

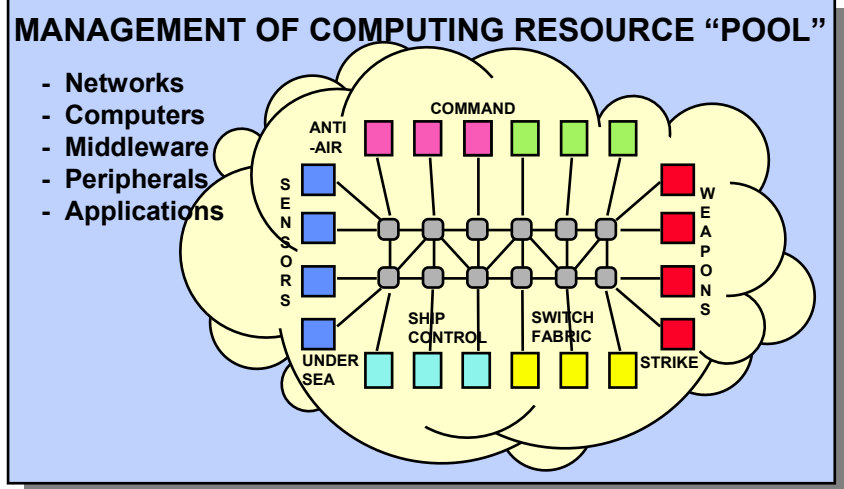
# ***Architectural Considerations for Validation of Run-Time Application Control Capabilities for Real-Time Systems***

**Paul V. Werme, NSWCDD**  
**Antonio L. Samuel, NSWCDD**

**DISTRIBUTION STATEMENT A.** Approved for public release; distribution is unlimited.

- **Background / Overview**
  - Resource Management (RM) Overview
  - Certification Overview
  - Certification Issues for Dynamically Allocated Systems
- **Verification and Validation (V&V) Research Efforts**
  - Compositional V&V Approach
  - Run-Time Validation
- **Proof of Concept – Application Control Component**
  - Requirements Analysis
  - Architectural Analysis
  - Design Guidance
  - Prototyping Efforts
  - Standards
- **Next Steps**

# Resource Management Environment

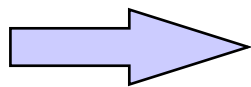


### SHIPBOARD COMPUTING MANAGEMENT

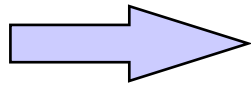
- DISTRIBUTED
- FAULT-TOLERANT
- REAL-TIME
- SECURE
- SCALABLE



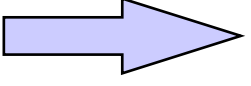
MISSION NEEDS



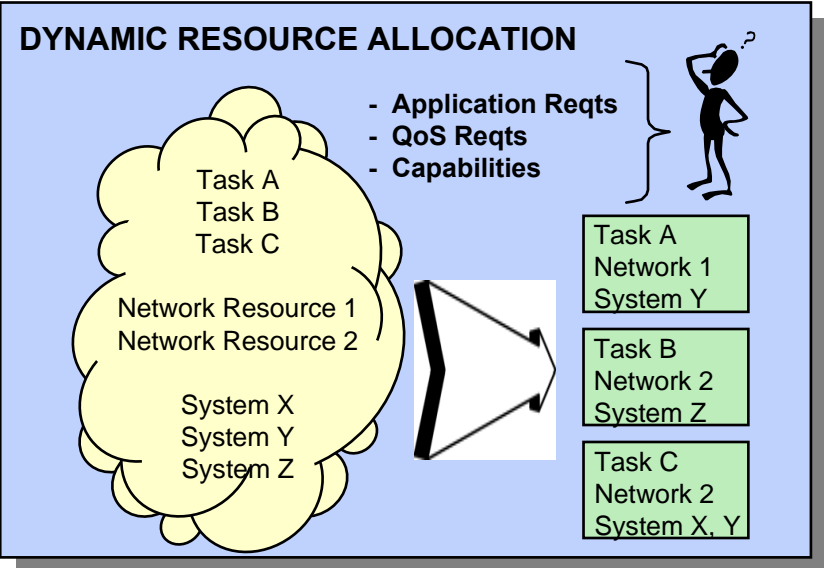
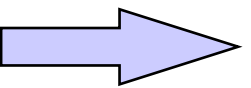
COTS



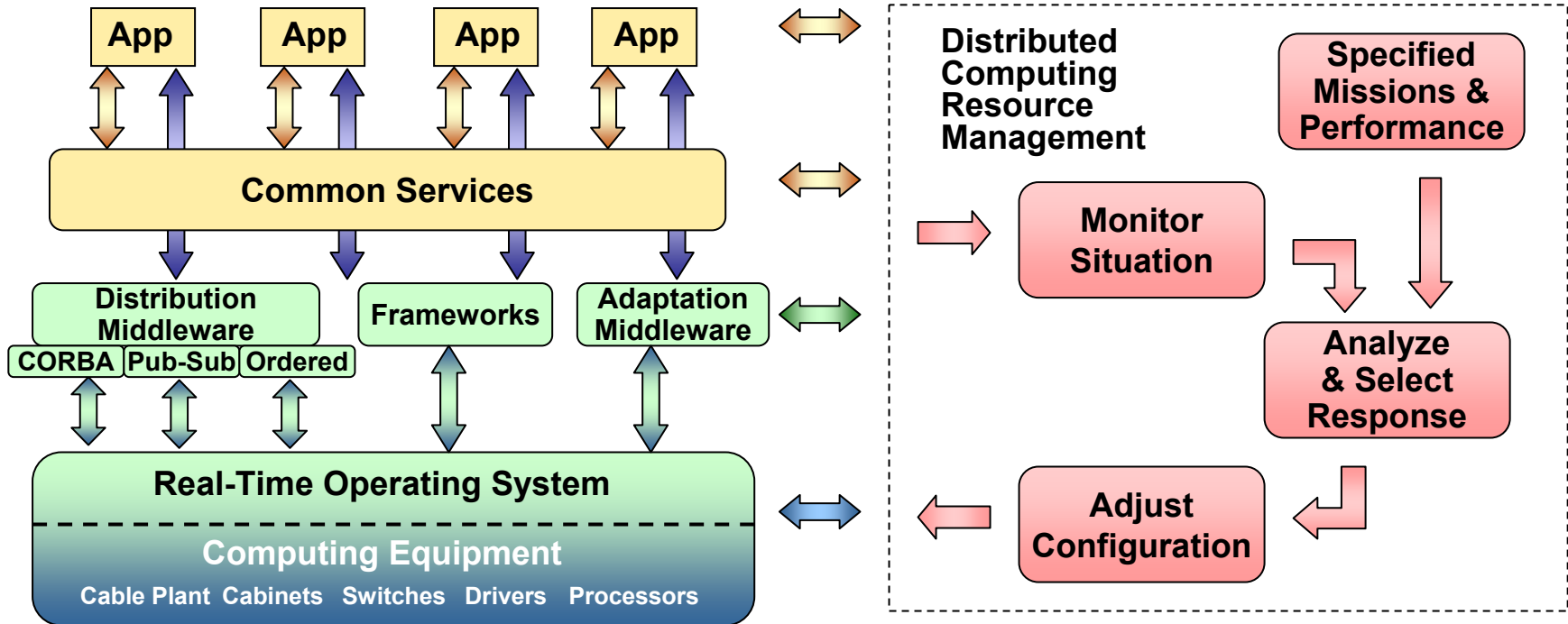
STANDARDS



LEGACY SYSTEM TRANSITION SUPPORT



# Open Architecture (OA) Resource Management



## DCRM monitoring capabilities

- Applications
- Middleware
- Processors
- Networks

## DCRM control capabilities

- Adapts system-to-mission priorities
- Maintains “load invariant” performance specified by system engineers
- Provides “self-healing” fault tolerance
- Enables “maintenance-free” operation

**Real-time, closed-loop control system ensuring a defined level of performance**

# Benefits

- **Operational benefits**
  - Provides **increased survivability** of mission-critical functionality
    - Dynamic reconfiguration of systems in response to battle damage and equipment failures
  - Allows systems to **adapt to changes in mission priorities**
  - Allows **adaptive response to loading**; real-time scale up/down
  - Allows **better utilization of spare computing resources** across systems
    - Makes possible the notion of “maintenance-free deployments”
  - Supports **reduced manning** levels by automating control (startup, shutdown, and failure recovery) and monitoring of systems
  - Allows **reduction in the number of computing resources**.
    - Reduction in space, weight, and corresponding cost for each naval vessel
- **Acquisition benefits**
  - **Functionality (dynamic control mechanisms) can be separated from system functionality**
    - Supports OA Goal: “Bring your software not your hardware”
  - **Simplifies the design and development of applications**
    - **Near Load-invariant design: Developers only need to worry about their “problem”**
      - DCRM technologies handle the system dynamics
    - **Results in overall lower development cost.**
  - **Replacement of tightly-coupled RM point solutions**

**The use of RM technologies by the Navy can result in significant operational and acquisition benefits.**

## *But...*

- **Current system certification methodologies assume static allocation of resources and static failover approaches.**
  - Even within these systems, it is often not possible to fully test all potential alternate configurations.
- **Dynamic allocation and reallocation of resources can result in an exponential number of potential configurations.**
  - The impact on areas such as testability, test coverage, event reconstruction, quantification of risk, determinism, safety, and security are not well understood.
  - Theoretical foundations and approaches are needed.
  - Design guidance, best practices, and instrumentation and data collection plans are needed.

**New methodologies are needed for certification of non-statically allocated systems.**

# System Certification

- Certification is a bounded statement of risk, identifying the capabilities and limitations of the system, resulting from the technical evaluation of the system's effectiveness.
- Certification efforts are based on rigorous assessment (IEEE1012) of product and process addressing the following criteria:

- Safety
- Mission readiness
- Quality indicators
- Installation readiness
- Sustainability
- Standards compliance
- V&V status – includes test and evaluation (T&E) activities
- Plans, process, and schedule status

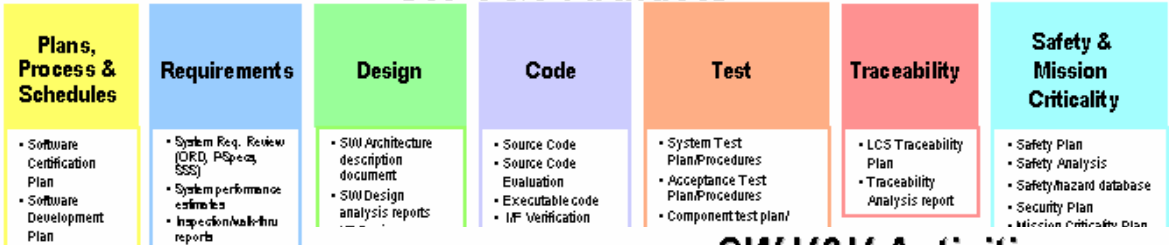
What is known about the system?

What is the level of confidence that the critical defects have been identified?

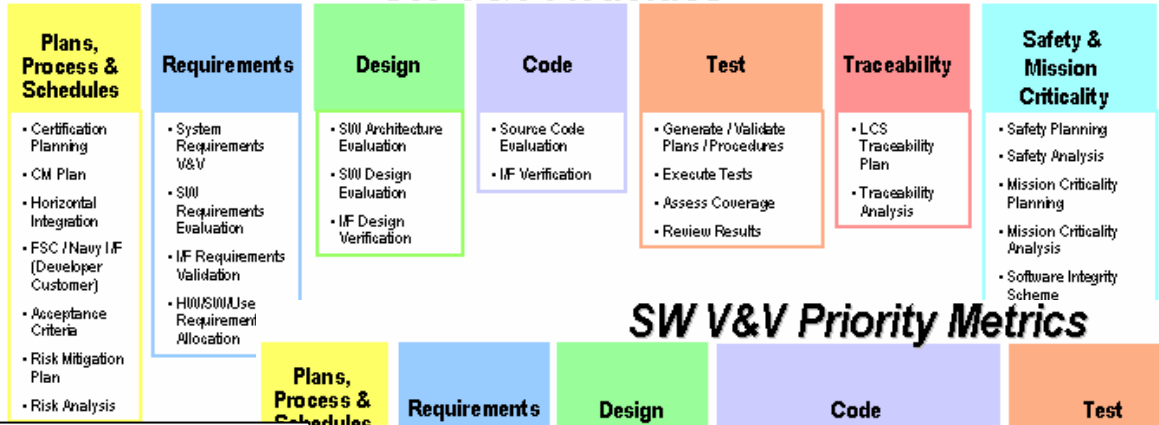
Certification is **both** a process of assuring and an act of attesting to system readiness.

# Detailed V&V Activities and Artifacts

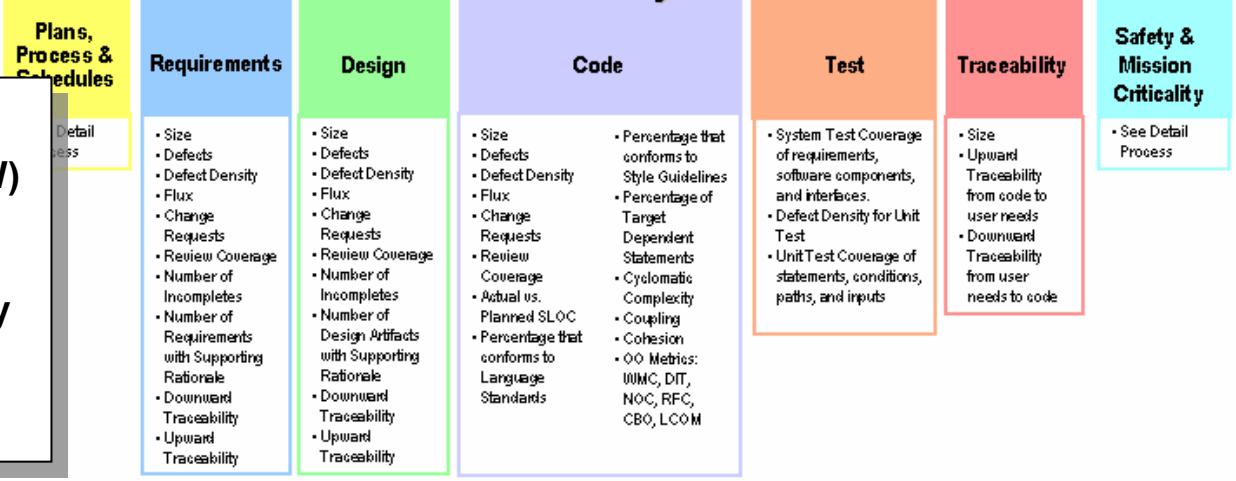
## SW V&V Artifacts



## SW V&V Activities



## SW V&V Priority Metrics

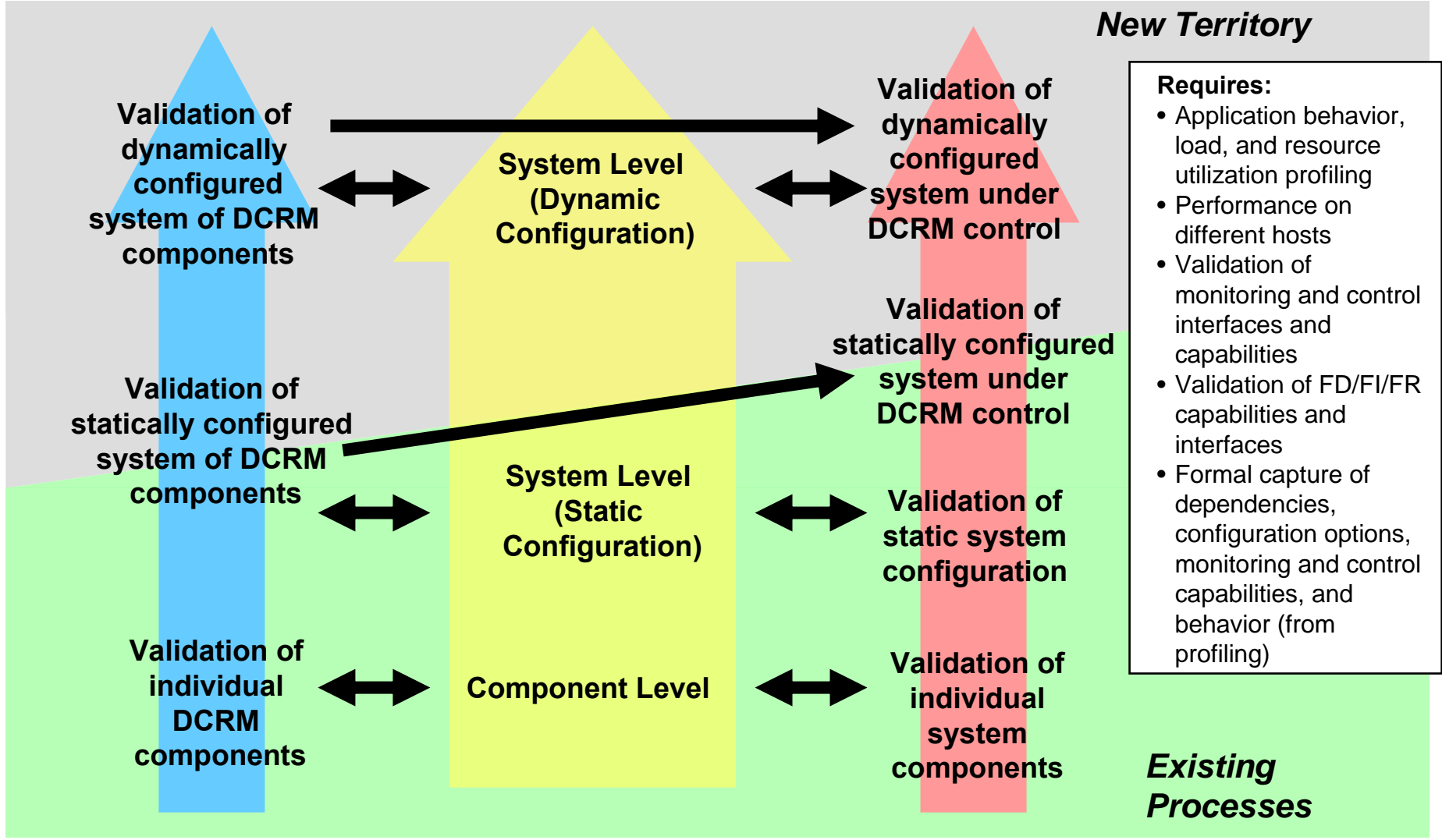


- V&V efforts are ongoing throughout the software (SW) development life cycle.
- V&V efforts for non-statically allocated systems must be bounded and affordable.

# *Research Approach*

- **Certification / V&V methodologies**
  - **Compositional V&V approach**
  - **Model and Test-based V&V approach**
  - **Run-Time Validation**
- **Detailed component analysis**
  - **Application Control selected for analysis**
    - **Key infrastructure service**
    - **Bounded scope**
  - **Requirements analysis**
  - **Architectural analysis**
  - **Design guidance**
  - **Testbed / prototyping efforts**

# Iterative Compositional Static-to-Dynamic V&V Approach



DCRM Focus      Hardware (HW) / Operating System (OS)      System Focus

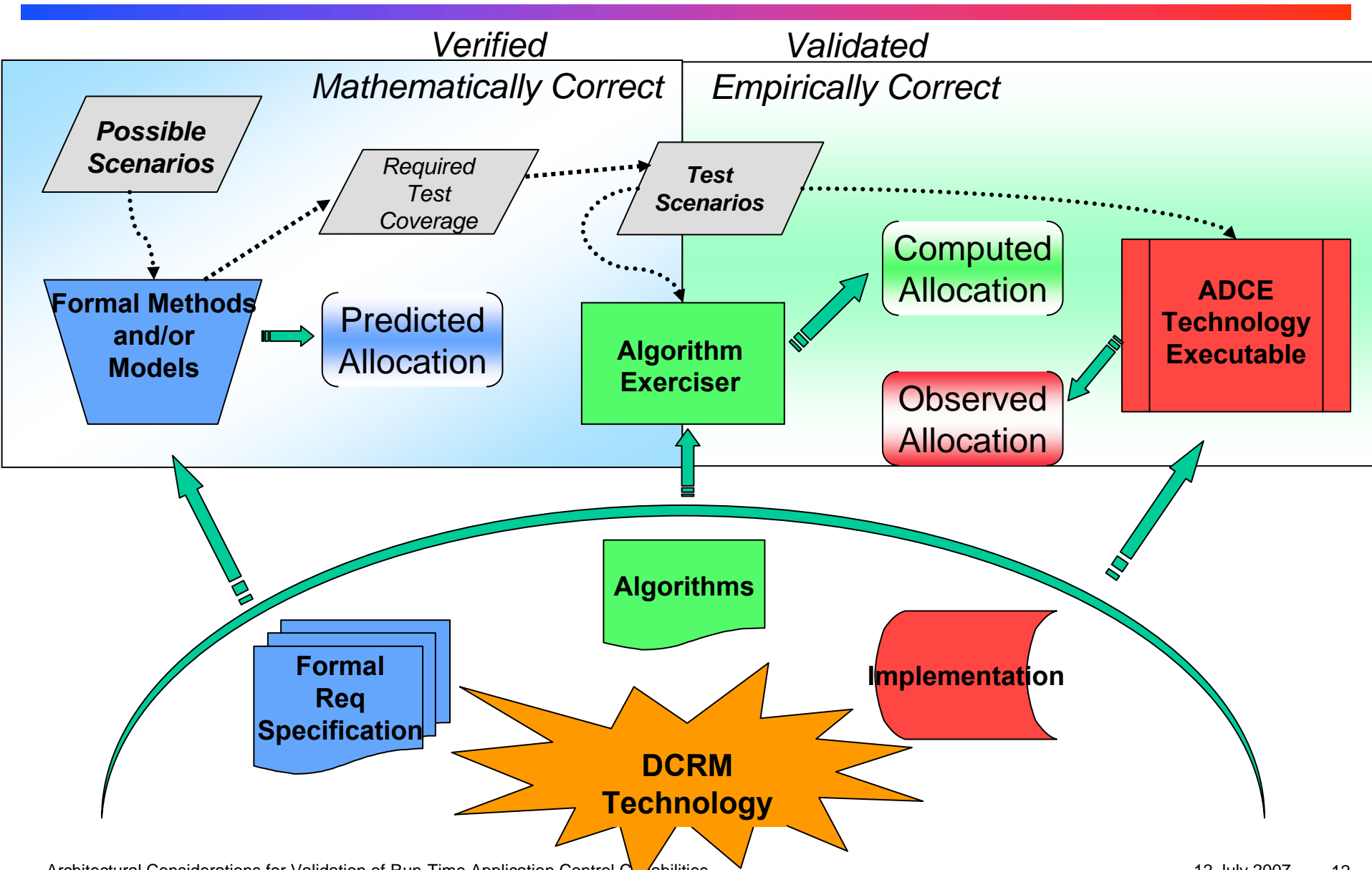
**Equivalence Classes**

# HW / OS Equivalence Classes

- **Equivalency of sets of HW and OS resources / "Virtual Homogeneity"**
- **Need clear definition:**
  - True equivalency
  - "Good enough" equivalency – How do you define "good enough?"
- **Monitoring and verification – What level is needed?**
  - Examples: bad switch ports and memory configuration deltas
- **Profiling and run-time assessment**
- **Role of application profiling**
  - Cross-hardware profiling
  - Container impacts
- **License issues and limitations**
- **Connectivity assessment**
  - Understanding topology and impacts
- **Role and limitations of location transparency**
- **Real-time priority issues**
- **Recognition of unintended and unexpected consequences**
  - Resource contention
  - Pathological interactions
- **Understanding survivability impacts and dispersion impacts**

- **Essential for Certification**
  - Secondary focus in our efforts
- **Requires extensive instrumentation at all levels:**
  - OS
  - Network
  - Middleware / containers
  - Applications
  - RM infrastructure

# Notional DCRM V&V Approach





# *Notional DCRM V&V Approach*

- **Limiting the scope of changes**
  - **Do not want changes to ripple in a manner that invalidates significant prior V&V results**
    - **Goal is to limit change to the minimal set of functional and nonfunctional requirements/design/code directly impacted.**
    - **Architectural constructs supporting explicit modularity and separation of concerns will be needed.**
  - **Some form of sensitivity analysis will be necessary.**
  - **Changes to SW development process may be necessary**
    - **Example: “Validation-critical” modules or requirements that conform to “validation-critical” design guidance and verification criteria**

# Run-Time Validation

- **Approach: Instrument the infrastructure services and applications to allow validation of functional and nonfunctional requirements at run time.**
  - System reasons about what **should** happen.
  - System determines via instrumentation what **did** happen.
- **Useful in contexts where the impact of actions is dependent on a large number of variables, some of which may not be known or are poorly understood**
  - **RM capabilities fall in this category.**
    - Interactions and relevant data at the OS, network, middleware, container model, application, and RM infrastructure levels are complex and not fully defined.
- **Minimalist instrumentation approaches should be defined and implemented to bound processing and communication overhead.**
- **Issues are raised in regards to the added complexity and potential for errors created by the validation code and infrastructure.**
  - In general, however, this would appear to reduce to roughly the potential for errors within the test harnesses and errors during post-test analysis processing.
  - Further evaluation is needed to determine the extent and impact.

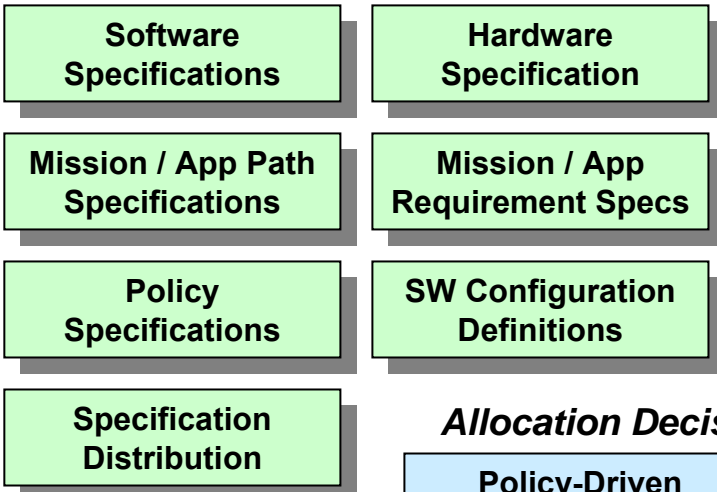
# Application Control Analysis

- **Analysis of Application Control scope**
  - **Key service within an RM infrastructure**
  - **Scope:**
    - **Application deployment**
      - Multiple application models (e.g., process, JVM, AppServers, etc.)
      - Multiple OSs
    - **Application life cycle control**
      - Start, stop, configuration
    - **State coordination**
    - **Fault signaling coordination**
      - Failure signaling from / to applications,
      - Signaling from HW/network monitoring
      - Signaling from external fault detectors
    - **Application status and state monitoring and reporting**
      - Including state-dependent startup/shutdown control
    - **Domain-specific application controls**
- **Goals:**
  - **Proof of concept for Run-Time Validation approach**
  - **Requirements, use case, and architectural analyses in support of**
    - **Assessing feasibility of a Compositional V&V approach**
    - **Assessing requirements and architectural option spaces in regards to impact on generation of design guidance and propagation of changes within a Model and Test-based V&V approach.**

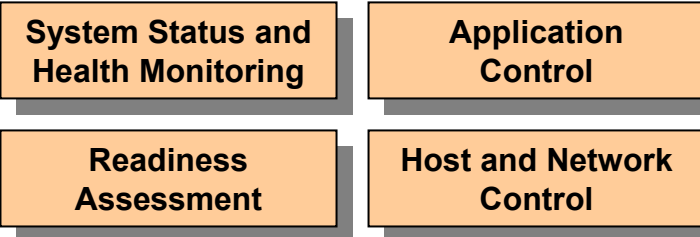
• Capabilities defined as minimum Application Control scope

# Application Control Within DCRM Functional Decomposition

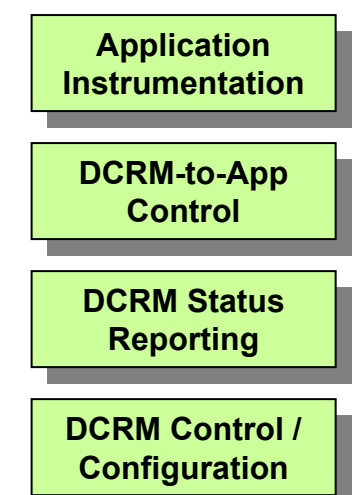
## Specifications



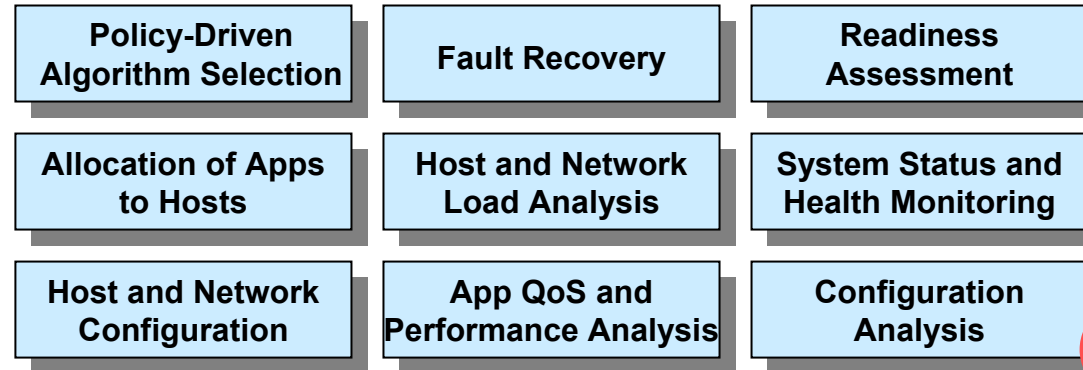
## Visualization / Display Interfaces



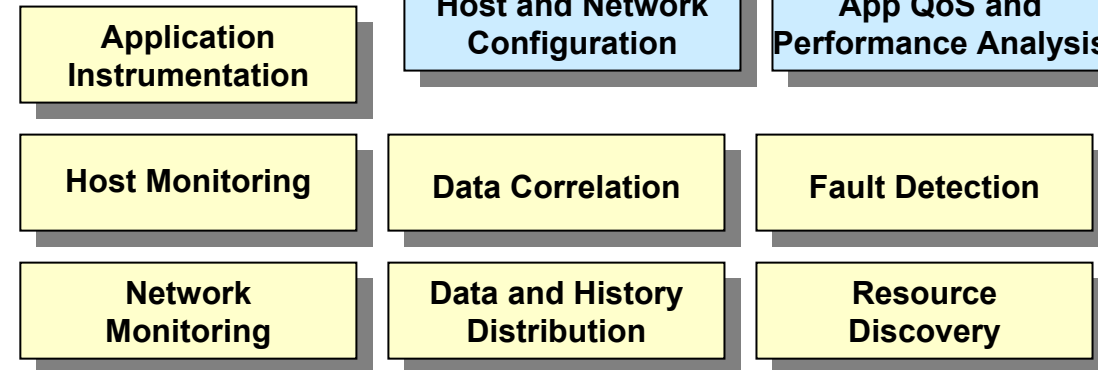
## External Interfaces



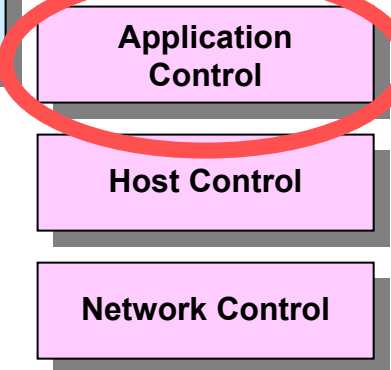
## Allocation Decision Making



## Monitoring



## Control



# *Requirements and Architectural Analysis*

- **Approach**
  - **Requirements analysis – three requirement sets**
    - **Minimum Application Control capabilities**
    - **Extended Application Control capabilities**
      - Multiple app models, fault tolerance support, specification support
    - **Full Application Control capabilities**
      - Policy-driven, domain-specific controls
  - **Architectural analysis – three architectures**
    - **Script-based**
    - **Distributed agent-based**
    - **Hierarchical agent-based**
- **Analysis of the matrix of architectural approaches vs. requirement sets is ongoing.**
  - **Analysis focusing on**
    - **Impact on Run-Time Validation**
    - **Scope of change propagation within the Model and Test-based V&V approach**

# *Application Control Testbed*

- **Initial tests using the OA Prototype Program Control capabilities**
  - Running on Solaris and Linux
  
- **Initial Run-time Validation tests**
  - OS-level validation of process configurations
    - Program Control-based startup and configuration (command-line parameters, environment variables, pgid, etc...) of processes
    - /proc filesystem monitoring of processes and configuration information
  - Detailed latency measurements for startup of multiple processes
    - Processes with and without interdependencies
  - Initial internal monitoring of Application Control state management during fault scenarios

# Next Steps

- **Analysis of COTS Application Control products**
  - Analysis of capability, instrumentation, and architectural design deltas
- **Analysis of Application Control-related standards**
  - OMG Application Management and System Monitoring Specification
  - SAForum Application Interface Specification
  - OMG Deployment and Configuration Specification
  - DMTF CIM / WBEM
  - Others...
- **Detailed prototyping and analysis as a proof of concept for the Compositional V&V approach**
- **Future investigation to formalize guidance and approaches for isolating change within Model and Test-based V&V approaches**

**In order to take advantage of emerging technologies that allow dynamic allocation and control of system resources, new approaches are needed for the testing, validation, and certification of dynamically configurable systems.**

# *Acknowledgements*

- **We would like to express our thanks to the following people for their valuable insights and assistance in support of this analysis:**
  - **Carol Galloway (NSWCDD)**
  - **Earl Sam (DCS Corporation)**
  - **Willie Sullivan (NSWCDD)**
  - **George Snider (NSWCDD)**

**DISTRIBUTION STATEMENT A.** Approved for public release; distribution is unlimited.

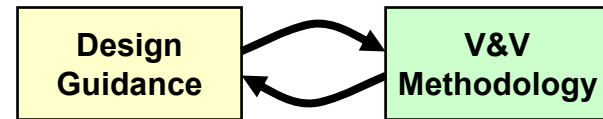


# *Backup Slides*

# Role of Design Guidance

- **Four design guidance categories:**
  - Highest Payoff {
    1. **Guidance applicable across DCRM V&V approaches and DCRM architectures**
    2. **Guidance for a specific set of DCRM V&V approaches, which is applicable across DCRM architectures**
  - Lower Payoff {
    3. **Guidance for a specific set of DCRM architectures, which is applicable across DCRM V&V approaches**
    4. **Guidance for a specific set of DCRM V&V approaches, which is applicable only to a specific set of DCRM architectures**

- **Iterative refinement process required:**



- **Detailed analysis of Application Control architectures, requirements, and interactions is ongoing**
  - **First step for assessing:**
    - General vs. specific guidance
    - Mapping from requirements to modeled capability behavior
    - Interactions with system applications and other DCRM components
    - Interface / data needs
    - Role of a notional architecture

# *Application Design Guidance / Best Practices for DCRM Control*

- **Application startup and shutdown under DCRM control**
  - Initial configuration of applications via command-line parameters and/or environment variables
  - Ability to cleanly shut down applications via signals or scripts
- **Application-level instrumentation**
  - Performance and status data
  - Application state and state changes
  - Internal processing loads
  - Occurrence of key events
  - Internally detected error conditions
  - Internal view of interface statuses (network and middleware)
- **Load-invariant application design**
  - Rather than load-adaptive design
  - Allows DCRM to optimize system and application performance
- **Minimal coupling with control infrastructure**
  - Required by control theory approaches
- **Application specifications**
  - **Hierarchical software specifications**
    - Software systems, software subsystems, applications
  - **Application startup and shutdown information**
    - Processes to run
    - Command line arguments (dynamically configurable)
    - Environment variables
  - **HW and OS requirements**
  - **Middleware and software dependencies**
    - Startup, shutdown, and run-time dependencies
  - **Application states and state transitions**
  - **Performance profiles and Quality of Service (QoS) Requirements**
  - **Path structure and data flow**
  - **Survivability / fault tolerance / scalability capabilities**
  - **Security capabilities and requirements**
  - **Static and/or dynamic fault configurations**