

# Vetronics Technology Testbed: Experience Report

William Pritchett and Brian Wood

DCS Corporation  
1330 Braddock Place  
Alexandria, VA 22314  
703.683-8430

wpritch@dcscorp.com | bwood@dcscorp.com

## 1. INTRODUCTION

The Vetronics (**vehicle electronics**) Technology Testbed (VTT) is a research project being developed by the U.S Army Tank-Automotive Research Development and Engineering Center (TARDEC). The main objective of the VTT is to demonstrate the capability of one crewmember to perform the functions of both the vehicle Commander and Driver. The demonstration was required to take place while operating over military-significant terrain and while performing a military-significant mission. In order to accomplish this objective, TARDEC adapted and integrated crew station designs (Figure 1) developed under the Crewman's Associate Advanced Technology Demonstrator (ATD).



Figure 1. VTT Crewstation

In addition, hardware and software developed under the INter-Vehicle Embedded Simulation Technology (INVEST) Science and Technology Objective (STO) was used to create realistic simulated operating and training scenarios for the crewmembers. Another goal of the

project was to demonstrate the technology in a platform comparable (mobility, size, weight, etc.) to the currently envisioned Future Combat Vehicle (FCV).

Several key technologies and capabilities were incorporated into the VTT and include the following:

- Drive-by-wire capability
- Day and night operation
- Indirect vision as the primary vision
- Three-dimensional (3D) audio system
- Speech Recognition and Generation
- Multi-function displays with touch screens
- Embedded simulation as an enabling technology for embedded training, mission rehearsal, battlefield visualization and after action review
- WSTAWG Operating Environment (OE) [1]
- Joint Variable Message Format (JVMF) [2]
- Communication via voice intercom and SINCGARS radios

## 2. SYSTEM ARCHITECTURE

The VTT electronics consist of four VME chassis connected by a Fibre Channel high-speed databus [3]. Two of the chassis are designated as crew vetronics stations (CVS) and each contain two Dy 4 179 (PowerPC) single board computers (SBC) and three Dy 4 783 graphics cards to control the multi-function displays. Another VME chassis is designated as the drive-by-wire vetronics station (DVS) and uses one Dy 4 179 SBC to control the actuators used in the drive-by-wire subsystem. The remaining VME chassis is designated as the command and control vetronics station (C2VS) and uses one Dy 4 179 card to control the embedded simulation and mapping subsystems. All SBCs were configured with 64 MB of RAM and 32 MB of non-volatile flash memory.

## 3. SOFTWARE ENVIRONMENT

The software developed for the VTT is a combination of Ada 95, C, and C++. Ada 95 was used to develop the majority of the infrastructure components to include the operating environment, the portable graphics library, the station management subsystem, the built-in-test (BIT) subsystem, the data logging/instrumentation subsystem, the mapping subsystem, and the Command and Control (C2) messaging subsystem. C++ was used to develop the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page.  
To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.  
SIGAda 2001 09/01 Bloomington, MN, USA  
© 2001 ACM 1-58113-392-8/01/0009...\$5.00

lethality, mobility, survivability and C2 Soldier-Machine Interface subsystems. C was used for the drive-by-wire subsystems. The language mix was somewhat arbitrary and the choice of language was largely left to the subsystem developer. The overall main program, however, was developed in Ada95 to reduce mixed language integration issues.

The software for the VTT was first developed on Sun workstations located in two geographically dispersed laboratories. The use of a portable OE and portable graphics library enabled developers to first compile and test their application code in the workstation environment and move to the PowerPC/VxWorks target with a simple recompilation. A common OE interface isolated operating system dependencies from the applications and enabled them to be ported from Solaris to VxWorks with no application software changes. A common graphics interface isolated graphics system dependencies from the applications and enabled them to be ported from an X-Windows environment to a Dy 4 RTGS environment with minimal application changes.

The VTT software runs on top of the VxWorks real-time operating system. The Ada 95 software was first developed on the Sun using the GNAT Ada 95 compiler while the C/C++ code used the gcc compiler. The Green Hills Ada 95 compiler was used for the target (VxWorks) environment while the C/C++ code used the gnu cross compiler supplied with VxWorks.

#### 4. METRICS

An approximation of the source lines of code breakdown for the VTT is as follows:

Subsystem	Language	Total Lines	Semicolons
VTT Subfunctions	Ada	92,000	24,000
Operating Environment	Ada	58,000	19,000
Graphics Library	Ada	150,000	40,000
Map Server	Ada	136,000	30,000
Utilities	Ada	36,000	9,500
JVMF Message Parser	Ada	161,000	50,000
VTT Subfunctions	C/C++	27,000	4,900
<b>TOTAL</b>		<b>660,000</b>	<b>177,400</b>

Please note that the operating environment, graphics library, map server, utilities, and JVMF message parser were developed for other projects and reused on this project. The source lines listed are for the entire subsystem, and do not reflect the portion of the subsystem actually used by the VTT. The executable image for the VTT (PowerPC/VxWorks) is approximately 29 MB.

#### 5. LESSONS LEARNED

The biggest lesson learned was the validation that Ada continues to be a viable language and can be successfully used on large, mix-language development programs. The development team encountered no major software-related problems during the development. The only real lessons learned or issues of note encountered during this project fall into two general categories: 1) mixed language programming and 2) embedded development. All mixed language issues were resolved early in development. The use of Ada 95 as the main program eliminated any problematic elaboration issues. Nearly every Ada 95 package exported a C interface through child package named *package\_name.c\_wrapper*. The biggest problem encountered with the language mix was the use of arrays. All of the Ada arrays were constrained and care had to be taken when passing these to and from C. Any manipulation of the data types between the two languages was accomplished in the *c\_wrapper* packages.

Regarding embedded development, the use of fixed interfaces in the OE to isolate operating system dependencies and graphics library to isolate graphics hardware dependencies greatly enhanced portability and enabled development to occur on inexpensive Sun workstations long before the target hardware arrived. Additionally, the use of static configuration files enabled facilitated a single, common executable image that was loaded on each SBC. The specific subfunctions (e.g., lethality subsystem) to be run on the board were specified in the configuration file. This let the integration team shift functionality around based on hardware availability and performance requirements. A drawback of this approach, however, was the resulting image size was larger than it would have been had the images been tailored to each SBC.

#### 6. ACKNOWLEDGEMENTS

The VTT was a collaborative effort between teams from TARDEC, DCS Corporation and General Dynamics. Thanks to all of the dedicated individuals from these organizations that made this project a success.

#### 7. REFERENCES

- [1] U.S. Army Weapon Systems Technical Architecture Working Group, *Operating Environment API*, Version 1.0, June 1998.
- [2] Variable Message Format (VMF) Technical Interface Design Plan (Test Edition), Reissue 3, 17 June 1998.
- [3] ANSI X3.230-1994, Information Technology – Fibre Channel – Physical and Signaling Interface (FC-PH).