Open Architecture Applied to Next-Generation Weapons

Leo J Rose[a], Jonathan Shaver[a], Quinn Young[b], Jacob Christensen[b]
[a]Air Force Research Laboratory, 101 W Eglin Blvd, Suite 304, Eglin AFB, FL, USA 32542;
[b]Space Dynamics Laboratory, Utah State University Research Foundation, 1695 North Research Park Way, North Logan, UT, USA, 84341

ABSTRACT

The Air Force Research Laboratory (AFRL) has postulated a new weapons concept known as Flexible Weapons to define and develop technologies addressing a number of challenges. Initial studies on capability attributes of this concept have been conducted and AFRL plans to continue systems engineering studies to quantify metrics against which the value of capabilities can be assessed. An important aspect of Flexible Weapons is having a modular "plug-n-play" hardware and software solution, supported by an Open Architecture and Universal Armament Interface (UAI) common interfaces. The modular aspect of Flexible Weapons is a means to successfully achieving interoperability and composability at the weapon level. Interoperability allows for vendor competition, timely technology refresh, and avoids costs by ensuring standard interfaces widely supported in industry, rather than an interface unique to a particular vendor. Composability provides for the means to arrange an open end set of useful weapon systems configurations. The openness of Flexible Weapons is important because it broadens the set of computing technologies, software updates, and other technologies to be introduced into the weapon system, providing the warfighter with new capabilities at lower costs across the life cycle. One of the most critical steps in establishing a Modular, Open Systems Architecture (MOSA) for weapons is the validation of compliance with the standard.

Keywords: MOSA, Open Architecture, Flexible Weapons, Interoperability, Composability

1. INTRODUCTION

An important aspect of AFRL's Flexible Weapons initiative is the ability to create a family of weapons using common functions and components. While there may be a penalty paid in individual unit costs or performance compared to a closed, tightly coupled / integrated weapon, it is believed that the overall cost of ownership of the Open Architecture based weapon system will be greatly reduced, considering the ease of component interchanges, technology refresh rates, product improvements, hardware testing, and software updates. In order to achieve this desired end-state, AFRL needs to create a Plug-n-Play environment, based on modularity, in a comprehensive system of systems through the establishment of an Open Architecture.

Modularity requires encapsulating functionality within a physical unit with clearly defined interfaces. This can present a challenge when functions in different regions may be tightly coupled. A sensor, for instance, may be separate from the processing element which needs to be "aware" of the sensor's characteristics in order to accurately transform the sensors raw data into measurement information. One form of modularity would allow the definition of a distributed subsystem, in which functionality is distributed across physical elements with inter-module communication described in generic terms, with the understanding that functionality at either end of the communication paths would be responsible for transforming data into a usable form by other subsystems. This concept enables the definition of a flexible system architecture while protecting proprietary data that may need to be exchanged between nodes in a larger system.

2. OPEN ARCHITECTURE

Modular designs encapsulate selected functionality into separate physical units with clearly defined interfaces. What functions are encapsulated and where physical boundaries are drawn are driven by the goals for modularity. Clearly defined goals for modularity help design teams to understand how to evaluate different architecture options. Drivers for modularity may include technology risks, new technology insertion, mission adaptability, maintenance, cost, and protection of vendor intellectual property. A modular open system architecture reduces overall costs through reduced non-recurring engineering, reduces development timelines by building on existing infrastructure, reduces the scope of and increases automation of integration and tests, improves upgradability and maintainability through modularity, and

Open Architecture/Open Business Model Net-Centric Systems and Defense Transformation 2014,
edited by Raja Suresh, Proc. of SPIE Vol. 9096, 90960K · © 2014 SPIE
CCC code: 0277-786X/14/$18 · doi: 10.1117/12.2055266

improves inter-compatibility within a system of systems. Characteristics of an open architecture include packetized, networked data transfer which allow data to seamlessly move throughout the network, open standards that allow each component within the system to work together, and unit and system verification / certification, allowing independent development of components.

2.1 Architecture Overview

A system architecture for an aircraft and weapon can be depicted as in Figure 1 below. A weapon system, consisting of a sensor, aperture, processor, payload, propulsion, and actuators must physically and logically connect to the aircraft to make up a system of systems. In this case, the suspension and release equipment provide for the physical connection and the stores management system provides for the logical connection.

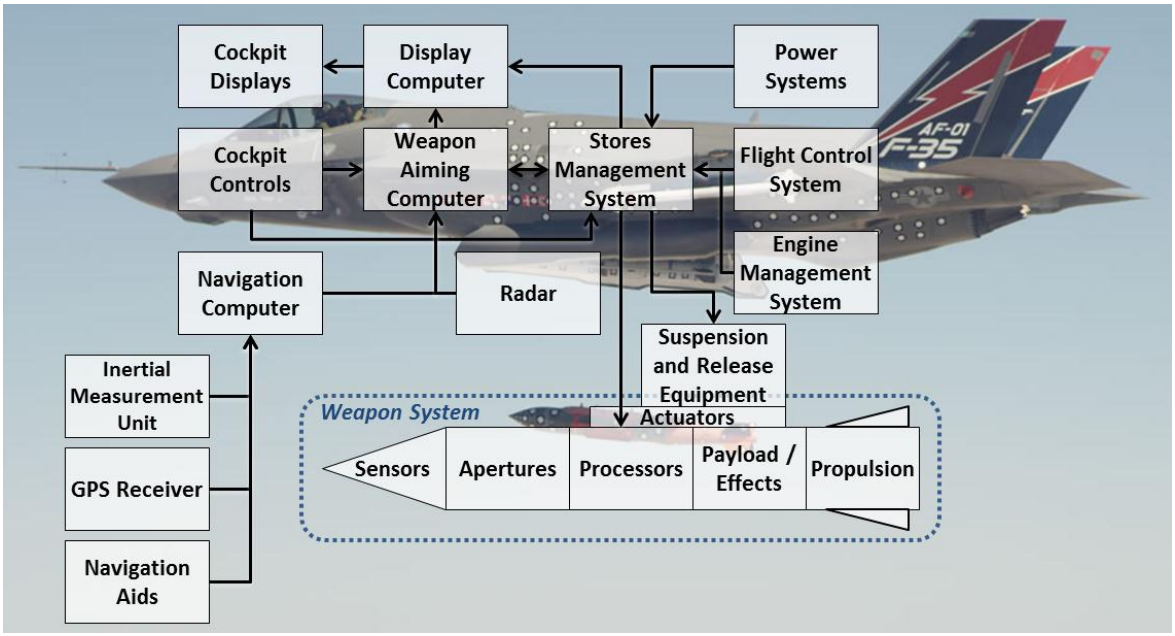


Figure 1. System Architecture for Aircraft / Weapon

2.2 Incorporating a Modular Open System Architecture

A Modular Open System Architecture (MOSA) can enable future "plug-and-play" weapons systems. MOSA provides the standards and infrastructure for the capability. The vision for a plug-n-play weapon system incorporates several features:

- Weapon systems have standard interfaces to the aircraft and an electronic data sheet that includes a machine readable (electronic data) interface control document (ICD)
- The Stores Management System (SMS) is able to query all connected munitions (weapons), receive their electronic data sheet, and automatically populate the type, capabilities, data needs, and other parameters associated with the munition
- The SMS knows from the electronic data sheets the parameters needed to define performance enveloped, operational constraints, and display parameters
- The Display Computer uses the data in the SMS to generate munition-specific cockpit displays (the munitions electronic data sheet includes sufficient information to enable this, parameterized to maintain portability and adaptability)

MOSA provides the common interface and the ability to self configure and self recognize, essentially automating tasks that are done manually or semi-manually (one platform at a time) in current systems. This is shown in Figure 2.

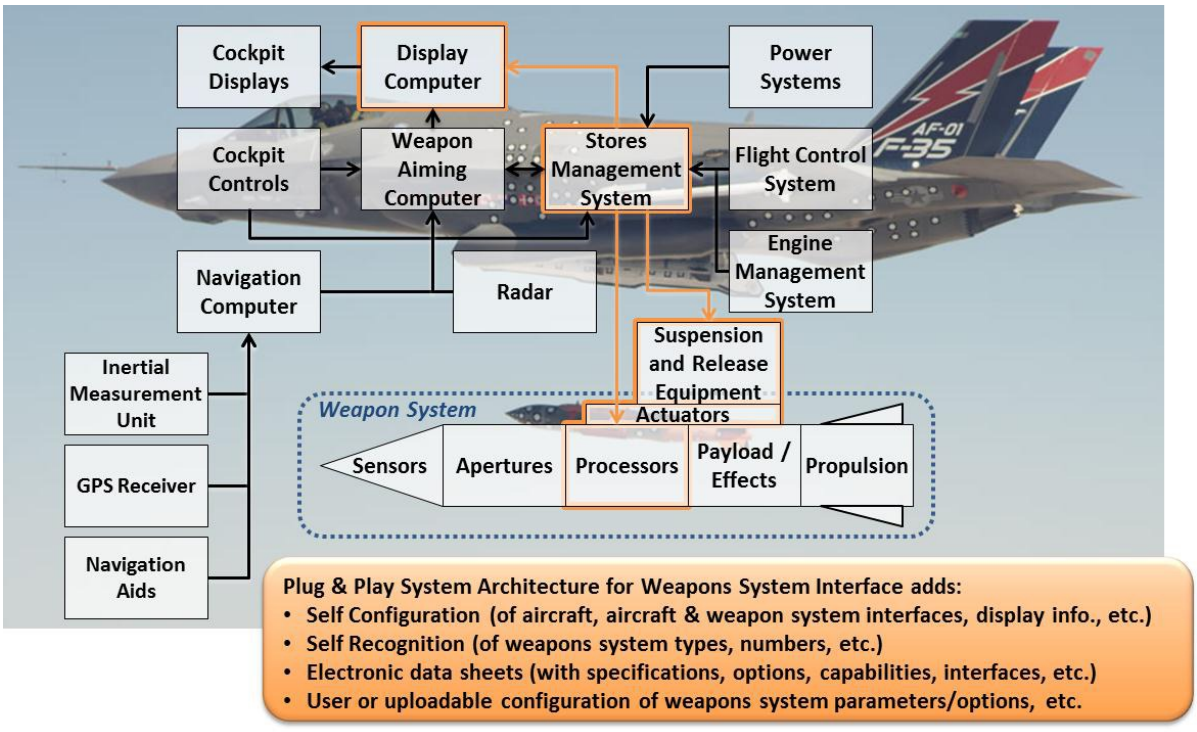


Figure 2. System Architecture and the Stores Management System

2.3 Protecting Intellectual Property Within a MOSA

One major concern in dealing with MOSA is the notion that an open system removes the ability for businesses to maintain their competitive edge and that their intellectual property will be given away to others. The change to an open system is better characterized as defining the interfaces, not the activity inside the functional components within the system. In an open system, the interfaces must meet the standards – as shown with the green arrows in the diagram below, but inside the "box" you could have either a proprietary or non-proprietary component, depending on what fills the need the best. Pictured in Figure 3 below, is a random division of functions that can all work together because of the open interface, but that could be either proprietary or non-proprietary

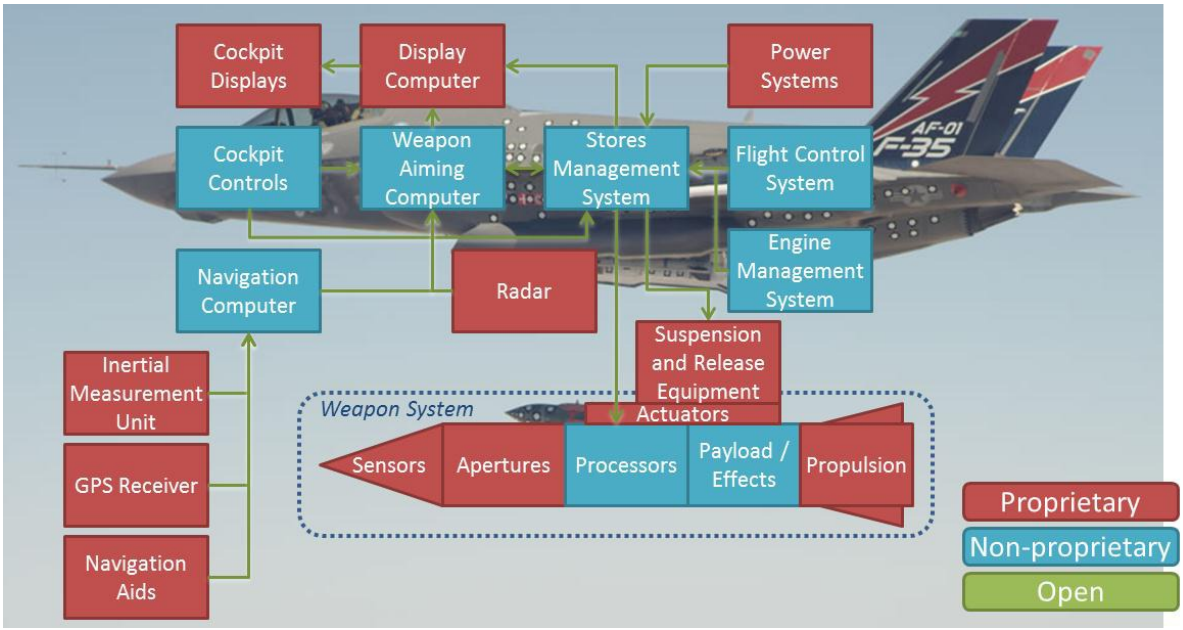


Figure 3. System Architecture and Proprietary / Non-Proprietary Functions

Through a MOSA approach, interchangeability, reduced non-recurring engineering (NRE), and other benefits can be realized without eliminating the intellectual property inside the individual components. This protects the competitive edge and incentives to innovate that are important for business and Government.

2.4 Functional Decomposition

Functional decomposition is a method for determining logical separation of functions into groups of physical modules. Careful functional decomposition reduces coupling between modules, and enhance the flexibility of the overall system, as illustrated in Figure 4. For Flexible Weapons, the architecture under development will look at logical functional and physical boundaries to separate the modules. A notional architecture, illustrated in Figure 5, shows how modules can be organized to enable the desired composability of the system. Various interchangeable modules are available to compose a system with the needed seeker, guidance, effect, and aerodynamics for a particular mission type.

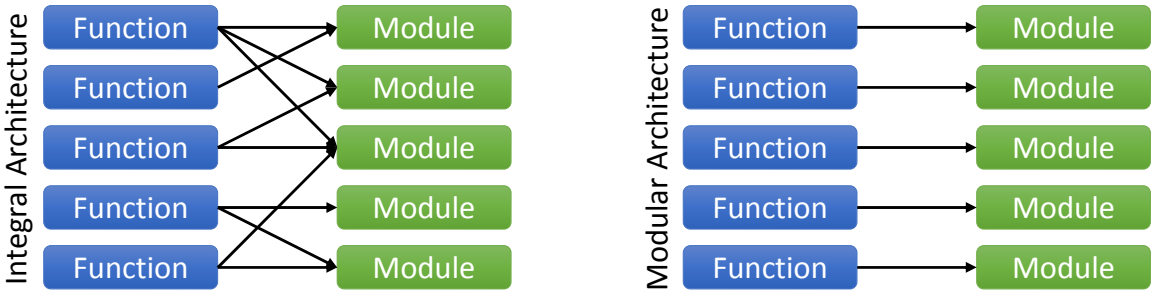


Figure 4. Functional Decomposition Provides a Way to Define Physical Modules Whereby Coupling is Minimized and Modularity is Maximized

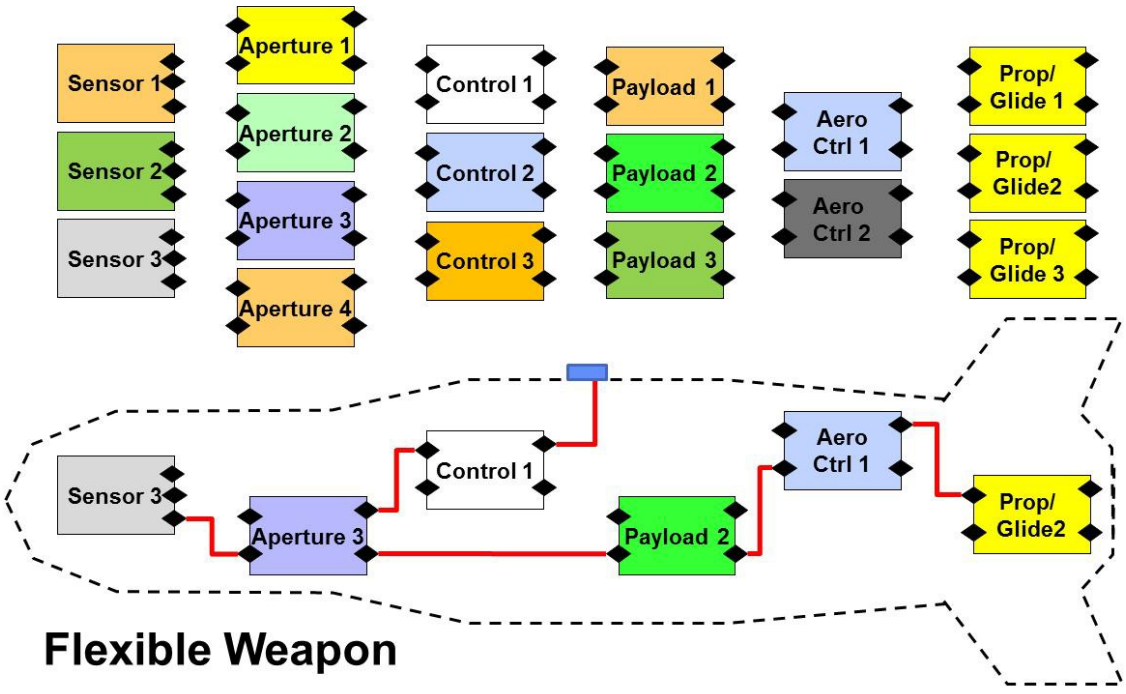


Figure 5. Flexible Weapon Notional Functional Decomposition

2.5 Architecture Development Plan

In order to successfully develop an open system design for weapons, an iterative, two-cycle approach will be applied. The first cycle will include the first revision of architecture development, while demonstrating modularity with bench-top hardware and preliminary software. The second cycle matures the architecture while implementing the first revision in the second demonstration. This two-cycle approach is shown graphically in Figure 6 below.

The first demonstration provides lessons learned early enough to feed into the architecture development, and builds momentum, while the second architecture provides a reference implementation and demonstration of the architecture with more flight-like components in preparation for transitioning to the next phase.

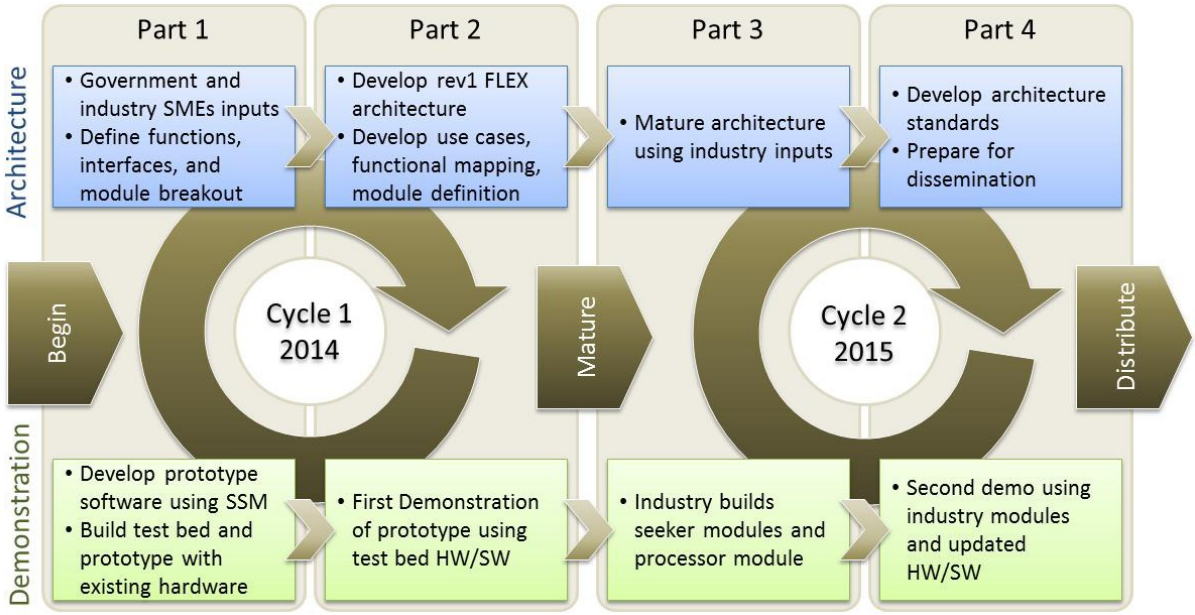


Figure 6. Flexible Weapon's Two-Cycle Development Plan

3. MOSA

Modular Open System Architecture (MOSA) provides Flexible Weapons the framework for incorporating desired attributes. The building blocks of the MOSA puzzle include software modularity, hardware modularity, open standards and interfaces, and a networked data transport mechanism. Figure 6 illustrates these building blocks and the kinds of attributes they provide.

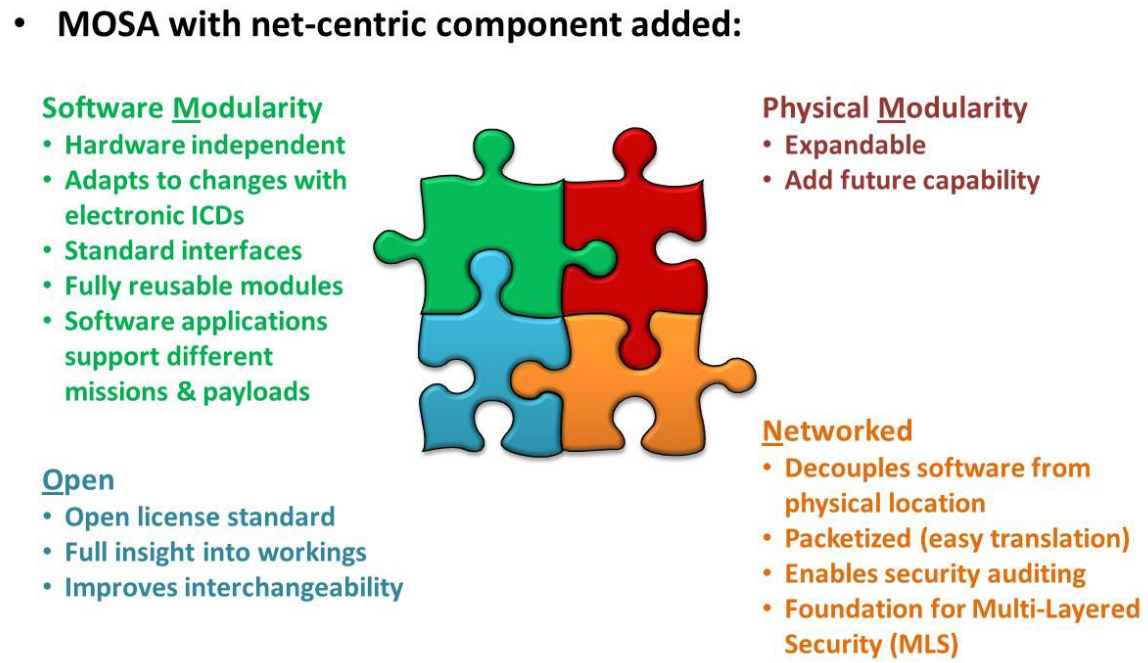


Figure 7. MOSA Building Blocks

3.1 Software Development and MOSA

In following a MOSA approach, AFRL is building on the billions of dollars spent by the computer and networking industry that has already matured the architecture and the tools. AFRL can adopt methods for software development, data transfer, and open interfaces from the best proven practices of industries that already are familiar with this approach, such as PCs, cell phones, etc. This implementation does not result in scraping all the software we've already built, only changing how it works together and with the hardware so that we improve the reusability and build on a common infrastructure.

One of the improvements in the software architecture comes from using middleware. Having the middleware in the software stack decouples the software applications (algorithms, controls, etc.) from the hardware, as illustrated in Figure 8. That means the software is more portable (can be used on different hardware) and doesn't have to change every time the hardware is changed. Reuse of existing software allows more experience with individual code (find bugs, know how it works & reduce learning curve), reducing risk, as well as reducing testing time and costs (because the software changes less frequently).

- **Building on billions of dollars of investment in smart architecture**

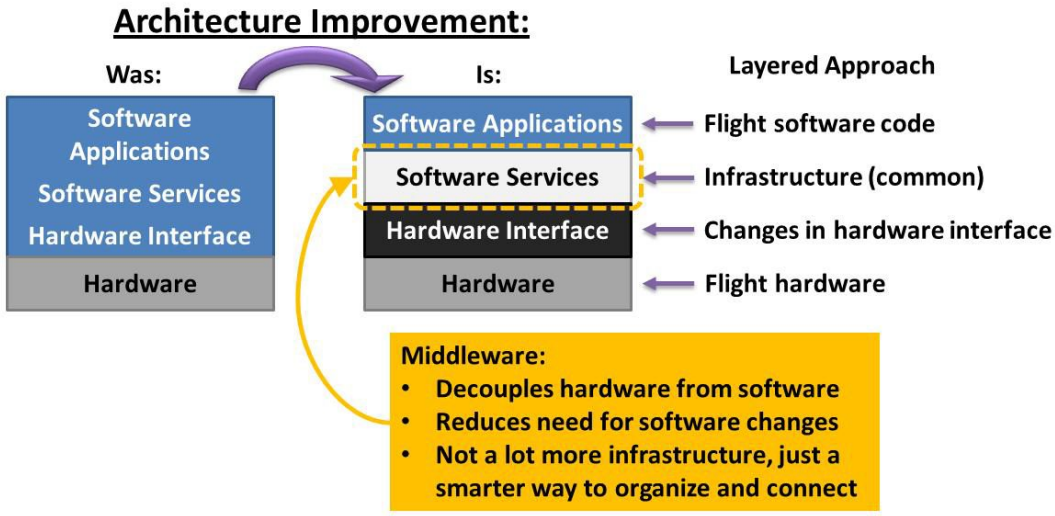


Figure 8. Smarter Software Architecture Improves Software Reuse

3.2 MOSA Improves Software Reusability

We often say that roughly 80% of software is reused from one program to another. Unfortunately, that typically means that 80% of each software component is reused, while the remaining 20% requires changes to adapt to new hardware. In the old paradigm, introducing new hardware for a new mission means the interfaces between hardware and software changes (they are usually hard-coded, at least to some extent). This is costly in recreating the hardware/software interface – this is like creating Facebook and then rebuilding the internet and rewriting a web browser just to access it. Businesses have figured out how to use the existing infrastructure on the internet and web browsers to eliminate the interface problems.

The industry is very good at developing software the traditional way but there is a better way that will further reduce the cost of software. Software is often the highest technical risk area and by reducing the amount of software that is written for each system, the risk is reduced. With software reuse, there is a reduction in both the non-recurring engineering (NRE) and the testing costs.

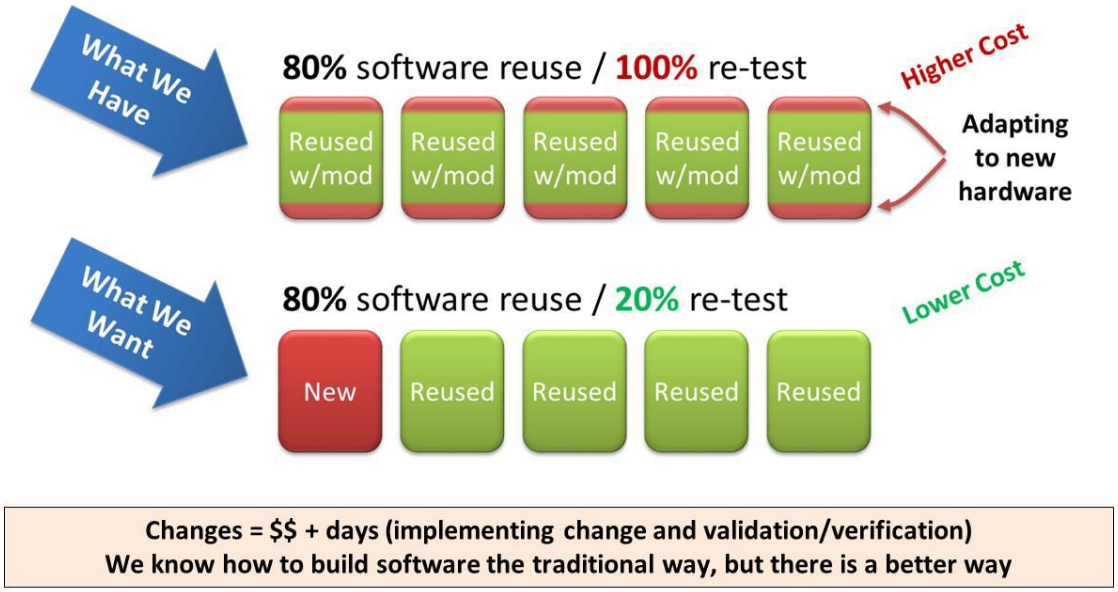


Figure 9. Software Reuse Saves Time and Money

3.3 Packetized Data Transfer Improves Modularity

Packetized data transfer is one way to modularize the data path, and when coupled with encapsulation, significantly simplifies how we handle data. This is the way terrestrial data networks work – like sending email from a smart phone, a desktop machine, or a web browser: each system can send data across different kinds of networks and still work seamlessly with each other.

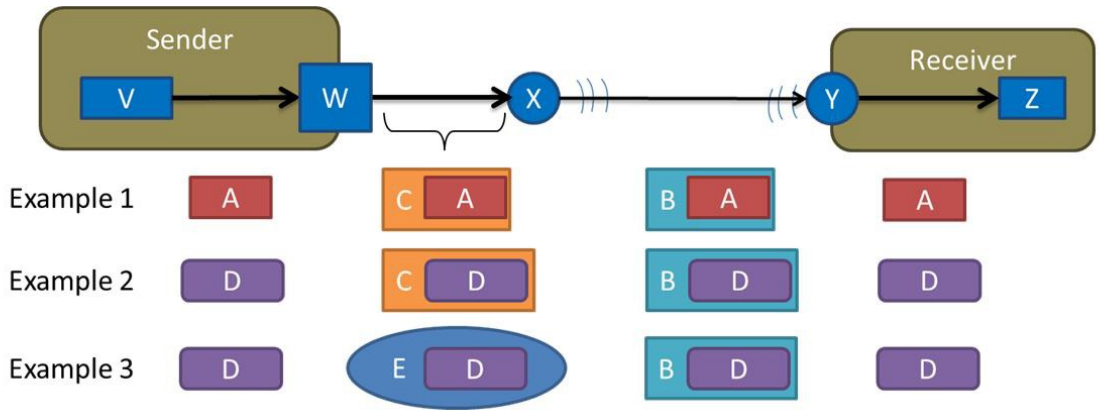
Figure 10 illustrates how packetized data transfer enables a modular data path.

As a first example of a packetized data path, the data packet, "A" is generated from the source, or "Sender", is encapsulated when transferred from the internal network ("v" to "w") to the first external network piece (going from "w" to "x"). When transferred to the next external network piece ("x" to "y") there is a different protocol and packet type, but "A" is just encapsulated in the new type ("B"). Upon reaching the destination it is delivered as "A", just like when it was generated.

In a second example you can see the modularity: if you change the source packet type from "A" to "D", the rest of the network can remain the same and still get the data where it needs to go.

In a third example, changing the packet type for the "w" to "x" network from "C" to "E" type packets, the encapsulated "D" packet doesn't have to change at all. It just works.

The architecture minimizes change propagation and maximizes compatibility – which is why the internet works this way, too.



- **Use packet encapsulation to minimize change propagation**
- **Packet encapsulation enables modularity in network communication**
- **Packet formats can change, other formats stay the same**

Figure 10. Packet Encapsulation

The basic architecture shown in the previous illustration may look more familiar if you look at a container transportation system, illustrated in Figure 11 below. They are the same basic architecture.

Packaging bits into sub frames and frames, then creating packets out of them is like putting books (data) into boxes (sub frames) and boxes into pallets (frames) then pallets into a container (packet).

Once packetized, the method for delivery can be varied from trucks, to cranes, to boats, to trains, and the same packet just rides along whatever the method. It can then be unpacked at the end and is still the same data.

To show the modularity, which allows upgradability or variability, imagine the ship is obsolete and is replaced by a containerized aircraft. The transportation process doesn't change, and the packets don't change. The books don't care if they are transported via boat or plane, they just get where they need to go.

Once the system is put together, the end user doesn't need to understand the complexities of the system. It just works.

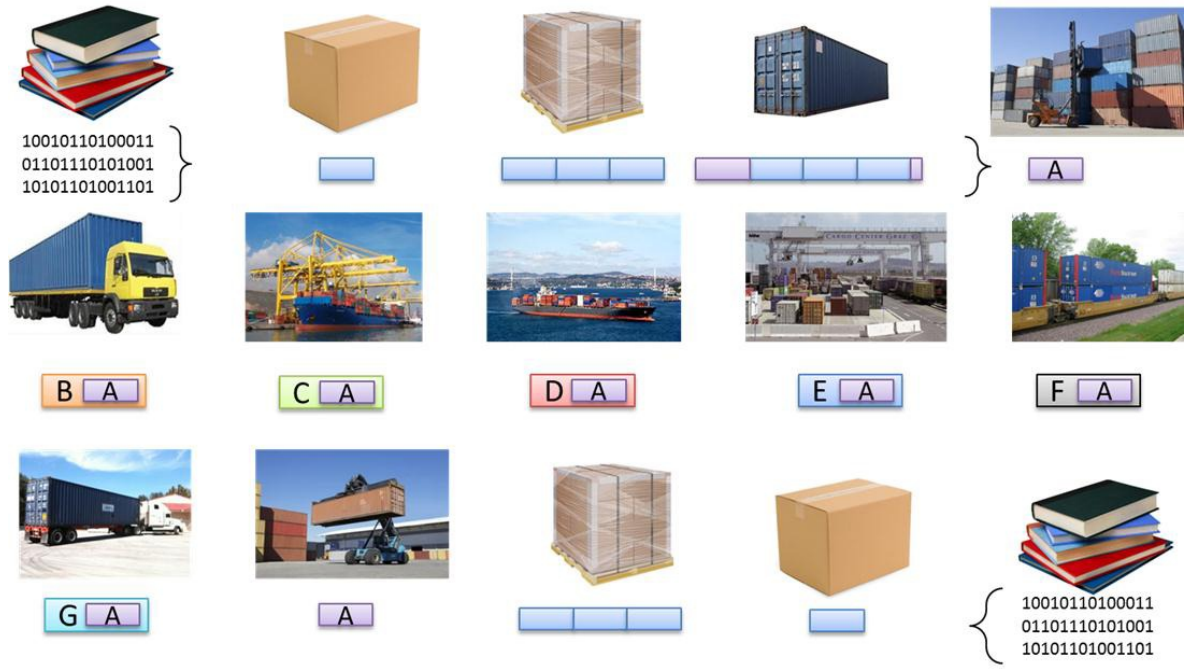


Figure 11. Packet Movement Through the System

3.4 MOSA Can Provide Additional Benefits

Figure 12 illustrates several additional benefits and characteristics of a MOSA-based system.

In the left side of the figure, you can see that the MOSA portions of the system can work with heritage components (actual hardware, or networks, or other software) as long as the heritage interface (red line) is provided by the MOSA components. This flexibility is helpful in rolling a new architecture into an existing system.

Other added functionality, shown on the right side of Figure 12, is also possible in a MOSA system. In this case, the link between A and B is replaced by software applications that look like A and B, but enable splitting out A's interactions into multiple B units, or adding a logging function to the network traffic.

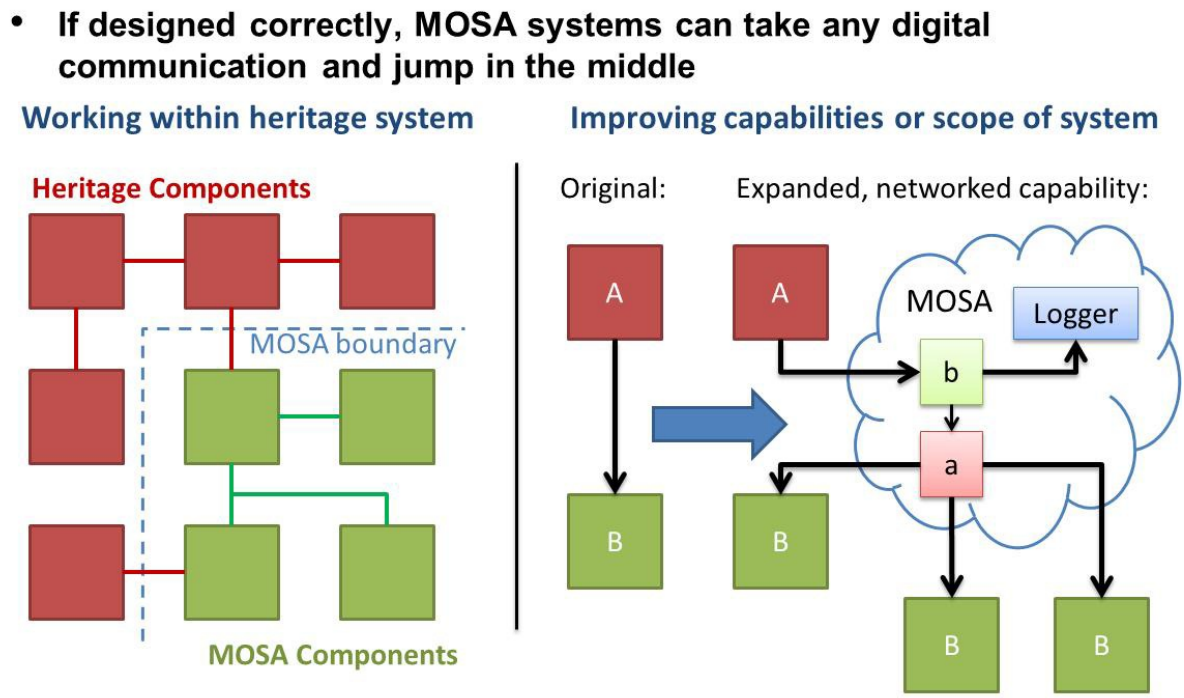


Figure 12. MOSA-Based System

4. CONCLUSION

The overarching objective for Flexible Weapons is to replace current inventory weapons that will not fully utilize the increased capabilities of 6th generation platforms, with a single weapons kit made up of flexible, open architecture components. Flexible Weapon will develop a common architecture to enable modular subsystems to achieve flexible weapons capability while allowing technology refresh at the pace of technology discovery in an affordable and sustainable design. The various combinations of weapons to address multiple missions must be 100% compatible with 6th generation delivery platforms (fighters, bombers, RPAs) and backwards compatible with 4th and 5th generation platforms.