Enabling Interoperability Via Software Architecture

John A. Hamilton, Jr., Ph.D.
Lieutenant Colonel, US Army
Joint Forces Program Office
Mail Code 05J1, SPAWAR
4301 Pacific Highway
San Diego, CA 92110-3127
drew@drew-hamilton.com

Jeanne L. Murtagh
Major, US Air Force
Software Professional Development Program
School of Systems and Logistics
Air Force Institute of Technology (AFIT)
Wright-Patterson AFB, OH 45433-7765
murtagh-jl@acm.org

Abstract

This paper will address the critical role of software architecture in achieving large-scale system interoperability as well as initiatives underway to promote architectural-based interoperability solutions for the Unified Commands. Software architecture is the means to define systems composed of systems. This definition is critical to achieving interoperability. Joint Publication 1-02 defines interoperability as "the ability of systems, units or forces to provide services to and accept services from other systems, units or forces and use the services to enable them to operate effectively together [JP 1-02, 1994]."

1. Introduction

In order to achieve interoperability, compatible systems, doctrine and policy must exist. The technical challenges to interoperability can be daunting -- particularly when a new requirement is established that requires existing (legacy) systems to interoperate. Military forces do not operate as a fully connected graph; modern warfare does not require every system to interoperate. Joint doctrine is the key to determining interoperability requirements. Doctrine tells us how to fight and how we fight determines interoperability requirements. Policy sets the bounds on acceptable doctrine.

[Alberts *et al.* 1999] discuss military capability packages in terms of DOTML-P (doctrine, organization, training, materiel, leadership, and personnel.). From an interoperability standpoint, it makes sense to focus on doctrine, organization and materiel, which Alberts et al. attribute to the characterization used by US Atlantic Command (now US Joint Forces Command.) The development of meaningful interoperability requirements is based upon:

- Doctrine: to identify why we interoperate.
- Organization: to determine who interoperates.
- Materiel: to provide the technical "how" we interoperated.

Doctrine and organization determine operational interoperability requirements. These operational interoperability requirements determine system interoperability requirements. Materiel solutions to meet system interoperability requirements must span programs, services and system versions.

Interoperability implies the existence of diverse systems that need to exchange data and services. Much is written about "systems of systems." A prime example of a system of systems is the DOD Global Command and Control System (GCCS). GCCS integrates several applications. The DOD GCCS program is managed by the Defense Information Systems Agency. Each service has its own service implementation of GCCS with service-unique functionality and applications added.

System interoperability is what makes heterogeneous systems of systems a reality. All of these systems are composed of hardware and software. Hardware is not easily changed. Furthermore, fielded hardware systems often cannot be wholly replaced. Therefore as a practical matter, interoperability is more easily achieved through software and so therefore that is the focus of this paper.

Diverse hardware-based communications systems require an overall software architecture in order to interoperate. As noted in IEEE Standard 12207.0-1996 *Software Lifecycle Processes*, software architecture describes the top-level structure of the over-arching system and describes the software components [IEEE 1998]. Specifically, developers adhering to the standard are required to development and document a top-level design for the interfaces external to the software item and between the software components of the software item. This is an essential first step in achieving interoperability between any two systems.

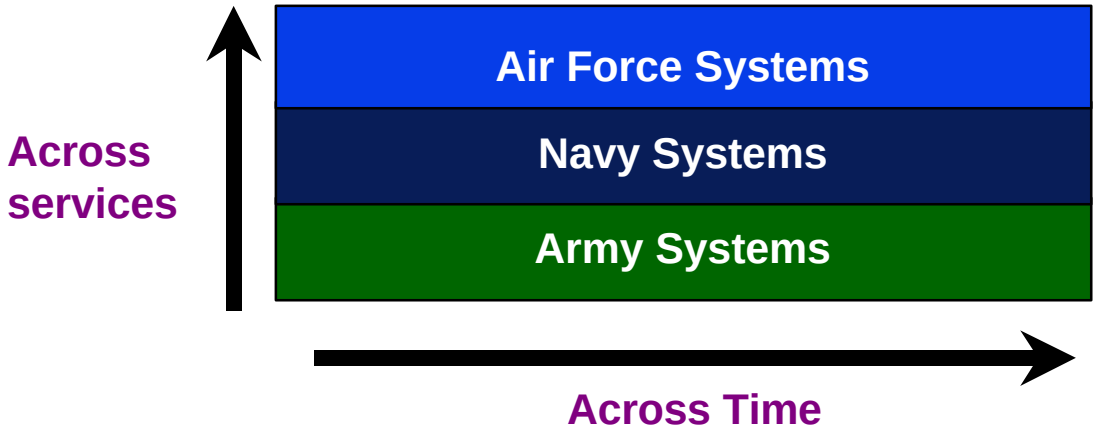

Figure 1. Two Dimensions of System Interoperability.

Software architecture development and implementation is complicated when the systems belong to different organizations. Joint interoperability, that is, interoperability between different services, is challenging. Sustained joint interoperability cuts across two dimensions: laterally between services and horizontally over time as shown in Figure 1. Once two systems become interoperable, there is no guarantee that they will remain interoperable if they are upgraded asynchronously. Recognizing these challenges, the C2 system acquisition commands of the Army, Navy and Air Force have developed an innovative initiative to promote joint interoperability.

2. **A Joint Interoperability Initiative from the Services.**

The commanders of the service C2 acquisition centers, Communications and Electronics Command, Fort Monmouth (CECOM), Space and Naval Warfare Systems Command, San Diego (SPAWAR), Electronic Systems Center, Hanscom, AFB (ESC), formed the Joint Command and Control Integration Interoperability Group (JC2I2G). The JC2I2G exists to promote joint interoperability and change processes and structures by initiating "bottom up" change to implement Joint C2 integration and interoperability, and by supporting the unified commands in resolving interoperability issues of service-specific systems. Recognizing the pivotal role the US Joint Forces Command (USJFCOM) as the Joint Force Integrator, the Director, J6 of USJFCOM serves as principal member of the JC2I2G.

The JC2I2G proposed and the Under Secretary of Defense for Acquisition and Technology, Dr. Jacques S. Gansler, approved the establishment of the CINC Interoperability Program Offices (CIPO) at each C2 acquisition center and the establishment of the Joint Forces Program Office (JFPO). The CIPOs now play a major role between the originators of joint requirements and the designers of service C2 systems. The primary purpose of the Joint Forces Program Office is the horizontal integration of the CIPO efforts across the Unified Commands in direct support of US Joint Forces Command. The support relationships are outlined in Figure 2.

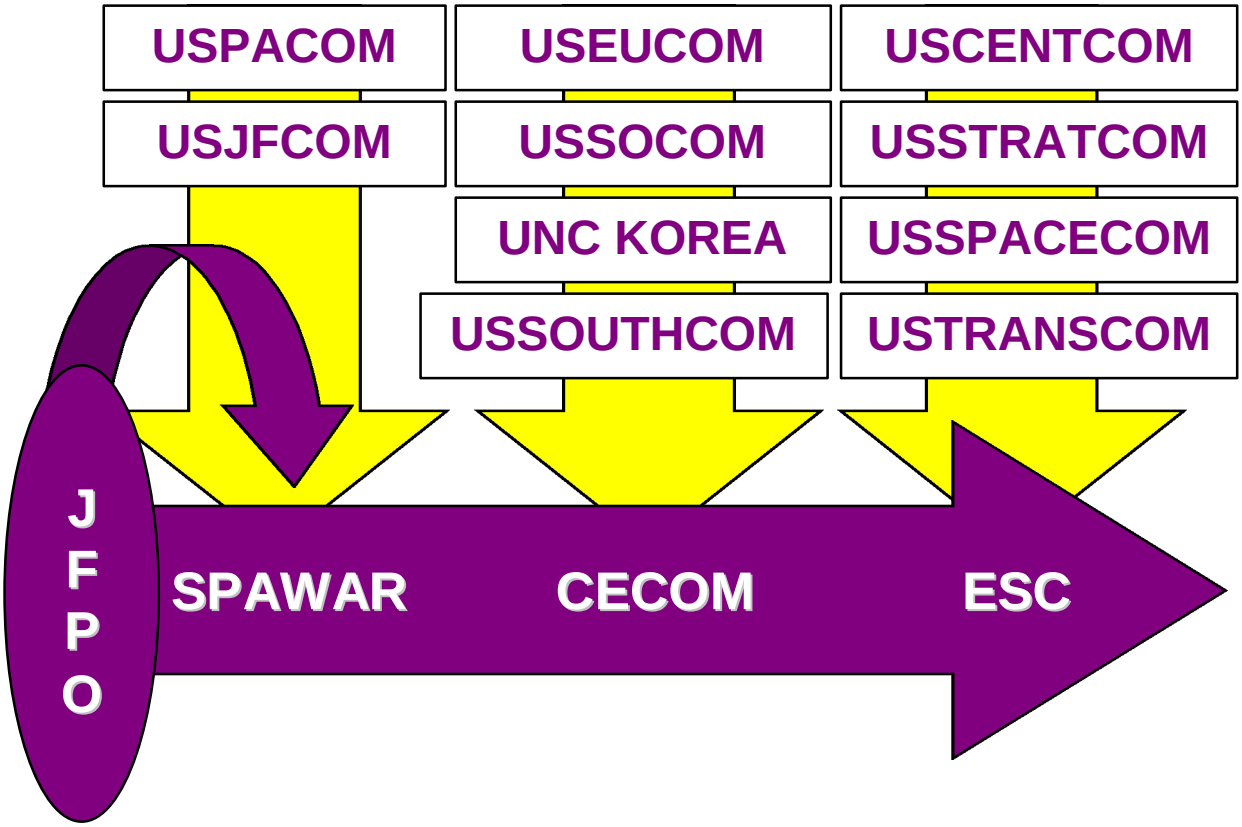

Figure 2. Alignment of Unified Commands and Supporting CIPOs.

Although each CINC is supported by a single CIPO, the reality is that the interoperability problems will be solved in the system commands, regardless of which CIPO staffs the action. For example, consider an Army / Air Force interoperability program raised in the Pacific Command. The SPAWAR CIPO takes the issue back to the appropriate Army and Air Force PMs for action as illustrated below in Figure 3. It is not terribly important which CIPO initiates the action. The key capability is the reach back to the service program managers.

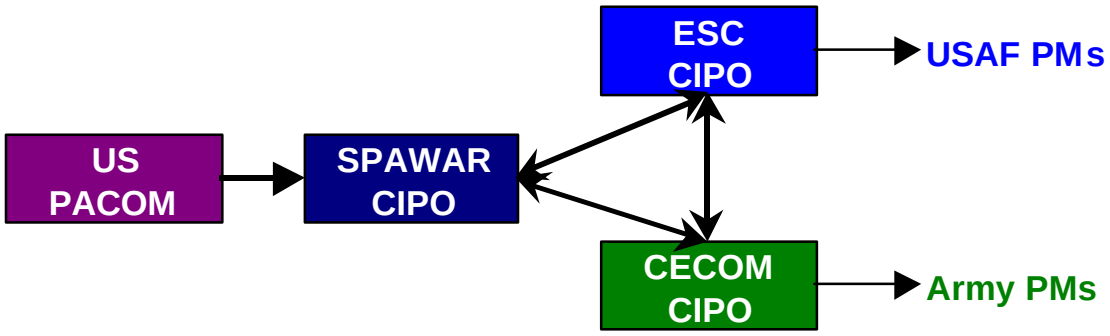

Figure 3. Outline of CINC/CIPO/PM Interaction.

The JFPO is a tenant of SPAWAR, *but* a JC2I2G organization. The JFPO serves to:

- Act as CIPO coordinating authority to identify cross-CINC joint interoperability issues and synchronize cross-CINC solutions where feasible.
- Provide technical/engineering support to USJFCOM in its role as executive agent for Joint Forces Integration, to include providing USJFCOM with technology insertion recommendations.
- Support USJFCOM in assessing joint interoperability during MNS/ORD/CRD requirements and milestone reviews.
- Support USJFCOM in tracing future C4 systems requirements to other CINC needs and solutions.
- Oversees JFCOM CIPO support.
- CIPO liaison to ASD(C3I), Joint Staff, DISA and other Defense agencies.

The special relationship between the JFPO, the Joint Interoperability Test Command (JITC) and the JFCOM J6 provides the ability to do better requirements engineering in the development of capstone requirement documents (CRDs) and operational requirement documents (ORDs). At the request of the JFCOM J6, the JFPO evaluates proposed interoperability requirements via a three parallel processes.

1. JFPO conducts a technical evaluation of the interoperability requirement.
2. The JFPO provides the interoperability requirement to each of the service CIPOs for technical review.
3. The JITC evaluates the testability of the proposed interoperability requirement.

The role of the JITC deserves special attention. Evaluating the testability of a requirement is sound engineering practice and must be accomplished before moving into the design phase.

The CIPO/JFPO structure provides the Unified Commands with the engineering expertise to catch and correct errors in requirements before these errors propagate throughout the rest of the system. As the CINC staffs develop and refine *operational requirements*, technical expertise is required to develop the *system requirements*. As interoperability requirements between systems are developed and validated, a high-level software architecture is needed for the system acquisition commands to develop *designs* that will be interoperable.

3. From Operational Requirements through System Requirements to Architectural Design

Requirements engineering is the first step towards achieving system interoperability. Requirements engineering provides the basis for a software architecture which is essentially a high-level design. It can be argued that the next logical step is horizontal integration as illustrated in Figure 4.

Figure 4. From Requirements Engineering to Horizontal Integration.

The inclusion of Joint Information Exchange Requirements (JIERs) in operational requirements documents (e.g. ORDS and CRDs) is required by CJCSI 3170 [CJCSI 1999]. From these JIERs, system performance parameters are identified. These parameters are used to derive many of the system's technical requirements. These requirements must be incorporated into the system's architectural design. It is particularly important that the *software* architectural design be robust and flexible since most future requirement changes will need to be implemented through extension of this design.

How does one evaluate requirements development? [Kotonya & Sommerville 97] provide some guidelines based on the model in Figure 5.

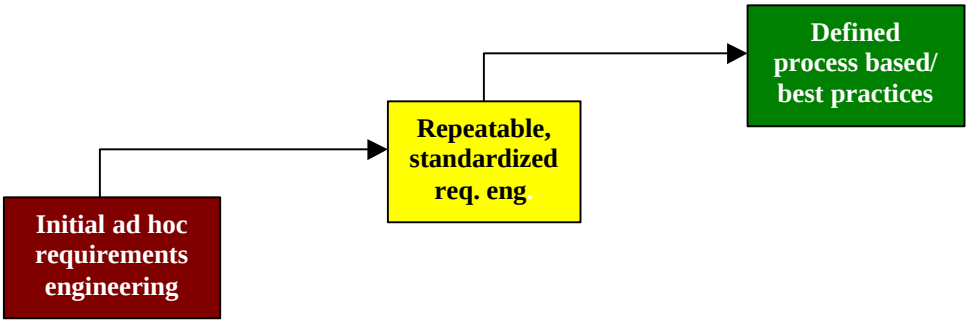

Figure 5. Requirements Engineering Process Improvement.

Kotonya and Sommerville describe the following three levels of requirements engineering maturity:

Level 1:

- Define a standard document structure
- Uniquely identify each requirement
- Define policies for requirements management
- Use checklists for requirements analysis

Level 2:

- Use scenarios to elicit requirements
- Specify requirements quantitatively
- Use prototyping to animate requirements

Level 3:

- Reuse requirements
- Specify systems using formal specifications

It is difficult to assign DOD requirements generation a "level score" nor is it important to do so. In some areas, such as standard document structures, policies, and using scenarios, the DOD excels. The effective use of formal specifications and checklists varies from organization to organization. The successful development of requirements, evaluated for their testability, technical feasibility and ability to satisfy operational needs leads to the development of a high level design.

As noted in IEEE Standard 12207.0-1996 *Software Lifecycle Processes*, software architecture describes the top-level structure of the over-arching system and describes the software components [IEEE, 1998]. Specifically, developers adhering to the standard are required to develop and document a top-level design for the interfaces external to the software item and between the software components of the software item. This is an essential first step in achieving interoperability between any two systems. Software architecture is the high-level design developed from the requirements. [Hofmeister *et al.* 2000] write that software architecture is the purposeful design plan of a system.

The definition of software architecture above is somewhat different what appears in many DOD architecture efforts. There is a strong school of thought that architecture should capture every detail about every subsystem. This is low-level design, not high-level design. Further the costs of such an architecture strategy should not be underestimated. In a recent architecture project, using the current DOD-approved system to produce the minimum set of required products, it took more than 2100 man-hours to document 100 systems.

As [Bass *et al.* 1998] point out, "although computer programs can be combined in more or less infinite ways, there is something to be gained by voluntarily restricting ourselves to a relatively small number of choices when it comes to program cooperation and interaction." Simpler is often better from a design standpoint.

One way to simplify the complexity of the system interactions is to reduce the number of external interfaces between systems. Consider the example in Figure 6. The diagram on the left shows a collection of "systems of systems." The "fishbone diagram" over GCCS-M represent

the subsystems that together make up GCCS-M. For METOC and GCCS-M to interoperate requires external interfaces unless the systems are integrated. The interfaced systems on the left have $(N^2-N)/2$ possible interfaces. On the right is an illustration of SPAWAR's plan for an integrated product line. An integrated product line means the fleet receives a single product delivery, with no after-delivery interfaces between systems. Horizontal integration is achieved by low-level design which conforms to the software architecture

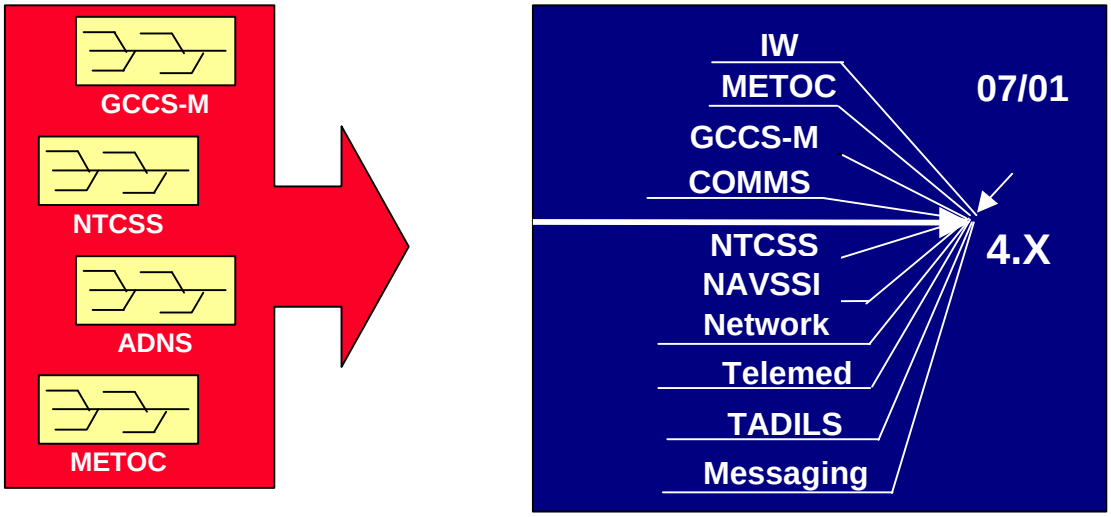

Figure 6. Horizontal Integration Example.

4. **Conclusions**

We have described the role of software architecture in the achievement of joint interoperability. We discussed the role of the CIPOs and in particular, the JFPO, in brokering interoperability solutions for the Unified Commands from the service C2 system acquisition commands. In this capacity, the JFPO aids in the transformation of CINC-generated interoperability requirements into architectural designs by the system acquisition commands. Bridging the gap between the unified commands and the service C2 system acquisition commands makes it possible to integrate service C2 systems into interoperable systems of systems.

Currently, program managers (PMs) field individual systems to support service components. In the near-term we can expect PMs to field integrated product lines to support their service components. When integrated product lines become the norm rather than the exception, we can expect the application of horizontal integration in the joint arena.

The Joint Forces Program Office, with the Joint Interoperability Test Command and the Joint Forces Command are working together to improve joint interoperability requirements. Improved requirements can provide the basis for a software architecture that is the first step towards achieving horizontal integration.

5. **References**

[Alberts *et al.* 1999] David S. Alberts, John J. Garstka, Frederick P. Stein, *Network Centric Warfare: Developing and Leveraging Information Superiority*, DOD C4ISR Cooperative Research Program, National Defense University, Washington, D.C, 1999.

[Bass *et al.* 1998] Len Bass, Paul Clements, Rick Kazman, *Software Architecture in Practice*, Addison-Wesley, Reading, Mass., 1998.

[CJCSI 1999] Chairman of the Joint Chiefs of Staff Instruction 3170.01A, *Requirements Generation System*, The Pentagon, 10 August 1999.

[Hofmeister *et al.* 2000] Christine Hofmeister, Robert Nord, and Dilip Soni, Applied Software Architecture, Addison-Wesley, Reading, Mass., 2000.

[IEEE, 1998] IEEE Standard 12207.0-1996, *Software Lifecycle Processes*, the Institute of Electrical and Electronics Engineers, New York, 1998.

[JP 1-02, 1994] Joint Publication 1-02, *DOD Dictionary of Military & Associated Terms,* 1994.

[Kotonya & Sommerville 97] Gerald Kotonya and Ian Sommerville, *Requirements Engineering*, John Wiley & Sons, New York, 1997.

6. **Authors**

Lieutenant Colonel John A. (Drew) Hamilton, Jr., US Army, is the Director of the Joint Forces Program Office. Previously he served as the Research Director for the Department of Electrical Engineering and Computer Science at the US Military Academy, as Chief of the Ada Joint Program Office. Chief, Officer Training Division at the Computer Science School, Fort Gordon. Lt.Col. Hamilton has a B.A. in Journalism from Texas Tech University, where he was commissioned in Field Artillery; an M.S. in Systems Management from the University of Southern California and an M.S. in Computer Science from Vanderbilt University and a Ph.D. in Computer Science at Texas A&M University. Lt.Col. Hamilton is a graduate of the Naval War College with distinction. His book, Distributed Simulation, written with Major D. A. Nash and Dr. Udo W. Pooch, was published by CRC Press.

Major Jeanne L. Murtagh, USAF, is currently the Director, Software Professional Development Program (SPDP), School of Systems and Logistics, Air Force Institute of Technology. Previously, she served as an assistant professor of computer science at the United States Military Academy, an Acquisition Engineering Lead Instructor, Acquisition Training Branch, 3440th Technical Training Squadron; Computer Systems Program Manager, Wright Research and Development Center; Computer Systems Lead Engineer, Mission Avionics Division, Avionics Laboratory, Wright-Patterson AFB; Software Manager, Joint Tactical Information Distribution System (JTIDS) Joint Program Office. Major Murtagh was commissioned as an acquisition engineering officer upon her graduation from Rensselaer Polytechnic Institute with a B.S., Computer Science. She has a M.S. in Computer Science from Boston University and is a certified Level III acquisition professional. ..