ESS needed to accommodate multiple instances of the RSTA simulation due to the multiple vehicle experiment configuration.

Since the RSTA being modeled was normally not used as a live video feed sensor, but rather as a static imaging sensor, ESS design allocated a single video channel to serve all instances of RSTA simulation. As depicted in Figure 5, a client-server distributed architecture was employed to enable a single RSTA simulation server to provide the RSTA interface data for multiple RSTA simulation clients.

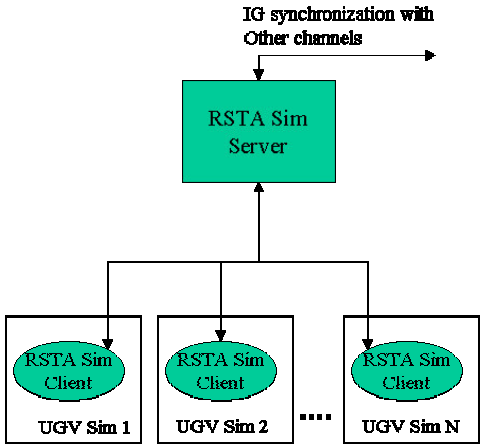

Figure 5: RSTA Simulation Approach

Also, an algorithm was devised to mimic the target discrimination confidence levels of current ATRs. This algorithm was not based on any scientific validation. However it was an educated guess at the target discrimination probabilities as a function of measurable parameters in the simulated world. Using methods described by Shumaker, et al [7], and the venerated observations by Johnson [4], the probability curves as functions of pixels on target were devised as shown in Figure 6. The implementation plan for this approach was to determine the number of pixels on target (with obscuration) via a number of costly Image Generator (IG) manipulations. This was certainly a simplistic view of ATR performance since the orientations, range , and other factors were not considered. However, given the difficulty in attempting to predict ATR performance as discussed by Ratches et al [6], the approach taken for ESS was a reasonable compromise.

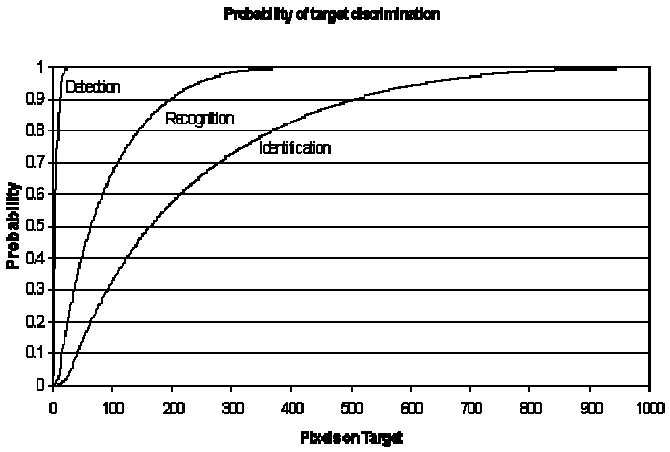

Figure 6: ATR Modeling

Despite the effort expended in designing a higher fidelity RSTA simulation, actual implementation and execution suffered from limitations unforeseen. First, the sheer size of the synthetic environment database used in the CAT experiments caused the IG to spend large amount of time loading terrain and textures every time the view point changed from one vehicle to another. Since the RSTA simulation was required to build representations of sensor scan "mosaic" images, multiple view angle screen shots from each vehicle location was needed. This operation meant frequent replacement of terrain and textures from cache, or even from disk, which translated to large delays. These delays were acceptable when simulating the scan operation of a single RSTA, since the real RSTA also took comparable amount of time to complete a scan. However, when multiple RSTA scan operations were requested simultaneously, the ensuing delays were unrealistically large. In addition, the complexity and overhead of the RSTA simulation server increased exponentially due to the management of mutual exclusion between the dynamic server threads for each concurrent instance of RSTA scan. Thus, the resources of the RSTA simulation server were fully taxed and unable to provide the basic RSTA functionality in a timely manner when multiple RSTA operations were requested. This in turn, prevented the implementation of the ATR performance modeling that was planned, since that would have added even more demands on the already busy RSTA simulation server IG.

The RSTA simulation for the next generation ESS may attempt to incorporate multiple video channels for the RSTA simulation server. Ideally a simulation video channel would be dedicated to each instance of a simulated RSTA. However, that could make the ESS footprint unacceptably large for the "embedded" environment. Therefore, some mechanism to share existing channels for RSTA simulation may warrant some investigation. Adding more RSTA simulation channels is also an option, but it is the less desirable one. Further, a different approach to the RSTA ATR performance modeling may be needed. One possible option is to utilize multiple Line-Of-Sight (LOS) checks on the target rather than counting pixels on the target to derive some measurement of level of "visibility." Since the LOS checks and range checks in simulation world does not require reloading of the scenery, this may be more feasible.

WHEELED VEHICLE SIMULATION

For the UCD virtual experiments it was necessary to model the mobility characteristics of both the Stryker ICV and the LSI's ARV concept. Previously under the VTT program, the ES Team built a simple tracked vehicle model for the Bradley fighting vehicle. The mobility characteristics of a wheeled vehicle are significantly different than that of a tracked vehicle. Though a high fidelity model wasn't needed, a reasonable facsimile of the Stryker mobility was desired in order to give the soldier a reasonable level of consistency moving from the SIL to the physical vehicle. Additionally, an ARV/UGV mobility model was needed as well. Because the ES team doesn't have particular expertise in mobility, and given the short schedule, it was decided to approach the TARDEC Ground Vehicle Simulation Lab (GVSL) on availability of models. A Stryker 8x8 for the own ship and a suitable simple dynamics model for the UGV application were available and obtained from the GVSL. Each model type utilized the GVSL API for exchanging the parametric data with the ES Team application code. This data was used by the World process to update the vehicle's position and orientation on the virtual terrain. See Figure 7: CV & UGV Mobility Models. However, problems were encountered in the usage of the UGV mobility model.

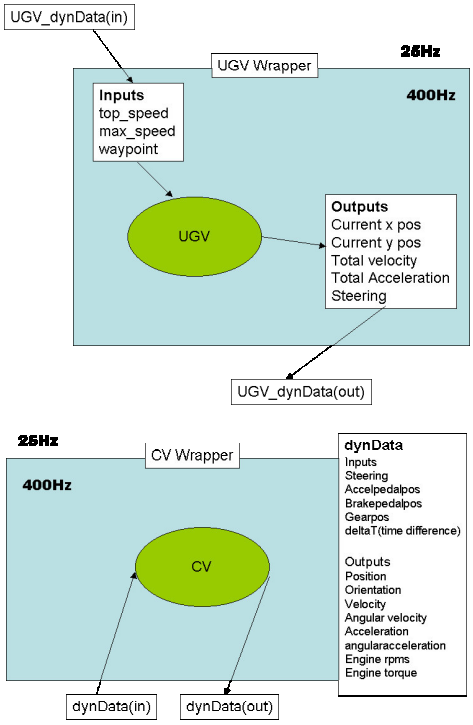

Figure 7: CV & UGV Mobility Models

Periodically during an exercise, the UGV visual model would "warp" to a distant location on the terrain database. The World process would occasionally receive faulty x/y UGV position data from the UGV process and place the visual model at this faulty location during it's next draw cycle. Checks were put in place to catch the occurrences by comparing current reported x/y location to previous x/y location and looking for large deviations. However, faulty location data was still occasionally being sent.

The cause of the problem may be in the conversion of data types being passed to the GVSL mobility model. Data types were converted from double to float and passed to the model. In the conversion process, decimal places are truncated. These missing decimal places would in time culminate in an illegal division operation within the model, causing the erroneous data.

This situation could have been avoided through better system testing and interface control. The models were tested in the lab prior to integration but no problems were found. It wasn't until extended operations were being conducted with the target scenarios that this issue was noted. Since mission scenarios were not available until near demo time, the issue was not discovered until soldier runs were being conducted as objective scenarios typically took hours to conduct, allowing more time for bugs to surface. For future efforts, it is desirable to test at length on the target scenarios as far in advance as possible. Also, greater care should be taken in defining or adhering to the interfaces for externally supplied software components.

VISUALIZATION APPROACH

The CAT ESS visualization approach was largely based on the architecture utilized in the VTT. The CAT ESS utilized the Carmel Applied Technology, Inc. (CATI) X-IG™ synthetic environment visualizer that was also used in the VTT. This IG provides the scalability and the diversity of functionality the program required. Although there were limitations, particularly in the maximum size of the terrain database it could handle, this IG was chosen to be the IG for CAT because of the familiarity with the product given the rapid development schedule.

In the CAT ESS, the synthetic environment view channels were synchronized to a single "world awareness" via a "master" channel that was updated by a visualization manager application in ESS as shown in the top picture of Figure 8. This channel arrangement was adequate in VTT since the VTT crew stations shared a single visualization "mode." When the VTT system switched from normal operation to battlefield visualization, both crew stations switched from the "real world state" data to the "perceived state" data (Situational Awareness (SA) knowledge acquired via C2 communications). However, the CAT crew stations required the ability to visualize either SA data or real world data independent of each other. Implementation of this capability in the ESS required a decentralized IG channel management as depicted in the lower half of Figure 8. This IG channel management approach was a radical departure from the existing ESS architecture; and it would have required an extensive overhaul of the ESS software architecture to implement. Due to schedule limitations, the decentralized IG management was not implemented for the CAT 2002 – 2003 experiments. Re-architecting of the software is being planned for the next phase in the CAT program.

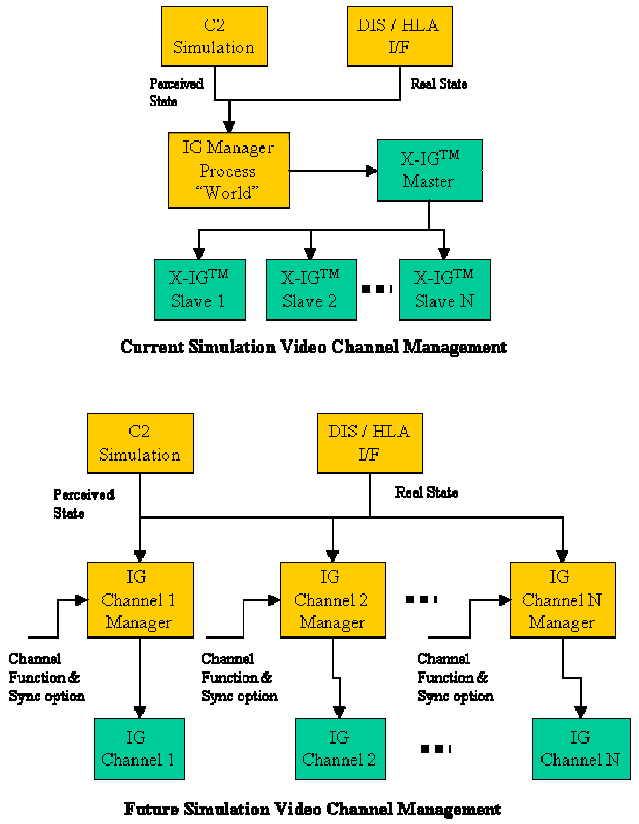

Figure 8: Image Generation Architecture

The redesign of IG management will be key to the success of the next generation of ESS. The network centric paradigm of FCS will demand more and more flexibility from the ESS to support the multiplexed controller to vehicle connectivity depicted in Figure 3. The centralized architecture of the single vehicle simulations has been patched and hacked to work thus far for the multi-vehicle system. However, the ESS needs to be redesigned with a decentralized architecture, including a distributed IG management, to continue stable integration with the evolving FCS vehicle programs. Otherwise, the relevance and utility of an ESS as a tool for Simulation Based Acquisition (SBA), and as an operational tool for FCS will be difficult to maintain.

SCENARIO DEVELOPMENT

For both the CAT and UCD experiments militarily significant scenarios were needed. Using scenarios that resemble a true-to-life battlefield situation gives experiments a higher degree of validity.

Scenario development for the CAT was a several step procedure. First, CAT employed a Subject Matter Expert (SME) to develop the scenarios (or vignettes) on paper. An SME can generate scenarios with the look and feel of something likely to be encountered in the real world as they have experience in that environment. Initially, the GDLS SMEs produced 11 scenarios for the CAT program, developed with and approved by The Unit of Action Maneuver Battle Lab (UAMBL). This was later reduced to one master scenario that could be tailored to the needs of the individual CAT or UCD experiment. A scenario can also be constrained by the size of the digital terrain database employed. Ultimately, the amount of space available to maneuver on will affect the mission as this limits the number of entities that can be used. The SME can also take this into consideration when developing the vignettes.

The next step was to construct the scenarios in a Semi Automated Forces (SAF) program on a Correlated Terrain Data Base (CTDB). For the CAT this was the OneSAF Testbed (OTB). OTB controlled the actions of the entities not under direct control of the ESS application code, telling them when to move or shoot for example. OTB communicates with the vehicle and ESS application code via a Distributed Interactive Simulation (DIS) protocol.

Lastly, an iterative process of running through the scenario on the visual terrain database was conducted to make sure the scenario played out as intended. Typically the CTDB is a lower resolution than the terrain database, so entities may behave slightly different than intended as they have visibility on the flatter CTDB that they wouldn't have in the visual terrain database. This enhanced visibility may make the SAF controlled entities react differently, such as moving to attack.

Also, the SME will be able to see how the scenario plays out from the point of view of the soldier at the crew station. The higher the resolution of the terrain database, the more terrain features will be captured. Given a very hilly terrain, a low-resolution database will miss many of the sloping terrain features and yield a line of sight to an entity that wouldn't normally exist on the real terrain. With a high-resolution terrain database, the terrain features will be captured, but the probability of performance degradation increases. The result of this trade-off may result in a lower resolution database providing more visibility than on the real terrain. This can affect the way the scenario plays out as an enemy entity might be seen by the soldier when it wouldn't be normally, changing the way he performs the mission.

The primary challenged faced during CAT scenario development was determining the way a UA will fight. No one really knew exactly how a UA equipped with semi-autonomous ARVs would fight. The doctrine that describes combat using these assets is still in development. For the CAT, the best educated guess was made (with the involvement of the user community) given current tactics as derived from the FCS ORD and O&O. This made vignette development more time consuming than originally thought.

THE DIGITAL TERRAIN DATABASE

Developing the terrain database was a significant activity during CAT development that required adequate lead-time to perform. A plane with the appropriate stereo imaging camera equipment had to be scheduled and Ft. Bliss fly-over time approved. Then the raw data obtained from the fly-overs had to be converted into an OpenFlight format for use by the IG. This process itself can take several months to complete. Complicating matters, the plane scheduled to perform the fly-over sustained damage just prior to its flight, requiring the fly-time to be rescheduled. This added several more weeks to the schedule.

After the initial database was built, the optimum size had to be determined. The processing power just doesn't exist to render a DTED level 5 database for the entire McGregor Range area in real-time without diminishing frame rates. There are effectively two choices: to decrease the size of the database and maintain the high resolution; or increase the size but sacrifice the resolution. The right mix of terrain database size and resolution had to be found so as not to hurt performance. For CAT this ended up being a 13Km x 9Km area at 10m resolution. This was achieved by a trial and error method of adjusting database size and polygon size until an optimum combination was found. This trial and error method consumed a large amount of time.

An important lesson learned for future efforts is to not underestimate the amount of time developing the optimum terrain database can take. Not only do the fly-overs and initial database generation take a long time to complete, but the optimization process can take nearly as long. Also, other tasks such as scenario generation depend on use of the terrain database. It is suggested that this be one of the first tasks undertaken in construction of the future ESS.

INTER-PROCESS COMMUNICATIONS

The ESS made good use of a custom distributed communication service called, the Process Interface Unit (PIU) Comm object to pass data between the various ESS internal processes. The PIU Comm object utilizes inter-process communication (IPC) APIs for System V shared memory and message queues to pass continuous and event data between internal processes.

Currently, data transmission between the vehicle and ESS occurs via the "A-Kit/B-Kit" centralized interface in the format specified by the A-Kit/B-Kit Interface Control Document (ICD). This practice is highly effective but does add one more layer of communications management at the A-Kit/B-Kit interface level. ESS processes exchange data via the PIU Comm. The A-Kit interface process must then take this data and format it IAW the ICD then transmit it to the A-kit. Currently this is done utilizing the TARDEC implementation of the Weapon Systems Technical Architecture Working Group (WSTAWG) Operating Environment (OE) open systems distributed communications API. The reverse is performed on the A-Kit side sending data to the B-kit. See the top picture of Figure 9.

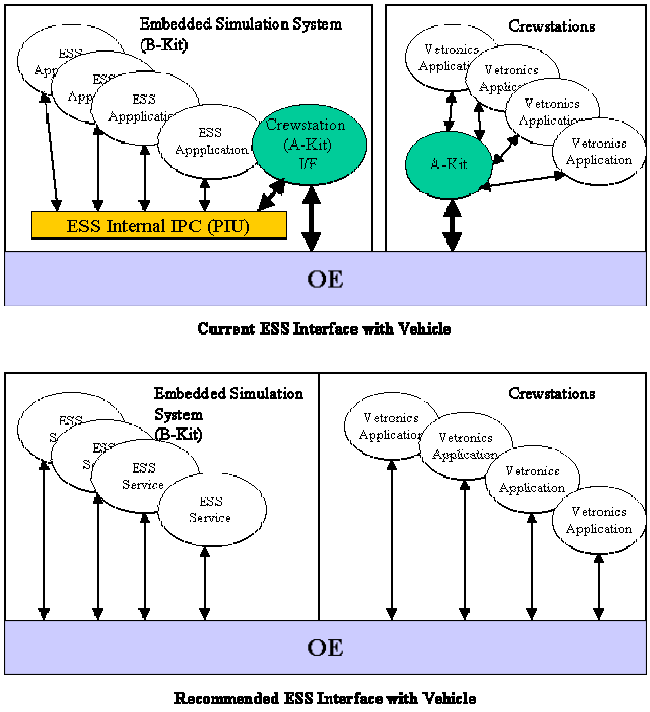

Figure 9: ESS Communications Interface

In the future it is envisioned that the centralized communication mechanism of A-Kit/B-Kit concept will be replaced with the more seamless direct coupling of ESS with vehicle systems as seen in the bottom half of Figure 9. This could happen through the use of the WSTAWG OE, or by some third party middleware. Since each process already has to send and receive messages using the PIU, a layer of management can be saved if they write directly to the shared memory that the A-Kit can access, thereby negating the need for an A-Kit interface process to do the same. This alternative will be explored more in the future.

HARDWARE CONSIDERATIONS

Typically the ES Team has used Commercial-Off-The-Shelf (COTS) material for B-Kit construction. This gives the team flexibility to change hardware or software with little to no down time, and is much lower in cost. Two versions of hardware were produced for the CAT program, one completely unmodified COTS solution for use in a static SIL environment, and a modified version for dynamic use in the CAT Stryker vehicle. The ESS for

the CAT vehicle was modified in form and added stiffeners, standoffs, and heat sinks to each output channel to increase their resistance to shock, vibration and high temperatures. The boxes were then placed in a shock mounted transit case for further protection. See Figure 10 for an example. Each ESS channel is a Racksaver™ 1U box containing a Tyan™ dual processor (1.6 GHz) motherboard and a TI 4600 graphics card. The ESS hardware performed very well over the course of the experiments. However, some considerations should be noted for follow on efforts.

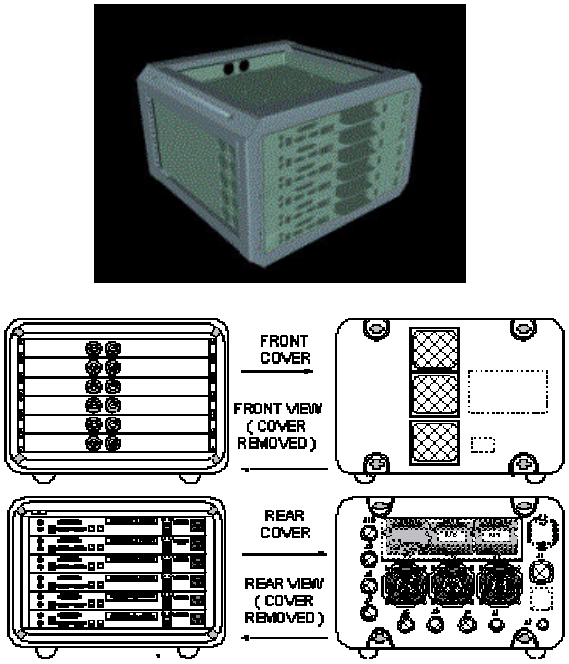

Figure 10: CAT Rugged ESS

Each complete ESS system was equipped with a National Instruments Field Point™ unit that could shut down the ESS if temperatures inside any of the boxes reach a programmable threshold level. The Field Point™ hardware was tested and verified in the lab. However, the software was not able to be tested prior to field exercises and was falsely shutting the unit down due to an extra byte in the data field. The problem was later corrected. More upfront SIL test time should have been built into the schedule for hardware/software stress testing and evaluation.

Also, one lot of the SIL Racksaver™ Tyan™ motherboards came with a factory defect in the AGP port enable control signal, causing a loss of video. Each motherboard had to be replaced as time permitted. Additionally, lack of airflow through the SIL boxes caused occasional machine lockups. This was corrected by providing additional airflow. These problems lead to some down time both in trying to trouble shoot what was happening, and in sending the boxes back to the shop for motherboard replacement. Unfortunately, these problems were not discovered until extensive use of the boxes had begun. As above, more thorough testing in a complete hardware and software environment should be built into the schedule to identify these problems as early as possible.

For the future, efforts will be made to reduce the general size of the dynamic ESS. The overall size of the complete transit case unit was fairly large and consumed some amount of interior vehicle space. A smaller box will yield more flexibility in vehicle mounting in and around crew stations and other components. A single processor motherboard will be explored. Moving to a single processor system will not only reduce the size of the motherboard, but significantly reduce heat inside the box as well.

Finally, a method of wirelessly logging into the embedded ESS boxes may be considered. Inside the vehicle, access to the boxes can be limited. An exterior port on the vehicle would only allow access when the CAT vehicle is stationary. The ability to gain access to the ESS from outside a moving vehicle would be a benefit, such as making SAF changes on the fly.

CONCLUSION

Several broad lessons learned can be derived from the CAT and UCD ESS development effort. First and foremost is to establish a common understanding with the primary contractor. Assumptions, designs, interfaces, and objectives should be agreed upon as far in advance as is possible. Defining and locking in requirements as far ahead of time increases coding productivity and reduces risk. Identifying and initiating procurements of long lead items such as crew station hardware or databases also reduces risk and adds valuable integration and test time.

REFERENCES

1. Virginia Tech Computer Science web site: http://courses.cs.vt.edu/csonline/SE/Lessons/Waterfall/Lesson.html

2. Old Dominion University Computer Sciences Web site: http://www.cs.odu.edu/~zeil/cs451/Lectures/01overview/process2/process2_htsu2.html

3. Software Engineering, Ian Sommerville, Addison-Wesley Publishers Ltd., 1996

4. J. Johnson, “Analysis of Image Forming Systems”, Proceedings of Image Intensifier Symposium, October 1958.

5. J. Johnson, W. Lawson, “Performance Modeling Methods and Problems”, Proc IRIS, January 1974.

6. J. Ratches, et.al., “Aided and Automatic Target Recognition Based Upon Sensory Inputs from Image Forming Systems”, IEEE Transactions of Pattern Analysis and Machine Intelligence, Vol 19, No 9, Sept. 1997.

7. 1997D. Shumaker, J. Wood, C. Thacker, **Infrared Imaging Systems**, DCS Corporation, 1988.

CONTACTS

Chris Mocnik is a Project Engineer at the U.S. Army’s Tank-Automotive Research Development & Engineering Center, where he has been employed since 1991. He is currently the Associate Team Leader of the Embedded Simulation Team. Mr. Mocnik worked as a Product Engineer for DaimlerChrysler from 2000-2002 releasing engine control modules for truck platforms. Mr. Mocnik holds a BS degree in Electrical Engineering from Lawrence Technological University Southfield, MI, and a MS degree in Computer Science and Engineering from Oakland University Rochester, MI.
Email: mocnikc@tacom.army.mil
Phone: (586) 574-5491
Fax: (586) 574-5008

Tim Lee is a computer engineer at DCS Corporation where he has worked since 1990. He is currently the head of the Support Systems Branch which is involved with the TARDEC Embedded Simulation projects. Mr. Lee holds a BS and a MS degree in Electrical Engineering from University of Virginia and Virginia Tech, respectively.
Email: tlee@dcscorp.com
Phone: (703)683-8430
Fax: (703)684-7229