2017-03

Transformation of Test and Evaluation: The Natural Consequences of Model-Based Engineering and Modular Open Systems Architecture

Guertin, Nickolas H.; Hunt, Gordon

Monterey, California. Naval Postgraduate School

https://hdl.handle.net/10945/58871

Transformation of Test and Evaluation:

The Natural Consequences of Model-Based Engineering and Modular Open Systems Architecture

Nickolas H. Guertin, PE

(703) 350-1061

Nickolas.Guertin@Hotmail.com

CAPT Gordon Hunt, USN

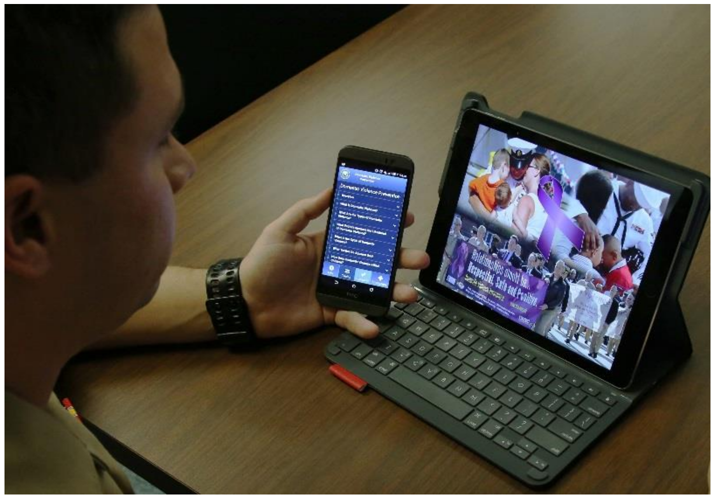
(650) 743-1040

Gordon@skayle.com

Flexible, Fast, Responsive

Our sailors fight with complicated things that have to be responsive and robust

Our lives are filled with complicated things that are responsive to our needs



The Defense Marketplace is Due for Transformation

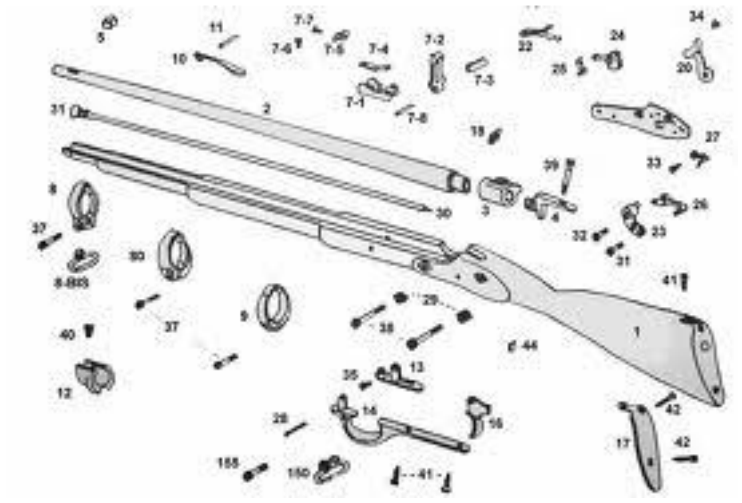
- Products take to long to get to the user
- Capability is not delivered modularly
- Destabilizing forces abound
  - Modularity
  - Ubiquitous technologies
  - Demands for different performance outcomes
- We have seen these dynamics before
- Can accelerate to a better approach if we act

Our Paper Addresses

- Things that limit DoD transformation success
    - Gaining the benefits of modularity
    - Generating enterprise value
    - Reference Frameworks vice program-specific approaches
    - Create interoperable data, vice only open interfaces
    - Improving cost-performance of integration
    - A holistic test strategy, starting with the architecture
- End the systems of systems integration nightmare

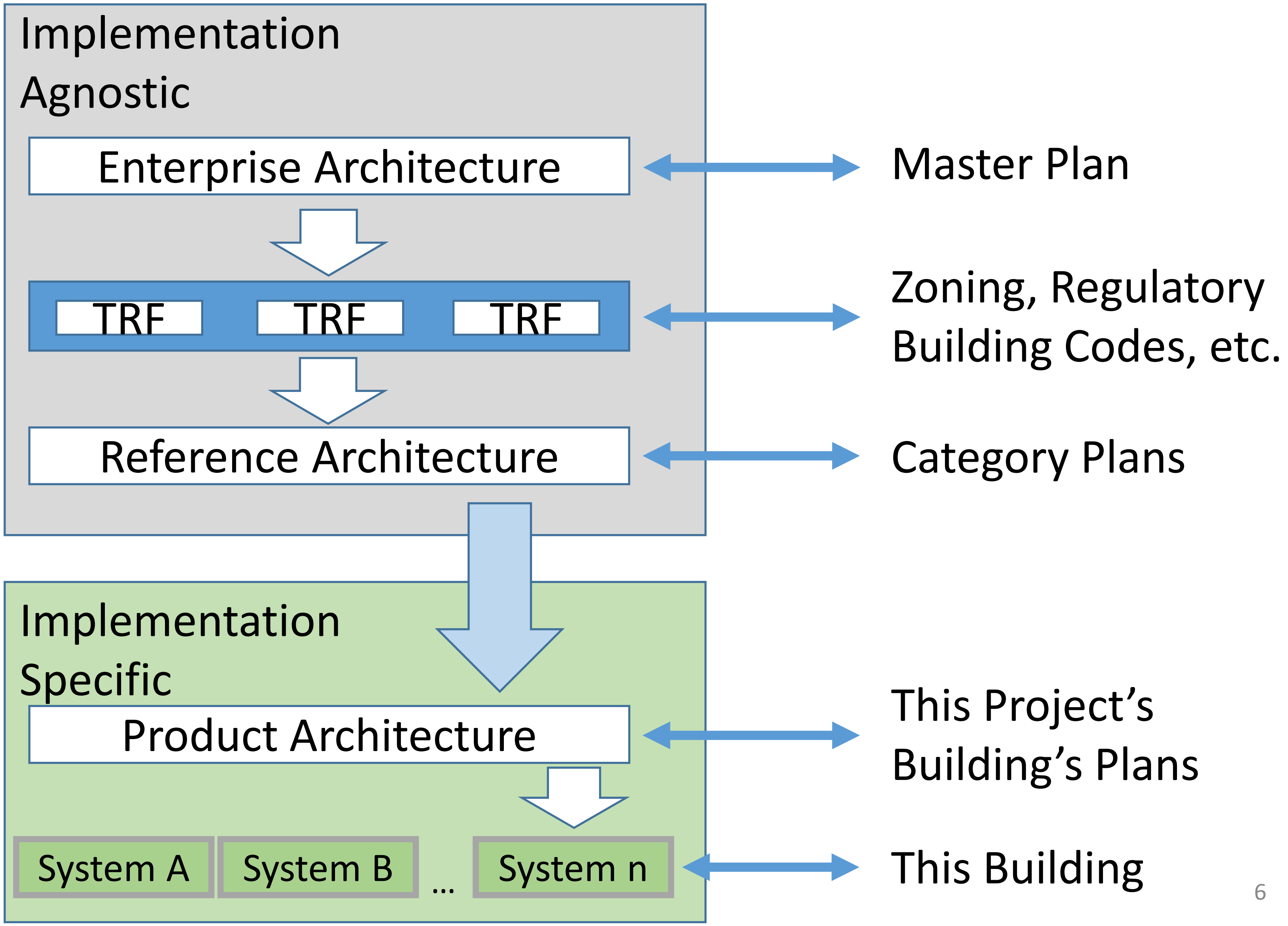
Eli Whitney and Software

- Environment where modules can be replaced or added
  - Rules of Construction
  - Consistent approaches
  - Preserving Creativity
- Screwing components together
  - Loose coupling and high cohesion
- Achieving Robust outcomes
  - Leveraging practices
- Continuous capability change



- Complexity management and affordable, rigorous testing

The Building Code Analogy



Cyber-Physical "Building Codes"

Cyber-Physical Concepts Execution & Implementation	Core Architectural Tenets	**Reference Architecture Category**
Hardware and Networks	Deployment	**Hardware**
Documentation, Configuration, Intrinsic Knowledge of Meaning	Knowledge Information	**Data**
Software Environment, Development Aids	Applications Infrastructure	**Software**
Defined Interfaces Standards (commercial and defacto) DoD Specifications & Requirements	Standards Interfaces Messages	**Functional**
Acquisition, Contracting and Requirements & Specifications	Business Model	**Governance**

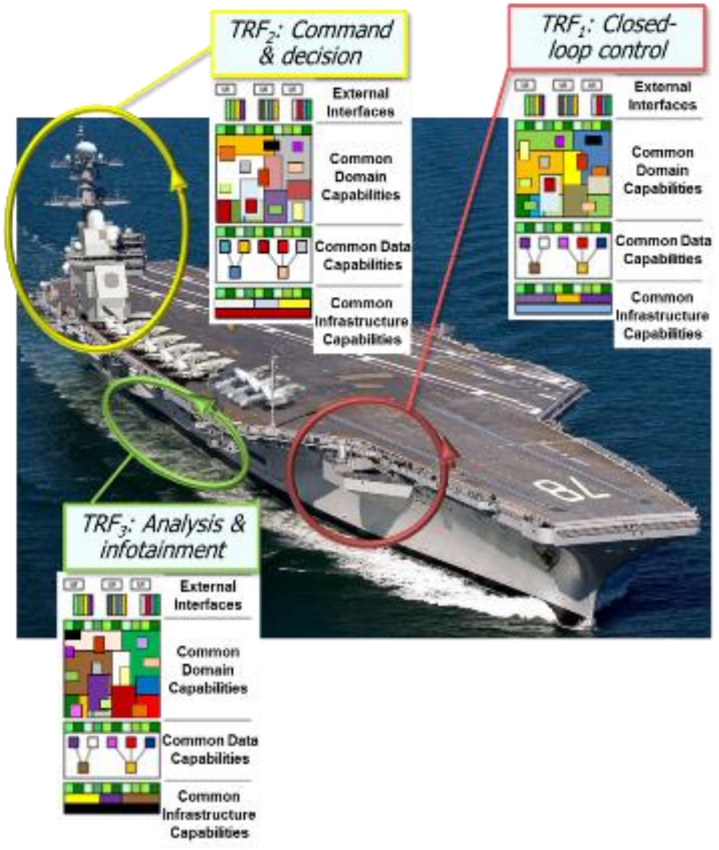
The Power of Technical Frameworks

Closed / Custom / Proprietary → Open

Proprietary	iPhone™	Android™	FACE™
Custom Apps	Open Apps	Open Apps	Open Apps
Custom Middleware	Custom Middleware	Custom Middleware	Open Middleware
Custom OS	Custom OS	Semi-Custom OS	Open OS
Custom Hardware	Custom Hardware	Open Hardware	Open Hardware

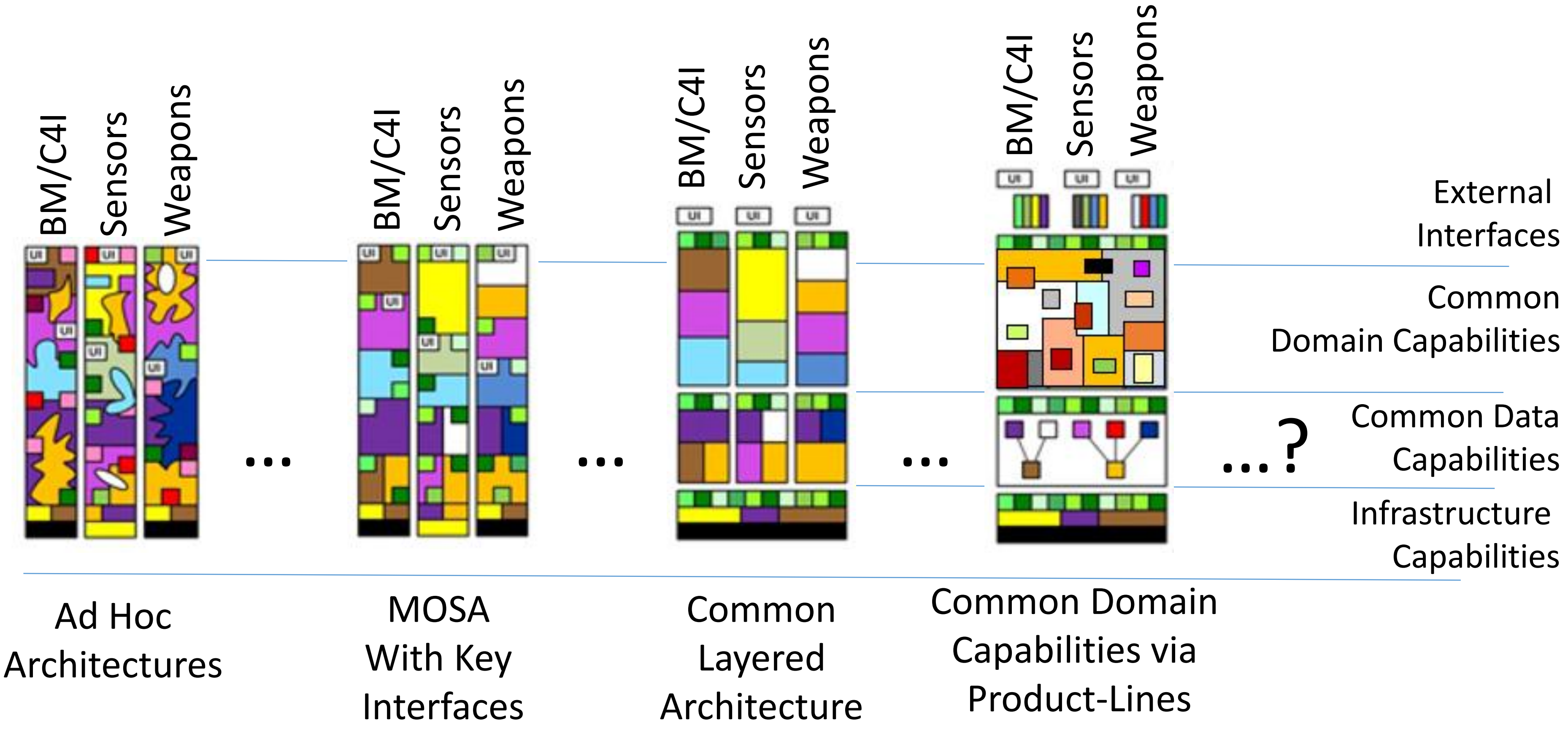
Technical Reference Frameworks (TRFs)

- TRFs are key to use of OSA
  - e.g., FACE, UCS, HOST, & SPIES
- Navy has many TRFs
- Build Reusable Modules of Capability
- Account for programmatic realities
  - New programs begin with them
  - Legacy program transition over time

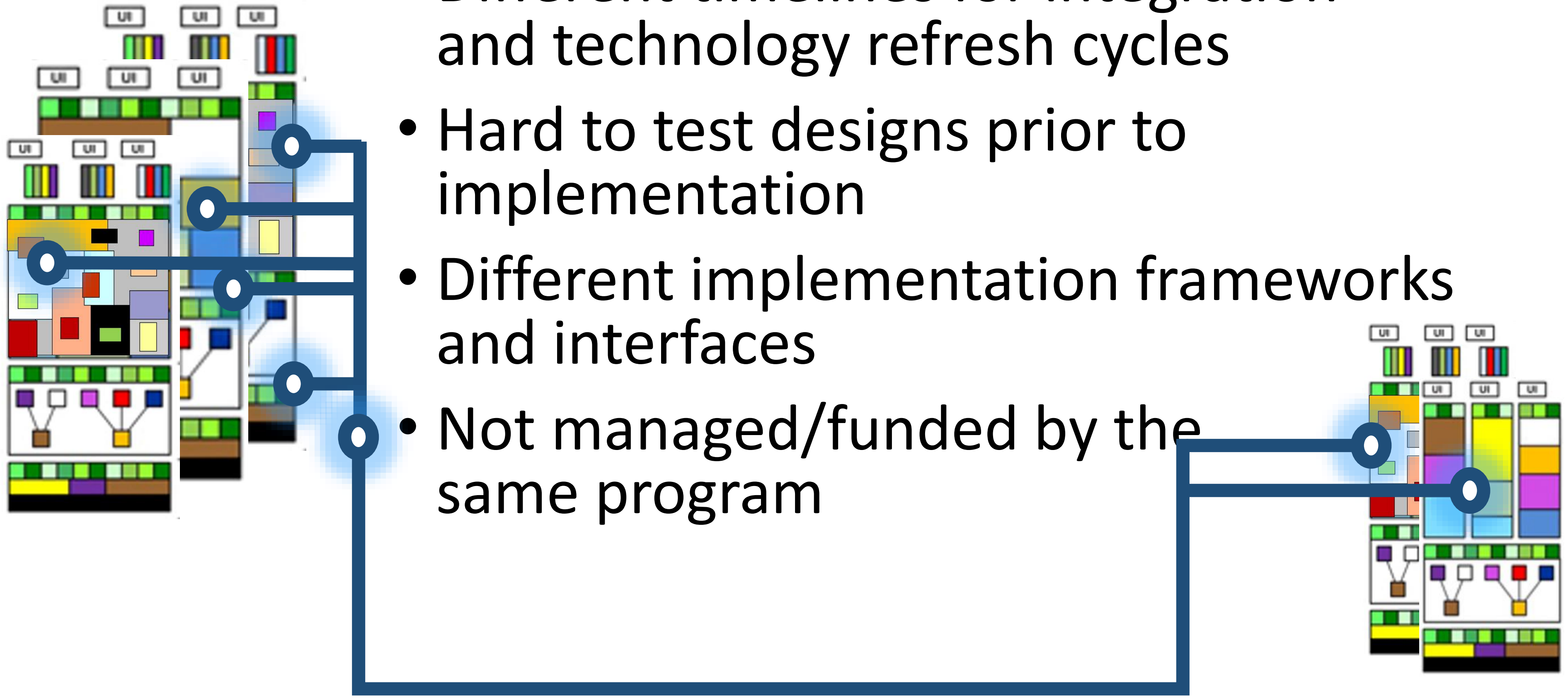

TRF₂: Command & decision
External Interfaces
Common Domain Capabilities
Common Data Capabilities
Common Infrastructure Capabilities
TRF₁: Closed-loop control
External Interfaces
Common Domain Capabilities
Common Data Capabilities
Common Infrastructure Capabilities
TRF₃: Analysis & infotainment
External Interfaces
Common Domain Capabilities
Common Data Capabilities
Common Infrastructure Capabilities

Gaining benefits of TRFs need an enterprise approach

Historical use of Frameworks: The Evolution of Complex Systems

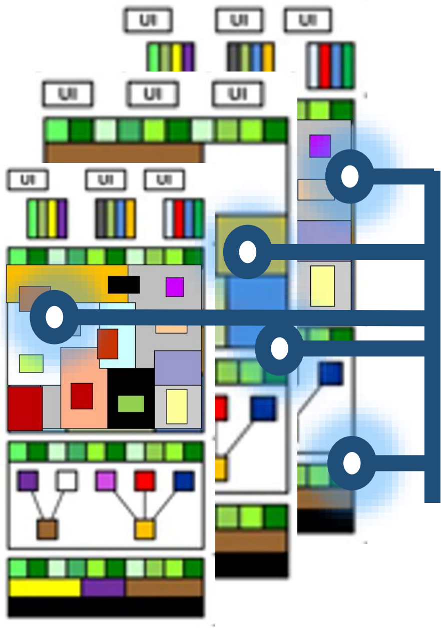

BM/C4I
Sensors
Weapons
UI
...
...
...
...?
External Interfaces
Common Domain Capabilities
Common Data Capabilities
Infrastructure Capabilities
Ad Hoc Architectures
MOSA With Key Interfaces
Common Layered Architecture
Common Domain Capabilities via Product-Lines

The Challenge of System(s) Integration



- Different timelines for integration and technology refresh cycles
- Hard to test designs prior to implementation
- Different implementation frameworks and interfaces
- Not managed/funded by the same program

Addressing the Challenge



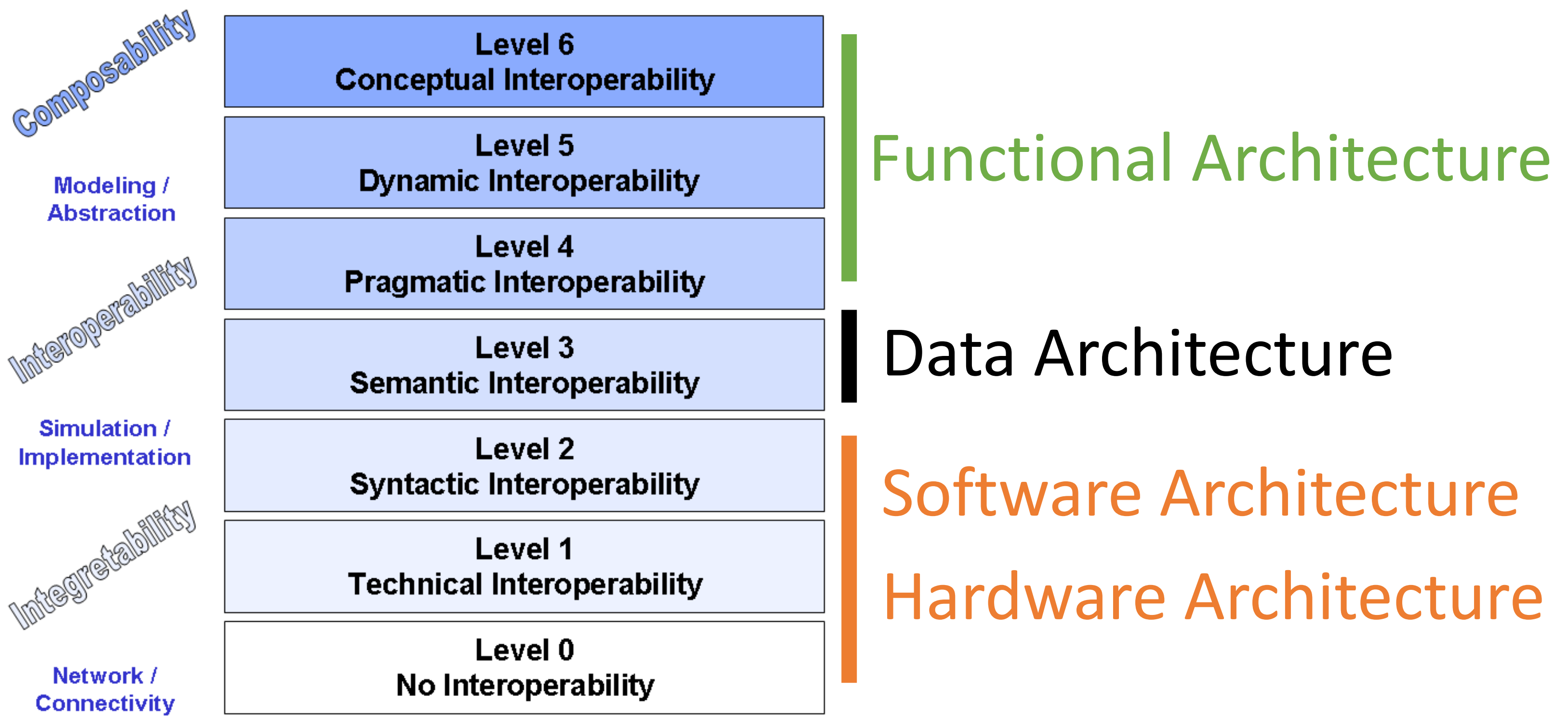
What we need:

- A common way to specify an interface
- Temporal and scale requirements
- Apply the right protocol for the job
- Configuration & deployment needs vary
- Architecture that's explicitly specified

How we get to the root:

- Content, context & behavior of data
- Scale testing and integration to new problems and situations

Architecture & Interoperability

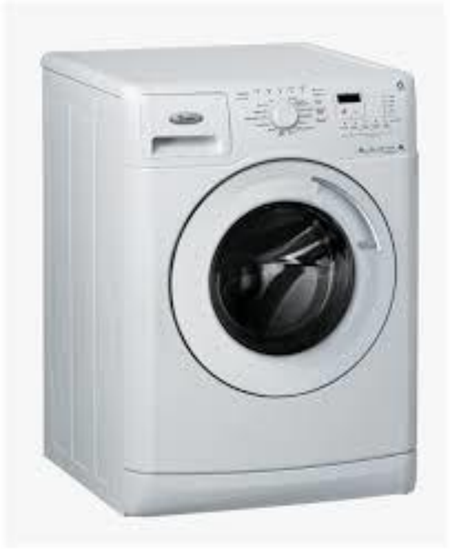


Composability
Modeling / Abstraction
Interoperability
Simulation / Implementation
Integretability
Network / Connectivity
Level 6 Conceptual Interoperability
Level 5 Dynamic Interoperability
Level 4 Pragmatic Interoperability
Level 3 Semantic Interoperability
Level 2 Syntactic Interoperability
Level 1 Technical Interoperability
Level 0 No Interoperability
Functional Architecture
Data Architecture
Software Architecture
Hardware Architecture

Semantics and Data Architecture An Example

The procedure is actually quite simple:

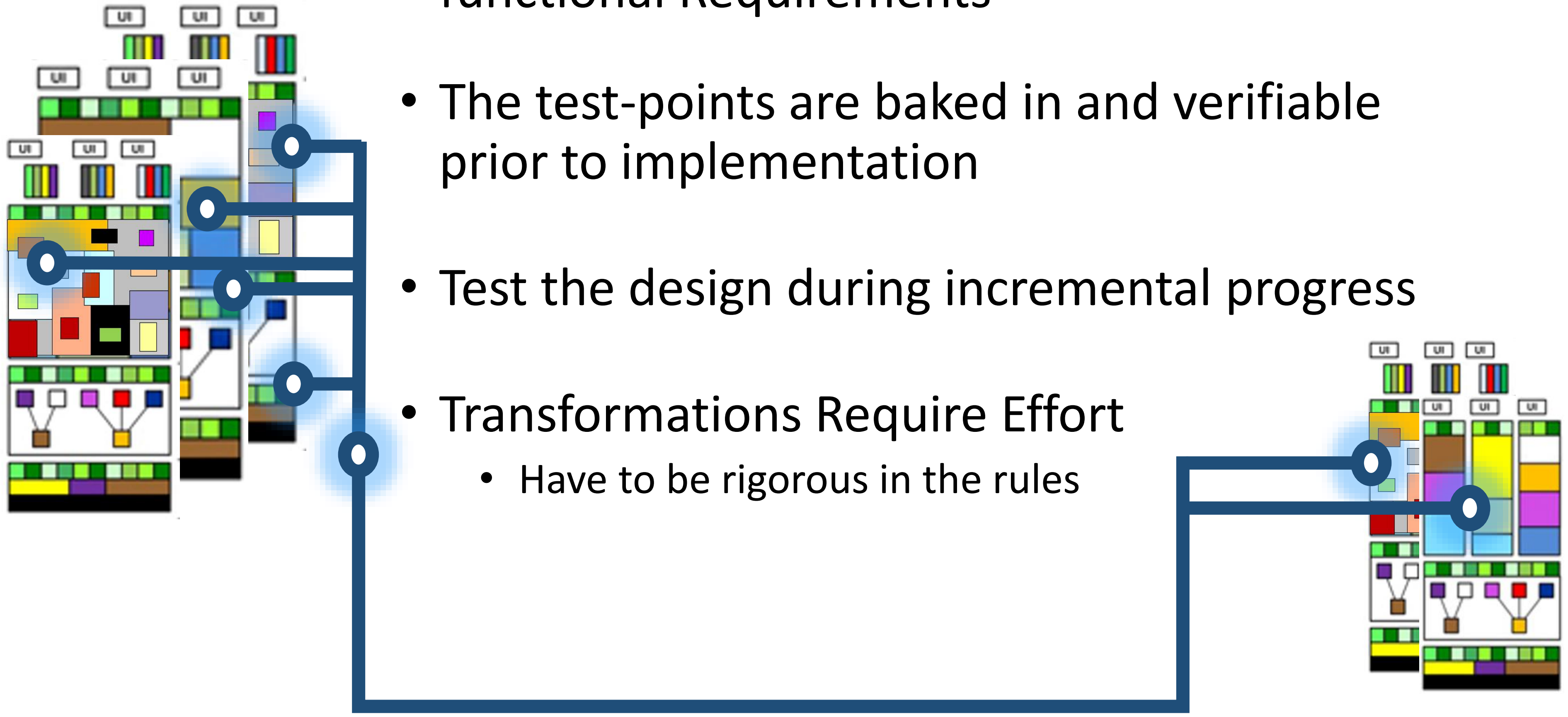
- *First you arrange things into different groups.*
- *Of course, one group may be sufficient depending on how much there is to do.*
- *Go somewhere else if there is a lack of facilities.*
- *It is better to do too few things at once than too many.*
- *In the short run this may not seem important but complications*
- *At first the whole procedure will seem complicated.*
- *Soon, however, it will become just another facet of life.*
- *It is difficult to see any end to the necessity for this task in the in*
- *After the procedure is completed one rearranges the materials into different groups*
- *Then they can be put into their appropriate places.*
- *Eventually they will be used once more and the cycle will then have to be repeated.*



- Bransford & Johnson (1972)

How we get there

- A testable architecture, including “Non-functional Requirements”
- The test-points are baked in and verifiable prior to implementation
- Test the design during incremental progress
- Transformations Require Effort
    - Have to be rigorous in the rules

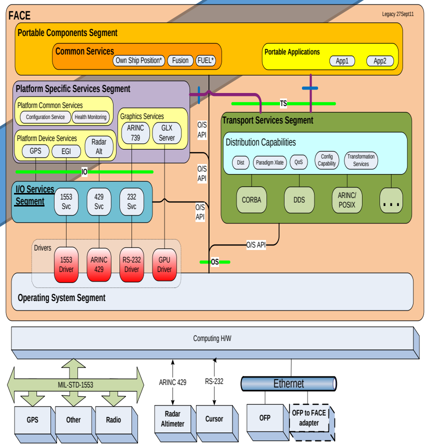


Applying Architecture

Product Architecture

Start testing with a Testable system, the game is over.

Reference Architecture



Start testing with a Testable Architecture, Checkpoints throughout development

Technical Reference Framework

Abstraction Layer

Functionality

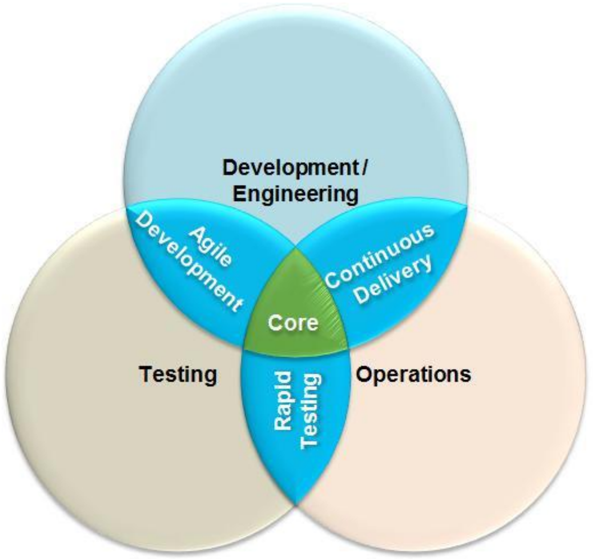
Data Exchange

Devise I/O

Enterprise Architecture

Adapt the Classic DoD Approach

- Apply Continuous Engineering practices
- Decompose Capabilities into modular components
- Reuse where possible and appropriate
- Use automated testing extensively
- Adapt the development lifecycle and have T&E community set the architecture rules



Enterprise Business Challenges

- *Match the Speed of Need*
- *Eliminate waisted effort*
- *Build so the user focuses on fighting*
- *New Strategies for Sustainment*
- *Rapid Delivery*

Actions

- *Use Architectures that are testable, flexible and decoupled*
- *Delivery modular capability*
- *Integrate innovation from anywhere*
- *Provide robust and secure products*