A Next Generation Avionics Software Architecture

The ECOA® Programme

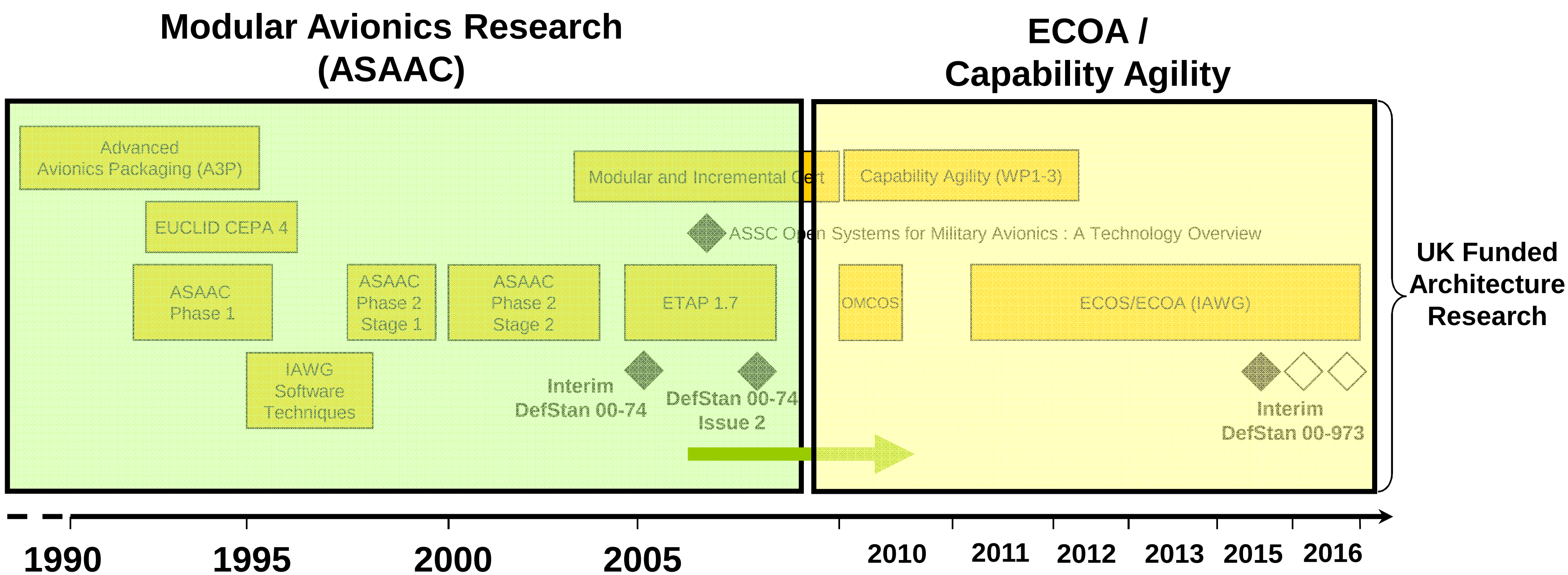
Paul Moxon, BAE Systems MAI

on behalf of the ECOA® team

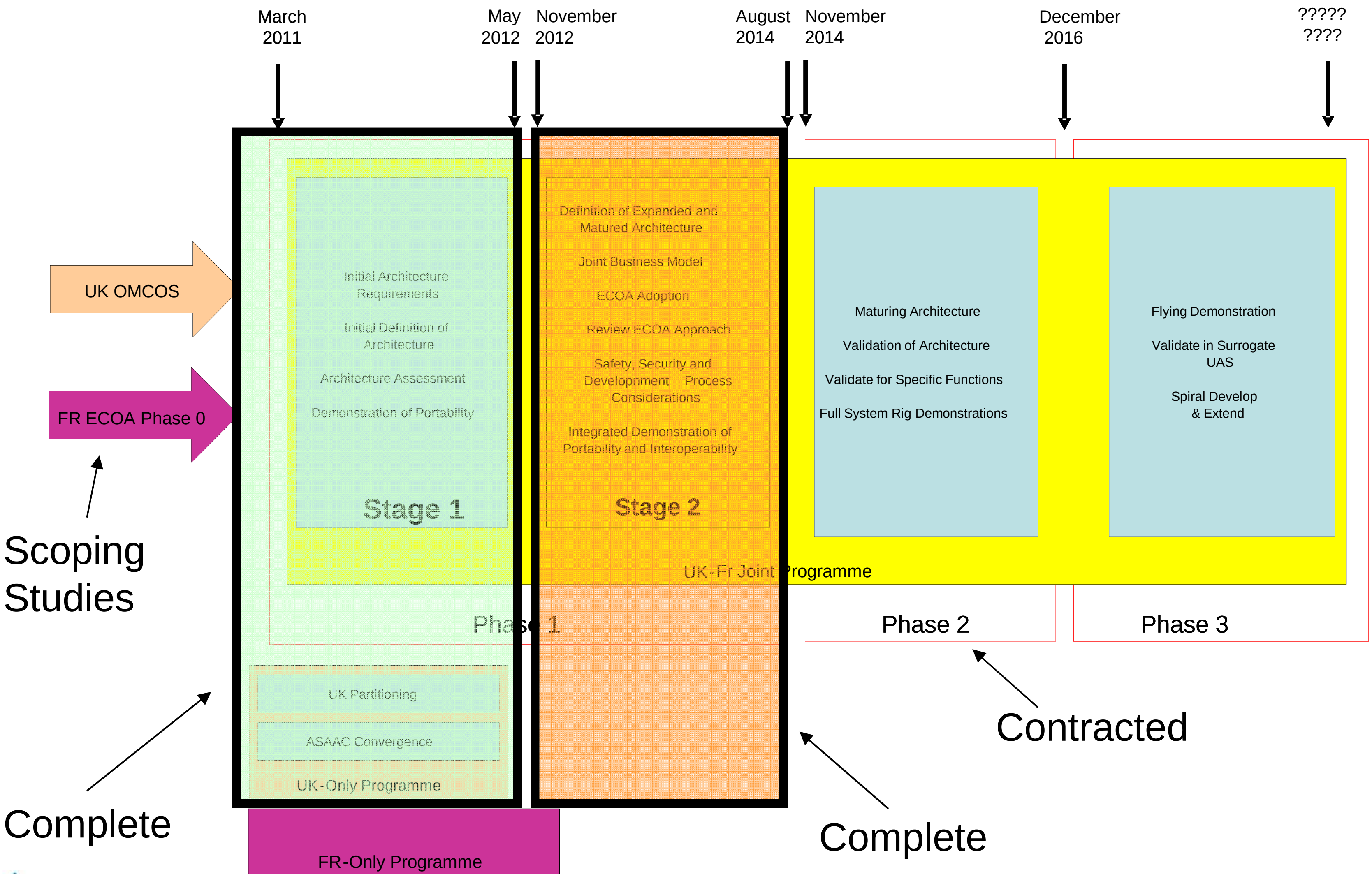
European Component Oriented Architecture

- Joint UK-French research project
  - previously known as ECOS – European Common Operating System
- Funded by the Ministries of Defence of both countries
- Work undertaken by both UK and French defence companies
- In the UK: BAE Systems (MAI + ES), AgustaWestland, Selex ES, General Dynamics UK and GE Aviation
- In France: Dassault Aviation, Thales and Bull
- To reduce the cost and timescales for production and modification of complex, real-time aircraft software systems by facilitating software reuse.
- Production of a Standard based upon Architecture Specification developed within the programme

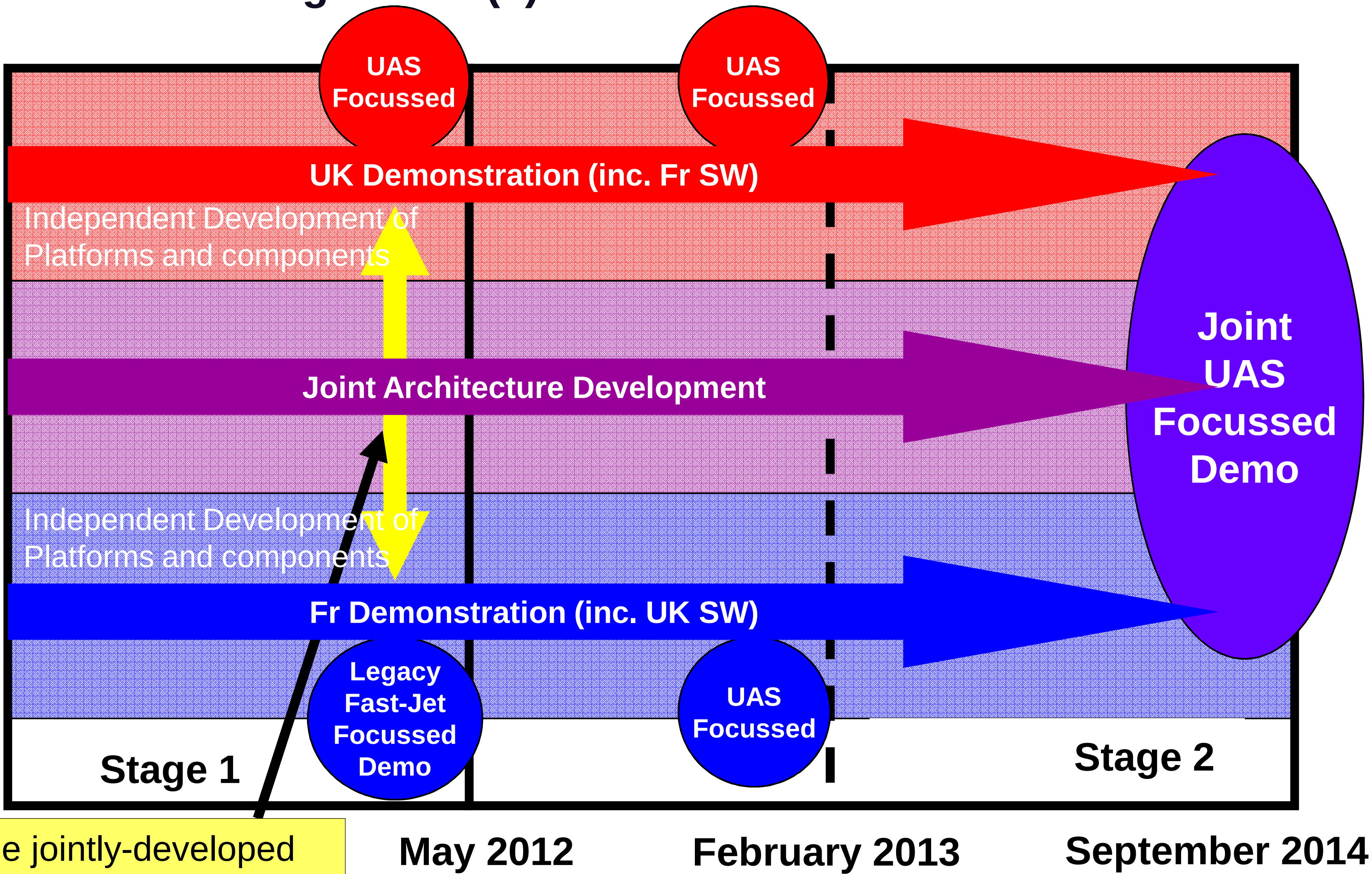
History


Modular Avionics Research
(ASAAC)
ECOA /
Capability Agility
Advanced
Avionics Packaging (A3P)
EUCLID CEPA 4
ASAAC
Phase 1
IAWG
Software
Techniques
ASAAC
Phase 2
Stage 1
ASAAC
Phase 2
Stage 2
ETAP 1.7
Modular and Incremental Cert
Capability Agility (WP1-3)
ASSC Open Systems for Military Avionics : A Technology Overview
OMCOS
ECOS/ECOA (IAWG)
Interim
DefStan 00-74
DefStan 00-74
Issue 2
Interim
DefStan 00-973
UK Funded
Architecture
Research
1990
1995
2000
2005
2010
2011
2012
2013
2015
2016

The ECOA Programme (1)



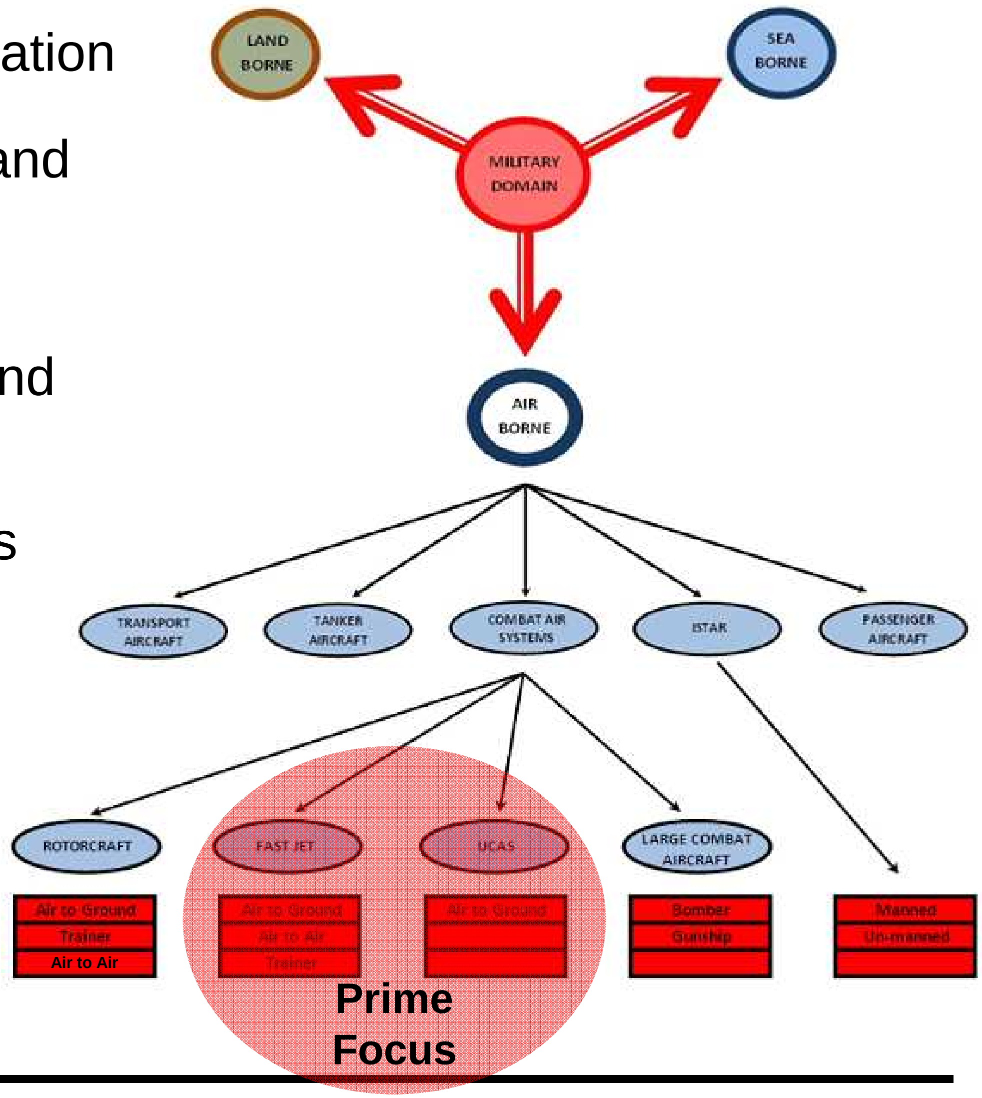
The ECOA Programme (2) – Phase 1



Testing the jointly-developed specification through component exchange between developer and integrator

Scope

- Comprehensive next generation software architecture specification, assessment and demonstration
- Prime focus on combat air mission systems for UAS and fast jet
- Applicable to other domains



Why ECOA is needed (1)

Ongoing

Reduce Through-Life Costs

Protection from obsolescence

Rapid product upgrade

Increasingly complex and large software-intensive systems

\+

Today and Future

Collaborative development

Workshare

Expanded software supplier base (e.g. Information Systems providers, SMEs)

Networked Systems of Systems

Enable Development (New build and upgrade)

ECOA

Why ECOA is needed (2)

1. **Reduce risk** in the integration of complex mission systems through enabling collaborative development

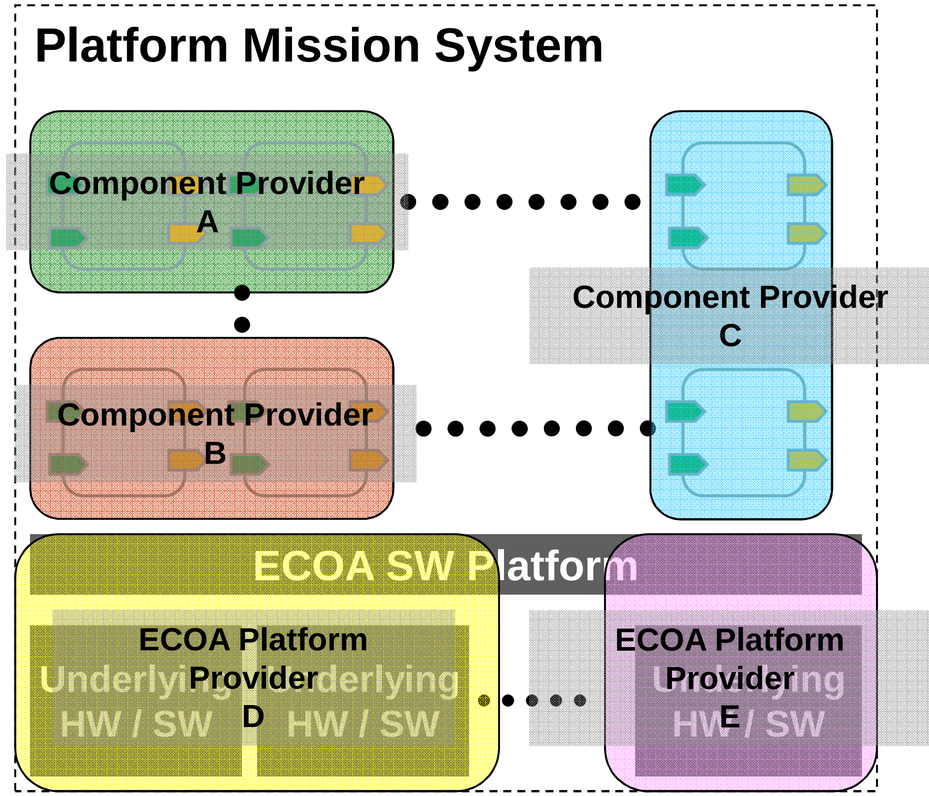
2. **Reduce development and through life-costs** of a collaborative development programme

3. **Reduce cost** through a business model of software reuse

4. **Improve competition** through broadening the software supplier base
   - Foster innovation through non-traditional suppliers
   - Create a structured market for avionics applications

5. Enable the development of **complex, networked systems**

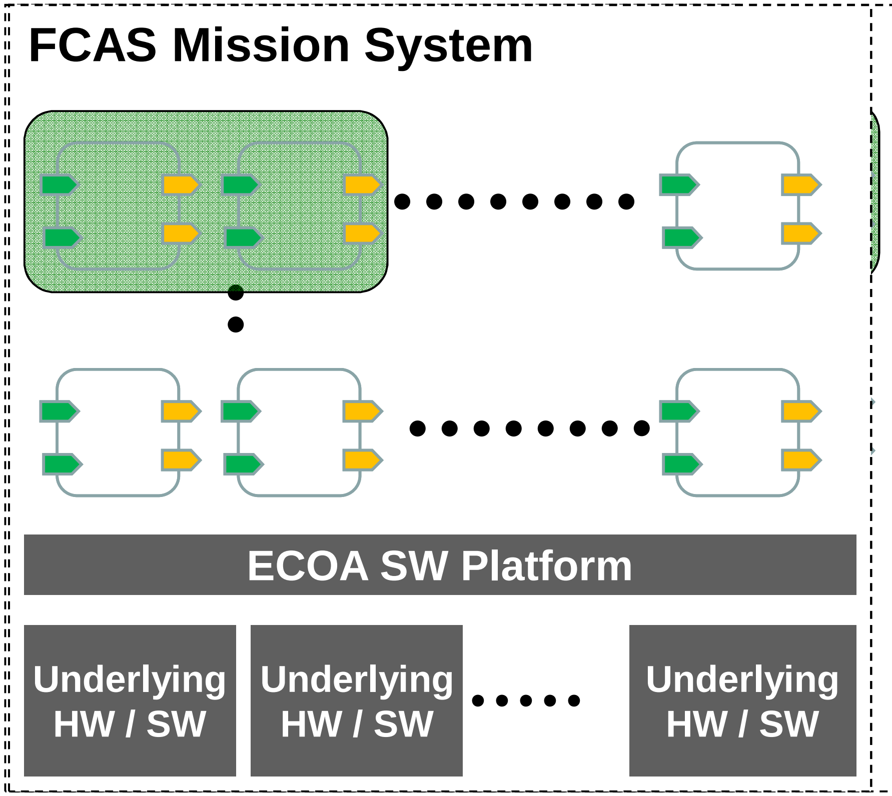
System development enabled by ECOA (1)



- HW/SW platform workshare decoupled from application HW/SW workshare
- Applications can be located on available hardware optimising size/weight/power
- System management built-in so co-ordinated fault and error management is possible

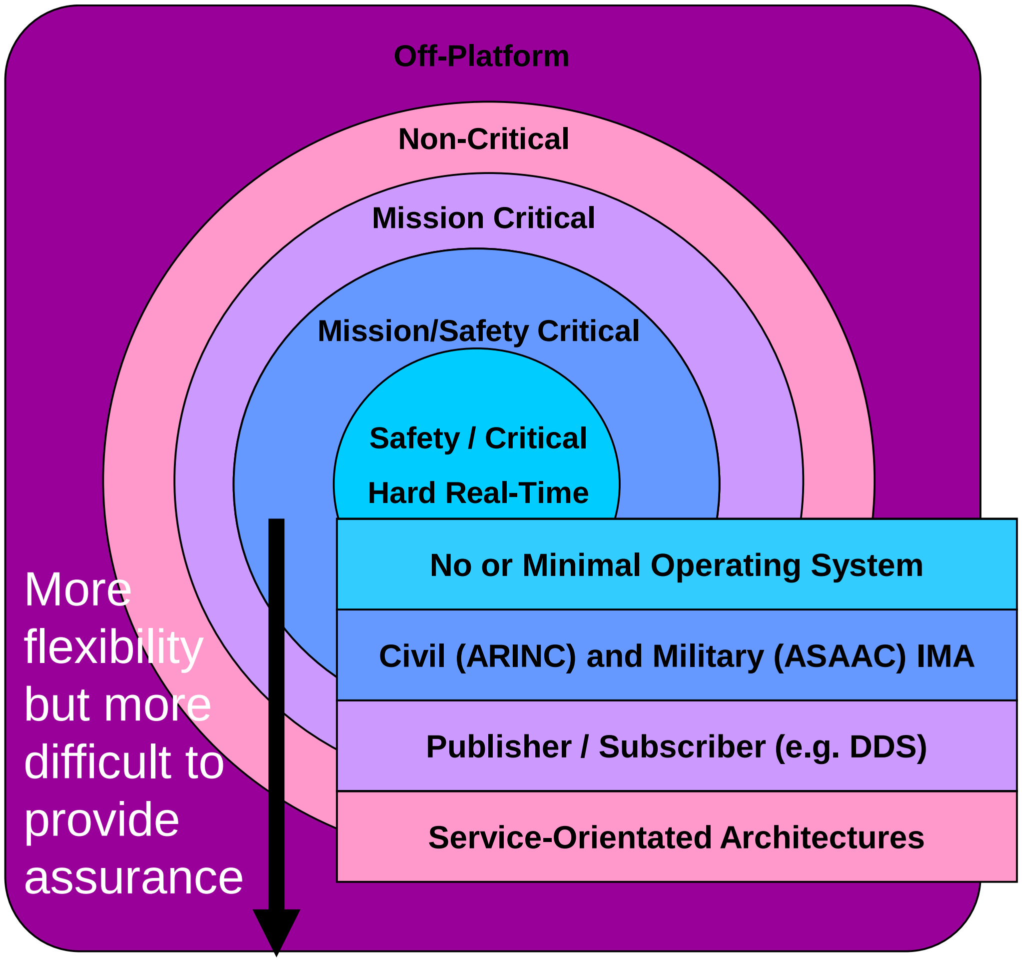
Workshare allocation is aligned with supplier capability e.g. HW/SW platform supplier, functional expertise

System development enabled by ECOA (2)



- Applications are loosely-coupled. This promotes reuse in different systems and contexts
  - Different platform
  - Platform Training system
  - In a ground-station as well as the air-vehicle

ECOA Flexibility



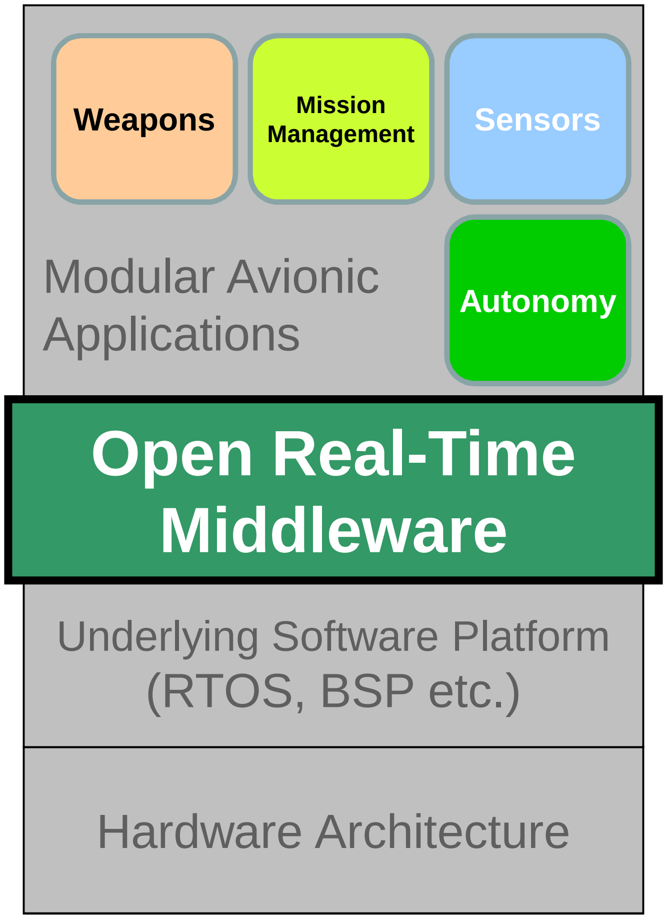
- ❑ OMCOS* found that loosely coupled Service-Orientated Architectures (SOA) and Publisher-Subscriber are important concepts that can provide the flexibility needed in increasingly complex, ad-hoc, networked systems
- ❑ Important for the UK to build on existing IMA concepts
- ❑ May be more difficult to provide high assurance for such flexible systems

ECOA Focus

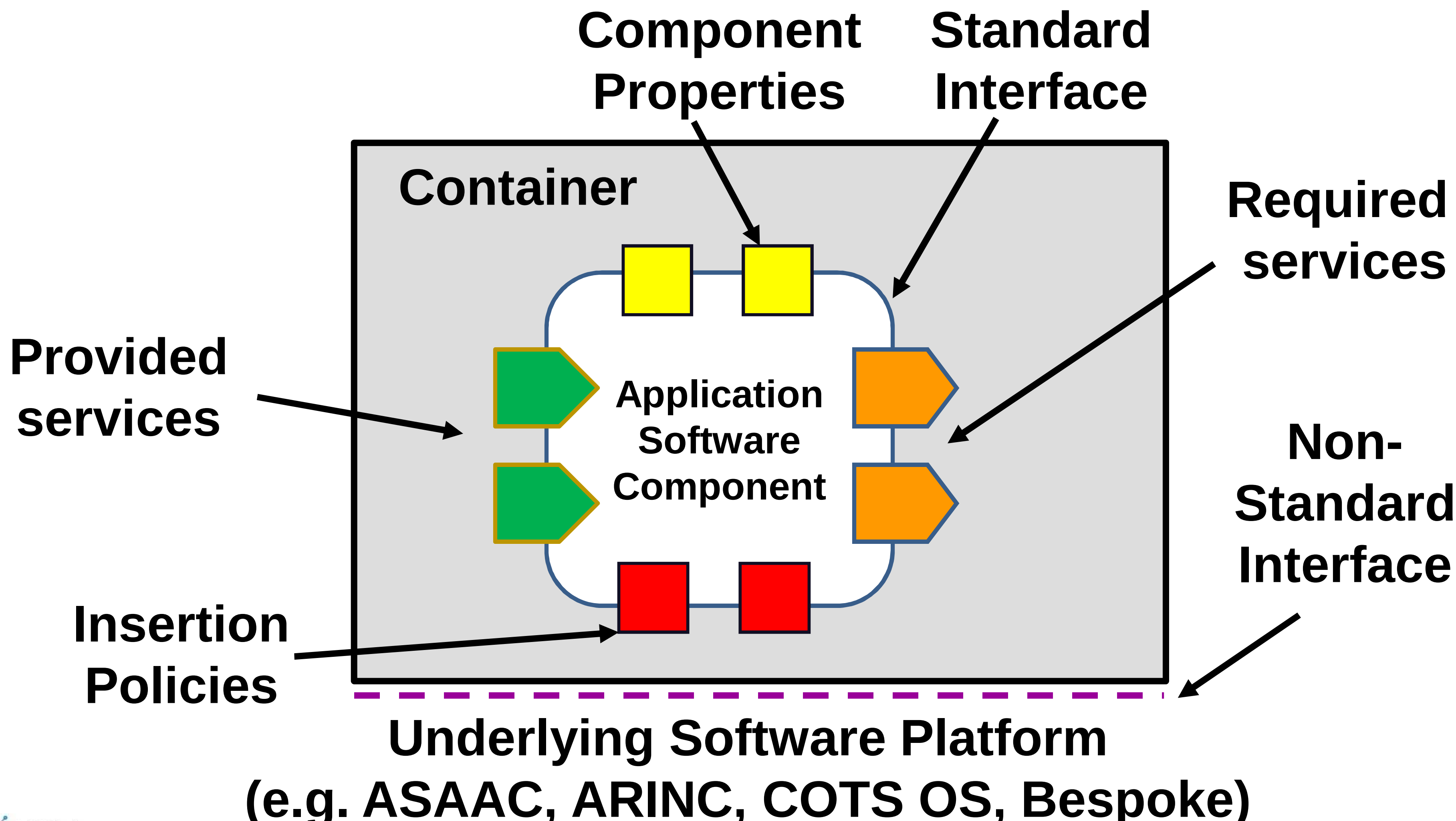
* Open Modular Common Operating System

An Open Real-Time Middleware

- Using Information Systems technology
- Component-Based Software Engineering (CBSE) and Service-Oriented Architectures (SOA)
- Supports Model-Driven Engineering (MDE)
- Allows use of code generation technology
- **<u>Not</u>** a traditional Operating System
- **<u>Not</u>** a replacement (or competitor to) COTS operating systems and platform software
- **<u>Not</u>** limited in usage to mission systems

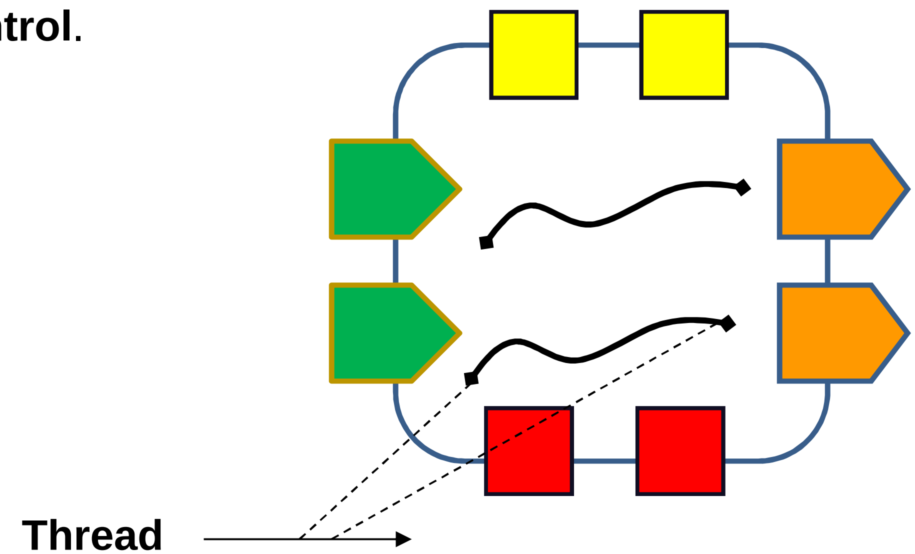


Components, Containers and Services (1)



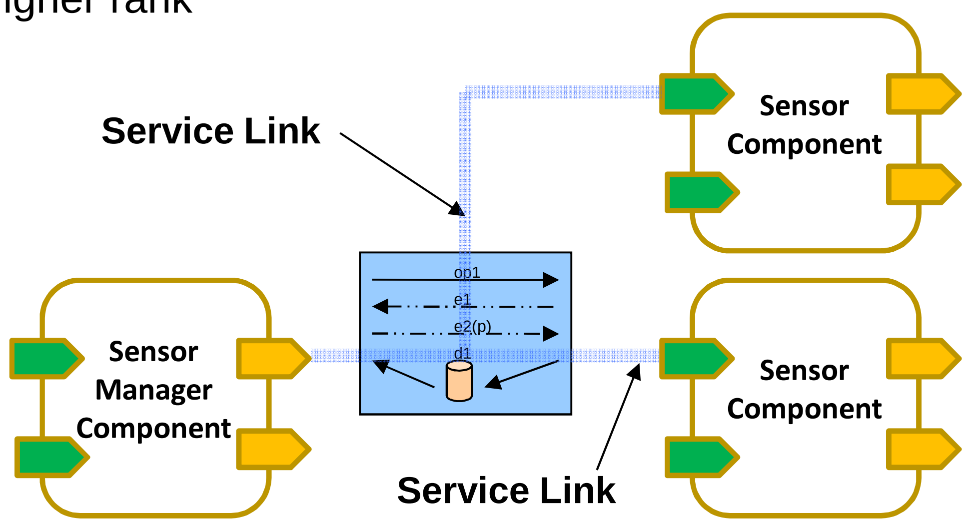
Containers, Components and Services (2)

- Components are **portable**
- Containers provide the glue code which interfaces to the underlying software platform
- Containers can be automatically generated to HW/SW platform of choice (e.g. ASAAC, ARINC)
- Component (application) software is executed from the container
  - Known as **Inversion of Control**.
  - Improves portability



Services, Links and Operations

- Services are defined as collections of related operations that are of 3 types:
  - Events (ie. Message passing)
  - Request-Response (akin to remote procedure call)
  - Versioned Data (simple publish/subscribe)
- A Service Link defines a connection between a service on one component with a service of another component
  - Service Links have a defined rank
  - Lower numerical value = Higher rank



Data Types

- All operations can carry typed data
  - e.g. An event carrying data
- ECOA has a set of predefined data-types
  - e.g. boolean8, uint16, float32
- Complex user defined data-types can be created using predefined ECOA types
  - e.g. A list of waypoints
- The service definition associates the data type to an operation's parameters

Discovery of Services

- The concept defined for dynamic discovery of services in ECOA enables dynamic connection of clients and servers within a given set of possibilities.
  - All possibilities (known possible service links, requirers and providers) are defined at design time and represented by the assembly schema.
  - Once an assembly schema is in place, no new potential services or providers are able to be discovered (and no new potential requirers may appear).
  - Within the set of allowed connections the infrastructure connects components based upon the availability of their services and the ranking of different connections.
- In addition to this and to keep the client side simple, the client is not allowed to dynamically define its required QoS
  - All servers shall provide required services(s) with compatible QoS.
  - The client is free to use a service if it is available, but is unaware which server is providing it
- Both concepts enable early verification of the system by defining the superset of all possibilities at assembly schema level.
- In future it may be possible to have different configurations, represented by different assembly schemas

Service Availability

- Within an assembly schema the availability of any particular service is declared by the providing component through a dedicated container API for the supervision module.

- Components that require services are notified of a change of availability (or provider if multiple ones exist) through another dedicated module API for the supervision module.

- This allows a level of 'dynamic' behaviour within the system and enables automatic selection of a service provider (based upon the rank assigned to the Service Link)
  - If a client is connected through one single required service to several servers via several links, the container selects the server based on the lowest rank value and availability of provided services.
  - If the provided service or server become unavailable, the infrastructure switches the service link to another server if one is available
    - At each switch, the client is aware of the change of provider and may decide to continue to use the required service or not.
    - If another server is not available then the client is informed that the required service is no longer available

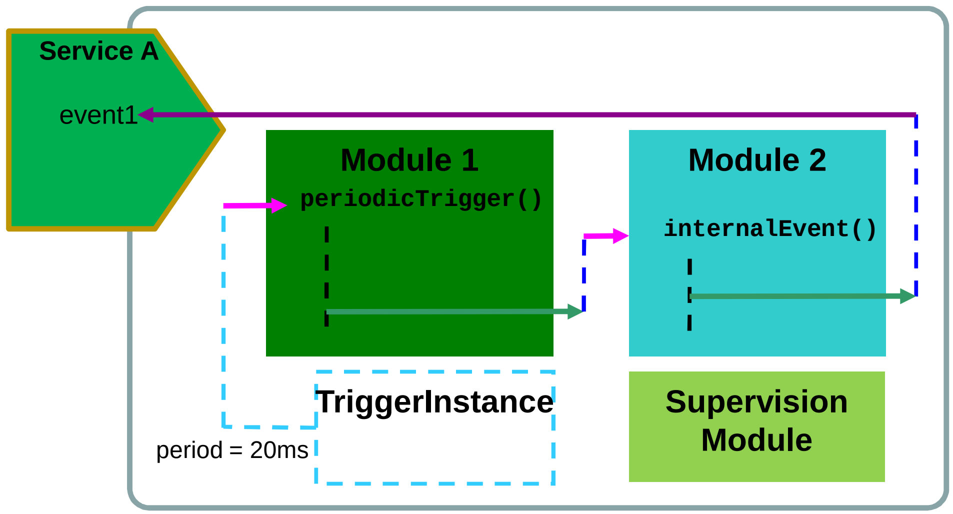
Component Implementation and Modules

- Components are implemented using modules which communicate via the same mechanisms as Services (events, request-response, versioned-data)

- A module's operations are guaranteed to be executed non-concurrently (hence no synchronisation issues)

- Modules allow multi-threading within a component

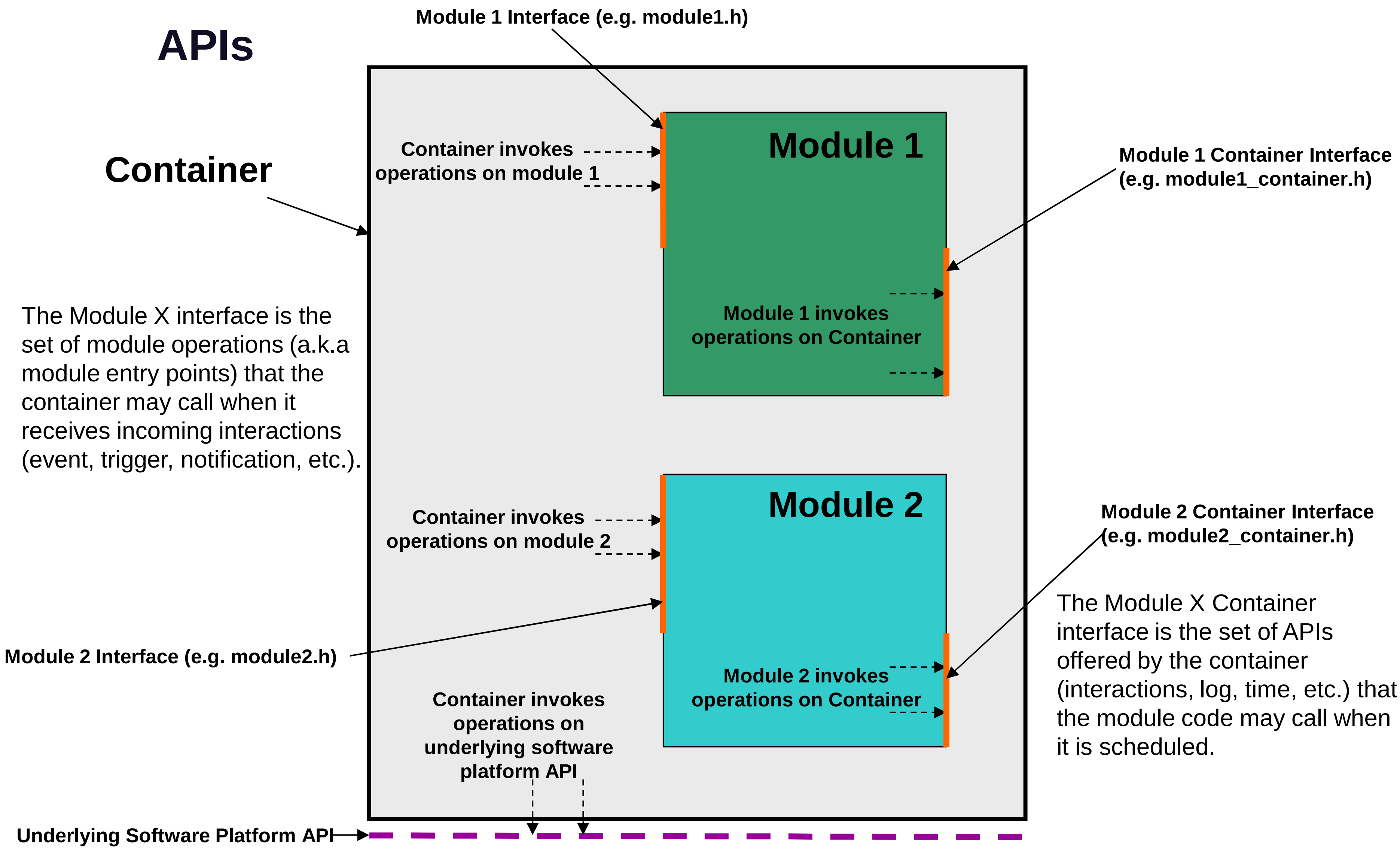
- ECOA makes use of 3 special module types for controlling execution within a component:
  - Supervision Module
  - Trigger Instance
  - Dynamic Trigger Instance



Modules

- Normal Functional Module
  - Used to implement component functionality
  - Code is supplied by application developer
- Supervision Module
  - Has additional APIs to manage other module types
  - Code is supplied by application developer, but may be initially generated from a template
- Trigger Instance/Dynamic Trigger Instance
  - A ‘virtual’ module that can be specified within a component implementation, but instantiated by the ECOA platform
  - Can be periodic (Trigger Instance), with a fixed period
  - Can be sporadic/aperiodic/single shot (Dynamic Trigger Instance), controlled by the module associated with it

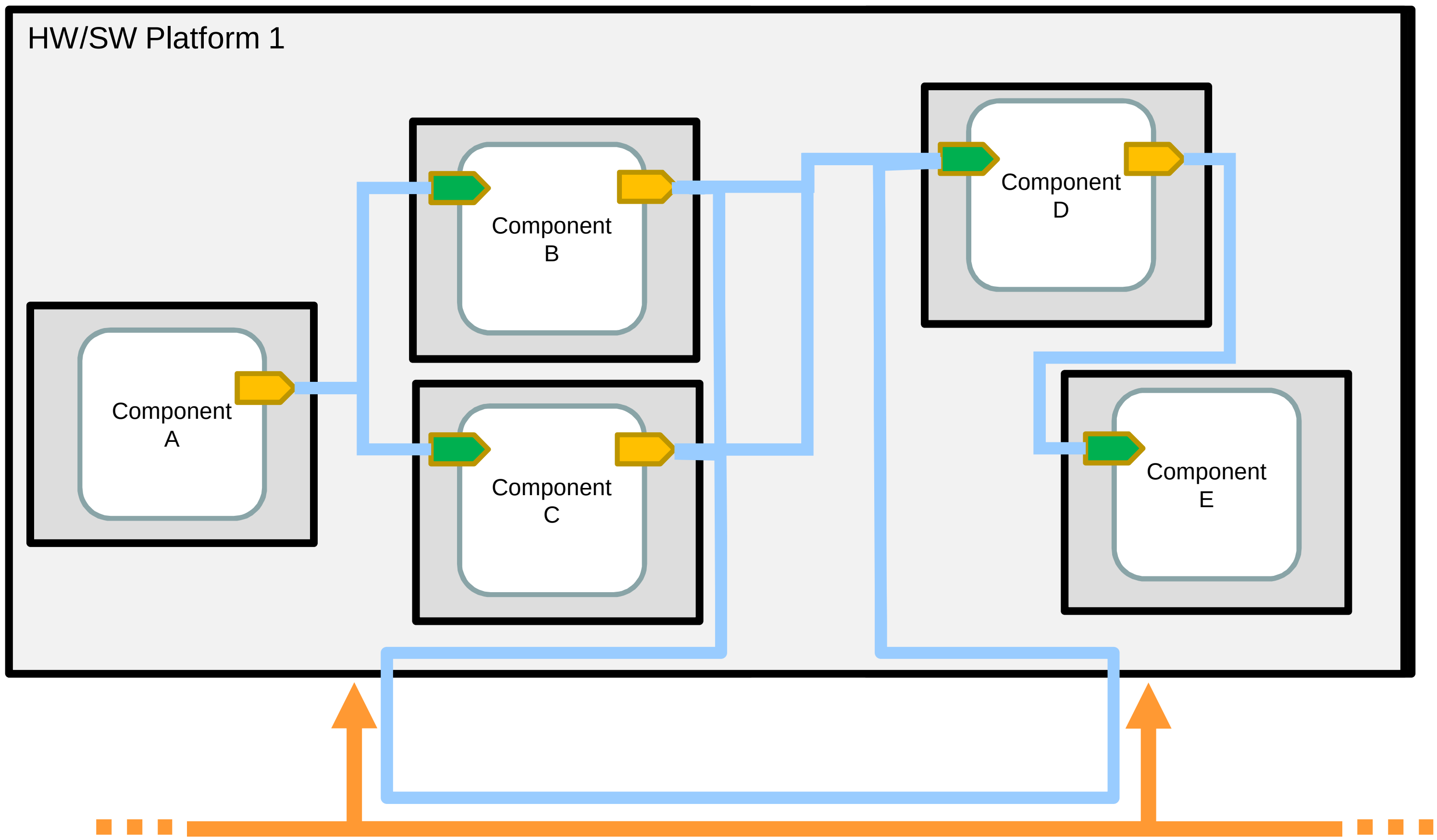
APIs



The Module X interface is the set of module operations (a.k.a module entry points) that the container may call when it receives incoming interactions (event, trigger, notification, etc.).

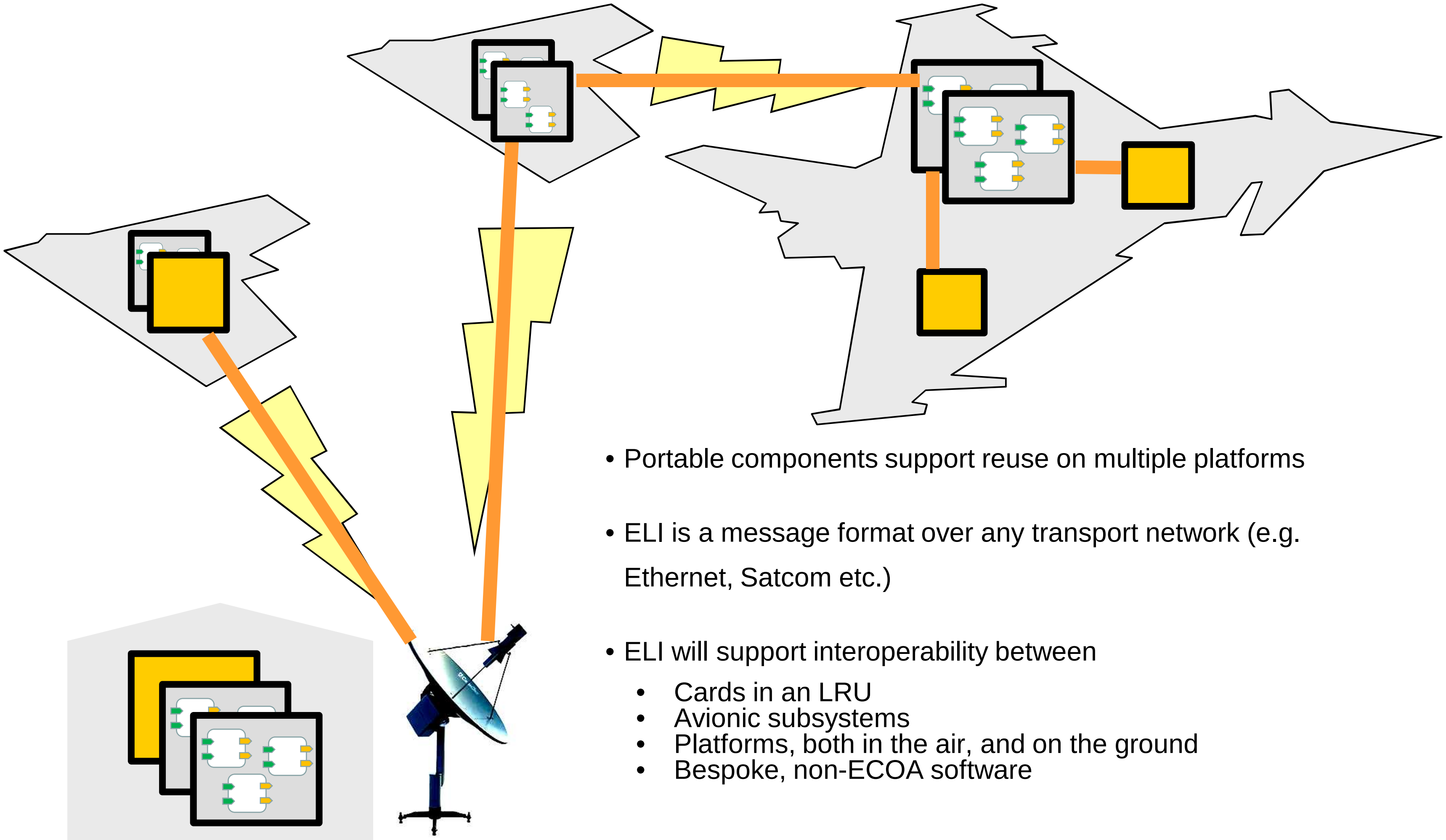
The Module X Container interface is the set of APIs offered by the container (interactions, log, time, etc.) that the module code may call when it is scheduled.

Building Systems out of Components



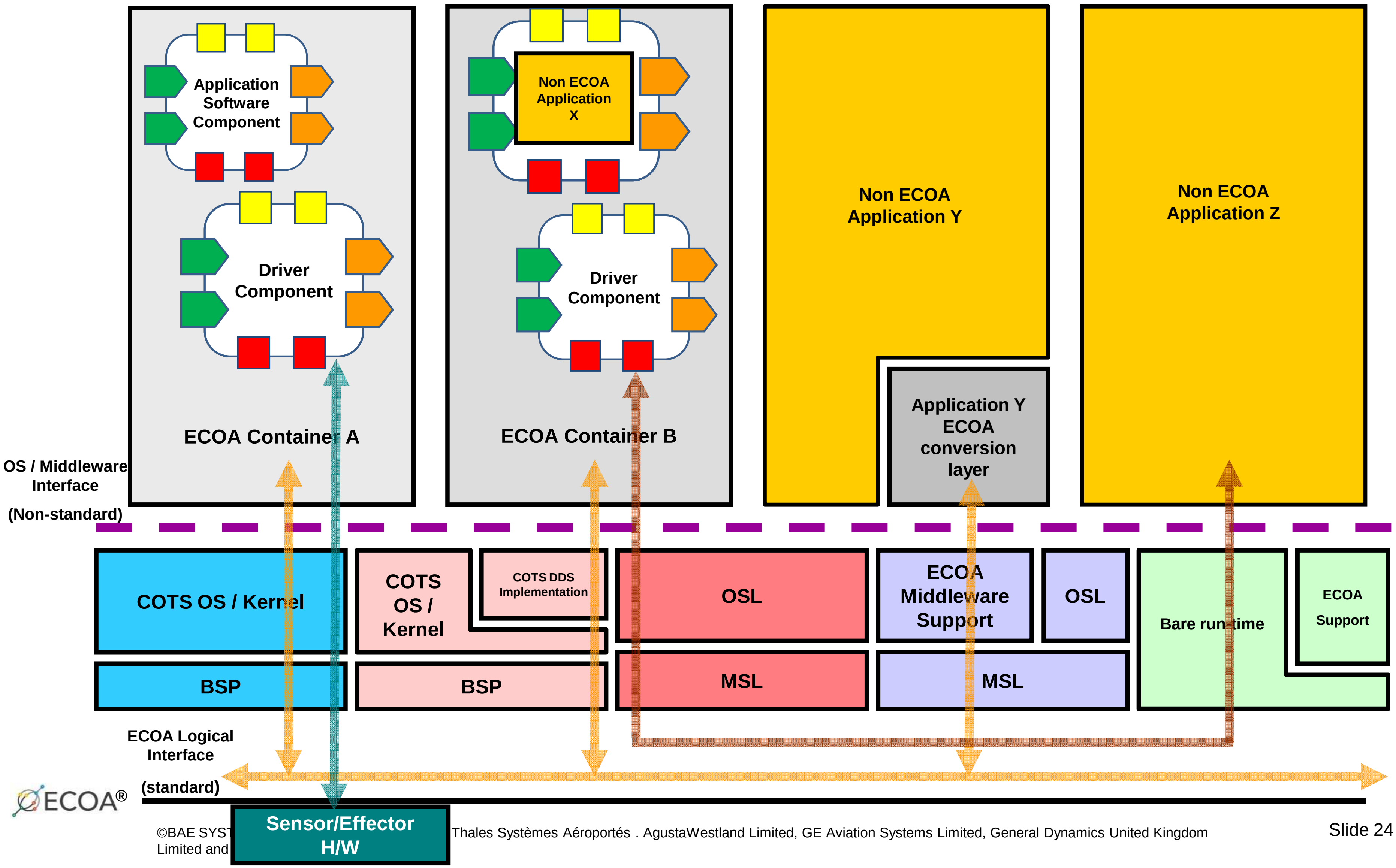
Interoperability - ECOA Logical Interface

Deployment and Interoperability



- Portable components support reuse on multiple platforms
- ELI is a message format over any transport network (e.g. Ethernet, Satcom etc.)
- ELI will support interoperability between
  - Cards in an LRU
  - Avionic subsystems
  - Platforms, both in the air, and on the ground
  - Bespoke, non-ECOA software

Integration with Legacy and Non ECOA Software

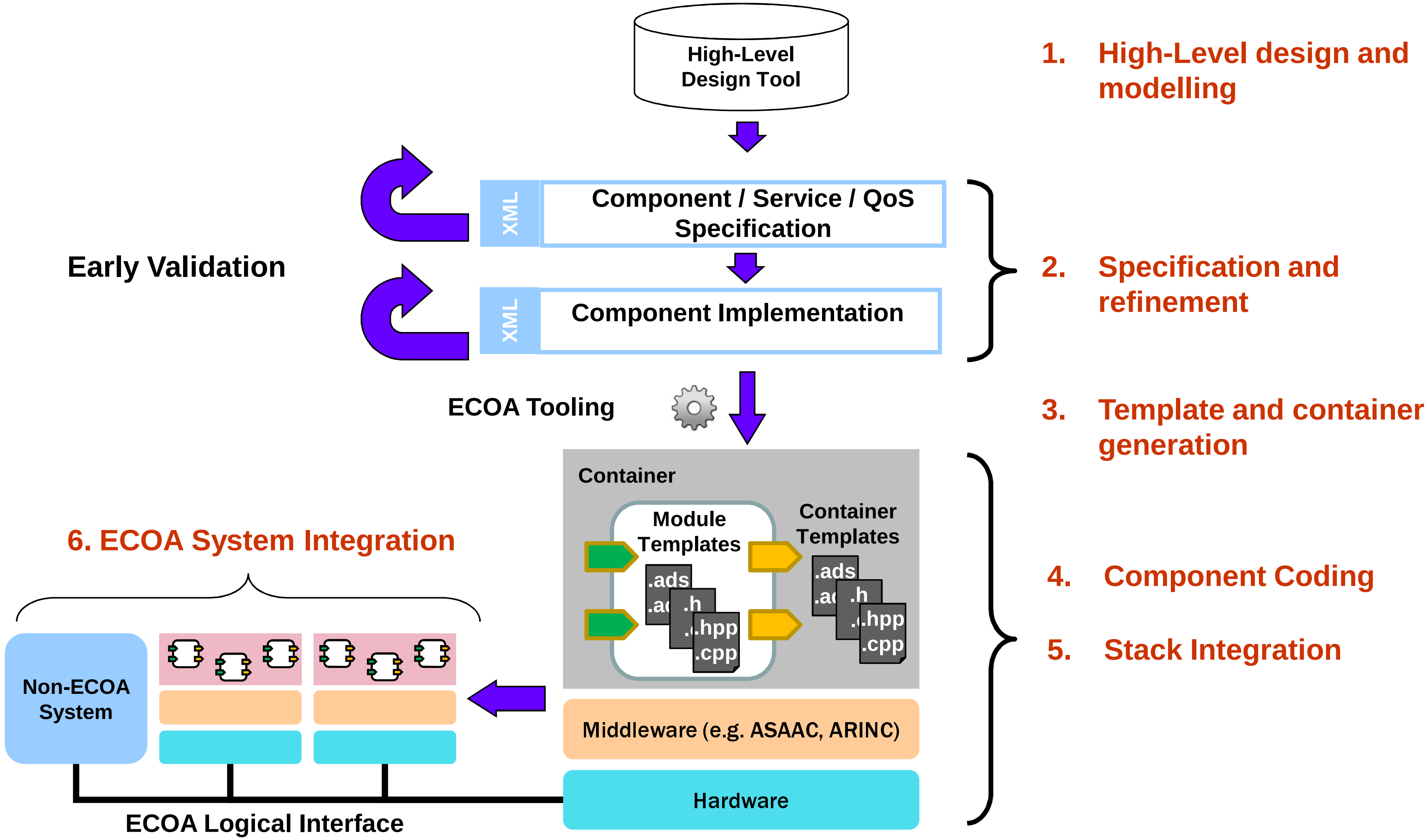


XML Metamodel and Bindings

- Standard XML data model (defined in the architecture specification) defines all aspects of the ECOA software architecture
  - Data Types
  - Services
  - Components and Modules
  - Quality-of-Service
  - Deployment of Software
- Standard programming language bindings
  - Defined in the specification
  - Currently support C/C++/Ada

It is this preciseness that promotes portability and enables the auto-generation of container code.

Partial Development Process



ECOA Architecture Specification and Standards

- Public release Architecture Specification documents available:
  - www.ecoa.technology
- Interim Def Stan® 00-973 “European Component Oriented Architecture (ECOA®) Collaboration Programme” is now available
- BNAE (Bureau de normalisation de l'aéronautique et de l'espace) standardization process still ongoing