



Naval Open Architecture

Family of Systems Engineering



December 2006

***Distribution Statement A: Approved for public release;
distribution is unlimited.***

***Kathleen Emery
PEO Integrated Warfare Systems
Technical Director, Acting***



OA requires a shift in the Navy's systems engineering approach to develop capabilities across multiple platforms / families of systems.

Open Architecture Model

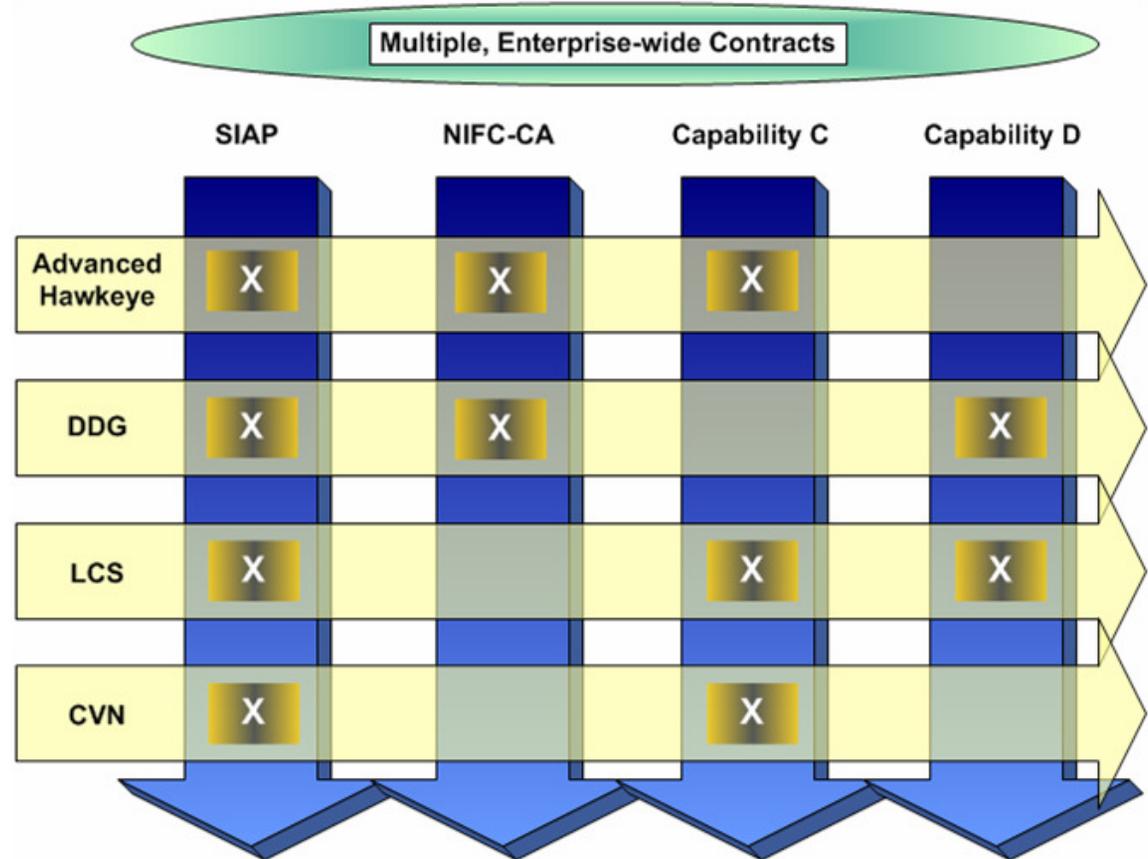
Business Model Attributes:

- Market controls evolution
- Total Ownership Cost emphasis
- License or Reuse software
- Leverage COTS or Reuse

System Model Attributes:

- Market-driven infrastructure
- Warfighter-driven capabilities
- Enterprise-wide business plan
- Flexible requirements
- Open system architecture
- Design for evolution (tech refresh)
- Planned, enabled, enforced reuse
- Early-managed obsolescence
- Spiral development

Capability / System-Based vice Platform-Based



Family of Systems - A set of systems that provides similar capabilities through different approaches to achieve similar or complementary effects.

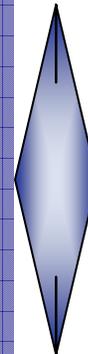
System of Systems - A set or arrangement of interdependent systems that are related or connected to provide a given capability. The loss of any part of the system will significantly degrade the performance or capabilities of the whole.



What Surface Navy Expects to Achieve from OA

Business Characteristics

- Increased **access** to technologies and products supported by many suppliers
 - Sharing design documents and source code with 3rd parties
- Business-case driven:
 - Software reuse
 - Commodity hardware and infrastructure
 - Capability-driven application portfolios
- Use of commercial products from multiple sources; competition for custom development
- Use of integrated product teams / peer reviews
- Increased competition to spur innovation
- OA language in legacy and new contracts



Technical Characteristics

- Component architecture to allow for affordable upgrade and reuse
 - Build once, field many times
 - Improved interoperability
 - Reduced training and support costs
- Open, published component interfaces
- Commodity infrastructure and shared application portfolios
- Layered, modular system designs to accommodate changing technology and requirements
- Widely adopted industry standards
- Spiral developments to enable technology / capability insertion

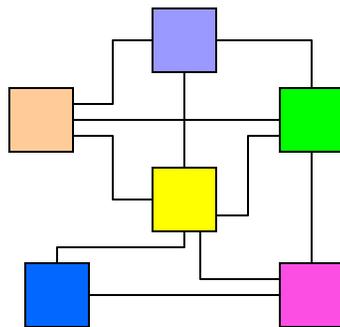
Characteristics will help drive costs down while increasing capability



Transform processes and standards from As-Is to To-Be

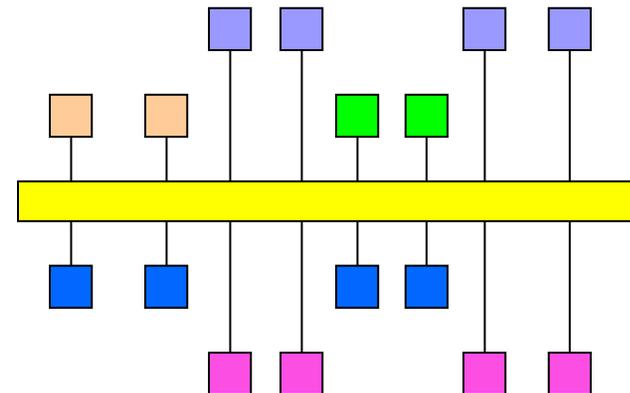
As-Is

- Monolithic ship-specific architectures
- Point-to-point interfaces
- Coupled hardware and software
- MILSPEC computing equipment
- Bundled developer/system integrator
- “Clone and own” reuse



To-Be

- Common combat system architecture with multi-platform common core
- Layered, componentized designs
- COTS standards-based environments
- Prime integrators leverage products from multiple developers
- Collaborative, peer-reviewed solutions
- PARM-managed component portfolio





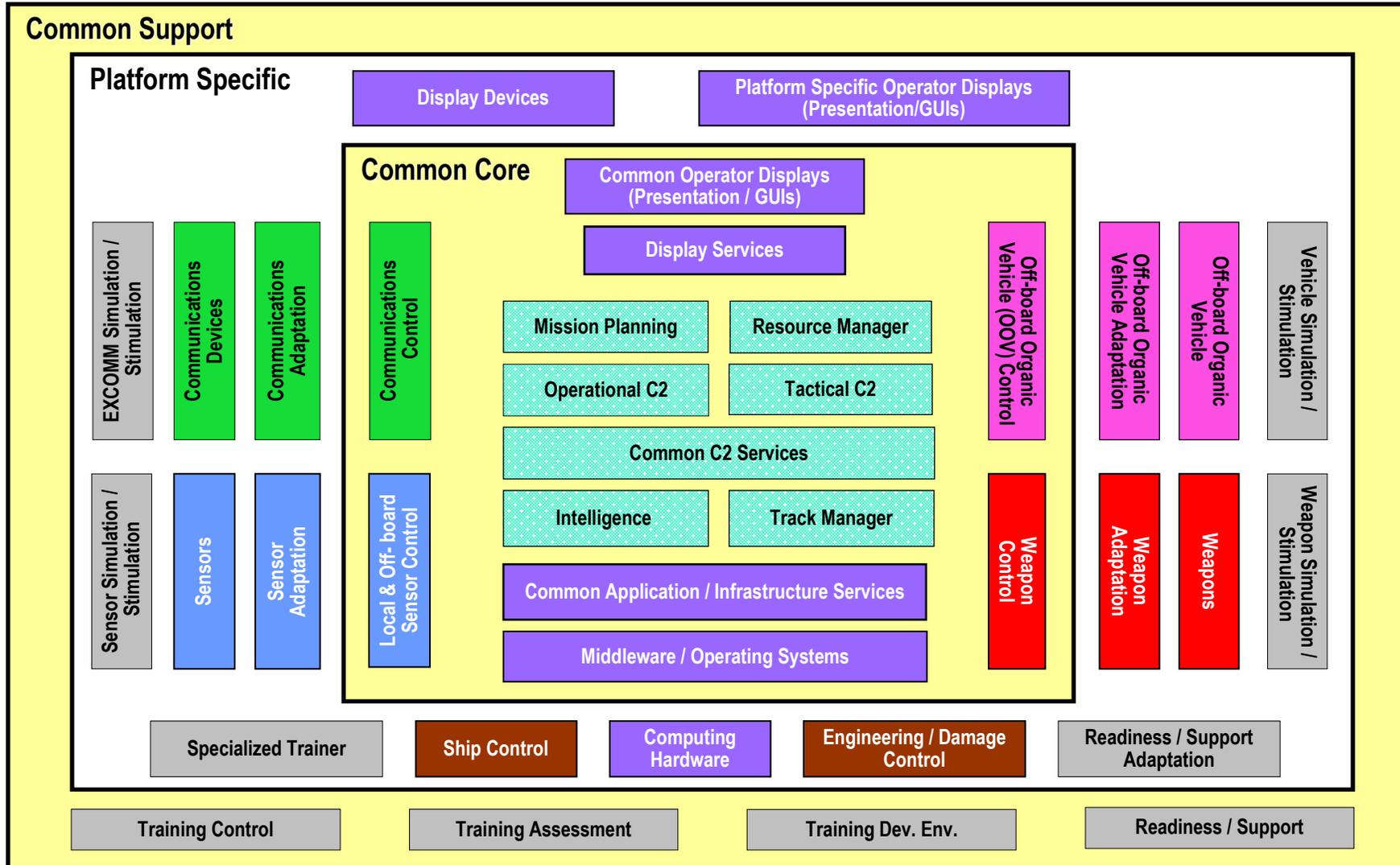
Central to the technical approach is a common software architecture

- In spite of different architectures, surface ship combat systems perform similar functions, sometimes with common implementations
- Define a standard common component architecture that:
 - Identifies the major components of surface ship warfighting systems, including:
 - detect-control-engage for multiple warfare areas
 - ISR, METOC, and C4I data integration
 - Mission planning and Command & Control (platform-level and netted force)
 - Readiness – logistics, training
 - Is decomposed to a level that can support:
 - Assignment to an organizational element (PM) responsible for development and maintenance
 - Mapping of existing systems within portfolio to identify shortcomings and duplication
 - Prioritizing investment decisions and POM issue requests
 - Provides a stable framework into which S&T activities can transition
- Transform from baseline-specific architectures to a common software architecture in concert with planned capability upgrades
 - Align existing development efforts to maximize reuse
 - Look for large-grained reuse to achieve most savings
 - Develop new capabilities in accordance with common architecture



Preliminary Surface Domain Enterprise Architecture/WBS

(Revision 0.97)



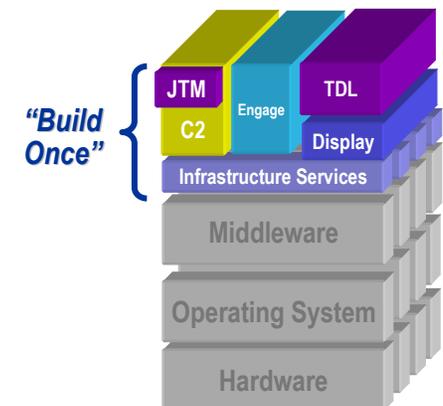
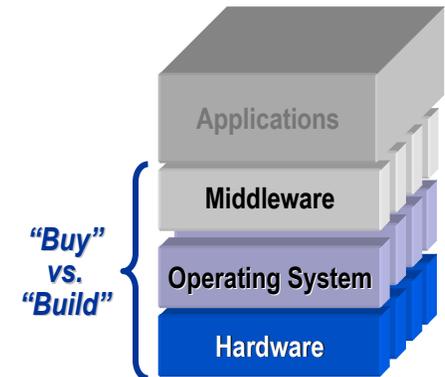
Domains





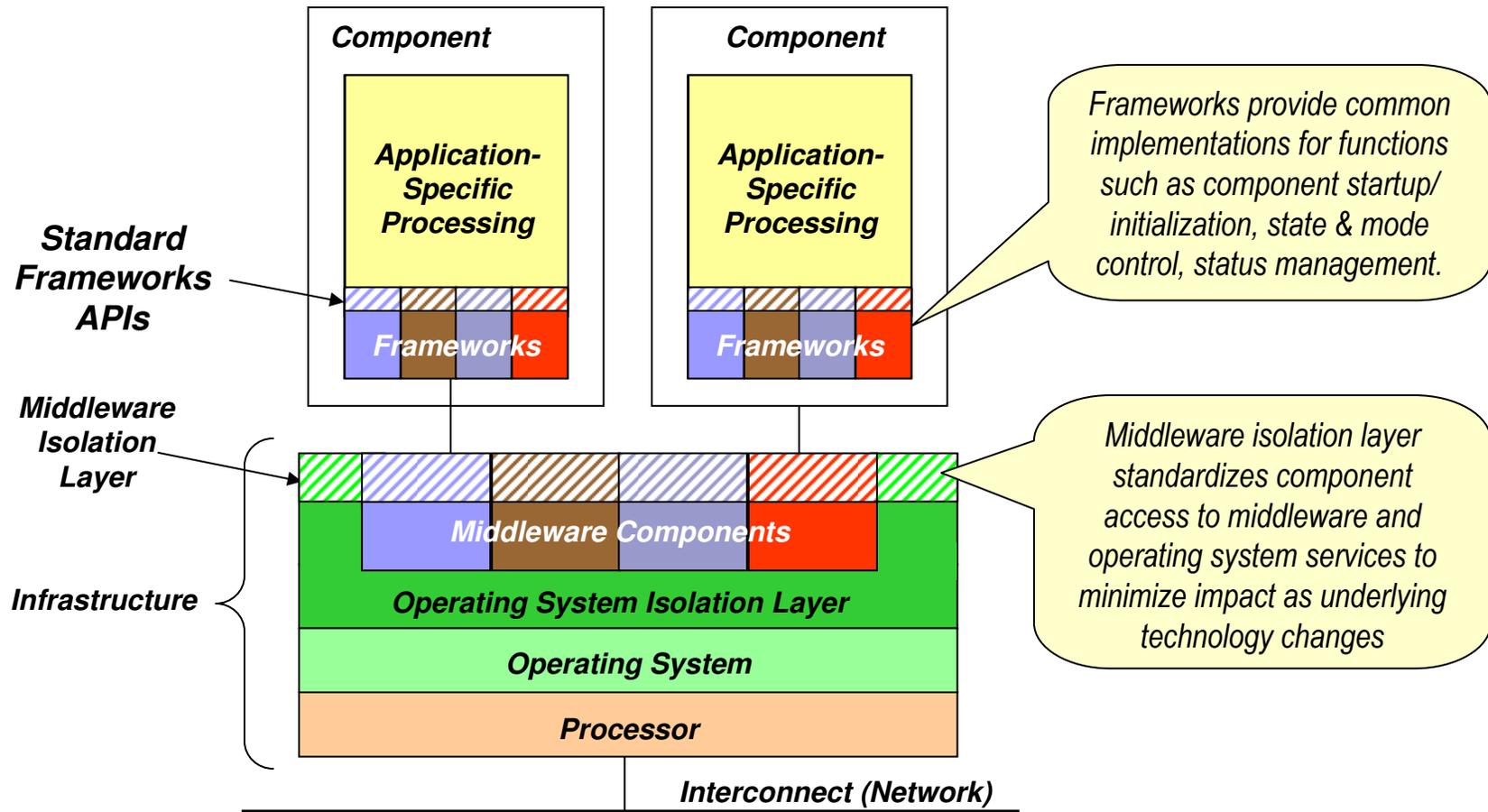
Designate Key Interfaces / Open Standards

- Standard Combat System Architecture defines key interfaces
 - Establish key interfaces to support extensibility and reuse
 - Interfaces based on common data model
 - Control interfaces / data model via Architecture Control Board
- Open system interface standards selected for commodities
 - Generally found in computing environment and common services layers
 - De facto standards selectable when open system standards don't exist and de facto standards are supported by multiple suppliers
- Custom interfaces specified where open standards don't exist
 - Interfaces must be openly published and non-proprietary
 - Defined to level required to assure interoperability of conforming products developed by different suppliers
 - Designed for extensibility and reuse
 - Subject to community peer review





Layering Supports Application Portability and Reuse

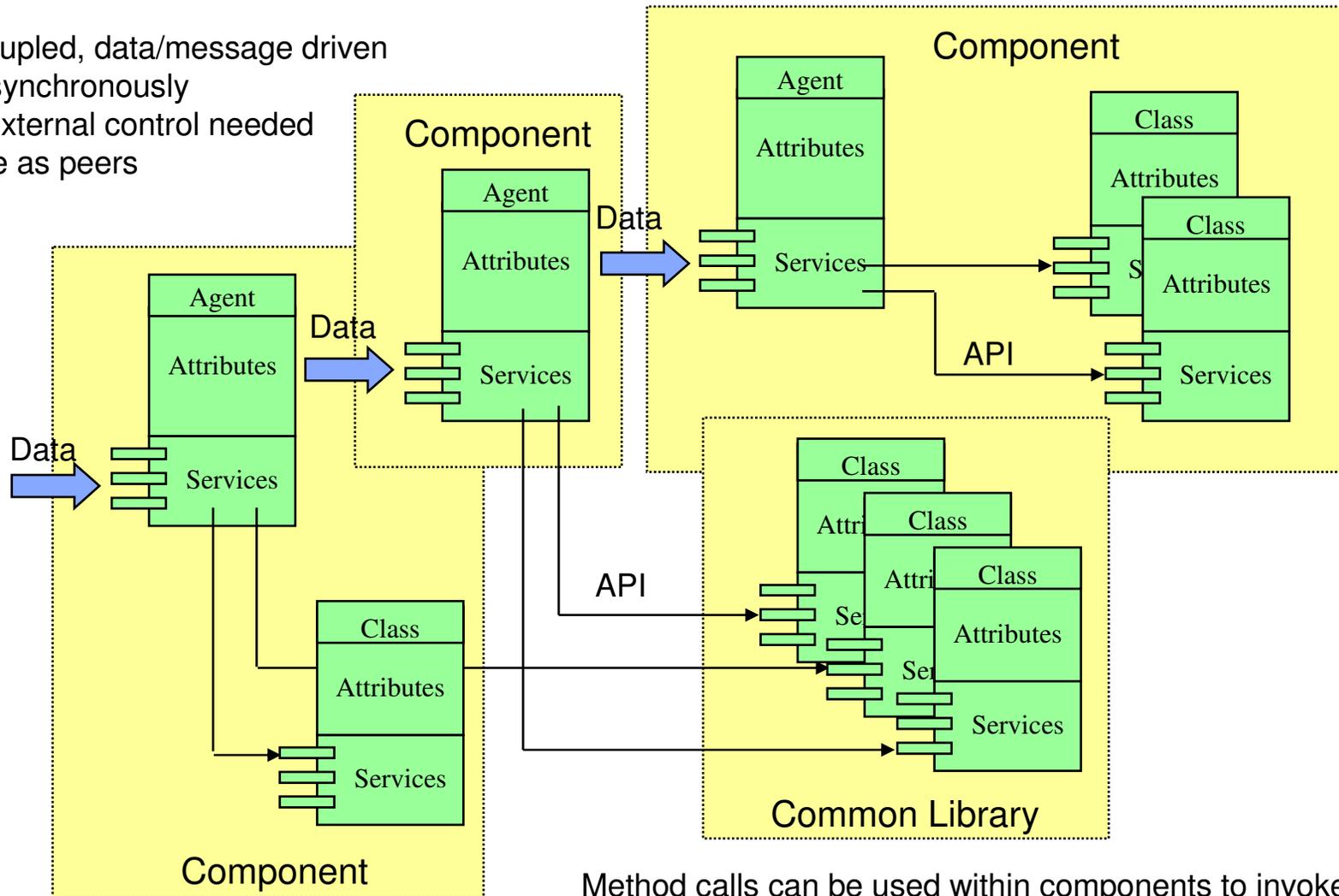


Common frameworks APIs will enable application reuse across programs. Common, standards-based Infrastructure will enable easier COTS refresh.



Decentralized, autonomous entities (collaborating peers)

Loosely-coupled, data/message driven
 Operate asynchronously
 Minimum external control needed
 Collaborate as peers



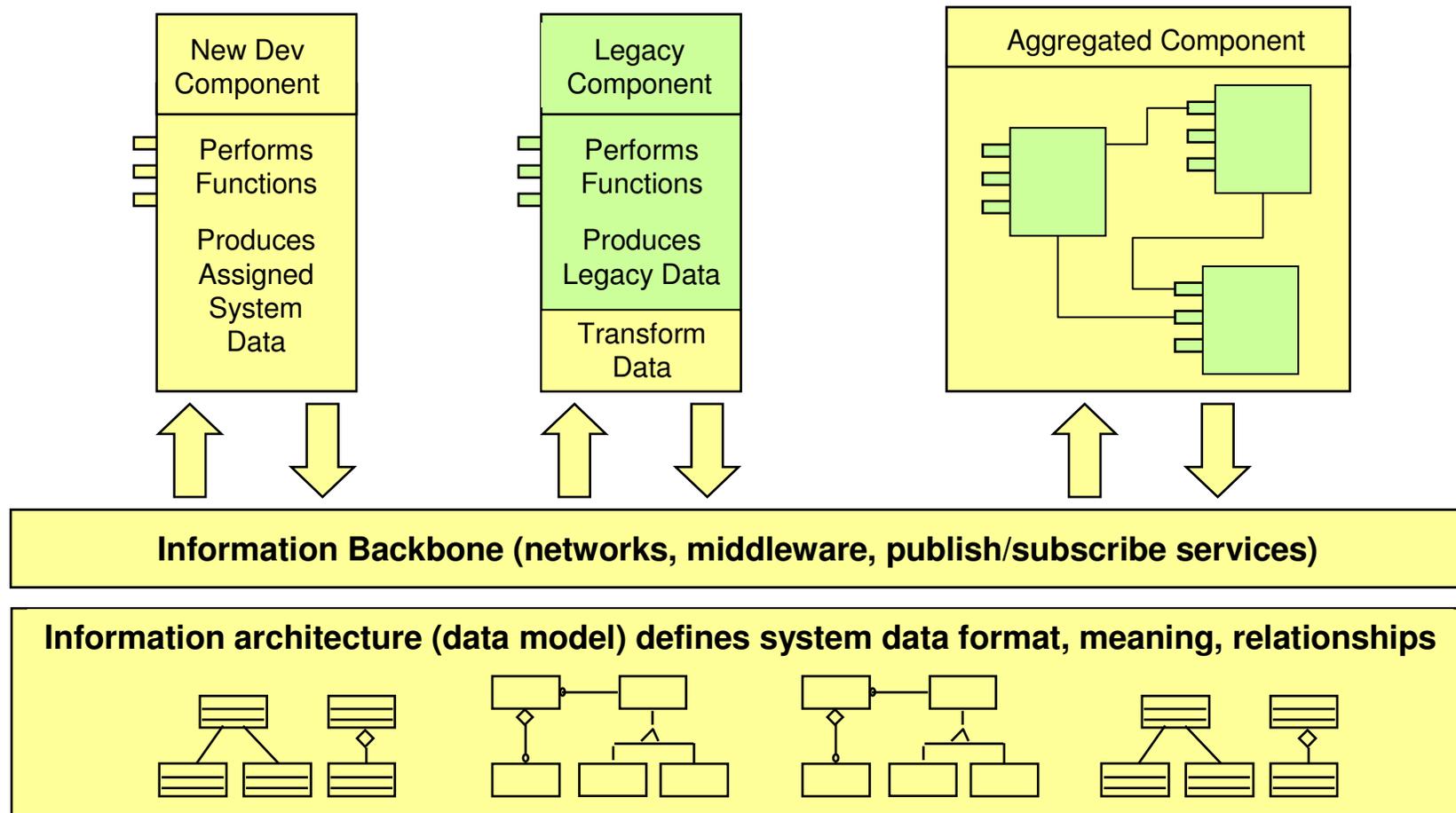
Method calls can be used within components to invoke services of other classes. Classes may be embedded within active components or provided as common library services for use by many components.



Distributed component architecture based on a common data model

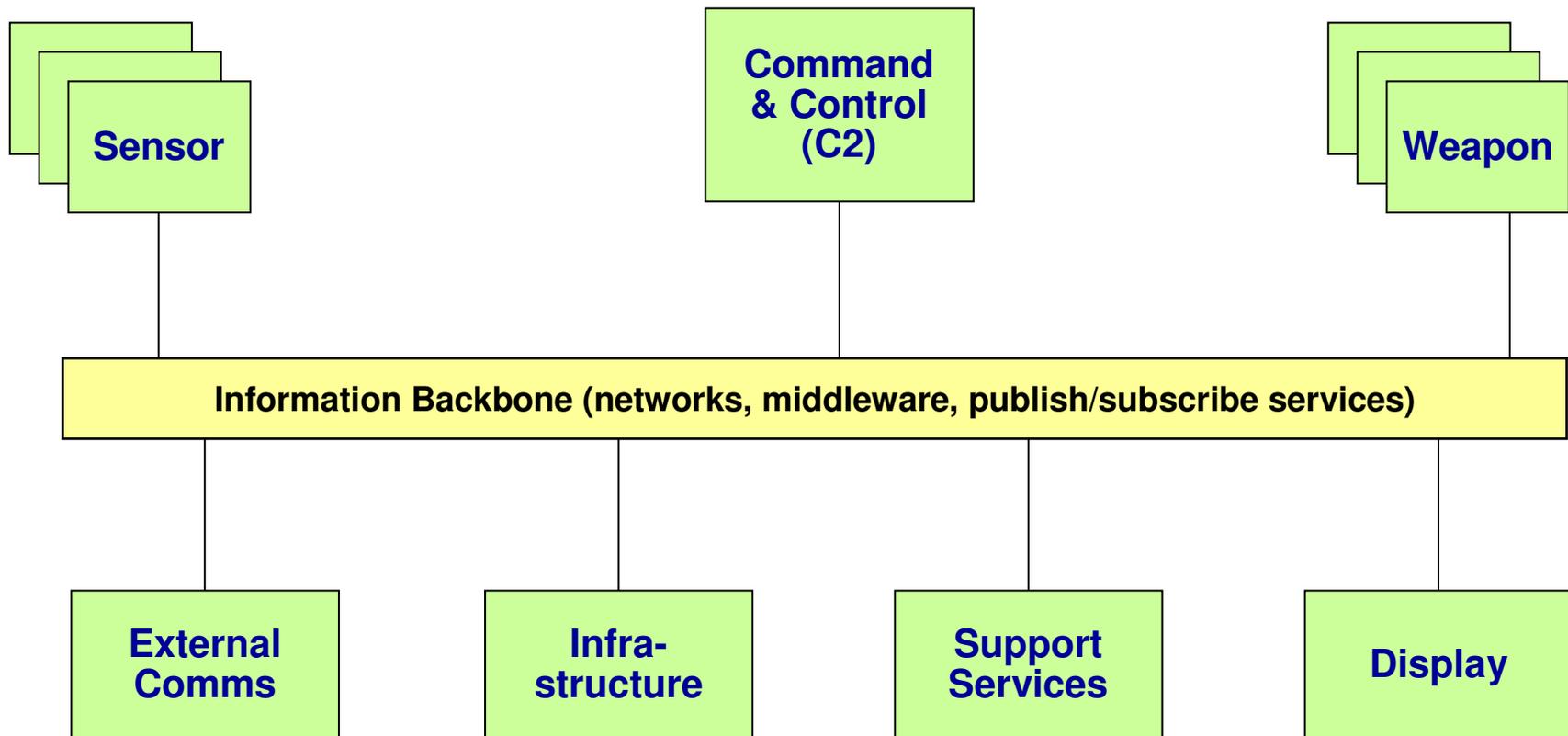
- Legacy Software
- Architecture-Compliant SW

- ◆ Components are responsible for managing/publishing assigned system data
- ◆ Components subscribe to the data they need to do their jobs
- ◆ Components can translate legacy software/data to match common data model
- ◆ Components can be any size, and can wrap non-componentized legacy software



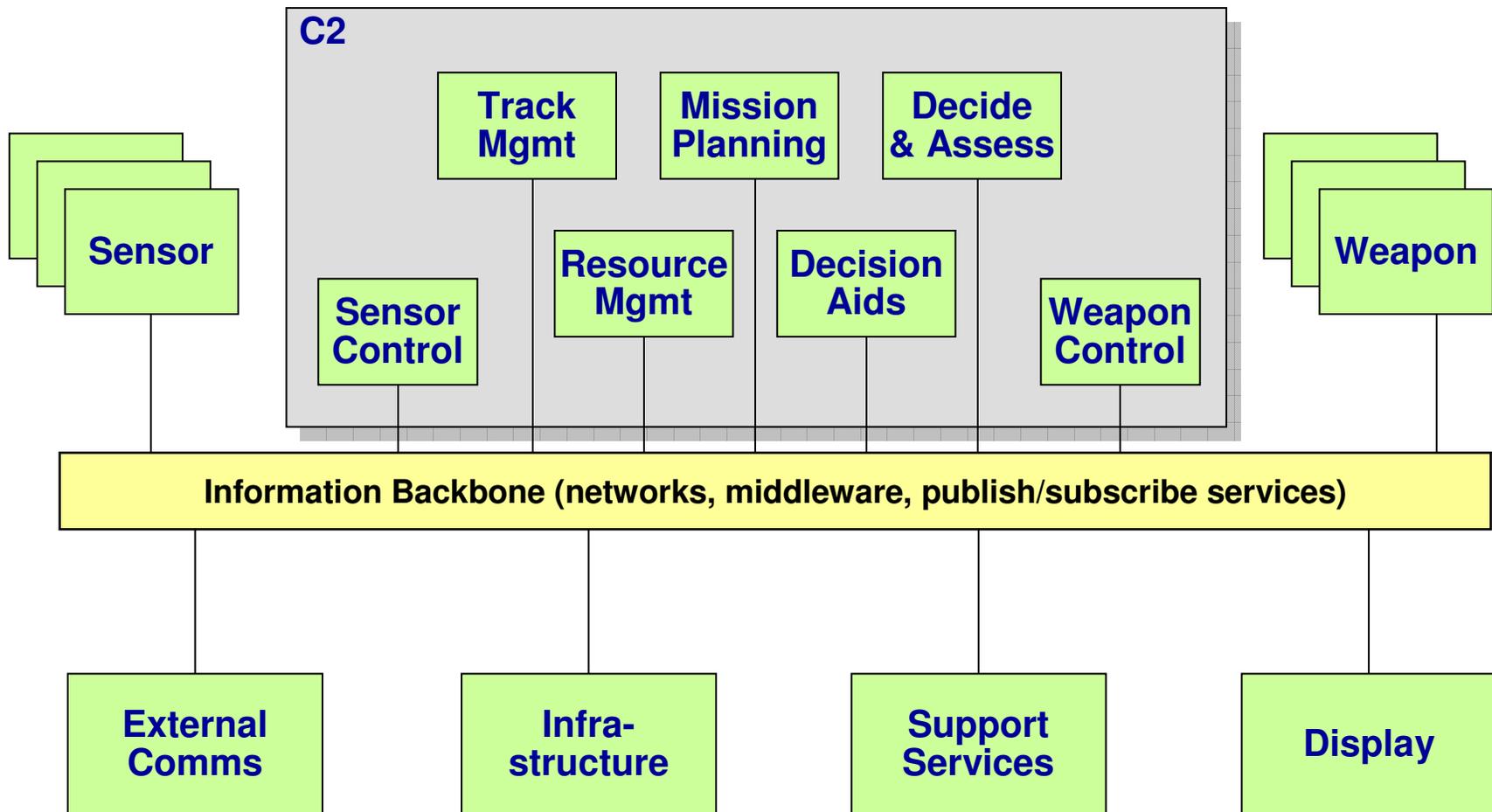


Components can be major segments of a Combat System



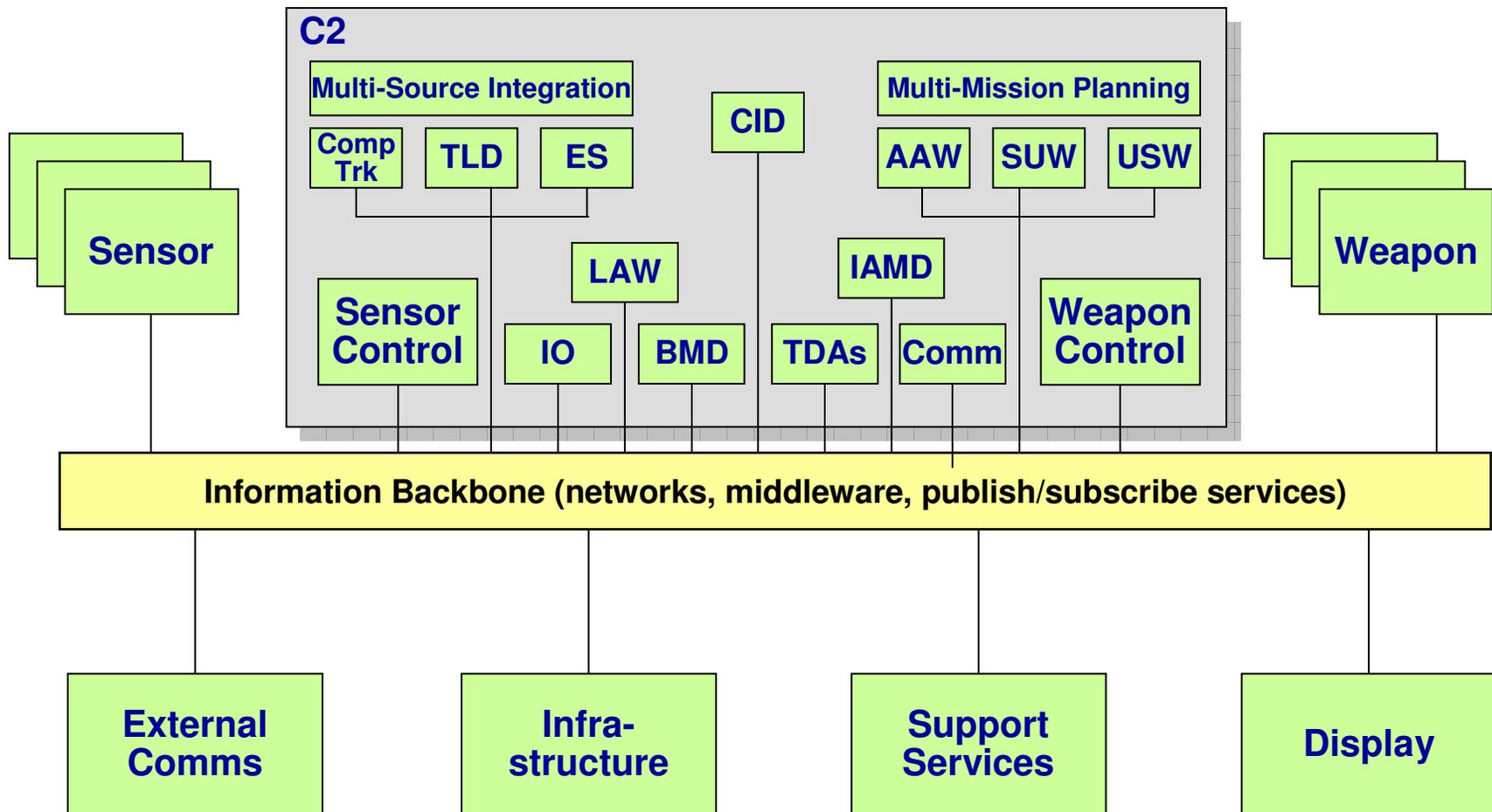


Components can be made more granular over time without impacting existing components through continued conformance to data model





Components can be made more granular over time without impacting existing components through continued conformance to data model





Modular Open System Architecture Design Precepts

- Layered architecture with clear separation of concerns
 - User presentation/visualization from business (application) processes
 - Common core application processes from element-specific processes
 - Application processes from supporting data and services
 - Supporting services from computing environment
- Technology-based levels compliant with well-supported industry standards
- Key techniques for defining an extensible component architecture include:
 - Object-oriented analysis – components that model real-world entities
 - Domain analysis – generalization and specialization of components reflects commonality and variability across a family of warfare systems
 - Information modeling – clearly defined system information (entities, attributes and relationships), with responsibility for maintaining any given data element residing in a single component



Product line development requires some additional products

Engineering Acquisition Products

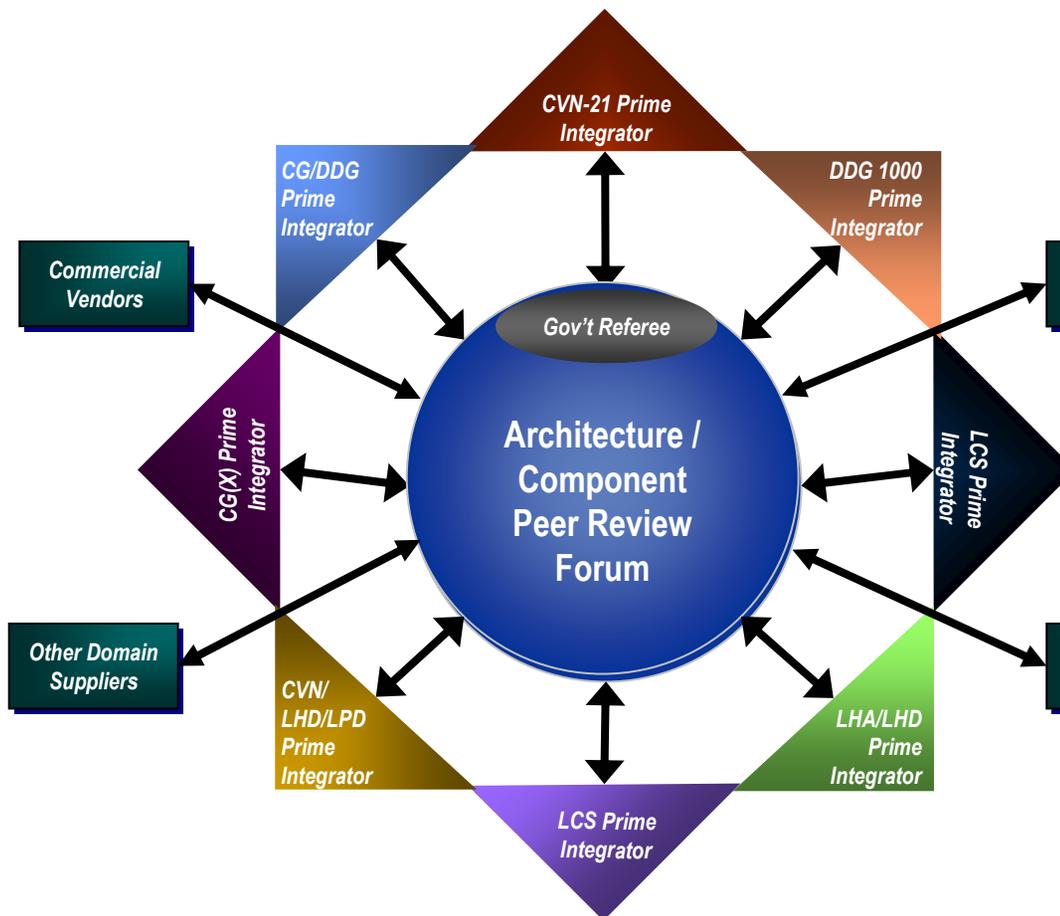
- Architecture Precepts Document
- Common architecture / data model
 - Component definitions
 - Significant system threads
 - Architecture Description Document
- Enterprise System Engineering Plan
 - Architecture Review and Control
 - Artifacts / Standards
 - Peer Review / Tech Review Team
 - CM / Acceptance / Certification
- Component Software Specifications
 - Functional & performance
 - Design & construction
 - Interface

Programmatic Products

- Capability Development Strategy
- Work Package Requirements
- Combat System Master Plan
 - Combat System Baselines
 - Integrated Schedule
- Migration Plan
 - Component Evolution Plan
 - Baselines and Schedules
 - Funding Alignment
- Core Asset Repository
 - Specs, design documents
 - Source code, build scripts
 - Models, simulations
 - Test plans, procedures, results
 - User's guides, training, ILS



Open development model balances cooperation and competition



- Gov't Purpose Rights (or greater) source code
 - Intent is to make available to integrators and potential component developers within SHARE
- Open access to facilities
 - Small businesses can demonstrate product capabilities
 - Integrate products with existing combat systems
 - Prototyping
- Open peer review process allows:
 - Primes to participate in product selection (retain end-to-end performance accountability)
 - Other contractors to compete for future extensions or subsystem integration efforts



Summary

- An objective component architecture based on a common data model will provide the foundation for developing common core assets
- A cooperative, yet competitive, culture will be required to achieve the desired reuse, efficiencies and innovation
 - Industry expertise leveraged via open peer reviews
 - “Badge-less” experts from within community recommend way ahead
 - Government facilitates/referees; industry delivers
- Competition can be held at different levels
 - New component development or upgrade of existing components
 - Combat system software integrator – common core
 - Independent verification & validation
 - End-to-end platform integrator
 - Enterprise architect and portfolio management
- Government acceptance of components before reuse
 - Certification in the context of each surface ship combat system
 - Accepted component artifacts maintained in common asset repository