










# 9LV Mk4

Use of DDS for system integration

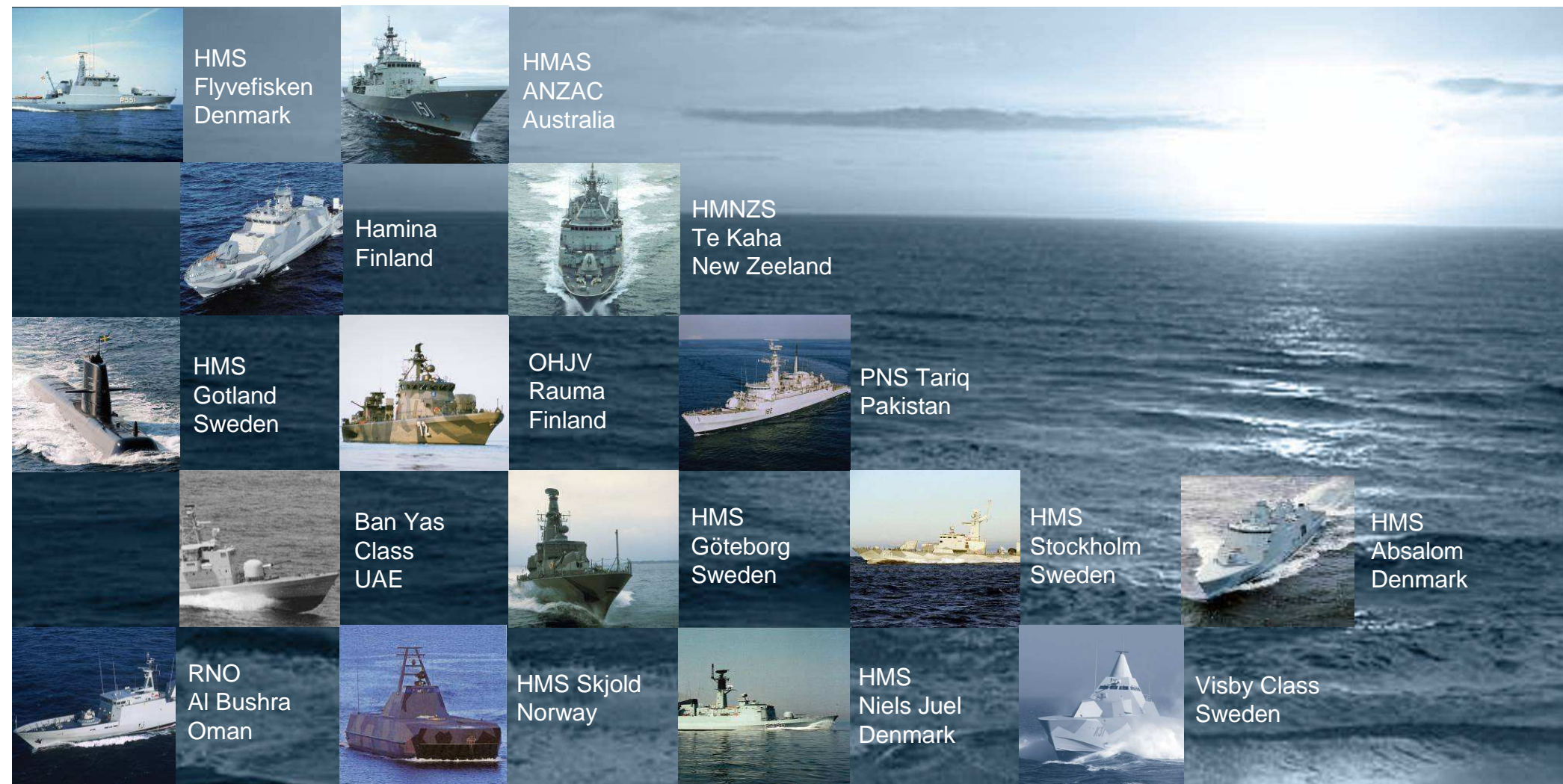


June, 2007  
Thomas Jungfeldt

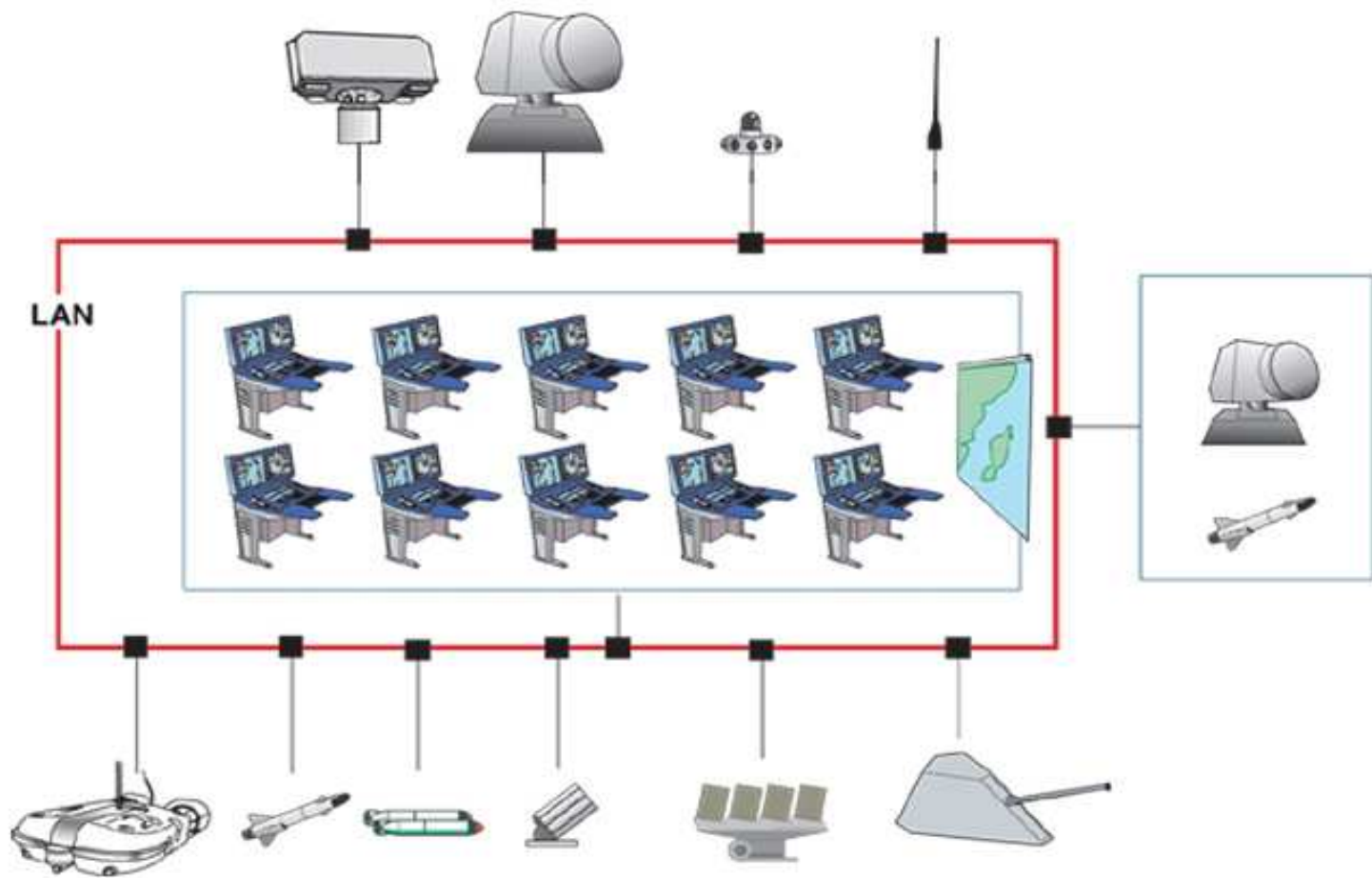
# Saab's capabilities

Support Solutions		Aviation		Sensor Systems
		Electronic Warfare		Simulation and Training
		Weapon Systems		Signature Management
		Command and Control		Communication
		Unmanned Systems		Space

# 9LV Ship projects (Mk3e & Mk4)



# Simple Naval CMS



# Important Customer requirements

- Adaptable to new threats.
- Adaptable to new tasks.
- Affordable.
- Advanced functions.
- Best of breed.



# Architectural goals

- Open Architecture / Open Standards

”Standards that are widely used, consensus based, published and maintained by recognized industry standards organizations” (The Open Systems Joint Force, US DoD).

- Service oriented
- Support modular system design with low coupling
- High degree of runtime configuration
- Rapid integration.
- Reuse across platforms and domains.
- Leverage civil investments.
- Easy to upgrade (functionality and performance)
  - Rapid capability insertion
  - Rapid technology insertion.
- Facilitate cooperation and competition (open market).

# Mk4 approach to OA

- Comply with defined standards
- Apply modular design and low coupling.
- Apply design patterns to isolate areas where standards are missing
  - Use existing SW (Mk3e or COTS) in these areas
  - Await standards
- Define and publish key interfaces using open standards and a formal semantic/syntactic model.
  - To ensure openness / flexibility.
  - To enable use of MDD.
  - To enable runtime configuration / consistency check.
- Apply design patterns with plug-able translators (data brokers) between data models
- Used well established concepts (e.g. j-messages) as a basis for information models

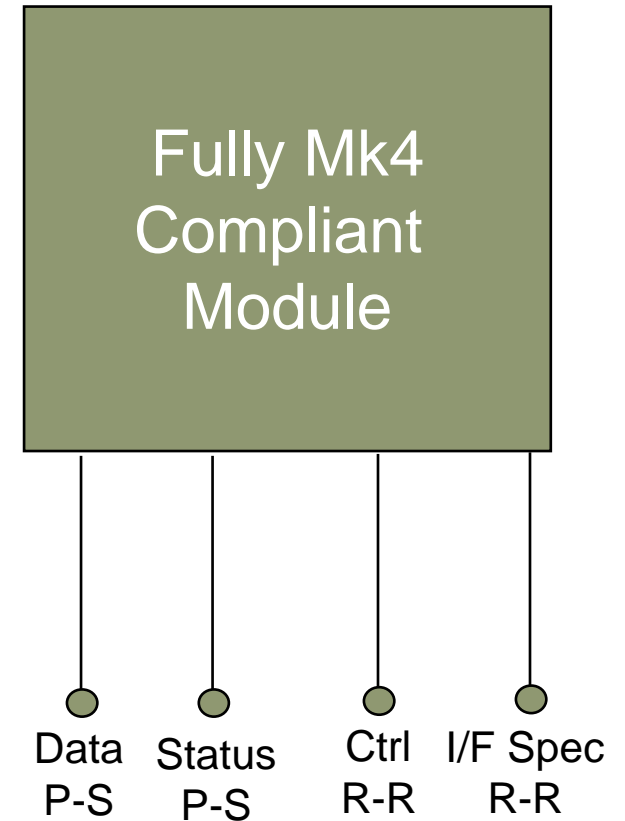
# Interface management

- Key interfaces should be identified.
- Put under control early in the system life cycle.
- Characteristic of key interfaces:
  - Common boundaries between system modules.
  - Provide access to critical data, information, services.
- Key interfaces are defined syntactic and semantically (information model) using standardized formal language (e.g OWL).



# Modular design principles

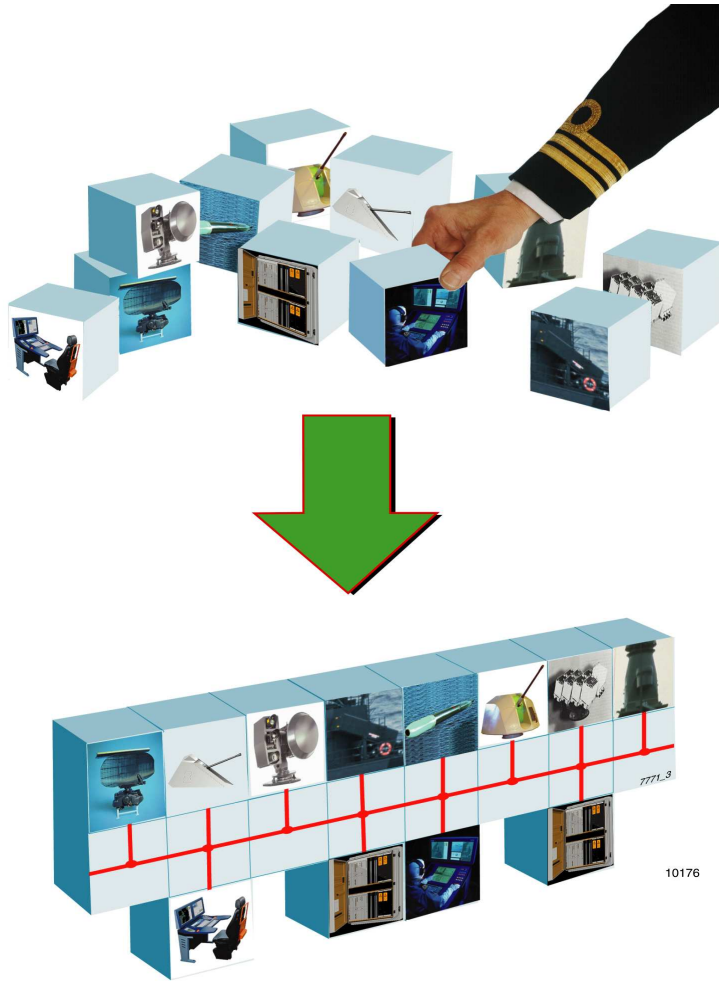
- Self containment
- Cohesiveness
- Encapsulation
- High binding



# Typical Modules

- Sensor (e.g. radar, sonar) (control).
- Director
- Missile (e.g. SSM, SAM) control.
- Platform sensor control
- Gun control.
- C3 functions
- Data link processor (I/f)
- DFE (Data Fusion Engine).
- ADC (Air Defense coordination).
- MTD (Main Tactical Display).
- JMTB (Java MMI Tool Box).
- General Services.
- Etc.

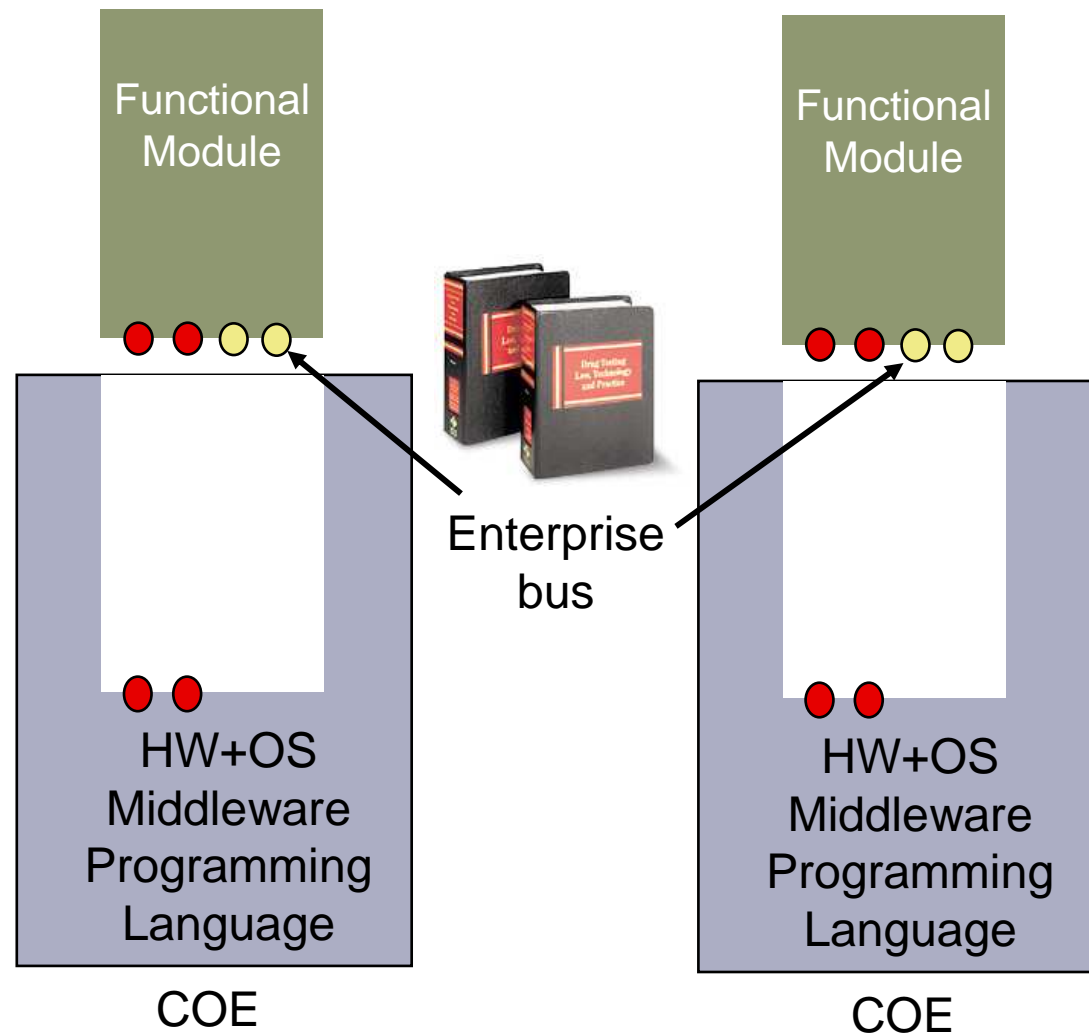
# 9LV Modular System Structure



- Flexibility in configuration
- Low project risk
- Enables incremental system growth

# Basic System building blocks

- COE Common operating Environment.
- Enterprise Bus.
- Functional Modules.



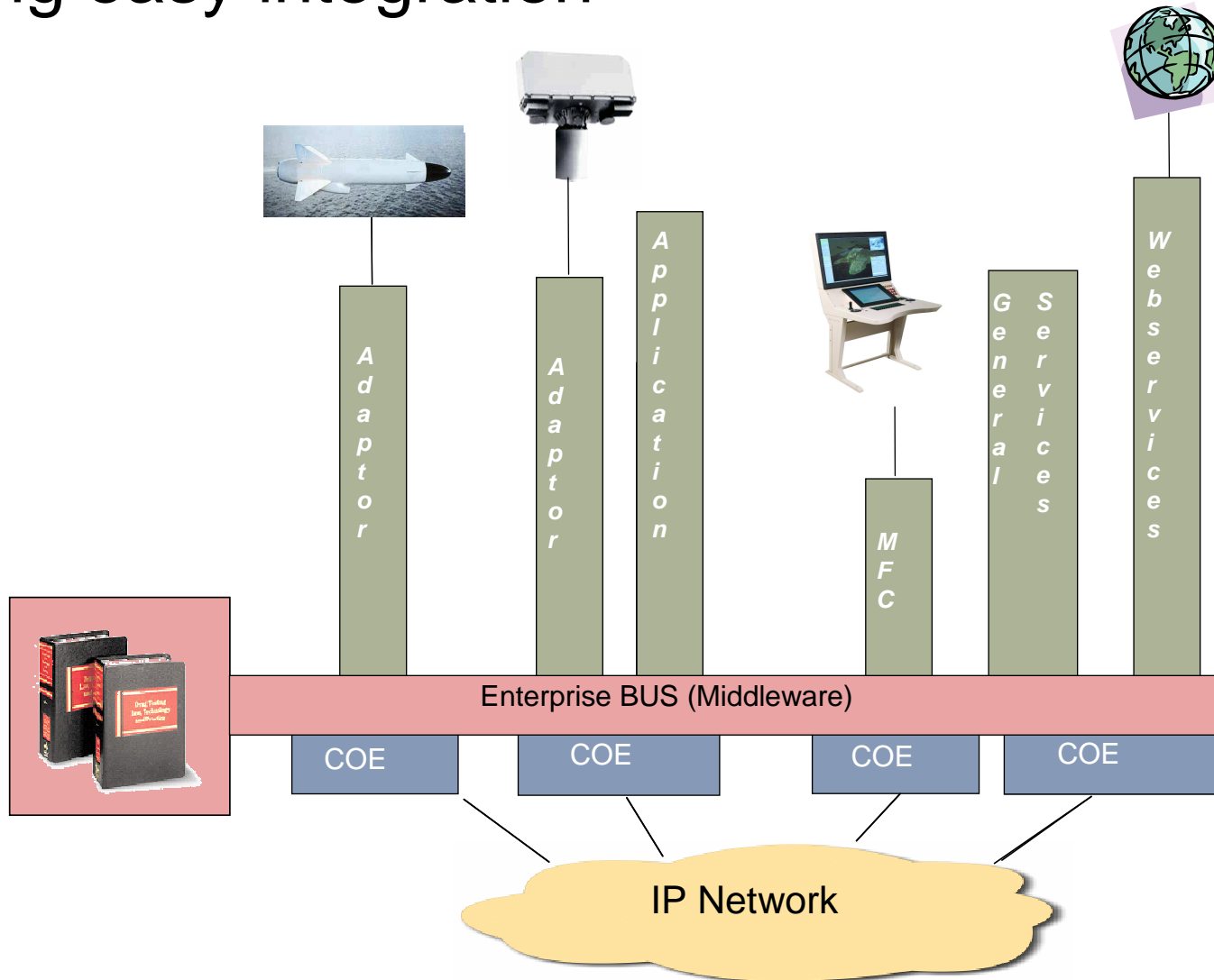
# Common operating environment

- An virtual execution environment for applications.
- Defined by open standards (IEEE, IETF, ANSI).
- Stable over time.
- Enables:
  - Reuse / portability of modules.
  - Heterogeneous systems.
  - Rapid technology insertion.
- Enhance life-cycle supportability.
  - Mitigate the risks associated with technology obsolescence.
  - Mitigate the risk of a single source of supply over the life of a system.

# Enterprise bus

- Provides the information highway that interconnects the modules in the system.
- Built on Publish-Subscribe (OMG DDS).
- All interfaces are defined in standardized formal languages.
- All interface definitions are available online in the system.
- Four distinct classes of interface are supported:
  - Business data (publish-subscribe).
  - Control (request-response).
  - Status / State (publish-subscribe).
  - Interface specifications (request-response).
- Enables NCW and rapid capability insertion

# Enabling easy integration



# Design guidelines

- Use DDS features as much as possible
- Use Content Based Addressing when possible (for group addressing)
- Use DDS Instances to differentiate between object of same class
- Limit number of topics
- Design messages with placeholders for extensions
- Specialize / extend messages though XML part
- Use partitions to isolate subsystems internals
- Use domains to create virtual systems e.g. for mixed mode simulation



# Design guidelines cont.

- No special provisions for HMI
- Avoid complex protocols on top of DDS
- Use stateless interfaces whenever possible
- Design for general case and specialize
- Model request-response on top of DDS
- Start with model and include requirements
- Generate artifacts from model

# Model driven development

A single high level information model is used to generate

- Interface specifications
- Interface code
- XML descriptions of services
- UML models
- IDL
- Test code

# Experiences

## Pros

- Good set of features
- Facilitates low coupling
- Facilitates interface specifications
- Facilitates integration

## Cons

- IDL has limited expressional power for interface contract
- Poor IDL support for versioning and backward compatible extensions
- Null values not supported
- Transition from message to data centric thinking is sometimes hard

?