This material is not complete without accompanying verbal presentation

Reference Architecture, Mission Threads and Software Integration

Stephen Beck

Jerry Jackson

Mark Kasunic

Tom Merendino

Bryce Meyer

Jeff Thieret

James Wessel

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM19-0025

Agenda

Simplified general example and review of Reference Architectures & Mission Threads

Transition into an example and discussion of the importance of a Reference Architecture and Mission Threads for describing an UAS (Unmanned Aircraft System)

Reference Architecture

Many definitions for “Reference Architecture” can be found in the literature. Many of those exist in the Software Engineering literature. However, widely agreed-upon definitions of Reference Architecture have been, and remain elusive

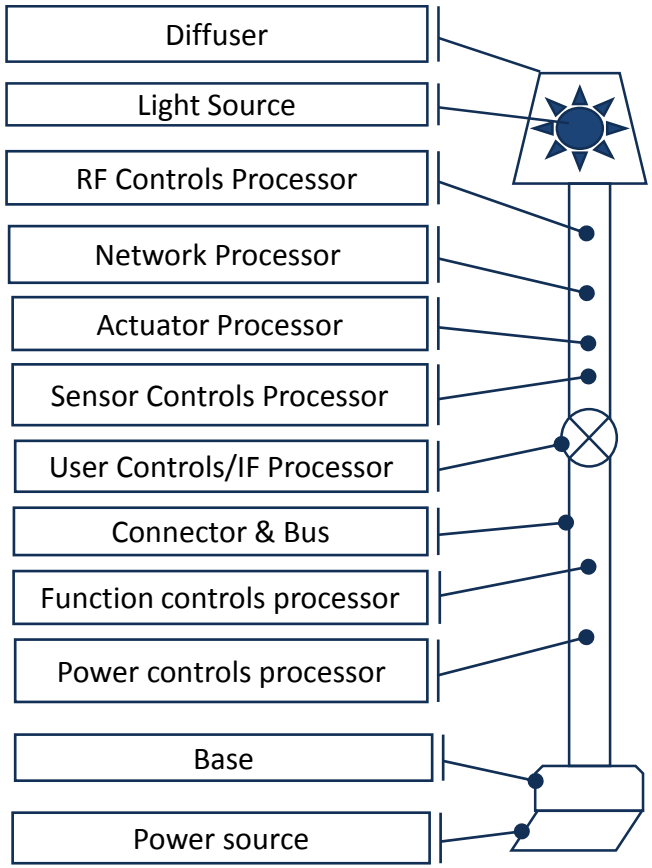
For present purposes, the DoD definition[1] for Reference Architecture suffices:

> “*Reference Architecture is an authoritative source of information about a specific subject area that guides and constrains the instantiations of multiple architectures and solutions*.” (from http://acqnotes.com/acqnote/tasks/reference-architecture-arch)

Still somewhat vague, but the important ideas that apply to software and hardware are that Reference Architectures:

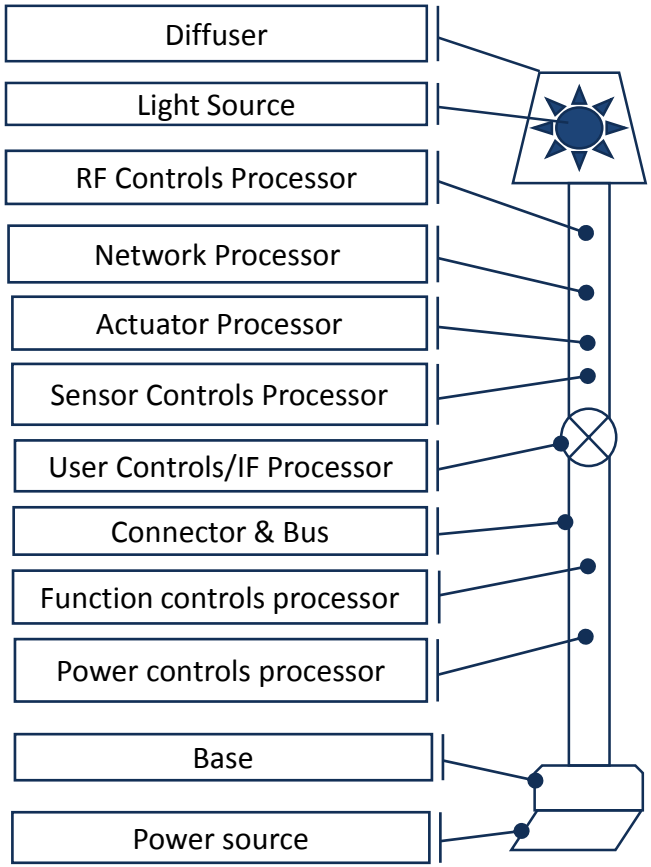
- *guide* and *constrain* Solution Architectures
- can be used to *reason about static structure and dynamic behavior* of solution architectures

Example Reference Architecture model for a "Smart Illuminator" (structural view)



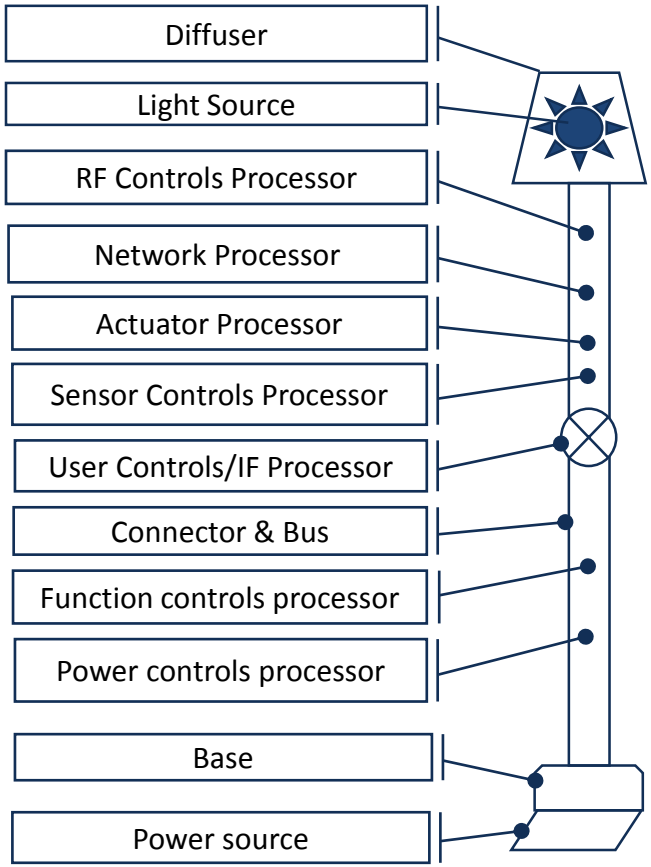
Several examples of system solution architectures (structural views)

How well do the Illuminator solution architecture structural views realize the reference architecture?



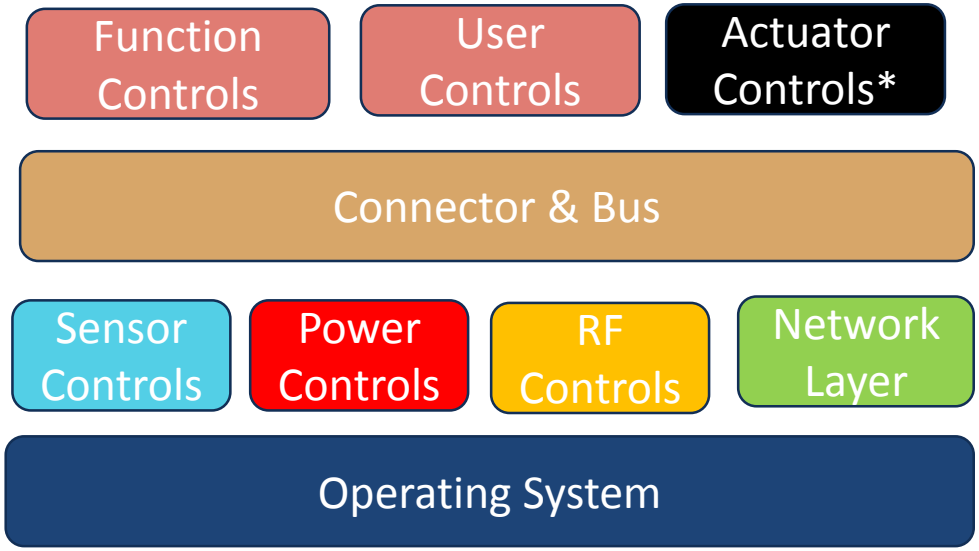
A reference architecture can be used by separate design teams to guide making compatible design decisions and identify interdependency of key design decisions

Examples of system software solution architectures (structural views)



Note that key software architecture decisions become constraints for downstream development teams.

Can you identify a few key constraining decisions within the reference architecture?



* Actuator Controls software consists of proprietary implementation with restricted license (black box)

Mission Threads and representation of an architecture's Dynamic Behavior views

Previous charts illustrated examples of structural views of an architecture

Mission Threads (MTs) are a useful technique for reasoning about the dynamic behavior views of an (Reference or Solution) architecture

- MTs – a dynamic sequence of activities / events occurring within a given context in which the structural entities of an architecture have participating roles
- Can leverage and tailor SEI's Mission Thread Workshop (MTW) methodology when beneficial to facilitate identification and prioritization of essential MTs that should drive the overall architecture

As needed, the context and scope of a MT can be narrow (i.e., several subsystems within a system) or can be much broader (i.e., end-to-end across Several platforms and Systems-of-Systems)

Representing MTs

MT's sequence of activities and events may be represented in several ways, i.e.,:

- Ordered list of steps, augmented by desired Quality Attributes (QAs such as Maintainability, Evolvability, Performance, etc
- Sequence diagrams
- Activity diagrams
- Process / workflow diagrams
- Communications diagrams, etc

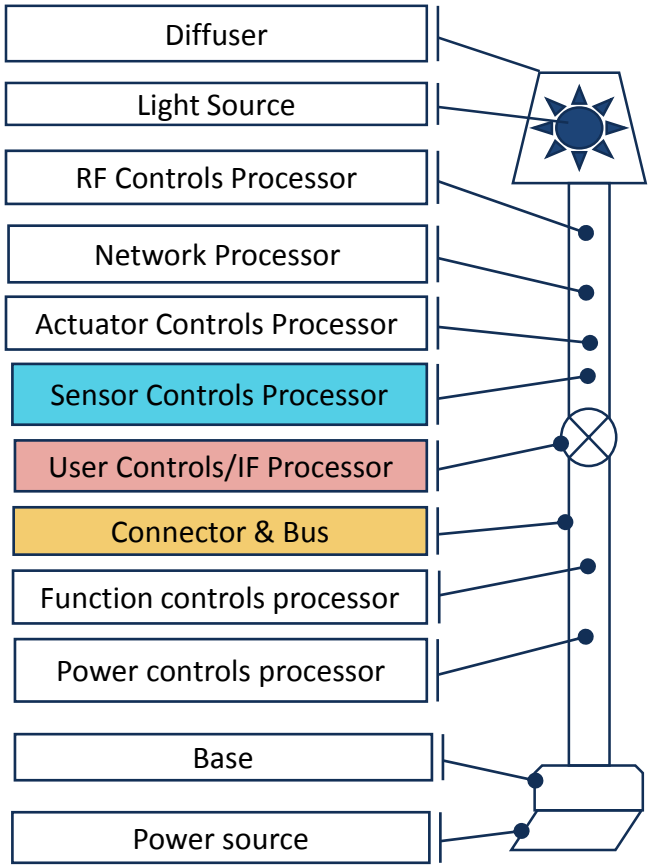
We can leverage useful MT description templates from SEI's Mission Thread Workshop (MTW)

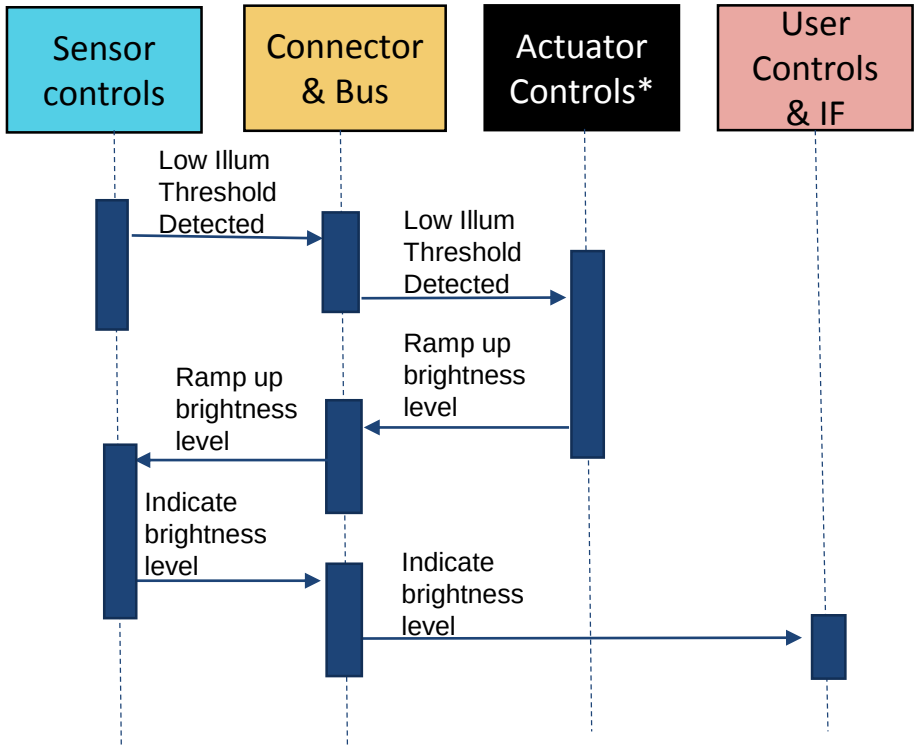
- The MTW primarily follows the first sub-bullet above to represent MTs
- Other MT representation diagrams can be used when needed as supplementary descriptions to improve clarity

Dynamic views – simplified example of "Mission Thread" for low light detected using Mission Steps representation

Mission Steps	Description (include time if applicable in each step)	Engineering Considerations, Issues, Challenges (This column will be filled in during the MTW)
1A	The Sensor Controls subsystem detects that a low-light condition has occurred. The Sensor Controls software forms a <<Low Illum Threshold Detected>> event message and send this asynchronous message to the "Actuator Controls" SW via the "Connector & Bus"	
1B	The "Connector & Bus" entity routes the <<Low Illum Threshold Detected>> event message to the "Actuator Controls" SW	
2A	"Actuator Controls" SW error-checks the received <<Low Illum Threshold Detected>> message. If check is okay, the "Actuator Controls" SW forms and sends the <<Ramp up brightness level>> command message to the "Sensor Controls" SW via the "Connector & Bus"	A system error log fault indicating an "Actuator Controls" SW command/response timeout has occurred. This fault needs to be generated and recorded if the "Actuator Controls" SW does not respond by sending the <<Ramp Up Brightness Level>> command message to the "Sensor Controls" SW within 500 milliseconds of having received the <<Low Illum Threshold Detected>> message. Who generates this fault and where is it stored?
2B	The "Connector & Bus" entity routes and delivers the <<Ramp Up Brightness Level>> command message to the "Sensor Controls" SW.	
3A	The "Sensor Controls" SW executes the received <<Ramp-up Brightness>> command message, computes the new brightness level, forms and sends an <<Indicate Brightness Level>> command message to the "User Controls & IF" SW via the "Connector & Bus".	
3B	The "Connector & Bus" entity routes and delivers the <<Indicate Brightness Level>> command message to the "User Controls & IF" SW.	

Dynamic views – simplified example of "Mission Thread" for low light detected using Sequence Diagram representation





How might individual development teams use such dynamic views to reason about potential architecture risks?

* Actuator Controls consists of proprietary implementation with restricted license (black box)

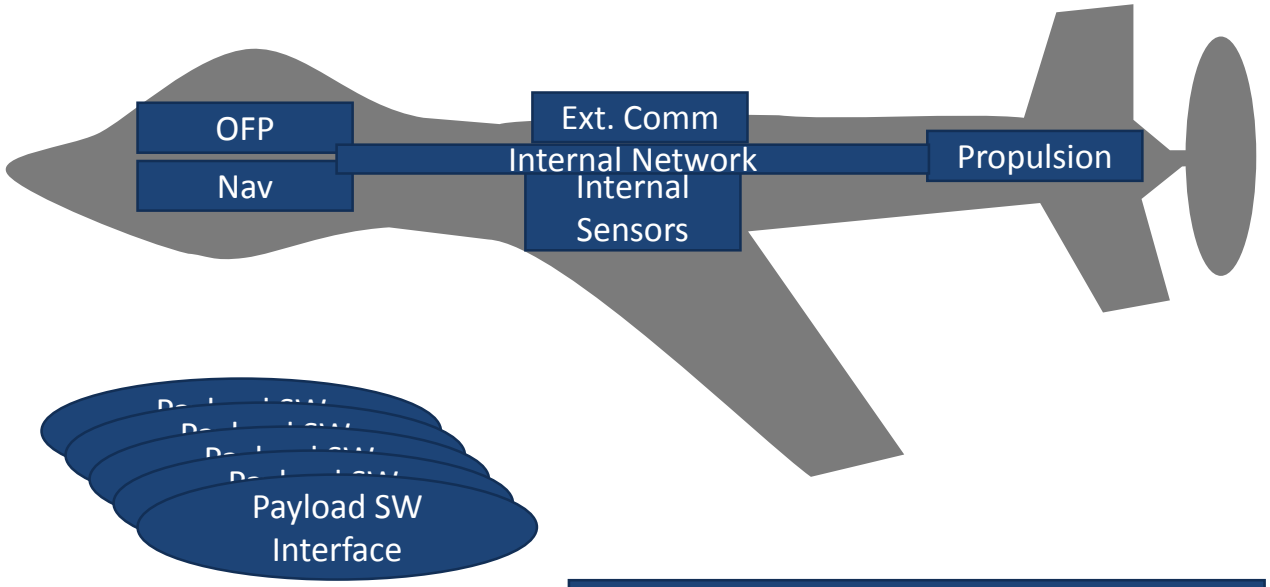
Key take-aways for utility of Reference Architectures and Mission Threads

- *They help to guide* and *constrain* Solution Architectures
- They help to *reason about static structure and suitability of candidate solution architectures to meet expectations of key system qualities*
- *They can be used to reason about dynamic behavior* of candidate solution architectures
- They can be used to help *identify dependent or coupled design decisions* made within different development teams

Benefit to the Program Office – Risk Avoidance

Actively reasoning about the system and software architecture enables the system AND SoS system engineers to assess a common, early and ongoing team-wide picture of integration risks – a significant cause of cost over-runs, schedule delays, and decreased system reliability and overall performance.

UAS System example contains significant amounts of SW that determines development and runtime behavior



Top Level Program Management and MBSE (w/integration in DevOps)

Flight/Mission/C4ISR/etc. Planning Services, Applications

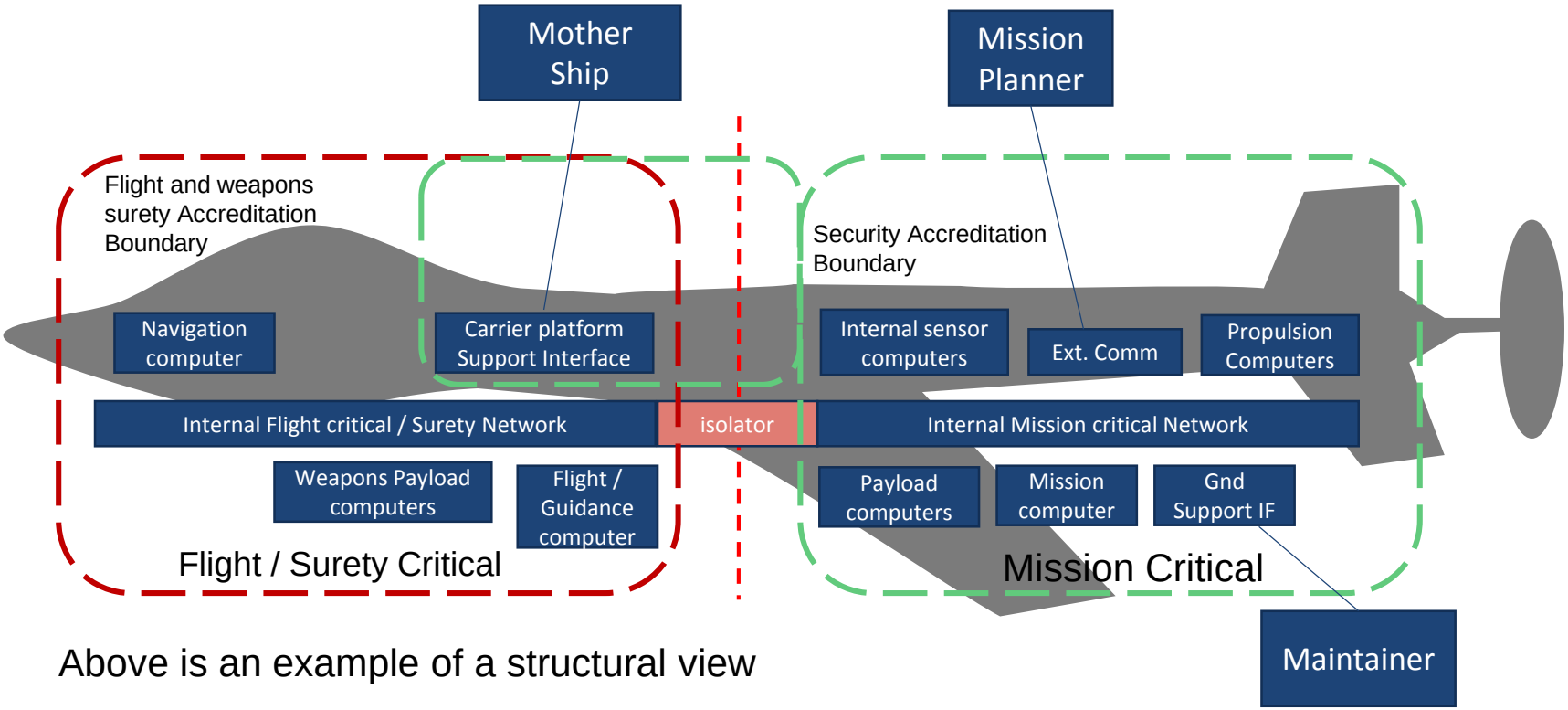
Independent (Semi Independent) V+V: 'lab' test software, test reporting, training s/w AT SCALE/RAINY DAY

Maintenance and Training Software, Maint. Equip. SW, VR, Docs, etc.

Modelling Software, Aircraft Test Software, hardware system tests

DevOps: Test Scripts and stubs, Environment, Cybersecurity, progress tracking, Test Asset Management/Ops, Reports, automated workflow, docs, dev training, in-stream integration, backlog (w/bugs) etc. etc.

Strawman Reference Architecture for UAS



Above is an example of a structural view

How can we focus on understanding desired system behavior?

- Focus articulation of the most important end-to-end system Mission Threads (MTs) that are in-turn augmented by the most important system Quality Attributes (QAs) affecting those threads
- Note that Mission Threads can be focused on Operations, Acquisition, Development or Maintenance / Sustainment scenarios or vignettes

Most Important end-to-end Mission Threads

A sampling of initial MT examples… work these with SMEs

Operational Vignette:

- Mission Plan Load
- Platform Launch / Take-off
- *Ingress* (further example in next few charts)
- Engage Target

Development Vignette:

- Successful Surety Accreditation
- Captive Carry IOT&E

Sustainment Vignette:

- Tech Refresh Insertion of Improved Navigation subsystem

What are the most important qualities to be promoted by the Reference Architecture?

Upon brief introduction to the program, it seems the top 3-5 QAs are likely to include these:

- Resiliency / Robustness (More general forms of Reliability)
- Successful Surety / Safety Accreditations
- Performance
- Evolvability
- Sustainability

Others instead / in addition?? … work with SMEs

Mission Thread “Ingress” – An Example of Vignette Summary (refer to charts 14-16)

Name	Ingress
Vignette (Summary Description)	Tactical Situation: The UAS has had its target mission and flight plans loaded, has been successfully launched from its mothership platform and is in high altitude guided flight. It begins maneuvering to low altitude flight as it transitions into the ingress area. The Navigation and Flight Guidance systems provide adjustment commands to Propulsion Subsystem to adjust the UAS final flight path to target The UAS navigates successfully to Target Engagement area and is ready to engage the target
Nodes Actors	• UAS • Mother Ship • Navigation Computer • Flight Guidance Computer • Propulsion Computer • Internal Flight critical / Surety Network • Internal Mission critical Network • Isolator
Assumptions	1. UAS has successfully executed Launch Procedures and inflight systems checks. All are good to go. 2. Communications links between UAS and Mother Ship are operational

Mission Thread "Ingress" – An Example of Mission Steps (refer to charts 14-16)

Mission Steps	Description (include time if applicable in each step)	Engineering Considerations, Issues, Challenges (This column will be filled in during the MTW)
1	Navigation Computer sends flight plan progress updates over internal UAS networks to Flight Guidance Computer and, via Isolator, to Mission Computer	1. Update messages are sent every 250 milliseconds. Is the isolator a potential bottleneck?
2	Flight Guidance Computer executes High Altitude Cruise controls (waypoint following, altitude monitoring and adjustment, sending time control adjustments via Isolator to Propulsion Computers)	1. Adjustments made by Propulsion Computers are made at each Flight Plan Waypoint
3	Mission Computer determines UAS has entered the Ingress area of the Flight Plan and sends command to Flight Guidance Computer via the Isolator and Internal Flight critical / Surety Network to start decent per Mission Plan	1. A non-surety computer is determining that a flight plan phase transition into the Target Ingress phase. The isolation mechanism must be approved by both Surety and Security Authorities
4	Flight Guidance Computer starts controlling the descent and entry into Low Altitude flight by sending commands to Propulsion Computer via Isolator	
5	Flight Guidance Computer monitors decent progress via updates from Navigation Computer and sends notification to Mission Computer via Isolator upon entry into Low Altitude Flight.	
6	Flight Guidance Computer monitors UAS progress of Low Altitude Flight Plan and sends adjustment commands as needed to Propulsion Computers over UAS internal networks via Isolator	1. Sending of Adjustment Commands via the isolator needs approvals by Surety and Security Authorities
7	Flight Guidance Computer sends notification to Mission Computer, via isolator and internal UAS networks, to start the Target Engagement phase of the Flight Plan	

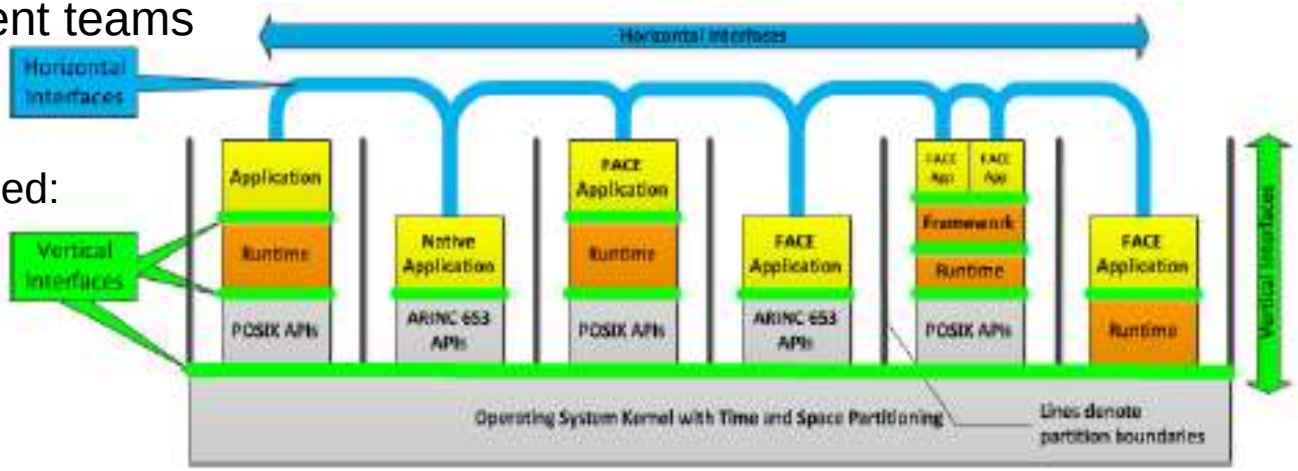
Mission Thread "Ingress" – Example of Over-Arching Quality Attributes (refer to charts 14-16)

Over-Arching Quality Attributes	
Name	Description
Resiliency / Robustness (More general forms of Reliability)	Identify the off-nominal system response should software errors occur or be detected during the decent phase of the Ingress Mission Thread. Should errors detected during a critical step of Thread operation attempt to be transmitted or stored somewhere?
Successful Surety / Safety Accreditations	Certain mission thread steps require that some messages, commands or events originate on a Mission Critical Computer part of the architecture and must traverse a the isolator and be consumed by an architectural entity that is part of the Surety / Safety critical part of the architecture. What are alternative architectural patterns might be proposed should it turn out that the Surety accreditation authorities will not accredit the design?
Performance	Can the isolator keep up with message throughput without exceeding Key timing constraints ?
Evolvability	How can we convince Surety and IA certifying authorities that the entire system should not need to be re-certified if we modify or add a largely independent subsystem
Sustainability	

Examples of potential UAS Software Architecture Patterns

Many key architectural decisions are needed to arrive at a solutions architecture Deploying software within the architecture (i.e., Chart 14) and selection of architectural patterns are among the most difficult decisions that cross-cut among development teams

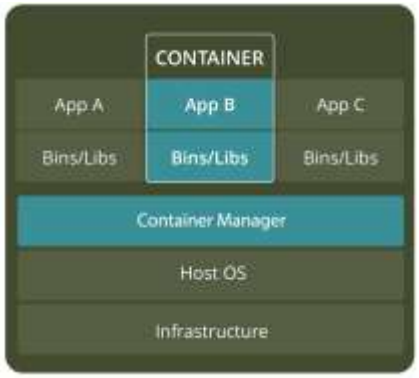
FACE / ARINC Based:

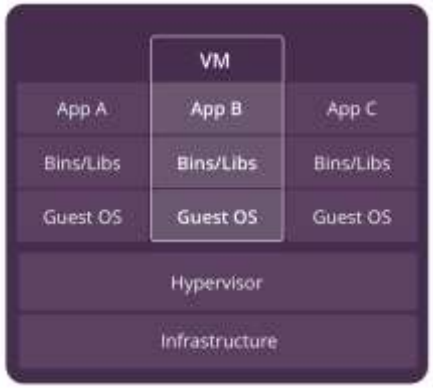


sigada.org/conf/hilt2016/paper-Tokar.pdf

Trending – Virtualization patterns based on VM Hypervisors or Containers:

https://www.backblaze.com/blog/vm-vs-containers





Use of FPGAs in the Solution Architecture

The functions provided by any of the processor units referred to in the prior examples may be implemented by firmware within FPGAs (firmware) or by software executing on processing units.

The design decisions made to trade off whether certain functionality is implemented via Firmware/FPGAs or via Software/Processing Units are key decisions to document and understand

- Documented visibility into the rationale for such decisions is crucial to enable sustainment of the platform's design for multiple decades.
- The firmware designed and implemented in FPGAs needs to be documented, version controlled and released using the same procedures as any other software that is part of the UAS platform.
- Outside accreditation of Firmware by Surety, Safety and IA authorities is to be expected as stringent as for Software

Summary

- Reference Architectures help to *guide* and *constrain* decisions made by downstream development teams
  - Those decisions you do *not* want to leave up to the downstream teams should be part of the Reference Architecture
- Reference Architectures can be used to *reason about static structure and dynamic behavior* of solution architectures
  - Such reasoning early in the design phase may be the only opportunity to discover development team interdependencies regarding key decisions
  - Reference Architectures can be used as a framework to assess the merits of candidate solution architectures

Questions?

Backup

SEI Mission Thread-related Definitions (DoD Context)

Vignette: A description of the geography, own force structure and mission, strategies and tactics, the enemy forces and their attack strategies and tactics, including timing. There may be associated Measures of Performance (MOP) and Measures of Effectiveness (MOE). A vignette provides context and scope for one or more *mission threads*.

Mission Thread (4 types):

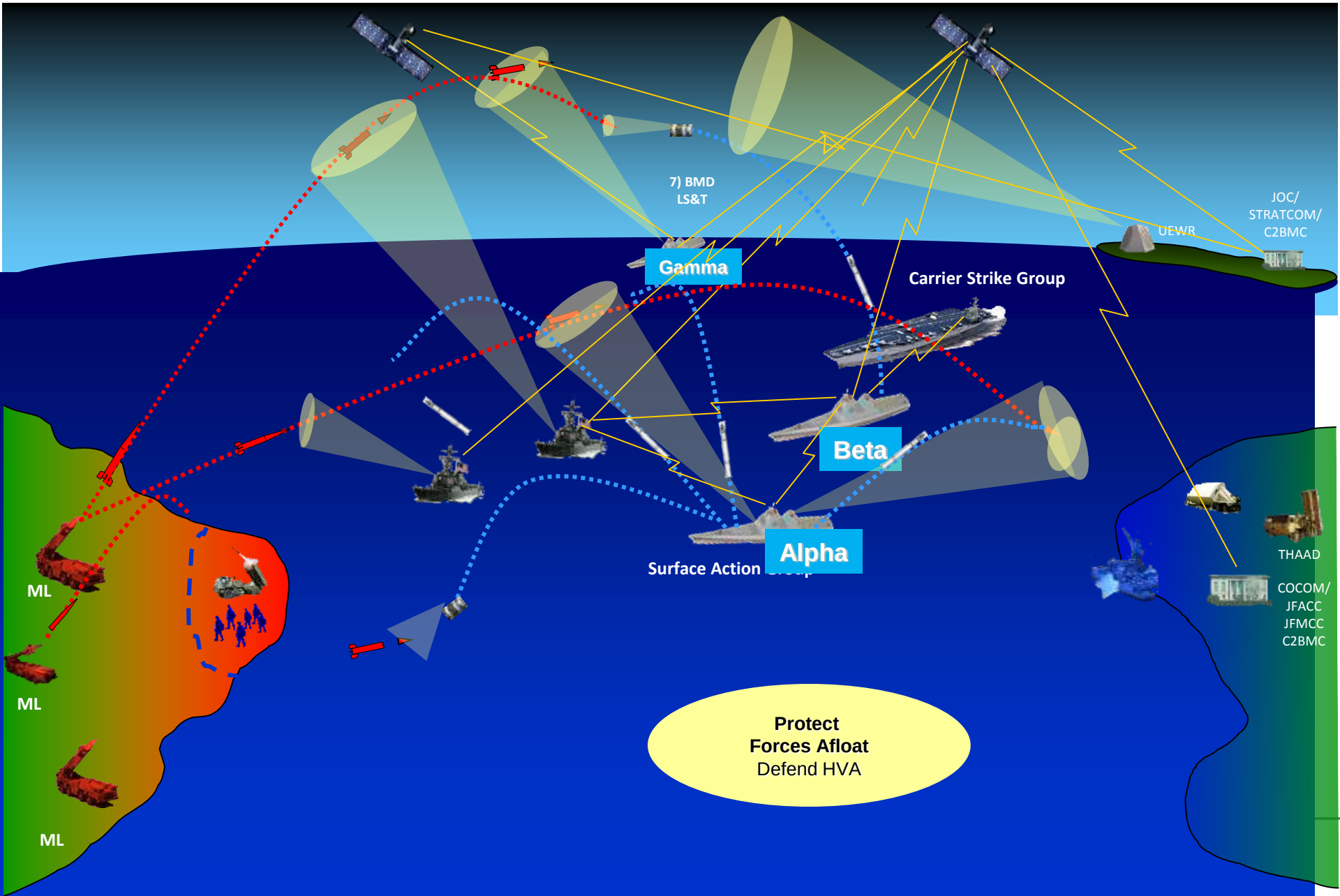
Operational: A sequence of end-to-end activities and events beginning with an opportunity to detect a threat or element that ought to be attacked and ending with a commander's assessment of damage after an attack.

Sustainment: A sequence of activities and events which focus on installation, deployment, logistics and maintenance.

Development: A sequence of activities and events that focus on re-using or re-engineering legacy systems and new adding capabilities

Acquisition: A sequence of activities and events that focus on the acquisition of elements of an SoS, and the associated contracts and governance

Air and Missile Defense (AMD) OV-1 Example



Vignette – Example Wording

Two ships (Alpha and Beta) are assigned to integrated air and missile defense (IAMD) to protect a fleet containing two high-value assets (HVA). A surveillance aircraft SA and 4 UAVs are assigned to the fleet and controlled by the ships. Two UAVs flying as a constellation can provide fire-control quality tracks directly to the two ships. A three-pronged attack on the fleet occurs:

- 20 land-based ballistic missiles from the east
- 5 minutes later from 5 aircraft-launched missiles from the south
- 3 minutes later from 7 submarine-launched missiles from the west

The fleet is protected with no battle damage.

From: Michael Gagliardi, SEI, “Mission Thread Workshop Planning”

Mission Threads Flow from Vignettes – Example (Non-Augmented)

1. 20 land-based missiles launched - X minute window
2. Satellite detects missiles - cues CMDR
3. CMDR executes re-planning – reassigns Alpha and Beta
4. Satellite sends track/target data - before they cross horizon
5. Ships' radars are focused on horizon crossing points

…

N Engagement cycle is started on each ship

N+1. Aircraft are detected heading for fleet

N+2. SA detects missile launches – tells CMDR

N+3. CMDR does re-planning - UAVs are re-directed

N+4. FCQ tracks are developed from UAV inputs