Analytic Certification Technologies for Military Avionics

RTSS 2012 AVICPS Position Paper

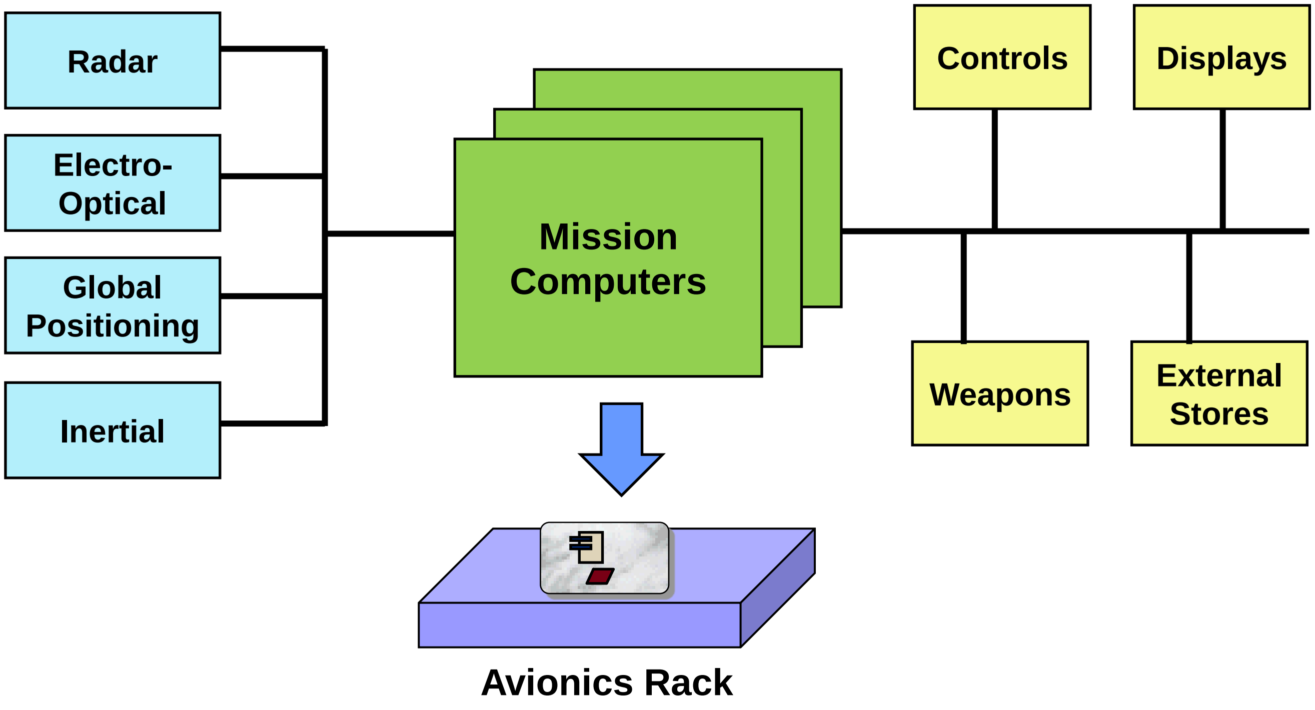
4 December 2012 – San Juan, Puerto Rico

Russell B Kegley, LM Fellow
Jonathan D Preston, LM Fellow
Lockheed Martin Aeronautics
Fort Worth, Texas

LOCKHEED MARTIN

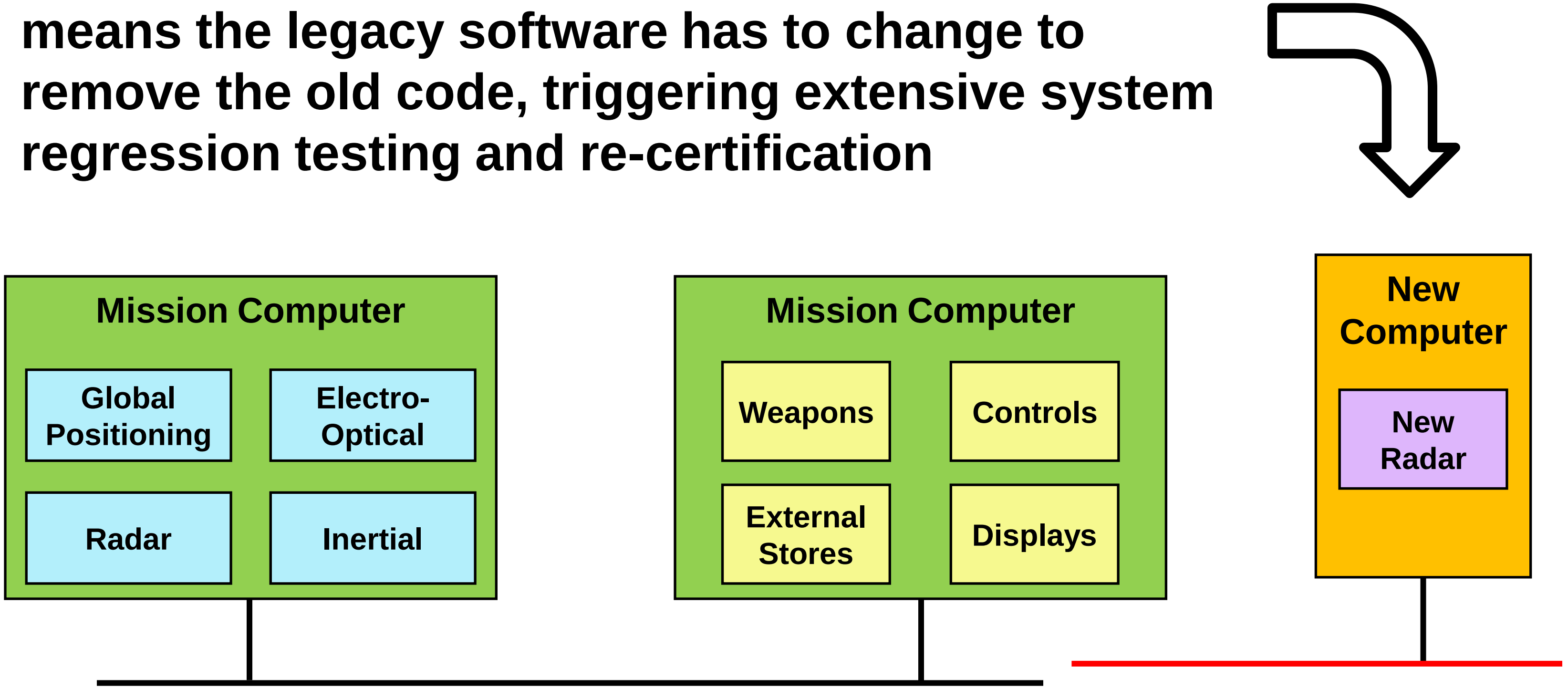
Historic Federated System Architecture

- Functions were federated into hardware subsystems which were largely independent
- Each function generally had a single processor to itself
- Thus, updates to one function or replacement of a single piece of hardware did not affect other functions very deeply



Current Integrated System Architecture

- Multiple mission functions run on each processor
- Overall timing is dependent on how each function executes
- Changes to one component ripple throughout the system
- Even re-hosting a function on a new computer means the legacy software has to change to remove the old code, triggering extensive system regression testing and re-certification

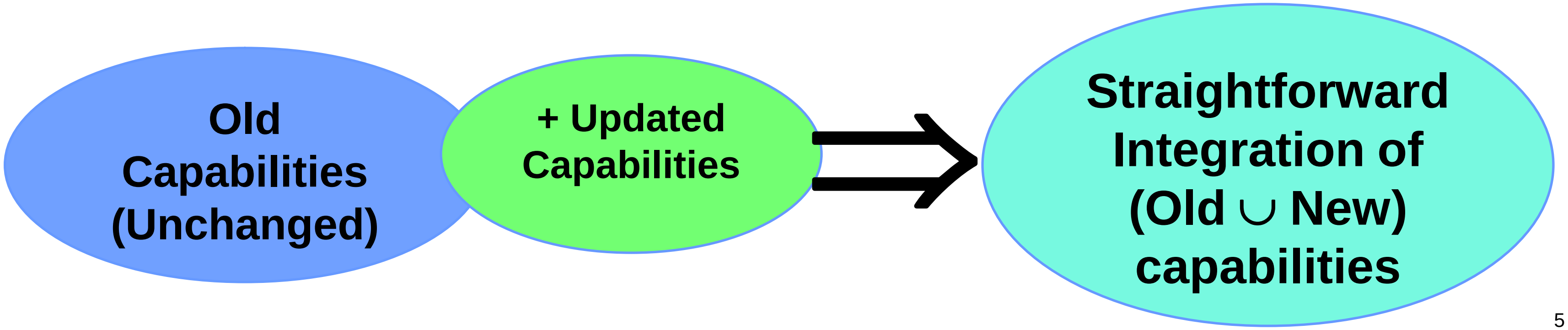


Summary: Legacy vs. Future Differences

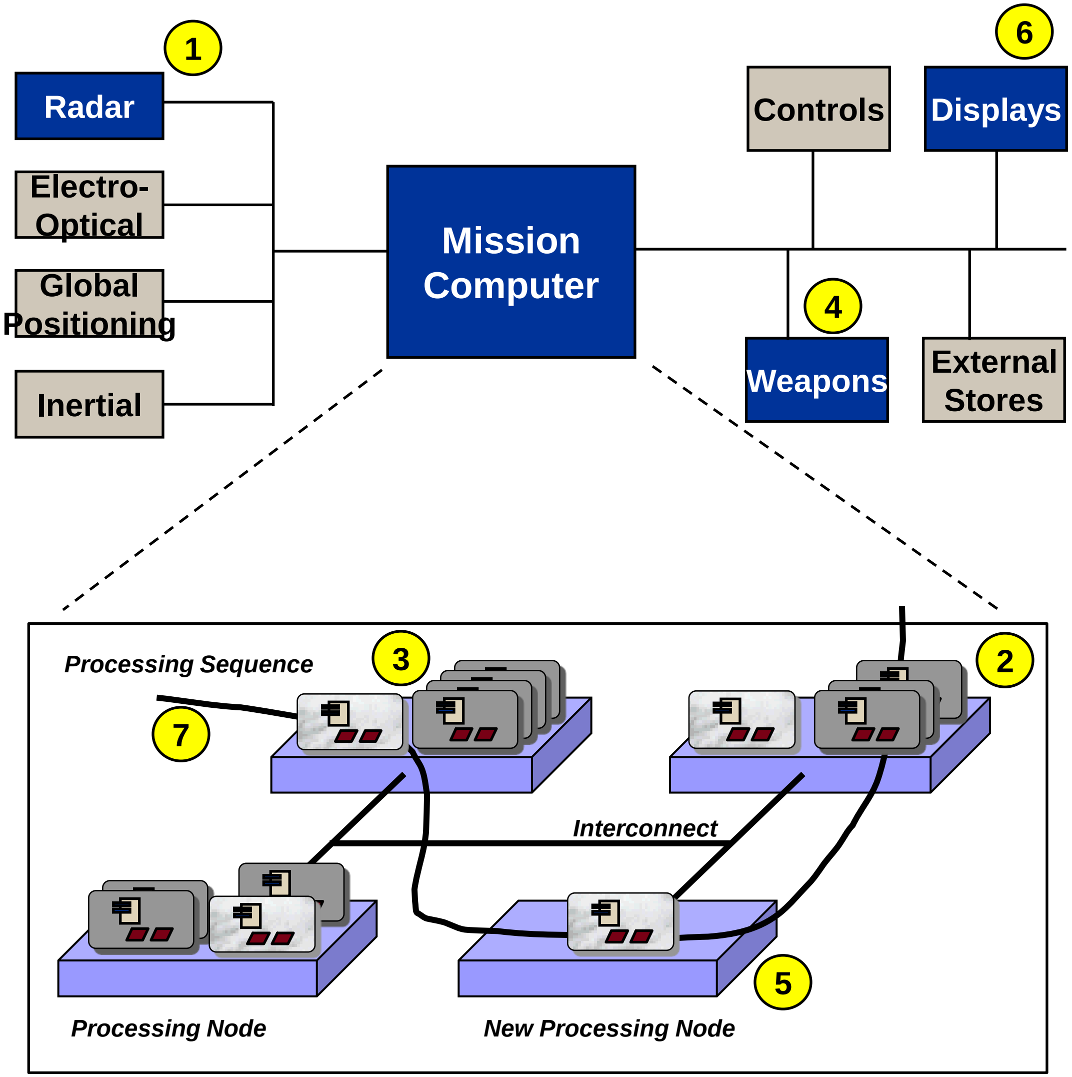
	Legacy System	Future System
Processing Hardware	• Single CPU per subsystem • Military-specific instruction set • Single-core CPU's	• Multiprocessor subsystem • Commercial CPU's (suitably hardened) • Multicore technology
Operating Systems	• Custom developed • Fixed rate scheduling • Timesliced temporal architecture	• Commercial High Assurance RTOS • Mix of rate- and event-based processing • Preemptive priority based scheduling
Networks	• Custom or MIL standard	• Commercial off-the-shelf
Applications	• High degree of hardware dependency • Minimal dependencies between applications • Functionally constrained by computing resources, inducing high degree of complexity	• Hardware independent • Increased interdependency between software functions • More complex capabilities leveraging increased hardware resources

Proposed Refresh Scenario

- Affordability a requirement for future technology refreshes
  - Upgrade budgets are expected to be much smaller
  - Fewer new platforms are expected to enter the inventory
- Strategy: Insert only the new capabilities
  - Insert new software, and optionally new hardware, that interacts with new components
  - Leave old software untouched by “virtually” removing it
- Benefits
  - Avoid re-development of existing software which still works
  - Minimize changes to existing software to lower retest costs
  - Elimination of very costly re-certification of old software
  - Focus development budget on providing new capabilities

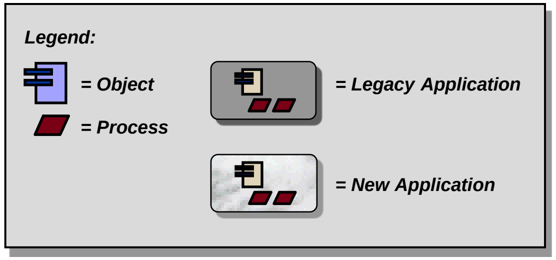

Old Capabilities (Unchanged)
+ Updated Capabilities
Straightforward Integration of (Old ∪ New) capabilities

Specific Challenges



Upgrade Scenario:

1. Replacement of Radar subsystem with a newer version having a different temporal architecture requiring “glue code” & analysis to accommodate
2. Modified processing sequence in Mission Computer resulting in unused code in legacy applications
3. Integration of new applications on Mission Computer having different design styles (e.g., event vs. rate based)
4. New weapon with different communication protocol or medium
5. New node in Mission Computer with different O/S and infrastructure code
6. New capabilities and/or information presentations in Displays
7. Reconciliation of Fault Logic all along modified & new Processing Sequences


Legend:
= Object
= Process
= Legacy Application
= New Application