See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/302936381

Modular Open System Architectures (MOSA) for Military Systems: Addressing Challenges of Complex Systems

Presentation · April 2016

CITATIONS
0

READS
20

2 authors:

Michael Scott Moore
Southwest Research Institute
36 PUBLICATIONS  84 CITATIONS

Kase Saylor
Southwest Research Institute
15 PUBLICATIONS  0 CITATIONS

Some of the authors of this publication are also working on these related projects:

Modular Open Radio Frequency Architecture (MORA) View project

Vehicular Integration for Command, Control, Communications, and Computers (C4), Intelligence Surveillance and Reconnaissance (ISR) Electronic Warfare (EW) Interoperability (VICTORY) View project

All content following this page was uploaded by Michael Scott Moore on 11 May 2016.

The user has requested enhancement of the downloaded file.
Modular Open System Architectures (MOSA) for Military Systems: Addressing Challenges of Complex Systems

SOUTHWEST RESEARCH INSTITUTE®

Interoperable Open Architecture Conference

Michael S. Moore
Kase J. Saylor

27 April 2016
Southwest Research Institute (SwRI) is a nonprofit R&D org. performing R&D in a wide range of military & commercial areas, including MOSA.

The VICTORY effort defines a network-based architecture and interface specifications for integrating electronic systems in military ground vehicles.

VICTORY is being implemented in products and by vehicle programs, and the artifacts are being made available to many NATO and TTCP countries.

Lessons learned through experience with VICTORY and other MOSA efforts can provide insights for other efforts.

Having many different MOSA approaches is appropriate (not conflicting), as there is no “one architecture to rule them all”.

Bottom Line Up Front

Statements I Hope You Take Away From This Brief

- Southwest Research Institute (SwRI) is a nonprofit R&D org. performing R&D in a wide range of military & commercial areas, including MOSA.
- The VICTORY effort defines a network-based architecture and interface specifications for integrating electronic systems in military ground vehicles.
- VICTORY is being implemented in products and by vehicle programs, and the artifacts are being made available to many NATO and TTCP countries.
- Lessons learned through experience with VICTORY and other MOSA efforts can provide insights for other efforts.
- Having many different MOSA approaches is appropriate (not conflicting), as there is no “one architecture to rule them all”.
Agenda

- Brief introduction to Southwest Research Institute
- MOSA & interoperability concepts
- VICTORY, as an example of a “successful” MOSA approach
- Lessons learned from VICTORY and other MOSA efforts
- Questions and discussion
Southwest Research Institute
Benefiting Government, Industry & the Public Through Innovative Science and Technology

• US Nonprofit R&D organization
• Development & transfer of technology
• Independent & unbiased
  – Not a federally funded R&D center
  – Not affiliated with a university
  – Technical expertise in broad range of areas
  – Not a “product company”
  – Both government and commercial R&D
  – Worldwide customer base

• Main campus in San Antonio, TX
  – 1200 acres, 170 buildings
• Broad range of technical disciplines
  • Deep Sea to Deep Space
• Approximately 3000 employees

10 technical divisions, including
  – Engine, emissions, and vehicle research
  – Fuels and lubricants
  – Defense and Intelligence Systems
  – Automation and data systems
Complex systems: why are they so costly?
Overall goals: what problems need to be solved?
Objectives: what specific things must be achieved to get there?
Technical approach: can MOSA and standards achieve objectives?
Complex Systems: Why are they so Costly?

Complex systems: Systems too sophisticated to be fully designed or developed by individuals or small teams. They must be decomposed into interacting elements, implemented separately, then integrated.

- “Divide & conquer” approach
  - Essentially “separate concerns”
  - Decompose architecture/design into elements with interfaces
  - Map/allocate requirements and functions to elements
  - Task different organizations with development of elements
  - Integrate after elements are largely developed

- What could go wrong?
What Does Go Wrong Sometimes, and Why?

- Complex military systems tend to exceed budgets and schedules, or not satisfy all original requirements
- Systems are not modular, and are difficult to maintain & extend
- Government is locked into a particular vendor or set of vendors

One Part of the problem:
- Architecture is manual and informal
- Element specifications can be misinterpreted by teams
- Element and interface specs have ambiguities, errors, or inconsistencies, and acceptance criteria are not “executable”

Human nature works against us
- Engineers interact better in small groups
- Micro-cultures form, and diverge
- Our languages are imprecise
- Understanding is the projection of the truth onto each individual’s “basis set” of pre-dispositions, talents, and experience
- Engineers look for a technical solution

Modularity & interoperability are hindered by ambiguous or unclear specifications. Misinterpretation of design spec. semantics is prevalent.
What is the Goal?

- **Goal 1**: Be able to decompose systems into elements, assign functionality, and define interfaces, and have the elements work in the expected way when we put the elements together

- **Goal 2**: Be able to recompose systems based on existing elements to create new integrated capabilities (plug and play, roughly speaking)
  - Why? Reduce life-cycle costs, increase capabilities and flexibility, etc.

- **Modularity** and **Composability** are key objectives
  - Components that coordinate through interfaces, and can be composed into systems that creates a predictable, coordinated behavior

- **Open Interoperability** is also a key objective
  - Exchanges of information (data) with a *common syntax and semantic*
  - Other interactions (*control, process flow, alerts*, etc.) for coordination
  - *Open*: interfaces are completely documented and accessible within the community of interest, and can be implemented by different organizations without royalties
Why are We Not There Already?

Many reasons: one is Informality of Architecture & Specifications

- System Architecture is still largely an art
  - Relatively ad-hoc and unpredictable
  - Requires precision, and discipline: not exciting to implementers
  - Existing languages and tools for systems engineering, architecture, & design are not yet complete, or in common use

- One missing piece of the process
  - Accessible, accepted language for architecture and specification with a commonly understood semantic (a “mathematics” for architecture)
  - Methods and tools for describing system architectures & designs that will improve the likelihood of achieving modularity and interoperability

- Research areas concentrating on formal languages, model transformations, and integrated tool-chains are perhaps close
  - It will still be necessary to form agreements, and create “model

Lesson learned from VICTORY and other MOSA efforts can provide valuable insights to how to succeed in other efforts
VICTORY

A Brief Overview
Provided as a “case study”
A Systems of Systems Engineering Problem:
Integrating C4ISR/EW and Other Electronics on Ground Platforms

Traditional Approach

“Bolt On” Mission Equipment Integration

VICTORY Approach

VICTORY Data Bus enables interoperability across C4ISR/EW and platform systems

Problems to Solve

1) Reduce SWaP-C impact of GFE/TPE over time
2) Enable new capabilities through interoperability: systems share data and are managed via a vehicle network – the VICTORY Data Bus (VDB)
3) Enable commonality: common specifications, software and hardware
VICTORY Technical Approach

Command, Control, Communication & Computers (C4ISR)
Intelligence, Surveillance, Reconnaissance (ISR)

- Include an Ethernet network (data bus) as part of vehicle
  - Integrate C4ISR/EW systems, interface with other electronic systems
  - Provide the plumbing for systems and components to interoperate

- Provide shared hardware and services as part of the data bus
  - Shared services, shared processing, and shared user interface hardware
  - Position, orientation, direction of travel
  - Management: configuration, control, health reporting

- Define components with open network-based messaging interfaces
  - C4ISR/EW, Platform, Weapons Systems elements interoperate on net

VICTORY is not about new technology. It is about adding in-vehicle networks to platforms, and agreeing upon open specifications for interfaces.
VICTORY Scope and Boundary

- VICTORY provides enablers to enhance integration & interoperability between electronics systems on Army ground vehicle platforms
  - Specifies network interfaces between C4ISR/EW systems
  - Sensors, components, sub-systems, and system interface
  - Specifies interfaces (bridges) to platform systems
  - System interfaces

- Scope / boundaries of the current VICTORY initiative (architecture A)
  - Stops at the edge of the platform in-vehicle network (intra-vehicle only)
  - Stops at the edge of “vehicle platform” systems
    - Bridges to automotive, weapons, and power systems
  - Does not integrate safety critical systems
  - Does not define common physical components or software applications
“Component Types” are collections of interfaces

Component types are organized into three hierarchical groups
Kinds of VICTORY Interfaces

- **Transport**: communication functions from the transport layer and lower, including the data formats, protocols, signaling, and perhaps physical media specifications (routing data w/QoS)
- **Data**: data definitions, message formats, encapsulation methods, and transport choices for publishing data on the VDB (pub-sub)
- **Management**: data definitions, message formats, and behaviors for configuration, control, and health functions (req/cmd-response)
- **Health Reporting**: data definitions, message formats, encapsulation methods, and transport choices for publishing health alerts (pub-sub)
- **Access Control**: protocols used for protecting access to data and other interfaces (Attribute Based Access Control model)
- **Auto-Discovery**: protocols for automatically identifying components at runtime on the VDB
- **Application Program Interface (API)**: properties and resources of the runtime system that will host software (minimal approach; libraries, runtime elements)
VICTORY Framework Products

- **Architecture**: defines common terminology, structures, component (types), and interfaces
- **Standard Specifications**: provide technical specifications for the items identified in the architecture
- **Compliance Test Suite (CTS)**: provides [for each component type] the “golden standard” test plan, report template and test tool for compliance testing
- **Reference Designs**: provide samples on how to use the standard specification
- **Validation Artifacts**: results of the initial implementation of the standard specifications
- **Reference Software Library**: reusable software to jumpstart implementation, that has been verified with the use of Compliance Test Tool (CTT)

The architecture documents the scope of interest, system structure, boundaries, “component types”, their functions, and their interfaces. It does not identify a specific design or set of HW/SW components. The CTT includes the specifications with browsing capability.
VICTORY Take-Aways

- VICTORY defines an architecture and standard specifications for
  - Network infrastructure (Ethernet-based)
  - Shared processing and shared services

- Network-based interfaces for systems and components to interoperate
  - Data interfaces: publish-subscribe
  - Management interfaces (configuration, control, & health): command response
  - Health reporting: publish-subscribe
  - Auto-discovery
  - Specify up to application layer (protocol, encapsulation, encoding, data model)

- Vehicles include a “data bus”, providing infrastructure
  - Integrates C4ISR and EW systems
  - Bridges to platform systems (automotive, weapons, power)

- VICTORY standard specifications are being implemented on Army platforms
Lessons Learned from MOSA Efforts

Series of MOSA-related projects executed by SwRI
Common patterns and lessons from successes / failures
Some Relevant MOSA-Related Efforts

Boeing 787 Data Acq. & Recording
- DAR: commercial flight test

Integrated Network Enhanced Telemetry
- iNet: military flight test, telemetry

VICTORY
- Military ground vehicle electronics integration

Modular Open Radio Architecture
- Modular RF systems

ROS-Industrial and ROS-Military
- Domain-specific “instantiations” of the robotic operating system (ROS)
Lessons Learned: Architecture

- **Define “architecture” clearly**
  - Identify elements and interfaces, and vet with the stakeholders
  - Common language: non-ambiguous, agreed upon terms
  - Domain specific (not generic) is best

- **Select the modular elements and level of abstraction wisely**
  - Line between architecture and design will vary depending upon the domain
  - Which elements are modular should be based on the goals, business analysis
  - e.g. VICTORY defines “on-the-wire” messaging, FACE defines SW layers

- **Define scope, and stick to it**
  - Concentrate on problems driving cost

- **Provide legacy system strategy**
  - Show how existing equipment adapts
  - Provide a pathway from the current practice to the objective architecture

- **Pitfalls**
  - Getting bogged down in developing new tools: content and common understanding is the goal
  - Trying to solve problems not driving cost in the domain of interest
  - Being too general: attempting to teach domain to think in software terms

Architecture provides the bones onto which specifications will be developed. It drives the standards development process, but must also be matured based on inputs from the community and the standards body itself. Do not think architecture is “done once”.

©SOUTHWEST RESEARCH INSTITUTE swri.org
Lessons Learned: Specifications

- **Agreements, not technology**
  - Resist the urge to find a new technology to solve the problem. Instead, develop trust and agreements between stakeholders.
  - Patience: engineers and scientists want to start development, but the specification (design) must come first.
  - Good enough trumps optimized

- **Make acceptance tests central**
  - Explicitly define acceptance criteria
  - Test early, and test often

- **Formalize interface specifications**
  - Information exchanges: data content and functions first, format and encoding later. Get the logical aspects down first, then move to details.

- **Pitfalls:**
  - Being a “first adopter” of a new, unproven technology, tool, or approach
  - Getting caught in an optimization loop
  - Getting side-tracked by “cool” technologies
  - Under-estimating the need and cost of defining acceptance criteria and tests up-front

Specify as little internal component behavior as possible to allow for innovation
Listen to domain experts, and put together “task teams” guided by core team to solve problems
Data models are important, and must be non-ambiguous
Lessons Learned: Organization

- **Champion** (government): holds the vision, works with a “core team” to define architecture and set up organization, and promotes to the community of interest

- **Steering body**: leaders from the “customer base”, adopters who will eventually write contracts for and accept delivery of elements, help guide

- **Architecture lead**: pin the rose on someone with experience and respect from community

- **Independent core support team:**
  - Paid & contractually held to deliverables
  - No perceived “stake” in the decisions

- **Include domain experts**
  - Specifications require input from domain experts for each type of interface

- **Pitfalls:**
  - Ignoring or alienating domain experts. You need their experience
  - Allowing a single vendor to provide a specification. The best specifications defined by vendors will be suspect
  - The government not having technical knowledge. Government must engage in and understand the technical details as much as possible
  - Unrealistic cost and schedule plan
  - Lack of understanding of the cost benefit

The work to form agreements on the system requirements, architecture, design, acceptance criteria, etc. is more about people & organization than technology
Lessons Learned: Trust

- **Environment of trust**
  - Ensure that all have access to the same information (a common portal or document base), with as wide access as is feasible
  - Hold “open” meetings with documented minutes and actions
  - Ask opinions in open environment

- **Consensus-based process**
  - Perhaps the most difficult aspect, consensus can only be achieved if all parties feel heard, and you are willing to be flexible without sacrificing vision
  - There is no “formula” for consensus, but you know it when you get it
  - Ensure all decisions are supported and documented, and shared with all
  - Ensure that the consensus opinion results in feasibility (validation)

- **Pitfalls:**
  - Do not let a single vendor or organization define specifications in closed settings, then “pull them out of the hat”

Success in developing MOSA solutions relies strongly on creating an environment of trust, open exchange of ideas, and consensus in the community, and then doing the work to document and manage change in a design.
Lessons Learned: Process

- **Plan for Validation**: quality control on specifications
  - “Validation” step in the process before “publishing” specifications
  - Different individuals developing and evaluating/validating specifications
  - Open, documented validation plan and results
  - Develop acceptance criteria (test plan) during validation. Thinking through how to test identifies defects and holes

- **Compliance / Conformance Verification Plan**: plan determining if implementations comply (or conform) to specifications
  - Provide detailed compliance test procedures and documentation expectations as part of the process
  - Plan for if and how compliance will be certified
  - Provide reference implementation and any tools developed during validation to the community

- **Pitfalls**:
  - Under or over specifications: ambiguity is your enemy, but so is over-specification
  - Finding balance is difficult, but key
  - Baselining / releasing too early
Lessons Learned: Take-Aways

- **Architecture:**
  - Choose modular elements wisely, manage scope
  - Define legacy system strategy

- **Specifications:**
  - Concentrate on agreements, not technology
  - Tests / acceptance criteria are crucial, and eventually embody the specifications

- **Organization:**
  - Government and industry buy-in
    - Independent core technical support team

- **Environment of Trust:**
  - Openness to all, and consensus process

- **Process:**
  - Mature specifications through validation
  - Plan for verification of compliance / conformance with tests
  - Provide reference implementation and tools
Bottom Line Up Front

Statements I Hope You Take Away From This Brief

- Southwest Research Institute (SwRI) is a nonprofit R&D org. performing R&D in a wide range of military & commercial areas, including MOSA.

- The VICTORY effort defines a network-based architecture and interface specifications for integrating electronic systems in military ground vehicles.

- VICTORY is being implemented in products and by vehicle programs, and the artifacts are being made available to many NATO and TTCP countries.

- Lessons learned through experience with VICTORY and other MOSA efforts can provide insights for other efforts.

- Having many different MOSA approaches is appropriate (not conflicting), as there is no “one architecture to rule them all”.
Questions? Discussion?

You can contact me at msmoore@swri.org with questions.