2021 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
SURVIVABILITY TECHNICAL SESSION
AUGUST 10-12, 2021 - NOVI, MICHIGAN

Modernizing Legacy Defense Systems to be Compliant with Open Architecture Standards

Kwapis, Alec [1], Barnett, David[2], Grager, Kyle[3], Johnson, David[1], Hildebrand, Stewart[1],

[1]DornerWorks, Grand Rapids, MI
[2]TenCate Advanced Armor USA, Goleta, CA
[3]US Army-Ground Vehicle Service Center, Warren, MI

ABSTRACT

This paper focuses on the complications and developmental strategy and approaches for updating legacy systems to MOSA standards. The Department of Defense has adopted Modular Open Systems Approach (MOSA) standards because it lowers risk and accelerates innovation by enabling interoperability between software components. The problem is many existing systems were developed and certified before the MOSA standard was implemented. A solution developed by DornerWorks and Tencate to bring a legacy system up to the MAPS standard is discussed to illustrate this. Tencate the developers of an active blast mitigation system (TenCate ABDS™) which effectively mitigates launch acceleration, jump height, flight duration and slam-down, caused by IED and mine blast to increase the protection of the vehicle's occupants. The ABDS was developed to be compliant to the latest DoD standards at the time. Fast forward a couple years and the development of the MOSA based Modular Active Protection System (MAPS) has caused Tencate to adapt their product to a MOSA standard to become MAPS compliant.

Citation: A. Kwapis, D. Barnett, K. Grager, D. Johnson, S. Hildebrand, "Modernizing Legacy Defense Systems to be Compliant with Open Architecture Standards", In *Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium* (GVSETS), NDIA, Novi, MI, Aug. 10-12, 2021.

1. INTRODUCTION

Modular Open Systems Approach (MOSA) is an initiative to increase the adoption of open architecture standards in weapons and defense systems. Open architecture standards help to increase (software) interoperability, allow for accommodation of future upgrades, reduce cost of acquisition, reduce complexity, avoid vendor lock-in, among other benefits. Some guiding principles for MOSA include subsystem separation, standards-based communication between subsystems, and utilizing consensus-based standards. Standards should have conformance

verification, and they should try to point to existing standards governed by a consensus body where applicable.

In 2019, the secretaries of the Navy, Army, and Air Force issued a joint memo[11] underlining the relevance of Modular Open Systems Approaches to weapons systems. Through partnerships with academia and industry, a number of standards have been developed (or are in development) to help with MOSA goals, including FACE[6], SOSA[7], OMS/UCI[8], and VICTORY[9] (acronyms detailed below).

Future Airborne Capability Environment (FACE)[6] has been gaining more adoption in more defense projects during recent years. FACE is a standard of standards for open software architecture. It specifies requirements for operating system, portable segment, I/O segment, platform segment, and transport segment.

Sensor Open Systems Architecture (SOSA)[7] is a standard of standards, encompassing hardware, electrical, software, and interconnect specifications. Standardizing Air Force sensor systems with ANSI/VITA 64 (also known as VPX)[10]. SOSA helps increase interoperability of subsystems in advanced sensor systems, while allowing for future upgrade capability. Software vendors can provide software in containers, virtual machines, or provide FACE conformant software for use in a SOSA system.

Open Mission Systems / Universal Command and Control Interface (OMS/UCI)[8] is an architecture specification that promotes interoperability and reuse for airborne systems.

Vehicular Integration for C4ISR/EW Interoperability (VICTORY)[9] is an open architecture initiative to enable interoperability of vehicle systems.

2. Complications in Legacy System Modernization

Modernizing legacy systems for open architecture standards may present several complications. Generally, there are two approaches that can be used in the modernization process: replacing the legacy system altogether or adapting and integrating the legacy system with modern standards. Replacing the legacy system altogether is risky for the multiple reasons [1]. One, there may not be a complete specification of the legacy system. As a result, creating a new system that is functionally identical to the legacy system cannot be done in a straightforward manner. Two business practices and the operation of the legacy system may be intimately intertwined. Replacing the system may require new business practices that could introduce undesired consequences. Three, new software development is generally risky because unanticipated problems could be introduced with a new system. Furthermore, the shortcuts and workarounds used in debugging the legacy system may no longer apply to a new system. For these reasons and more, it may be a better business strategy to adapt and integrate the legacy system instead of replacing the system altogether.

The process of adapting a legacy system to modern standards still presents several complications such as compliance, complexity, culture, and cost. Modifying the legacy system's existing code base to become compliant with modern standards will inevitably introduce the need to recertify the code base, which is both a time consuming and costly procedure. The adaptation process almost certainly introduces layers of complexity. For example, external communication interfaces on legacy systems may need to be updated to more modular, auto-negotiation-based interfaces, *e.g.*, serial-based protocols to Ethernet-based protocols. Furthermore, the approach used in adapting the legacy system may introduce

increased message passing between modules. While this does increase complexity for both endpoints, this aligns with the MOSA principle of functional decomposition, which is "… paramount in implementing a modular approach to system development" [2].

Culture is another complication that is somewhat shared with the approach of replacing a legacy system. Legacy system engineers, developers, and product managers are very familiar with their own system. The process of adapting a legacy system to modern open architecture standards introduce new technologies that the business must accept and become familiar with. However, this is a necessity in implementing MOSA standards, which state that a system should be designed and adapted in such a manner that supports the refresh and inclusion of innovative technology [2].

Lastly, cost is also one of the most important factors for businesses in deciding how to adapt a legacy system to modern open architecture standards. Fortunately, the cost of adapting a legacy system will most likely be lower than the cost of replacing the legacy system altogether. The following issues should also be considered: (1) How data will be integrated, (2) Connectivity between components in the architecture, (3) Message routing between components, (4) Validation and transformation of data in message passing, (5) Security mechanisms in the legacy system and the new system, and (6) Conformity to organizational and business standards [1]. The concept and impact of these complications should be taken into account when choosing a solution that best fits the adaptation and integration of a legacy system with modern open architecture standards.

3. Solutions to Legacy System Modernization

There are several solutions that can be employed to overcome the complications in adapting a legacy system to modern MOSA standards. One solution is to update the software of the legacy system to become compliant with the overall platform. However, one key requirement must be met in this approach: the hardware and software capability of the legacy system must meet the requirements of the new platform, such as containing all the necessary communication interfaces. If the legacy system does not meet this requirement, then new hardware must be used or additional hardware must be added to the legacy architecture, both of which have downsides. For example, switching to a different processor family with the required communication interface for the platform may be too difficult, depending on the size and complexity of the code base. While adding hardware to the existing architecture in the form of an extension card may be a more ideal option, this approach also shares downsides with the former. In both cases, the code base of the legacy system would have to be modified, which would require recertification. This increases costs for the vendor and takes time. Therefore, the approach of updating the software of the legacy system should only be considered in cases where it can be verified ahead of time that the legacy system will meet the requirements to become compliant and the code base is small and simple enough to allow for modifications.

Another approach to overcome the complications of adaptation is to add a separate, perhaps more capable, processor into the existing architecture of the legacy system. This processor would be specifically chosen to meet or even exceed the hardware and software capability requirements of the platform. The benefit of an additional processor is the creation of modularity by dis-aggregating tasks between the legacy processor and the new processor. Furthermore, any additions to the code base that are necessary to become platform compliant would be specifically for the new processor. While the addition of a separate processor into an existing architecture may be more ideal than simply updating the software of the

legacy system, there are also drawbacks. It may still be necessary to modify a portion of the certified code base to account for the addition of a separate processor. Fewer modifications would be needed to the certified code base compared to modifying the existing code base directly for platform compliance, however, recertification may still be inevitable. Therefore, the approach of adding a separate processor should only be considered when it can be verified ahead of time that the changes to the certified code base will be minimal and the legacy architecture supports the addition of another processor.

Another approach to overcome the complications of adaptation is the addition of an external conversion module. This module acts as a translator between the legacy system and the overall platform. The intention of this type of architecture is to reduce the need to modify the legacy system's code base as much as possible, if at all. This is evident from the module's role, which is to translate between the communication interface of the legacy system and the communication interface of the platform. The conversion module not only contains the logic for protocol translation, but also any other logic that is necessary for platform compliance that would otherwise not be ideal to incorporate into the existing legacy system. This type of architecture also allows for opportunities of future modernization to the platform's interface specification without having to modify the legacy system. However, if the platform's logic is tightly coupled with the legacy system and dependent on the updates to the platform's interface specification, this statement may not hold true in all cases. Nonetheless, the addition of a conversion module upholds the construct of reducing the need to modify the legacy system more so than modifying the legacy system's code base directly or adding a separate processor into the existing architecture.

4. MAPS – Deterministic Ethernet Core

Enabling vendors to assist in active protection by introducing their state-of-the-art technologies is the absolutely critical to continued evolution of protection for ground vehicles, as threats are also continuing to evolve. Modular Active Protection Systems (MAPS) was designed and developed with a Modular Open Systems Approach in order to support interoperability with many of vendor solutions readily available for both sensing and eliminating threats.

Low-latency and deterministic communication is critical for the success of the MAPS program for effectively integrating vendor technologies. Legacy Ethernet offers a high-speed, low-cost common network with proven support for modularity and interoperability. Deterministic Ethernet protocols adopted by the MAPS program complement those features by constraining latency and jitter and guaranteeing bandwidth availability. These standards defined in the MAPS Framework are not widely adopted in existing vendor solutions, which introduces complexity during integration.

Taking a Modular Open Systems Approach to addressing this deterministic Ethernet solution will ease the potentially large integration burden of implementing these protocols. Vendor technologies may communicate with a variety of different protocols, such as serial, PCIe, communication bus, legacy Ethernet, *etc*. In order to accommodate all of these different technologies, a GPR solution was developed that can easily interface with a plethora of communication protocols, while managing the deterministic Ethernet traffic and protocol integration.

Development of a solution to implement all of the deterministic Ethernet maintenance traffic and protocols, while still keeping the legacy system's native communication methods has proven to be exceptionally useful tool to the MAPS program.

This allows the Army to stay focused on the integration of technologies that enhance capability, as opposed to addressing the burden of compliance to the MAPS Framework. Repeatedly, this approach has avoided unnecessary cost, schedule, and risk to integration efforts.

5. Example Legacy System Modernization

One example of a legacy system that is undergoing modernization according to open architecture standards is the TenCate ABDS™ in accordance with the U.S. Army Modular Active Protection System (MAPS) program [3].

5.1. System Overview

TenCate ABDS is an active protection system that aides in the survival of a vehicle's occupants as they ride out the blast from an underbody improvised explosive device (U-IED). ABDS accomplishes this by reducing the vertical acceleration imparted onto the occupants to survivable levels [4]. An unmitigated high level of vertical acceleration would otherwise result in severe tertiary blast injury effects, including Pelvic, Thoraco-Lumbar Spine and Cervical Neck injuries. An added benefit comes from keeping the destroyed vehicle upright - multiple egress options remain available.

5.2. System Description

The legacy ABDS is an embedded system that consists of a bare-metal safety microcontroller that samples up to two dozen sensors and controls up to eight countermeasure initiator safe & arm devices. The sensors measure the vertical acceleration of the crew compartment. The countermeasures are explosive devices that, when detonated, apply an immediate downward impulse to counter the upward momentum of the crew compartment. Both the sensors and the countermeasures are distributed around the crew compartment. The legacy ABDS also uses a simplified operator interface that consists of switches and indicator lights in a small package to take up a small amount of the valuable real estate within the vehicle operator's cockpit. The ABDS context diagram in Figure 1 is a depiction of the boundaries of the system and the entities (vehicle, operator, IED blast effects, maintainer) that ABDS interacts with in the operational environment.

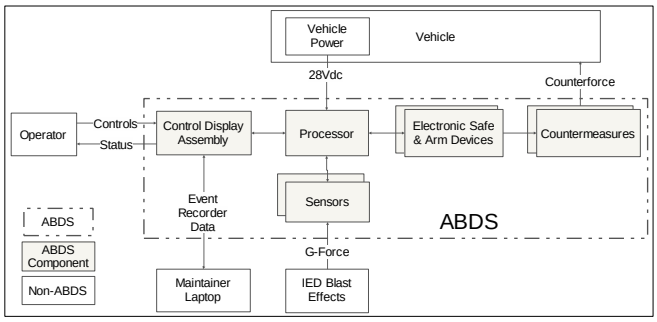

Figure 1: ABDS context diagram.

5.3. System Operation

An approximation of the blast effects on a vehicle are shown in Figure 2, which displays the Vehicle Underbody Blast Effects Timeline. In an underbody blast event, the initial explosive and soil debris begin to impact the vehicle roughly 1-2 milliseconds after an IED is detonated. The vehicles vertical acceleration peaks several milliseconds later at several hundred g's. Sensor data is rapidly sampled, filtered and processed by the system to detect a blast event within milliseconds. While the system is arming, sensor data is continuously analyzed for injurious acceleration levels. ABDS must continue to operate during an underbody blast while the vehicle is coming apart, so keep-alive power is necessary. ABDS countermeasures are initiated if injurious acceleration levels are detected. Otherwise, arming will be halted and arm voltages will be dissipated. ABDS is required to comply with the Safety Criteria for Hand-Emplaced Ordnance Design, MIL-STD-1911A [5].

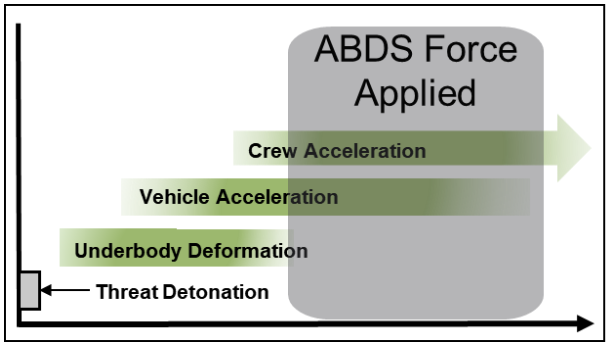

Figure 2: Vehicle underbody blast effect timeline.

5.4. ABDS Communication Interfaces

The types of operational communication links used in the legacy ABDS include the serial data link between the processor and the control & display unit, the serial data links between the processor and the countermeasure safe & arm device and the serial data link between the processor and the sensor unit as shown in Figure 3: Legacy communication links. The sensors and the countermeasures are both using time-slotted multi-drop communication links. Table 1 summarizes the message content for each communication link.

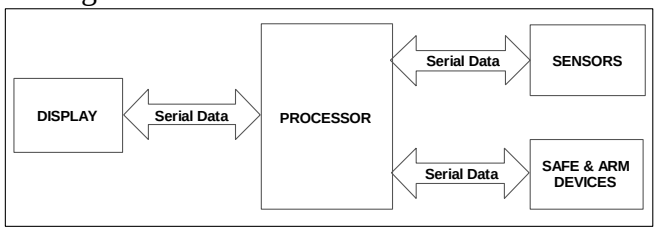

Figure 3: Legacy communication links.

Data Link	Message Content
Display Serial Bus	LED control, S/W version, status
Sensor Serial Bus	Sensor data, status, power, sync
ESAD Serial Bus	Status, authorizatrion command

Table 1: Legacy communication links - message content.

The functionality of the legacy ABDS user interface, which includes the application of power, the application of sequenced and timed arm-enabling and the reporting of system status, will now be handled by the MAPS User Interface Control Panel (UICP) and the External Maintenance Device (EMD). This allows the ABDS microcontroller assets that were allocated to the ABDS user interface to now be allocated to the FPGA processor interface.

5.5. Strategy for MAPS Compliance

The MAPS control over ABDS has been implemented in a gated approach. A gated approach ensures the MAPS base kit has authorization control over the ABDS arm-enable process but allows the ABDS microcontroller to have autonomy over the countermeasure fire control process. The decision to implement the gated-control approach was primarily driven by the latency for deterministic message delivery over the deterministic ethernet.

TenCate has incorporated a separate processor into the ABDS controller enclosure to implement the MAPS interface. The separate MAPS processor includes the MAPS Deterministic Ethernet Core described in section 4. The MAPS processor manages all ethernet traffic for ABDS and ensures the ABDS processor state machine remains synchronized with the MAPS states and modes. The ABDS-MAPS (ABDS-M) context diagram in Figure 4 is a depiction of the boundaries of the MAPS-revised system and the revised entities that ABDS-M interacts with in the operational environment.

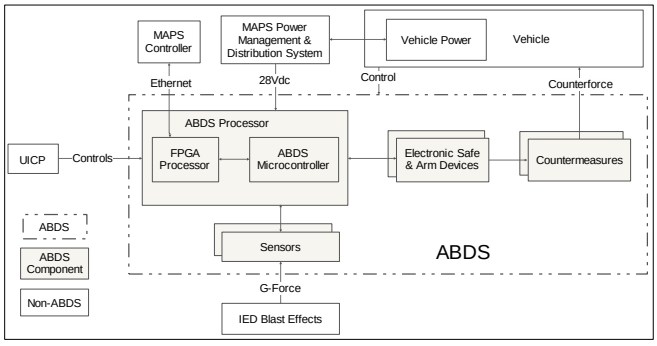

Figure 4: ABDS-M context diagram

The separate processor communicates with the ABDS microcontroller through a combination of serial communications and discrete input/output (DIO). Moding synchronization commands are sent from the MAPS Processor to the ABDS microcontroller over the serial data link. The serial communications also transmit ABDS microcontroller revision data and ABDS microcontroller diagnostic data. During ABDS microcontroller critical processing phases, the serial communications between the MAPS processor and the ABDS microcontroller are suspended. During this time the MAPS processor monitors the ABDS microcontroller moding via the DIO interface. During the suspension, if the MAPS controller commands ABDS to a safe state, the FPGA processor will use a discrete signal to the ABDS microcontroller to command the transition. Serial communications between the FPGA processor and the ABDS microcontroller are re-established upon return to a safe state.

5.6. ABDS Modernization Impacts

The interface between the ABDS microcontroller and the FPGA processor, although minimized, still required changes to 1) legacy software for adaptation of the serial interface that was previously used to communicate with the legacy control & display unit, 2) both legacy hardware and software for the addition of the DIO interface, 3) legacy hardware for the harness wiring for interconnection of the FPGA circuits and 4) the legacy processor enclosure for adaptation to mount the FPGA circuits. Incorporation of the FPGA into the legacy processor footprint allowed for overall reduction of the ABDS-M space claim and system weight by doing away with the legacy display. Primary power and keep-alive power increases for the FPGA circuitry were offset by the removal of the legacy display. Processor shock mounting was not re-evaluated for the 5% increase in processor unit weight. Legacy ABDS system modernization for compliance with the MAPS MOSA framework has not had an adverse effect on the ABDS blast-response timeline.

6. CONCLUSION

This paper presented strategies to modernize a legacy system for open architecture standards. The joint effort between TenCate and DornerWorks to modernize the legacy TenCate ABDS for the MOSA based MAPS platform was used as an example to illustrate the developmental strategy and the corresponding impacts of this strategy on the legacy system. In modernizing the ABDS to be compliant with the MAPS network, TenCate chose to implement a separate processor into their existing system architecture. The separate processor was tasked with implementing MAPS-specific logic using the MAPS Deterministic Ethernet core. The legacy ABDS software was modified in the modernization process to account for the addition of a separate processor, which was expected as a result of choosing this type of approach. This modernization effort has showed that there is not a single perfect approach in modernizing a legacy system, rather a single *best* approach. The best approach should minimize modifications to the legacy system's code base and should be architecturally appropriate for the system. An understanding of MOSA principles should not only guide legacy system modernization but should be used effectively in the creation of new defense acquisition programs. Effective use of these principles will inevitably lead to the success of defense programs.

1. REFERENCES

[1] Chowdhury, M., & Iqbal, M., "Integration of Legacy Systems in Software Architecture", 2004

[2]Deputy Director for Engineering, "Modular Open Systems Approach (MOSA) Reference Frameworks in Defense Acquisition Programs", May 2020 Retrieved from https://ac.cto.mil/wp-

content/uploads/2020/06/MOSA-Ref-Frame-May2020.pdf

[3]CRADA 18-99, Research and Development of MAPS Validation Testing Utilizing the MAPS Development Kit, Cooperative Research and Development Agreement (CRADA) Between Tencate Advanced Armor USA, Inc. and the U.S. Army Ground Vehicle Systems Center.

[4]M. Withun & P. Griffin, Griffin 2.0 - Development & Evaluation of an Occupant Protection Platform with Active-Blast Mitigation and Crew Floor Isolation, Proceedings of the 2016 Ground Vehicle Systems Engineering and Technology Symposium (GVSETS)

[5]MIL-STD-1911A, Safety Criteria for Hand-Emplaced Ordnance Design, 10 July 1998.

[6]FACE™ (Future Airborne Capability Environment) Technical Standard, The Open Group

[7]SOSA™ (Sensor Open Systems Architecture) Technical Standard, The Open Group

[8]OMS/UCI (Open Mission Systems / Universal Command and Control Interface)

[9]VICTORY (Vehicular Integration for C4ISR/EW Interoperability) Standard Specifications

[10]ANSI/VITA 46 VPX

[11]DoD Tri-Service Memo “Modular Open Systems Approaches (MOSA) for our Weapon Systems is a Warfighting Imperative”. January 2019. Retrieved from https://www.dsp.dla.mil/Portals/26/Documents/PolicyAndGuidance/Memo-Modular_Open_Systems_Approach.pdf