Countering Obsolescence via High Assurance

Jess Irwin, Northrop Grumman Corporation
Information Architect

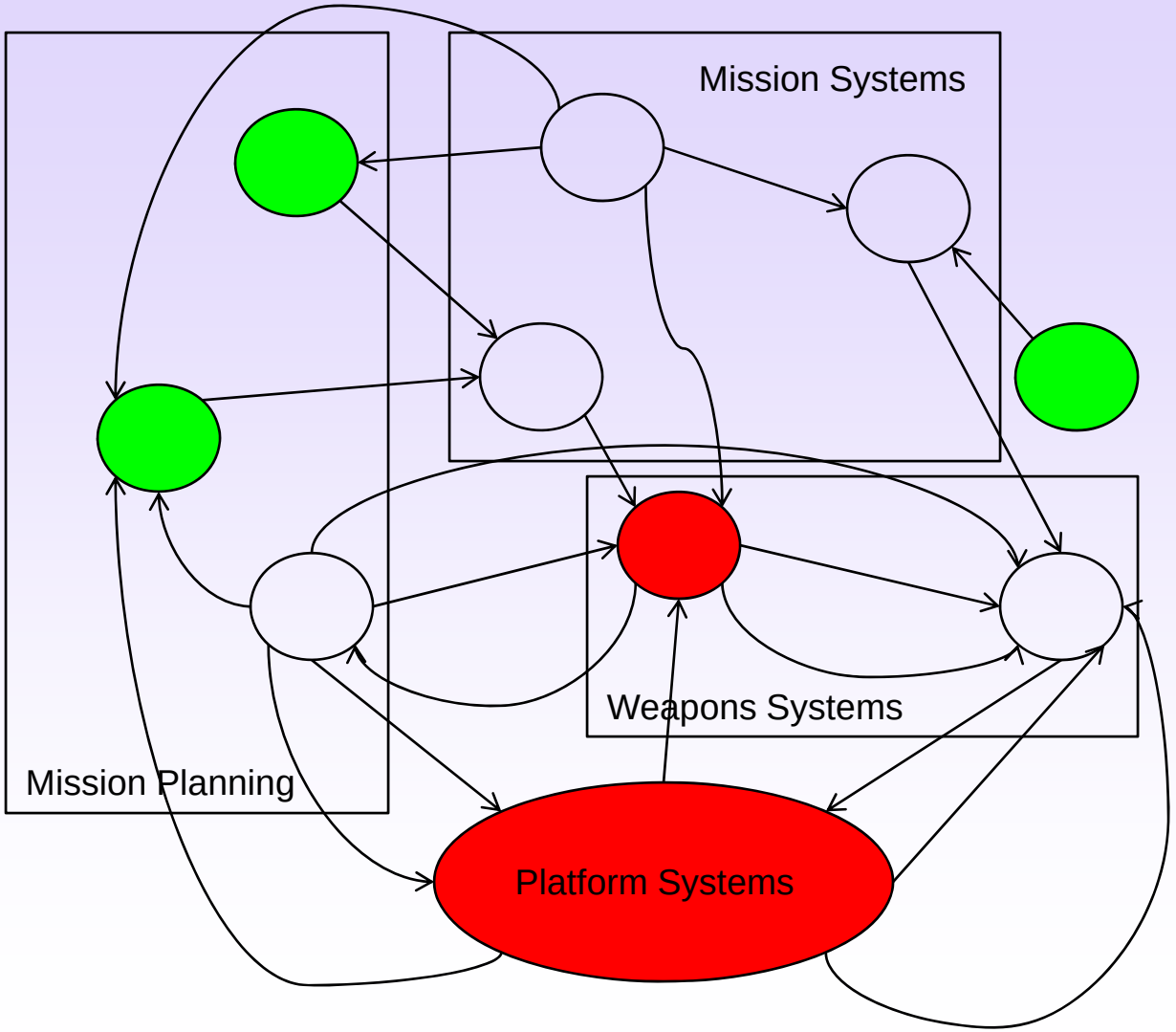
Gordon Uchenick, Objective Interface
Senior Mentor / Principal Engineer

W. Mark Vanfleet, National Security Agency
Senior INFOSEC Systems Security Analyst

What We are Going to Talk About

- We are going to talk about how High Assurance infrastructures can help counter obsolescence
- We are not going to define High Assurance or talk about how to achieve it
  - That's a different presentation

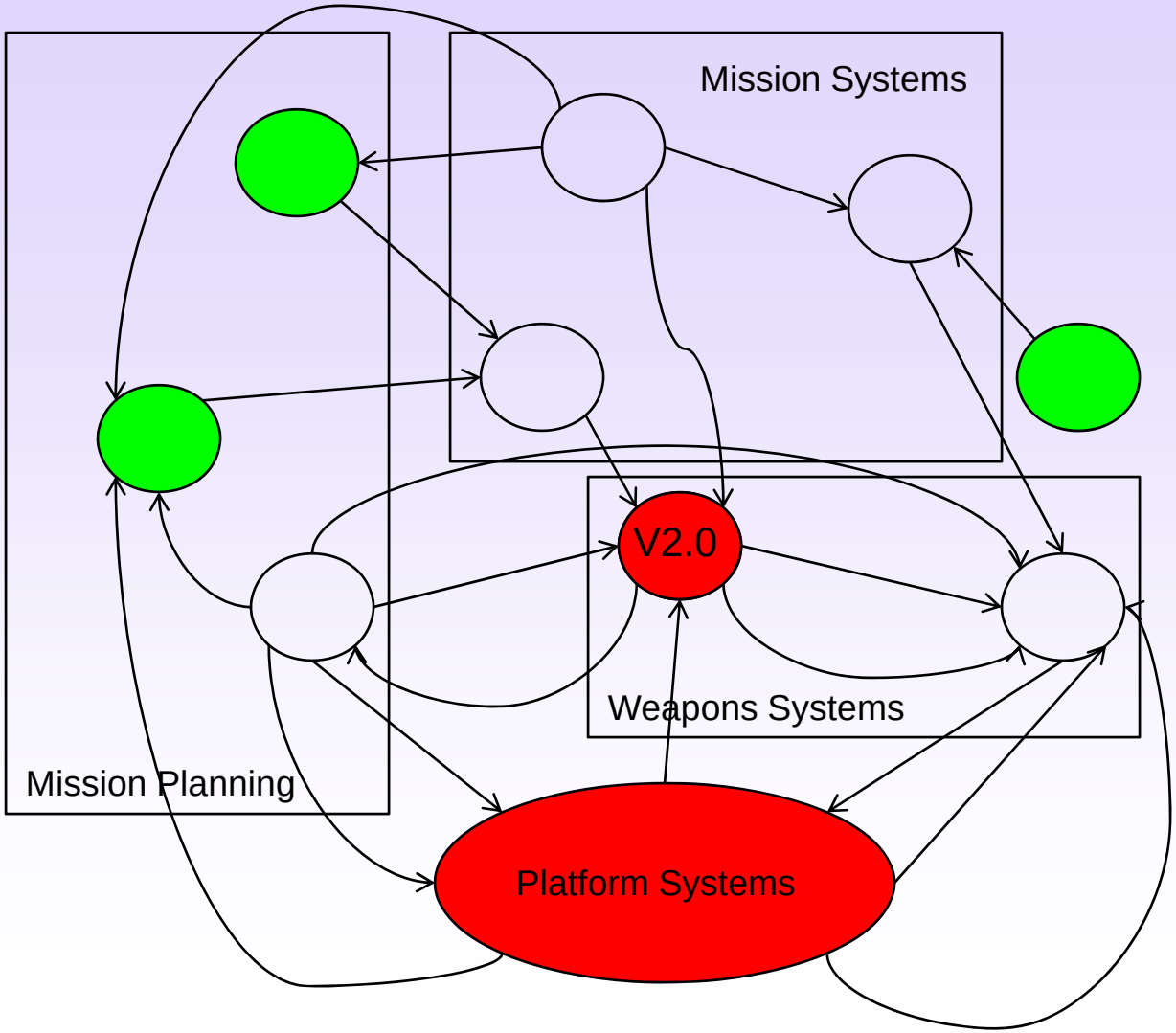
Modules Integrated Into System



Integration Done Right...

- System design decomposed into modules
- Module interfaces well defined
- Modules implemented and unit tested
- Modules integrated
- System functional test
- System penetration test
- Yippee! System accredited!

Just a Simple Tech Refresh



Tech Refresh Done Right…

- Interface specification unchanged
- New module unit tested
- Module reintegrated
- System function retested
- System penetration retested
- System reaccredited
- Refreshed system deployed and runs for years
- Then one day, for no apparent reason….

Why Does This Always Happen?

Why Is Tech Refresh Harder than Anticipated?

- Don’t really understand all the properties of all the modules
- Modules may be combined in unanticipated ways
  - We didn’t know all of the developer’s environmental assumptions
- Modules may not always work as anticipated
  - Unknown or unspecified behavior at edge conditions
- Modules could have latent bugs that did not appear in initial configuration

Unanticipated Side Effects

Module A is “good”

Module B is “good”

Module (A + B) does not work because A and B interfere with each other in some unanticipated way

Property is “Composability”

- Modules retain their original properties when integrated
- Properties (A) + Properties (B) = Properties (A + B)

How Can We Manage Side Effects?

"Paranoid" system engineering

- We know every line of code in every layer
- Architecture leverages platform's separation capabilities
- Interactions between modules are rigorously studied
- Damage limitation is a key part of system design effort

This kind of engineering is not new technology

- Have you have ever flown in an airplane?

We can't do this kind of engineering for every part of every system

- Expensive, difficult, slow, and just plain not much fun
- Assurance is about correctness, not budget or schedule

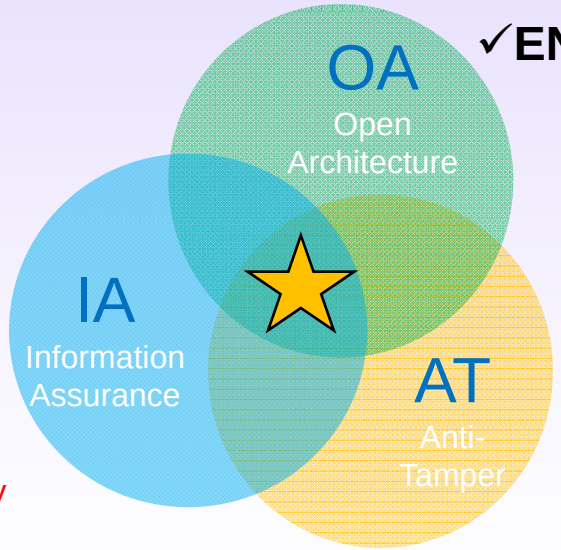
Obsolescence Issues

- Moore's Law
- GOTS vs. COTS
- Sole Source vs. Multi-source
- Supported (Licensed) vs. Communal (Unlicensed)
- Security vs. Functionality
- Customization vs. Mainstream
- Tight Control vs. Monolithic Control
- Easy to Lock Down, Niche Market, Small Vendor

  vs.
- Hard to Lock Down, Excessive Functionality, Large Vendor/Prime
- Made in the USA vs. Made in China, Russia, Japan, et.al.
- Growing Attack Surface over time

OA-IA-AT Principles

- ✓**MODULARITY & DISCLOSURE**
  - o Separable, Composable, Configurable
- ✓**RE-USEABLE COMPONENTS**
  - o Multiple Instantiation, Individually Upgradeable
- ✓**DESIGNATE KEY INTERFACES**
  - o Infrastructure and Enterprise API's
- ✓**INTEROPERABILITY & SECURITY**
  - o Operation and Evaluation across Primes
- ✓**ENABLING ENVIRONMENTS**
  - o Standardization, Role in Bigger Picture



- ✓**CONFIDENTIALITY**
  - o Critical Data **PROTECTED**
- ✓**AUTHENTICATION**
  - o Identity Confirmed
- ✓**AUTHORIZATION**
  - o Privilege Confirmed
- ✓**NON-REPUDIATION**
  - o Proof of Data Origin & Delivery
- ✓**INTEGRITY**
  - o Free of Unauthorized Manipulation
- ✓**AVAILABILITY**
  - o Critical functions **ALWAYS READY**

- ✓**DETERRENCE**
  - o Undesirable Consequences
- ✓**PREVENTION**
  - o Risk Avoidance / Management
- ✓**DETECTION**
  - o Visual, Alarm, Loss of Function
- ✓**RESPONSE**
  - o Destruction, Disabling, Zeroization

Open Architecture Separation Goals

- Isolate architectural components that evolve at different rates
  - Abstraction Layers isolate S/W (evolves slowly) from H/W (Moore's Law)
- Open interfaces that conform to business objectives
  - Open interfaces around enablers, support insertion of 3rd party enablers
- Use canonical modularity of the system
  - Most mature domains already have well defined H/W and S/W boundaries, need to be opened
- Architect for Composability, Separation, Reuse (multiple instatiation), Refresh
  - OA and IA both manage complexity
- Real Time Operating Systems (RTOS) must be COTS and unmodified
  - Hands Off the internals – allows non-disruptive H/W and peripheral upgrades
- Separate Infrastructure from Mission
  - Infrastructure enables composability and compositionality

Separation and Composition is the only game in town for managing complexity

- A good architecture simplifies the assurance case

High Assurance Infrastructures

- Provide trustworthy “boxes and arrows”
- Trustworthy boxes
  - No information can get into or out of the box
- Trustworthy arrows
  - Information only flows along designer’s arrows
  - There are no other arrows, not even accidental

Implementing High Assurance Architecture

- Start with intuitive boxes and arrows design
  - Boxes encapsulate data and control
  - Arrows are authorized information flows
- Some boxes are trusted to enforce safety or security policies
  - As small and simple as practical
- Rushby: Assume that boxes and arrows are free

High Assurance Resource Sharing

- Must implement boxes and arrows in an affordable manner by allowing modules to share resources
- Share resources so they don't violate safety or security architecture
  - This takes skill and experience
- Provide argument that modules combine to implement the architecture
- Goal is "Compositionality"
  - Properties of emergent system can be derived from properties of components

Weak Coupling

- Minimize number and complexity of connections
- Critical components are non-bypassable
- No gratuitous connections or information flows

High Assurance Design Principles

Strong Cohesion

- Things that belong together are together and work together
- Understandable, reliable, and robust
  - Well defined data and interface definitions

High Assurance Design Principles

Data Hiding

- Local data is only accessible locally
- Interfaces are respected and protected from bypass and tampering
- Use of global data is avoided / restricted / minimized

High Assurance Design Principles

Layering

- Each layer only enables the next higher layer
- Lower layer services are independent of higher layer services
- Lower layers must be
  - Application agnostic
  - Hardware independent
  - Support affordable tech refresh

High Assurance Design Principles

Abstraction

- Technology independence
- Interface does not expose implementation
- Life cycle protection
  - Maintainability
  - COTS Strategy

System Life Total Cost of Ownership

- Implementation
- Certification / Accreditation
- Deployment
- Operations, Maintenance, and Administration
- **Technology Refresh**
- **Growing Attack Surface over time**
- **Obsolescence Events**

Questions?

Objective Interface Systems, Inc.

13873 Park Center Road, Suite 360
Herndon, Virginia 20171 USA

Tel (703) 295-6500 Fax (703) 295-6501

www.ois.com