A Future Ground Combat Weapon System Software Architecture

Rakesh Patel/Peter Haniak
Vetronics Technology Center
USATACOM
Patelrak@tacom.army.mil
Haniakp@tacom.army.mil

Dr. Paul Richardson
Dept. of Electrical and Computer Engineering
University of Michigan-Dearborn
richarpc@umich.edu

In the past, ground vehicles have largely been mechanical systems with little electronic content. The increasing performance of computer technology coupled with decreased cost, however, has led to an ever-increasing usage of computer technology in ground vehicles. Though offering tremendous benefits, the use of computer technology in ground vehicles is not without problems. Rapid changes and advances in hardware significantly reduce the time in which the hardware becomes obsolete, thus forcing expensive and time-consuming changes to be made in the corresponding software. Additionally, as vehicles become more network-centric, many different subsystems are now being required to share software and exchange information at an ever-increasing rate. To meet the challenge future weapon system software architectures must continue to evolve and innovate to be completely open and allow greater portability and interoperability.

The US Army is leading the way in the new technology paradigm innovation by developing a future software architecture based on well defined, widely used, non-proprietary interfaces/protocols standards. Although the concept of standard interfaces is not new, standardizing interfaces for network centric warfare present unique and complex challenge problems. Among these problems is the challenge of meeting Quality of Service (QoS) requirements in dynamic, and demanding combat operations. These standards are defining all aspects of systems interfaces to facilitate new or additional systems capabilities and QoS requirements for a wide range of applications and environment. They also explicitly provide for expansion or upgrading through the incorporation of additional or higher performance elements with minimal impact on the system.

The software reference architecture (figure 1) specifies a software framework based on both the technical architecture and intelligent ground vehicle taxonomy, which can be employed to develop and describe an objective system(s) software design. The software reference architecture is organized in accordance with the SAE Generic Open Architecture (GOA) framework, such that each identified component may be located in terms of its intended usage within the application environment. The components within the software reference architecture are comprised primarily of Application Programmers Interfaces (APIs), which are accessed via the application software in accordance with the GOA model. To date there have been six API standards developed which include the following:

- The Weapon System Mapping Services (WSMS) API provides an application interface to the mapping subsystem within a system(s). The logical WSMS API interface is provided as a set of mapping services, which in turn are defined via functional specifications in concert with one or more language bindings. The WSMS services primarily provide interfaces to interact with and control the presentation of 2D and 3D raster, vector, and DTED maps; grid lines; and symbolic overlays within an application SMI environment.

- The Terrain Services API provides an interface to perform critical terrain analysis functions independent of the operating environment and application developer within a system(s). The logical terrain services API provides access to four primary application services as follows: Path Analysis, Range Analysis, Terrain Categorization, and Line of Sight.

- The Station Management API provides a conceptual “software backplane” to facilitate the integration, intercommunication, upgrade of application components within a system(s). The logical Station Management API interface provides access to three primary application services as follows:

Application Loading and Synchronization, Application Runtime Synchronization, and Application Subfunction Health Monitoring.

- The Performance, Analysis, and Measurement API provides an extensible tagged event/sequence generation, recording, and monitoring interface, which is mapped to the intelligent ground vehicle taxonomy to measure defined mission scenarios/tasks within a system(s). The logical Performance, Analysis, and Measurement API interface provides application access to data collection, data capture/monitoring, and data analysis services.
- The Operating Environment (OE) is the primary mechanism defined within the software reference architecture to abstract applications from hardware and software system dependencies. To this end, the OE encapsulates a system services core to facilitate application porting among diverse platforms and foster application migration within a system(s). The OE API defines an application interface to the OE via a set of services accessible from the application layer to access the underlying real time operating system as well as to communicate and share data with other intra/inter-system applications via a set of distributed XOS objects. The physical OE interface is defined by a POSIX profile for OS interfacing and a functional specification in concert with one or more language bindings for interfacing to distributed XOS services. The logical OE interface is defined by a binary encoding which is applied to achieve interoperability among OE implementations within a system(s)
- The Adaptable Graphics Interface Language (AGIL) API defines an interface between the system software application and a graphical engine. The primary purpose of the AGIL API is to isolate the graphics engine hardware and software dependencies from the application software. The AGIL logical interface is defined via a language binding to a set of packages, which in turn implement the AGIL services.

Many of these APIs have been implemented in both experimental and real systems. Development of this new architecture continues in coordination with a US Army led Weapons Systems Technical Architecture Working Group that includes Government, Industry, and academic members and the Crewmens Automated Testbed/Robotic Follower ATD architecture IPT. Working together, the Working group and IPT are forging the next innovation in combat software architectures.

Figure 1.

Software Reference Architecture

- Organized in accordance with the SAE Generic Open Architecture (GOA).
- Comprised primarily, but not exclusively of API components.
- Software framework is subset in accordance with the five primary system architecture partitions/classes.
- Built on key Middleware Initiative.