Component: NAVY

Topic #: N201-057

Title: Software Ecosystem Architectural Model and Application Program Interface for Common Core Combat System

Technology Areas: Info Systems

Acquisition Program: PEO IWS 1.0, AEGIS Combat System Program Office

ITAR: The technology within this topic is restricted under the International Traffic in Arms Regulation (ITAR), 22 CFR Parts 120-130, which controls the export and import of defense-related material and services, including export of sensitive technical data, or the Export Administration Regulation (EAR), 15 CFR Parts 730-774, which controls dual use items. Offerors must disclose any proposed use of foreign nationals (FNs), their country(ies) of origin, the type of visa or work permit possessed, and the statement of work (SOW) tasks intended for accomplishment by the FN(s) in accordance with section 3.5 of the Announcement. Offerors are advised foreign nationals proposed to perform on this topic may be restricted due to the technical data under US Export Control Laws.

OBJECTIVE: Develop a Common Core Combat System (CCCS) software application execution and cross-application coordination environment (ecosystem) and associated Application Program Interface (API) capable of supporting various DoD- or 3rd-party-sourced dynamically loadable real-time tactical combat systems client applications.

DESCRIPTION: The Navy requires expansion of its sea-based advantage through increased capability. This need is addressed by providing technology that has the potential to improve Combat Systems client applications real-time performance (and the overall tactical performance and effectiveness of the host surface platform), as well as enabling multi-platform software re-utilization and commonality. This may allow for a reduction in the number of platforms needed in a specific tactical arena by improving the overall tactical efficiency of the battlegroup as a whole, resulting in a subsequent improvement in overall affordability.

The current implementation of the AEGIS Combat System has fundamental architectural limitations deriving from its initial hardware and software design constraints. These architectural limitations have forced the use of inefficient “bolt-on” style modernization modifications and enhancements to meet evolving 21st century threats. Currently available commercial systems and software that might be considered for adaptation to partially address the Navy’s combat systems requirement for advanced situational awareness (SA) (e.g., the FAA Air Traffic Control System hardware and software) are dated in their designs. They lack the software flexibility and track monitoring capacity required to adequately address tactical requirements. Specifically, the currently available commercial technology mentioned above is limited in that it lacks the capability to track, identify, and manage complex air, surface and subsurface entities and threats present in a combat environment. Additionally, such commercial systems have no intrinsic ability to provide a real-time-response-capable software execution environment for DoD- or 3rd-party-sourced combat systems client applications. Since no viable commercial alternatives exist or thus cannot be adapted to address these needs, it becomes necessary for the Navy to pursue a different avenue of exploration.

The Navy needs a combat systems application software execution environment (i.e., software “ecosystem”) for use within a host CCCS (or AEGIS) environment. This software ecosystem will support various DoD- or 3rd-party-sourced dynamically loadable real-time tactical combat systems client applications, all potentially executing simultaneously within the software ecosystem. Here “dynamically loadable” means that a client application can be loaded and started at operator discretion within a currently executing combat system instance, without the need to reload, restart, or otherwise interfere with the current running status of the combat system. The software ecosystem must also provide these client applications with a common API enabling access to combat systems weapon, sensor, and Common Operational Picture data in a real-time manner. The innovative challenge in this design task derives from providing these services without jeopardizing or degrading the overall real-time performance of the other simultaneously executing client applications or the host combat system itself. A fundamental architectural redesign of the combat systems core is needed to enable the efficient, rapid, and cost-effective addition of new capabilities, such as multi-platform sensor and weapons coordination, off-board/organic on-the-fly sensor and weapons integration, built-in cyber resilience, and real-time fault recovery. To address this critical need, a new CCCS
architecture is currently in the planning stage, with the intent of providing a modular set of platform-agnostic common combat system services. This CCCS implementation, when supplemented by platform-specific sets of weapon, sensor, and communications capabilities modules, will constitute a new, modular, dynamically adaptable combat system design that can evolve to meet future emerging threats in a rapid and cost-effective manner.

Development of a CCCS software application execution and cross-application coordination environment (ecosystem) and associated API is critical to the future needs of the Navy. Such a capability will allow for the rapid integration of new tactical capabilities within a currently installed and running combat system instance. This software ecosystem is intended to serve as: (i) a core combat system architectural enhancement within the current AEGIS system; or (ii) a primary modular component within the newly proposed CCCS platform-agnostic combat system, intended for implementation on Future Surface Combatant (FSC) and other appropriate Navy platforms.

The CCCS modular ecosystem architectural framework and software component API should provide a comprehensive architectural model for a complete, versatile platform-agnostic “core” combat system, when taken in conjunction with: (i) an appropriate CCCS core combat system modular architectural framework and software component API; (ii) an appropriate CCCS multi-platform coordination architecture and inter-platform data exchange algorithm set; and (iii) an appropriate multi-platform coordinated/synchronized Distributed Common Operational Picture (DCOP) subsystem implementation. The term, “core” implies that this system architecture and implementation will provide those combat system services and capabilities that are considered necessary to satisfy common tactical warfighting requirements spanning various and diverse surface combatants. This core combat system implementation, when configured with the appropriate surface warfighting platform-specific sensor and weapons capability software modules, should be capable of fulfilling the functional warfighting capabilities and requirements needed to support the vast majority of U.S. Navy surface platforms well into the future. The CCCS modular applications program execution and cross-applications data exchange and coordination environment (hereafter referred to as the “ecosystem”) will provide a software execution environment supporting multiple dynamically loadable (i.e., run-time loadable) combat systems operator controlled and/or autonomous applications. The ecosystem will support application performance monitoring and control services as well as inter-application (ship and cross-platform or battlegroup) coordination and data exchange services, all provided via a well-defined and documented ecosystem API. The architecture of the API will provide a layer of software abstraction hiding any implementation-specific details of CCCS provided lower-level software communications and control services.

The CCCS ecosystem architectural model and software framework will support a software execution environment and API capable of supporting a scalable number of independently executing combat systems applications and autonomous AI-based software agents or entities, such as tactical and engagement planning tools, multi-platform weapons control and assignment tools, and AI-based situational awareness autonomous agents. Within the ecosystem, each application or entity will be capable (when granted the appropriate access permissions) of accessing: (i) ship- or battlegroup-hosted weapons and sensor system control elements system status and sensor data; (ii) DCOP data; (iii) inter-application cross-platform data exchange and messaging services; and (iv) host combat system (i.e., CCCS) ecosystem software management performance monitoring metrics and applications execution controls.

With respect to ship-based ecosystem applications accessing organic-hosted (such as ship) capabilities, all four of the access categories listed above should support real-time access and control. With respect to ship-based ecosystem applications accessing non-organic-hosted (such as battlegroup) capabilities, near real-time performance should be the goal. The ecosystem architecture and its API will support a Quality-of-Service (QOS) application and local and remote capabilities performance monitoring subsystem. This subsystem should be capable of providing QOS metrics for each off-board capability to any application granted access to that capability. The QOS metrics for any capability should be based on appropriate weapons communications control-loop delay and sensor data delay measurements derived from periodic automated communications-loop access senescence and jitter testing of off-board capabilities.

The ecosystem subsystem software architecture and API framework should be modular in nature. The installation, removal, activation, and deactivation of the ecosystem modular software within an executing CCCS implementation (or other host combat system) should have no adverse effect on the real-time performance of the CCCS system or the services it provides to the host platform and operator at the time those changes are implemented. An exemplar of this type of no impact behavior during runtime installation and removal of capabilities within an executing system can be observed in the Linux operating system kernel module control facilities (kernel 4.4 and above), e.g., the insmod, rmmod, depmod, lsmod, modinfo, and modprobe commands [Refs. 2, 3]. The process of installing, removing, upgrading, or otherwise controlling the ecosystem software implementation module within an executing CCCS installation should be easily executed by combat systems watch personnel, without the need for specially trained software maintenance personnel.

The CCCS ecosystem API architecture and software framework will provide a native suite of Ecosystem cross-application and cross-platform data exchange and message services. The API will provide a level of software abstraction, masking the underlying implementation details of the communications services framework of the CCCS (or host combat systems) that it utilizes. The
ecosystem API will also provide ecosystem resident software applications and entities with an access-controlled combat system weapons and sensor-system data real-time- or near real-time-capable interface. This API will be architected to provide a platform-agnostic abstracted weapons and sensor access mechanism capable of providing access to both organic and non-organic weapons and sensors utilizing common well-defined and documented data and control software structures, and access to the QOS metric subsystem described previously to quantify performance when accessing near real-time non-organic capabilities. The ecosystem API architecture will also implement a software applications access control subsystem, which will utilize a hierarchically structured set of privilege levels to control access to both organic and non-organic weapons and sensor systems data and interfaces.

Both the CCCS ecosystem architectural model and its associated APIs will be well-documented and conform to open systems architectural principles and standards [Ref. 4]. Implementation attributes should include scalability and the ability to run within the computing resources available within the AEGIS combat systems BL9 or later hardware environment.

The technology will be compatible with the C++ programming language and capable of running in a Linux (Redhat RHEL 7.5/Fedora 29/Ubuntu 18.4.1 or later) processing environment as a standalone application (i.e., no critical dependencies on network-based remotely hosted resources, save for sensor data emulators). The prototype CCCS implementation will demonstrate the following: first, the ability to start, stop, and control multiple operator-driven or autonomous combat systems applications or software agent entities; second, the ability of real-time performance when executing various data exchange operations between ecosystems-hosted applications and entities; and lastly, the ability of real-time performance when executing various data exchange operations between organic and non-organic ecosystem-hosted software applications and entities, and real-time or near real-time access and control of organic and non-organic sensor and weapons system emulators.

Any prototype produced during Phase II shall demonstrate that it meets the capabilities described above during a functional test to be held at an AEGIS or FSC prime integrator supported Land Based Test Site (LBTS) identified by the Government and capable of simulating an AEGIS BL9 compatible or newer combat system hardware test environment.

Work produced in Phase II may become classified. Note: The prospective contractor(s) must be U.S. Owned and Operated with no Foreign Influence as defined by DOD 5220.22-M, National Industrial Security Program Operating Manual, unless acceptable mitigating procedures can and have been be implemented and approved by the Defense Security Service (DSS). The selected contractor and/or subcontractor must be able to acquire and maintain a secret level facility and Personnel Security Clearances, in order to gain access to classified information pertaining to the national defense of the United States and its allies; this will be an inherent requirement. The selected company will be required to safeguard classified material IAW DoD 5220.22-M during the advance phases of this contract.

PHASE I: Design a concept for a CCCS software application execution and cross-application coordination environment (ecosystem) and associated API. Establish feasibility through modeling and analysis to show the concept will meet the required parameters in the Description. The Phase I Option, if exercised, will include the initial design specifications and capabilities description to build a prototype solution in Phase II.

PHASE II: Design, develop, and deliver a prototype software implementation of the CCCS ecosystem. Demonstrate that the prototype meets the capabilities detailed in the Description during a functional test that will be held at an AEGIS or FSC prime integrator-supported LBTS provided by the Government, representing an AEGIS BL9 or newer combat system hardware environment. It is probable that the work under this effort will be classified under Phase II (see Description section for details).

PHASE III DUAL-USE APPLICATIONS: Support the Navy in transitioning the technology for Navy use. Integrate the CCCS ecosystem software along with other CCCS compliant core capability and platform-specific capability software modules into a prototype combat system implementation, consisting of a Common CCCS experimental prototype, implemented on a virtualized hardware environment within an AEGIS compliant land-based testbed.

This capability has potential for dual-use capability within the commercial Air Traffic Control system in future development of an air traffic control system capable of rapid upgrade to handle increasingly complex traffic control patterns.

REFERENCES:


   doc.lagout.org/operating%20system%20/linux/Linux%20Kernel%20Development%2C%203rd%20Edition.pdf

   insights.sei.cmu.edu/sei_blog/2016/07/a-naval-perspective-on-open-systems-architecture.html

KEYWORDS: Common Core Combat System; Modular Applications Program Execution and Cross-applications Data exchange and Coordination Environment; Rapid and Cost-effective Addition of New Capabilities; Cross-applications Data Exchange and Coordination Environment; Communications Services Framework; Dynamically Loadable Real-time Tactical Combat Systems Client Applications