Comprehensive Architecture Strategy (CAS)

Version 4.0

Preparing Organization: US Army Combat Capabilities Development Command (CCDC)
Aviation and Missile Center (AvMC)
BLDG 5400, Redstone Arsenal, AL, 35808

Comprehensive Architecture Strategy (CAS)

Abstract

This document provides an introduction to a Comprehensive Architecture Strategy (CAS) for mission systems. The strategy integrates business and technical aspects to support the efficient development and sustainment of mission systems that meet business needs along with functional and performance needs. A managed architecture ensures components and systems are aligned to overarching business and technical objectives. This document also provides a common lexicon necessary to the understanding of the approach, and addresses, at a high level, the ideas necessary to shift current processes to support this strategy as well as guidance on how to document and apply those ideas.

Revision History

Date	Revision Description	Revision Number
12 February 2018	Reference Architecture section	v 1.0
9 April 2018	Add Objective Architecture section; restructuring and significant editing of existing sections	v 2.0
15 June 2018	Add separate section for element descriptions, Add System Architecture section	v 3.0
31 Aug 2018	Added references to work cited	v 4.0

Table of Contents

1. Executive Summary

The complexity and cost of mission systems are increasing at an exponential rate [Dvorak 2009]. These increases are due in large part to the proliferation of software-intensive capabilities and increasing demand for faster systems integration and superior performance while meeting stringent safety and security requirements [Tate 2017]. To a large extent, the Military Services develop military systems through independent procurements using a unique set of program requirements implemented by a single vendor. The architectures for these platforms are often a byproduct of system design. The system design activities often result in ad-hoc architectures that are the result of technical design decisions. A better approach is for the system design to follow an architecture deliberately created to meet the business concerns of the acquiring organization [GAO 2011]. The architecture ensures the business needs of the organization are met by placing guidance and constraints on the design. This improved approach allows an organization to procure systems that meet business and technical characteristics such as affordability, interoperability, robustness and consistency.

While the benefits of Open Architecture (OA) and mandates for its use have been discussed for decades [DAG; DODD 5000.1], experience has shown that singularly focused architectural approaches and broad mandates to "do Modular Open Systems Approach (MOSA)" have not provided the expected improvements in affordability, program schedules, and warfighting capability [Baldwin 2016; GAO 2013; Moore 2016; Patel 2013]. OA concepts require a more comprehensive, strategic approach to yield the desired benefit across multiple military systems and domains. Architectures and their development processes must be deliberately managed and strategically focused on the business and technical objectives of an organization [Muller 2007].

This document introduces a strategic comprehensive architecture approach based on the analysis and documentation of the Key Business Drivers (KBDs) and architectural drivers supporting the development of military systems that intentionally meet the business drivers of the military organization. These drivers form the requirements that support an organization's business and technical objectives for a predefined range of programs, systems or components. If properly managed, this approach will enable competition and reuse of products designed to a specification that complies with enterprise or family of system architecture constraints. The approach will help avoid some of the long term challenges of vendor lock, while supporting business drivers such as innovation and competition [Kendall 2014].

The content of this document is based on recommendations and best practices for architecture documentation gleaned from a vast body of well-known technical resources including:

- ANSI/IEEE Recommended Practice for Architectural Description for Software-Intensive Systems
- DoD Defense Acquisition Guidebook
- DoD Reference Architecture Description
- DoD System Engineering Guide for System of Systems
- Future Airborne Capability Environment (FACE) Technical Standard
- ISO/IEC/IEEE Systems and Software Engineering guidance
- Joint Common Architecture (JCA)

- SEI's Architecture Tradeoff Analysis Method (ATAM)
- Software Engineering Institute (SEI) Software Engineering and Product Line Framework materials
- The Guide to the Systems Engineering Body of Knowledge (SEBoK), v. 1.9

The CAS incorporates "best of breed" and lessons learned from these sources, supplemented by contributions from Subject Matter Experts who have engaged in in Software Architecture, Hardware Architecture, Functional Architecture, Data Architecture, Systems Engineering, Product Line Practices and similar architecture efforts. Specific references are inserted as appropriate and will be further matured and expanded as additional sections are added to this paper.

2. Introduction

To a large extent, the Military Services develop military systems through independent procurements, each based on using a unique set of requirements implemented by an Original Equipment Manufacturer (OEM)/Lead System Integrator (LSI), and their particular suppliers [DoD 5000.1]. Often, non-functional requirements such as interoperability, upgradeability and maintainability are not included as normative requirements in typical requirements documents [Kazman 2000]. This oversight results in the non-functional requirements only being achieved in an uncoordinated manner which negatively impacts the cost to operate, maintain and upgrade military systems from an enterprise perspective. This method of independent procurement has undesired side effects including long lead times, cumbersome improvement processes, limited hardware reuse, and a lack of software reuse, and often results in platform-unique designs that are costly to sustain and modernize [Kendall 2014]. The lack of managed architectures among our military systems results in organizations failing to achieve key qualities such as interoperability, while increasing development and integration costs and development cycle times [GAO 2011], Combined with the extensive testing and airworthy qualification requirements, these factors have begun to affect the ability of the military community to deploy new capabilities in a timely manner to the Warfighter [Baldwin 2016].

Every existing mission system has an architecture and every new mission system needs one [IEEE 2000]. Historically, the Government's focus has been on specifying Government Furnished Equipment (GFE), providing the platform's performance specification and leading requirements management (through design reviews such as SRR, PDR, CDR, FQT, etc.) of a mission system design [DoD 2008; SEBoK]. Integrators traditionally start with system performance requirements and develop methods to integrate components and subsystems (often defined in ways to maximize reuse of content that integrator controls or is familiar with) to achieve these requirements resulting in a system design not constrained by customer business and technical objectives [INCOSE 2015, ISO/IEC/IEEE 2015, ISO/IEC 2007]. This results in platform architectures that are an outcome of the system design process rather than deliberate artifacts to guide system development.

The Government has been slow to understand that architecture must be more than a by-product of the system design. The architecture plays a key role in achieving coordinated and consistent life-cycle characteristics (e.g. interoperability, scalability, portability, maintainability, modifiability, etc.) [Kazman 2007]. Architecture must be based on strategic decisions and provide tactics, patterns and methods that guide and inform design solutions at all levels of the system (sub-system, component, module, data,

etc.) to meet technical and business requirements [Bachman 2007]. Architecture is not a single approach or standard, it is a series of architectural decisions. The definition of KBDs and Key Architecture Drivers (KADs) drives the decision making process. The creation of an architecture that provides the tactics, patterns and methods to meet enterprise KBDs and KADs is one way to achieve the desired life-cycle characteristics [Bass 1999].

An enterprise architecture can provide a reference for preparing and performing a common baseline assessment of existing mission system capabilities and requirements across programs [GAO 2011]. The enterprise architecture provides a method for focusing all architecture and design decisions on meeting enterprise KBDs and KADs. The use of an enterprise architecture can enforce common, applicable standards and specifications and reduces the likelihood that a platform will develop a unique and difficult-to-support solution. The enterprise architecture constrains the set of possible systems that can meet the mission system requirements along with providing the architectural structure to be tailored for various families of systems [Muller 2007].

An architecture description is a strategic product describing the components and their relationships as well as how the architecture serves the business goals of the organization benefitting from the resulting system. [ANSI/IEEE 2000, Bergey 2005; ISO/IEC/IEEE 2011] Ideally, an architecture should be organization dependent, due to potentially conflicting business objectives, and flowed down from a governing agent. Since many organizations have an “enterprise” and “operations” structure, an architecture strategy should reflect, or complement the structure [GAO 2011]. Each level of the organization produces or procures several different products grouped into similar families.

Figure 1 illustrates how each organization has separate responsibilities and manages the appropriate level of the architecture to meet the business needs of the organization. The three architecture levels depicted in the graphic are described later in this paper. Documenting the relationship of the organization’s within each layer of the architecture, and between the architecture layers, will identify areas of commonality that can be realized across an enterprise as well as within the family of systems. Realization of the organization’s expected benefits from the architectural description can only occur when each layer is accompanied by the proper level of management required to meet the technical and operational needs of the organization.

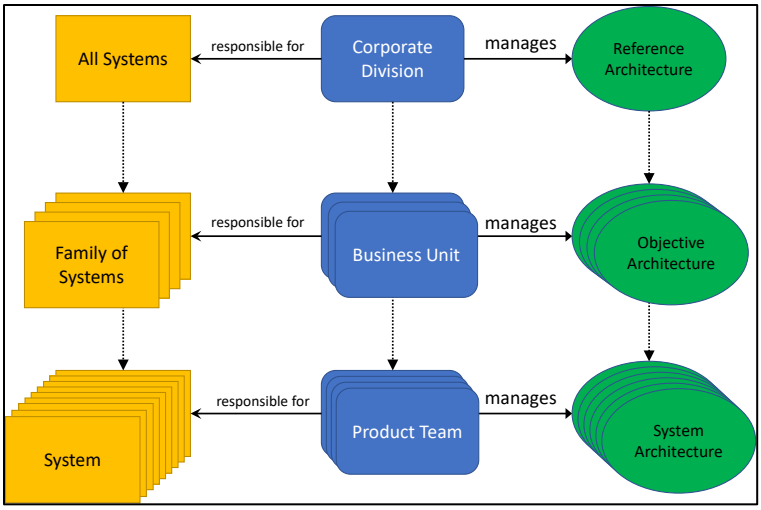

Figure 1 – Comprehensive Architecture Approach

3. What is the Comprehensive Architecture Strategy (CAS)

The Comprehensive Architecture Strategy (CAS) is an approach for organizations to determine the architecture mechanisms needed to meet high level business requirements and key business drivers in addition to the technical and performance needs. This approach will enable procured systems to be designed according to a prescribed architecture resulting in systems that meet common enterprise goals such as affordability, interoperability, maintainability and modifiability [OASD/NII 2010].

The CAS approach will help the Department of Defense (DoD) move away from unique, single-sourced solutions to manage the increasing cost and complexity of modern systems and to deliver enhanced, integrated warfighting capability. A CAS provides more effective management of cohesive OA concepts and enablers across a wider sphere of influence. These enablers are based upon proven concepts and patterns [Bachmann 2007; Buschman 2007], and apply a coordinated and intentional systematic use of Technical Reference Frameworks (TRFs) and open specifications to include physical hardware interfaces, software execution framework, electrical, mechanical, signal, and software component interfaces, software component behavior and the context provided by domain specific data models [Bergey 2005].

Following the CAS tenets results in documenting the selection and use of these open specifications to define a baseline and subsequent architectures, which can be managed and controlled by the government to meet their programs' needs. When developed mission systems and/or their components strictly adhere to these architectures, software and hardware assets may be reused or replaced more efficiently, which positively impacts the cost and schedule of providing new or enhanced capability [Bergey 2010; DSB 2018]. Within CAS, a Reference Architecture (RefArch) is an enterprise architecture that is tailored to the military domain. The RefArch will consist will consist of a purpose, guiding principles, policy and technical positions, patterns, and a vocabulary that result from the convergence of stakeholder, technical and business contexts across the enterprise. The RefArch will provide governance

as well as a superset of tools, methods, process and standards supporting key business drivers, policy and regulatory concerns [Muller 2007].

Effective architectures should align with the organizational levels of the acquiring organizations. [GAO 2011]. Since many government organizations are hierarchical, the Comprehensive Architecture Strategy aligns to a multi-level structure. Thus, the CAS presents a cohesive multi-level architecture relationship which includes the following architectures:

- A Reference Architecture (RefArch) guides and constrains the instantiations of all subsequent architectures and solutions that fall under the purview of an organization or enterprise. It provides a common lexicon and taxonomy based upon stakeholder mission, vision, and strategy in order to facilitate communication and alignment efforts of the current and future architectures [OASD/NII 2010]. It provides no performance specification and only provides implementation details when consistency and commonality is necessary, but otherwise, defers these decisions and constraints to subsequent lower level architecture analysis.
- An Objective Architecture (ObjArch) is a technology independent architecture derived from the RefArch. There can be multiple ObjArchs that tailor the RefArch to a more specific set of stakeholder and operational platform or Family of Systems (FoS) level requirements. An ObjArch represents an organizational level around which to identify and exploit opportunities for commonality that reflects a FoS approach. An ObjArch forms the basis for a flexible product line architecture [Northrop 2008] and provides guidance to constrain the tailoring of subsequent System Architectures to meet specific needs and requirements of the specific Mission System.
- A System Architecture (SysArch) is an architecture product developed from an ObjArch. Together with a performance specification, the SysArch expresses the overarching system requirements needed for procurement of system implementations. It represents the fundamental organization of a mission system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution. It is also known as “Design-to Architecture” [SeBoK].

Figure 2 provides a graphical example of the architecture levels and how they can be aligned to organizational levels.

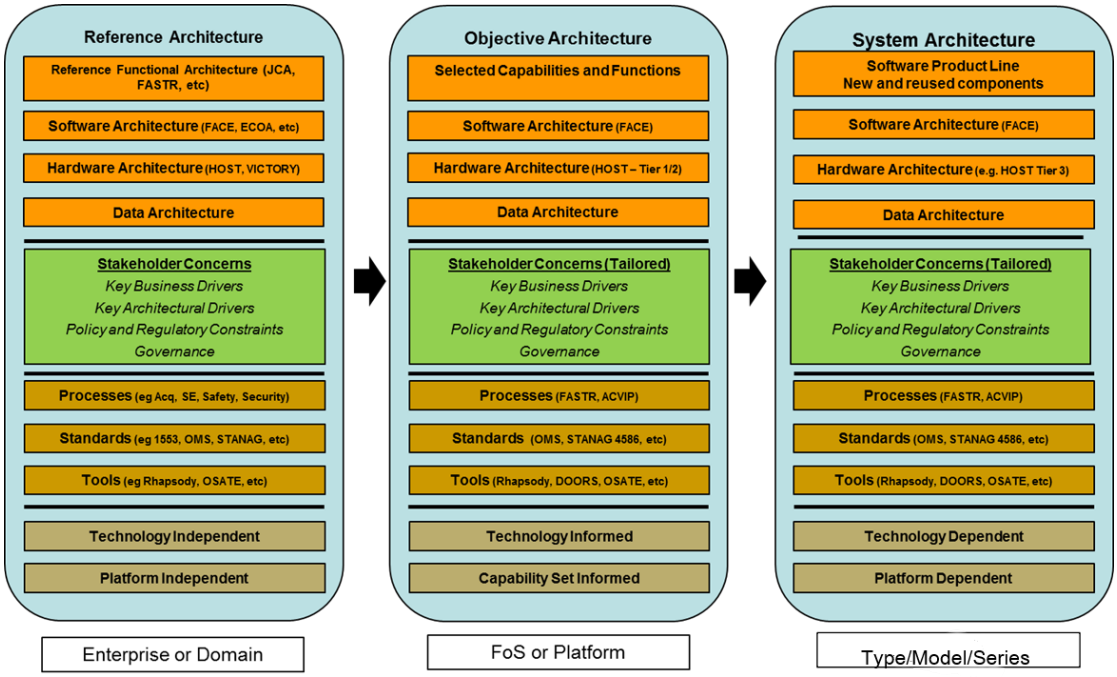

Figure 2 – Comprehensive Architecture Levels

The orange and green boxes in Figure 2 represent the eight (8) key elements that are present in each architectural level; the main difference between the elements at each level is the acquiring organization's tailoring or selection of options available from the higher level. These key elements are explained in further detail in Section 4.

The three levels of architectural analysis and documentation (RefArch, ObjArch, and SysArch) are necessary to provide the required guidance at each level of organizational control to achieve the desired technical and key business drivers established for each architectural level. This method of applying a multi-level architectural approach enables controlled elaboration and provides traceability between the outcome of decisions (choices) and the key business drivers behind the choices [Muller 2007].The tailoring process allows a large degree of flexibility and innovation at the ObjArch and SysArch levels to determine the right blending of open architecture and systems engineering principles to meet the high level business and technical drivers.

For example, in Army Aviation, and specifically, Future Vertical Lift (FVL), the three architectural levels align with the current FVL organizational and enterprise structure in the following manner:

- The PEO AVN or PM FVL RefArch could support the FVL organization across all aviation missions (bounded by the FVL capability sets) and subsequent Service platforms.
- ObjArchs will be derived from the RefArch and support the Capability Set (CS) structure (one ObjArch for one or more CS, or missions or other groupings as determined by analysis). Each ObjArch will form the basis of a flexible product line architecture and provide guidance to constrain the tailoring of subsequent SysArchs to meet specific needs and requirements of the specific Aviation Mission System.

- SysArchs (and resulting specifications) developed from an associated ObjArch, will support the development of each service's (Army, Navy, Air Force) variant of each FVL capability set.

4. Elements of each CAS Architecture Level

Each CAS architectural level contains a common set of elements, which are represented by the orange and green boxes in Figure 2. For each element, the specificity and options at each level depend on the stakeholder concerns at the corresponding organizational level. In general, the specificity will increase and the number of options available will decrease as the architecture specification moves from the enterprise level towards the system level. The following elements, defined in detail in the following sections, form the major areas of the each level specification. Of note, these elements are very similar to the concepts of capturing the vision, mission, strategic purpose, principles, and technical positions which are described in other Reference Architecture documents [Muller 2007; OASD/NII 2007].

- Key Business Drivers
- Key Architectural Drivers
- Policy and/or Regulatory Constraints
- Functional Architecture(s)
- Hardware Architecture
- Software Architecture
- Data Architecture (including Domain Specific Data Model)
- Governance (of the architecture, contract and acquisition guidance, data rights strategies)

4.1. Key Business Drivers

A Key Business Driver (KBD) is defined as a resource, process or condition that has major impact on the business and is vital for the continued success of an organization Key Business Drivers are a means of communicating stakeholder vision, guidance and critical business concerns. Clearly conveying the key objectives enables the selection of proven architectural concepts that meet the KBDs, independent of design and implementation decisions.

KBDs are generally determined by analyzing stakeholder inputs and grouping and prioritizing statements until a noticeable pattern emerges [Bass 1999; Kazman 2000]. KBDs can also be captured directly from the vision, mission, and strategy statements of the acquiring organization. Alternatively, they may be identified through analysis of the statements of business stakeholders captured in stakeholder concerns, use cases, or scenarios.

KBDs succinctly state the critical business drivers that must be met. All subsequent stakeholder objectives, constraints and quality attributes should ideally trace back to one of the KBDs. Quality attributes, defined below, such as modifiability and interoperability, are selected to support the KBDs independent of design and implementation concerns.

4.2. Key Architecture Driver (KAD)

The SEI defines a KAD as the combination of functional (operational) requirements, quality attribute requirements, and business requirements that shape the architecture or the element under

consideration. Typically, KADs are architectural requirements enforcing Quality Attributes (QA), or are derived from architecturally significant functional requirements, that trace or contribute to achieving two or more KBDs.

4.2.1. Quality Attribute

A Quality Attribute is a property of a work product or goods by which its quality will be judged by the stakeholders. Quality attribute requirements such as those for performance, security, modifiability, reliability, and usability will have a significant influence on the architecture of a system. It should be noted that these example QAs cross architecture, design, and implementation and policy boundaries. Each QA may also be applicable only at a certain architecture level. Selecting and documenting the QAs that are the highest priority of the Stakeholders forms the basis for the KADs.

4.3. Policy and Regulatory Constraints

Policy and regulatory constraints include National Defense Authorization Acts (NDAA), executive orders, legislative mandates, federal regulations, Service policies and other mandates or direct guidance that must be adhered to. Also known as "policy positions," they are included as architectural elements because they can directly impact (help or hinder) one or more KBDs or technical aspects. Policy positions may direct the use of specific technical positions or patterns or outline acquisition or business best practices such as data rights.

4.4. Functional Architecture

The Functional Architecture provides a method to document the functions or capabilities in a domain by what they do; the data they require or produce and the behavior of the data needed to perform the function. Note, at the RefArch level, the Functional Architecture is called the Reference Functional Architecture (RFA) because of its overarching and comprehensive content. At the ObjArch and SysArch levels, this becomes known simply as the Functional Architecture.

The FA should include a process which is used for logically grouping functions. These include functions that are candidate solutions to be developed as software or hardware implementations. The FA is complemented by a Data Architecture in order to provide the context and strong semantics necessary to ensure system interoperability.

4.5. Hardware Architecture

A Hardware Architecture specification describes the interconnection, interaction and relationship of computing hardware components to support specific business or technical objectives. A Hardware Architecture employs patterns, tactics and methods to address specific recurring architecture challenges with proven techniques, such as communication, topology, organization and interfaces. Implementation details and specific technologies are not typically included in a Hardware Architecture.

4.6. Software Architecture

A Software Architecture describes the relationship of software components and the way they interact to achieve a specific business or technical objectives. A Software Architecture employs patterns, tactics

and methods to address specific recurring architecture challenges with proven techniques such as communication, topology, organization, invocation, interface and state/mode maintenance. The Software Architecture provides a reference framework perspective for all implemented software components and provides a common architecture, data exchange and application of interface requirements.

4.7. Data Architecture

A Data Architecture provides the language and tools necessary to create, edit and verify Data Models. A Data Model captures the semantic content of the information exchanged. The Data Architecture promotes interoperability by eliminating ambiguities in the description of interface data. It is possible for a Data Architecture to provide common Data Models (e.g., FACE Shared Data Model (SDM)) to be extended when constructing other data model content. A Data Architecture can be applicable to many domains whereas a Data Model is generally restricted to a single domain of interest. These characteristics allow a Data Architecture to provide additional flexibility over a Data Model. An example is the FACE Data Architecture.

Domain Specific Data Model (DSDM) – A Domain Specific Data Model is a data model that contains the semantic information for a domain of interest. A FACE DSDM is defined as a data model, designed to the FACE Data Architecture's requirements, that captures domain specific semantics. A DSDM is used in developing system software components and can be the basis for additional software component level Data Models. A centrally managed Domain Specific Data Model can optimize the interoperability of functions within a centrally managed FA. The creator of the DSDM has control over which types of modeled elements are included to best meet interoperability goals of their domain.

One example implementation of the DSDM is the Unmanned Air System (UAS) data model, which is conformant to the FACE Technical Standard, Edition 3.0. The UAS data model provides a common foundation for software components leveraging the STANAG 4586 standard.

4.8. Governance

At each architectural level, the organization in control of the architecture must select and document the following at a minimum:

- Configuration management of the architecture
- Change management of the architecture, including the process for elaboration between the architecture layers
- Architecture Data Rights Strategy – specifically focused on what data rights are recommended for different components of the architecture to enable proper governance and management

The architectures must be actively managed by the appropriate organization to ensure that they are updated based on the evolution of the governing standards, emerging technology, changing stakeholder requirements and maturing business drivers.

5. Purpose of a Reference Architecture

The "DoD Reference Architecture Description" outlines the background, purpose, and structure of a RefArch which is defined as the "authoritative source of information about a specific subject area that

guides and constrains the instantiations of multiple, related architectures and solutions" [OASD/NII 2010].

The RefArch guides an organization in making decisions about the systems it acquires or uses and is the primary vehicle for communicating the KBDs of the acquiring organization, and the resulting KADs, to the development community. A RefArch provides a standardized structure and guidance to capture architectural decisions during the systems engineering process. A RefArch (i.e., a tailored enterprise architecture) is how an organization controls and manages the architectures of the systems it acquires or uses. The acquiring organization uses the RefArch to establish the architecture requirements and constraints for the specification of all platforms/systems within the oversight of the organization (i.e., fleet or enterprise). A driving factor for the creation of a RefArch is to improve the effectiveness of creating products, product lines and product portfolios. The RefArch guides and constrains the development of more detailed or refined architectures, essentially serving as a requirements specification for derived ObjArchs. Adherence to a RefArch enables maximum opportunities for commonality and interoperability across the entire area of interest [GAO 2011].

Documentation of the RefArch consists of a purpose, guiding principles, policy and technical positions, patterns, and a vocabulary that result from the convergence of stakeholder, technical and business contexts [OASD/NII 2010].

6. What is Documented in a Reference Architecture

The content of a RefArch is based on a number of considerations. First, the boundary of the area of interest (i.e., enterprise) needs to be established. Typically, this is based on organizational realm of control but may be driven by a wider community of interest. Establishing the enterprise enables bounding of what capabilities are required to satisfy all potential system capabilities within the RefArch's sphere of influence [GAO 2011].

Once the sphere of influence is established, then the enterprise business goals, and non-technical attributes, or "ilities", can be documented. Once such attributes are identified and considered, the appropriate patterns, processes, standards, tools and guidance to meet those attributes, and associated stakeholder goals, will be documented. These attributes can be selected and documented in a manner that is independent of implementation and based on pedigree of effectiveness.

A RefArch must be accessible and understandable for multiple stakeholders from senior leadership to program managers to engineers and suppliers. Therefore, the RefArch must be concrete and provide specific information. The challenge is to create a RefArch that supports development of multiple product lines by allowing multiple implementation options while still having concrete requirements and guidance to assist with lower level architecting. Each requirement offered within a RefArch should come with guidance as to why it was selected so that when decisions are made during development of a subsequent objective architecture it can guide users to a common solution [ANSI/IEEE 2000].

6.1. Key Business Drivers

The RefArch documents the KBDs of the enterprise and should note the Stakeholders who contributed to the derivation of the KBDs. The right mix of stakeholders, including those representing organizations at the ObjArch and SysArch levels as well as end users of actual implementations, will ensure a more comprehensive survey of desired attributes under the control of the enterprise. The enterprise level organization should look to establish business drivers that are identified as key to the organization's success.

The KBDs should define the strategic purpose and provide the context for the RefArch; they should explain why the RefArch is necessary in terms of scope, goals and objectives [OASD/NII 2010]. Each KBD should be well defined and documented with rationale. All subsequent stakeholder objectives, technical requirements and quality attributes should ideally trace back to one of the KBDs. Examples of KBDs are "increasing affordability" and "faster time to field."

6.2. Key Architecture Drivers (KAD)

The RefArch will document the KADs that were determined by stakeholders across the enterprise to meet the KBDs. The KADs are the highest priority architecture requirements enforcing certain QAs. The KADs for the RefArch are identified by translating business objectives, technical objectives and other Stakeholder concerns into a set of desired architectural QAs. The QAs are prioritized based on which ones support more than one KBD, with the top QAs becoming the KADs. KADs inform decisions when selecting architecture, design, and implementation mechanisms at each level of architecture. Identifying the KADs helps resolve conflicts and assists in evaluating the trade space between architectural and design mechanisms that are chosen during elaboration of the architecture specifications for the RefArch, ObjArch and SysArch resulting in the implementation of a system design. The RefArch will document the KADs in the form of architectural requirements that will yield desired QAs, such as maintainability.

6.3. Policy and Regulatory Constraints

The RefArch will document the organizational policy, regulatory constraints, mandates and other "fact of life" requirements that must be adhered to across the enterprise. Any technical positions and patterns resulting from policy positions should be documented in the RefArch. For example, a policy may require compliance with a specific industry or military technical standard. If there have been "trades" made to comply with certain policy positions, it is important to document these in the RefArch so they can be flowed down to the subsequent architectures to ensure consistency.

6.4. Reference Functional Architecture (RFA)

The Functional Architecture at the RefArch level is called the Reference Functional Architecture. Since it is a reference, it typically does not prescribe organization or dependencies between defined capabilities. The applicable RFA is selected during the definition of the RefArch. The RFA will indicate the library of functions that provides a superset of capabilities to meet operational requirements within the area of interest (e.g., UAS Control Segment (UCS) for ground control stations, Joint Common Architecture (JCA) for vertical lift). The capability requirements to scope the functional decomposition of modular capabilities will be bounded by the enterprise or domain area of interest. Selection of an RFA will need to be determined on enterprise goals. Reuse, portability, interoperability, adaptability, integrability,

upgradeability etc., are examples of these non-technical functions that will need to be weighed by the enterprise or domain leadership.

The RFA should include a process which is used for grouping functions that logically fit with the components likely to be used. If a model is used to capture RFA characteristics, it should depict data that is both required and provided by each capability; to build views in the model to depict this data; and to build projections from functional capabilities to the provided data entities.

At the RefArch level, the library of decomposed functions should be centrally managed by the owner of the RefArch to ensure maximum opportunity for reuse by a product family (ObjArch) or system (SysArch). These include conceptually defined functions that are candidate solutions to be developed as software or hardware implementations. As a reminder, the RFA only exists in the RefArch. The other architecture levels use the RFA to define an appropriate Functional Architecture.

6.5. Hardware Architecture

The Hardware Architecture at the RefArch level is an architectural mechanism to achieve quality attributes that meet the KADs. This mechanism can be realized by design or implementation mechanisms specified during the ObjArch and SysArch derivation process. At the RefArch level, a limited set of Hardware Technical Reference Frameworks (HW TRF) will be selected to minimize deployment options while enabling realization of the key business drivers of the enterprise. These standardized frameworks work in concert with an RFA and enable the decomposed functions to be deployed onto the architecture using standardized key interfaces.

6.6. Software Architecture

The Software Architecture at the RefArch level is an architectural mechanism to achieve quality attributes that meet the KADs. This mechanism can be met by design or implementation mechanisms specified during the ObjArch and SysArch derivation process. At the RefArch level, a limited set of Software Technical Reference Frameworks (SW TRF) will be selected to minimize deployment options while enabling realization of the KBDs of the enterprise. These standardized frameworks work in concert with an RFA and enable the decomposed functions to be deployed onto the architecture using standardized key interfaces.

6.7. Data Architecture

At the RefArch level, the RFA is complemented by a Data Architecture that provides the context and strong semantics on data exchanged. The enterprise or domain Stakeholders may specify and manage a Domain Specific Data Model (DSDM) in the RefArch to be used as the basis for information exchange. The Data Architecture requirements will constrain the ObjArch and SysArch definition and refinement of the Data Architecture requirements, including allowable refinements of the DSDM.

6.8. Governance

The RefArch will document the entity responsible for configuration management and change management of the RefArch and ObjArchs derived from the RefArch. The governance should include

what data rights are recommended for different components of the architecture to enable proper governance and management.

6.8.1. Guidance

In addition to documenting the eight RefArch elements, the RefArch should provide guidance on use of the architecture, best practices to derive or develop ObjArchs from the RefArch, acquisition strategy recommendations, and contract language guidance. Guidance will also include the recommended processes, standards and tools that assist in meeting requirements. Each requirement offered within a RefArch should come with guidance as to why it was selected as part of the RefArch so that when decisions are made during development of an ObjArch it can guide users to a common solution.

7. Purpose of an Objective Architecture

The Objective Architecture (ObjArch) defined by CAS is the architectural specification applicable to a group of systems such as a FoS, or a product line that is acquired, managed or controlled by or within an organization. The ObjArch provides more detail than, and is refined from, the RefArch, and includes support for the business drivers and the characteristics of the product line. It is the mechanism to provide consistency, cohesiveness and commonality between all products in the product line.

An ObjArch is an architectural product that constrains the development of individual, subsequent SysArchs. The ObjArch provides the first opportunity for high level decisions without specific functional allocation or system identification; as such it becomes the primary vehicle to ensure that all products in the product line, acquired by the organization, meet the organizational business goals. It communicates to the development community the architectural specifications that are common to a product line. The ObjArch contains architectural requirements, and may contain design and/or implementation requirements if they are necessary to achieving the business goals of the product line.

An ObjArch is focused on the architectural elements specific to a product line. These include constraints on the RFA, specification of Software and Hardware Architectures, a DSDM for the product line, and the allowed variation points and variation mechanisms for each of these. The ObjArch constrains the RFA to the specific functions (i.e., core assets) that are supported by the product line. The refinements to the Software and Hardware Architectures provide the specific frameworks and environments applicable and appropriate for the development of the product line. Of note, an ObjArch is a product line architecture, and depending on the product line, it may reference other product lines within it.

8. What is Documented in an Objective Architecture

The scope of an ObjArch is first determined by the RefArch from which it is derived and then further constrained to the product line that the ObjArch defines. An organization will have one ObjArch for each product line acquired, managed, or controlled. How the organization determines the product line is primarily a business decision, and will likely be representative of the management structure of the organization. Regardless of how the determination of product lines is made, the ObjArch documents the specification of the resulting related products. Once the product line is identified, the derivation of the corresponding ObjArch begins.

An ObjArch contains the same kind of elements as its parent RefArch, but with greater specificity directed at managing a product line for the organization. Each of these ObjArch elements is derived from the corresponding RefArch element through the refinement and the addition of detail. As with the RefArch, each requirement offered within an ObjArch should come with guidance as to why it was selected as part of the specification so that when decisions are made during derivation of a SysArch it can guide users to a consistent solution. The rationale for each selection should be well documented and traceable to the RefArch, which allows the owner to know what it affects upwards and downwards in the architectural levels.

8.1. Key Business Drivers

The KBDs defined in the RefArch are the foundation for the ObjArch KBDs. However, because the ObjArch organization may have more specific or additional business concerns, the KBDs of the RefArch may need to be tailored to accommodate the specific concerns of the ObjArch organization. Additional business concerns and their impact on architectural decisions should be captured in the documentation for the ObjArch so they can be flowed down to the SysArch level.

As an example, assume one of the RefArch KBDs is “Affordable Lifecycle Costs (ALC).” At the RefArch level, ALC is a comprehensive value assessment that considers the value of a capability, and its lifecycle sustainment costs. Minimizing costs is one way to achieve ALC while increasing value is another. At the RefArch level, the value-add of specific system capability is hard to determine. As a result, the RefArch is focused on architectural aspects that reduce cost and complexity. At the ObjArch level, it is easier to estimate the value of a given capability. This adds additional information to the value assessment, and may impact the types of architecture mechanisms that are specified by the ObjArch. Mechanisms that have a positive impact on operational value, in addition to reducing lifecycle costs, have higher value in the ObjArch than mechanisms that are neutral or negative.

Refinement of the RefArch KBDs at the ObjArch level should consider the additional information about the product line, and the impact of that information on the suitability of specific quality attributes and derived architecture mechanisms to achieve the KBDs in the context of the ObjArch. Figure 3 illustrates the relationship between a KBD, architecture mechanisms, architectural requirements and quality attributes. This information should be captured as rationale for the refined KBDs.

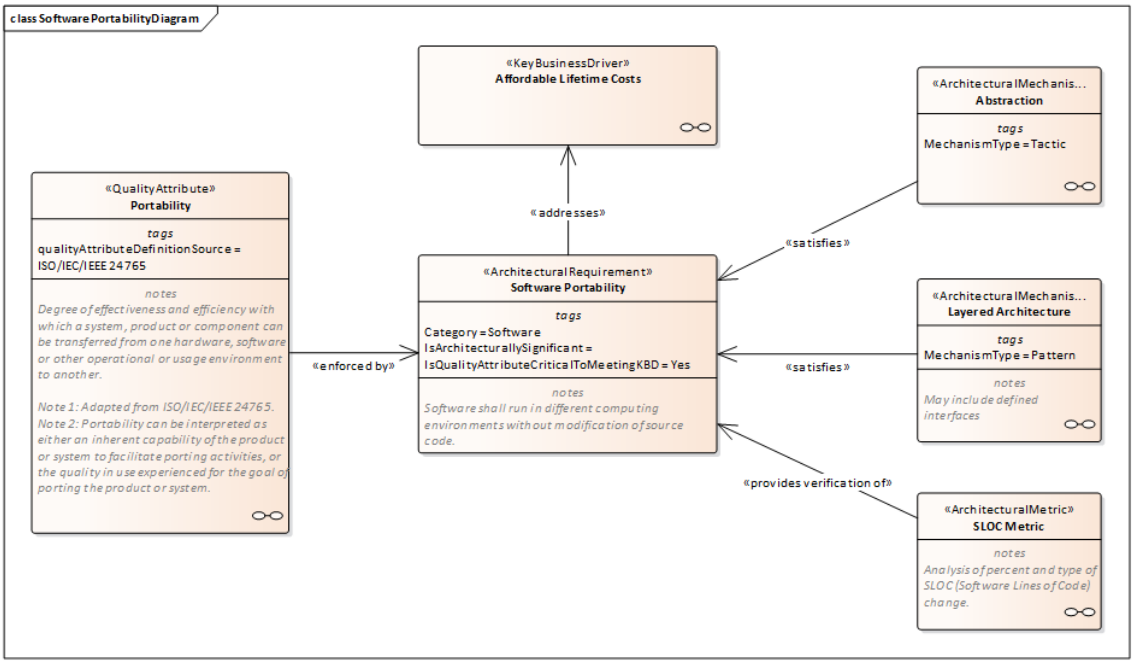

Figure 3 – Example KBD, QA and Mechanism relationships

8.2. Key Architecture Drivers (KAD)

Similar to the KBDs, the RefArch KADs need to be reviewed, and if necessary, refined to capture the specific architecturally significant qualities that will meet the organization's business goals for the product line specified by the ObjArch. Subtle refinements of the RefArch KBDs that may have taken place when deriving the ObjArch key business drivers may have significant impact to the KADs at the ObjArch level. Care should be taken in the review and refinement of the RefArch KADs to derive the ObjArch KADs to ensure that the ObjArch KBDs are met; in other words, refining the RefArch KADs must take place with the knowledge of the ObjArch KBDs. It is possible that a QA identified at the ObjArch level will result in the selection of specific design or implementation mechanisms to meet the product line objectives. Any QAs identified at the ObjArch must not contradict RefArch KBDs, KADs or QAs.

For example, if ALC is a RefArch KBD, it may be satisfied by RefArch KAD requirements that focus on KADs to reduce cost and complexity. At the ObjArch level, the ALC KBD can remain the same, but the rationale can be expanded to include the value of the operation of the product line. If a specific capability has a value greater than the total lifecycle costs of the system, the KADs may shift from "reducing cost and complexity" to "enhancing performance." This is one of the more obvious cases where the ObjArch KADs could be different than the RefArch KADs. It is necessary to review and possibly refine the RefArch KADs into different ObjArch KADs when creating the ObjArch specification.

8.3. Policy and Regulatory Constraints

The Policy and Regulatory requirements for the ObjArch are flowed down from the RefArch Policy and Regulatory requirements, with additional consideration given to any policies and mandates of the ObjArch organization. For example, organizational mandates to use "common" equipment or to adopt standards that are driven by non-technical needs or organizational policy are considered in this phase of

architectural analysis. These policy and regulatory constraints are one way that specification of "common" equipment or use of specific architectural mechanisms becomes part of the ObjArch.

8.4. Functional Architecture

At the ObjArch level, the refinement of the RFA defined in the RefArch will result in a Functional Architecture (FA) specification. The ObjArch FA contains the specification of functions (i.e., core assets) to be implemented by systems built to architectures developed from the ObjArch. Because of the central role of functionality in the specification of product lines, the refinement of the RefArch RFA into the ObjArch FA is one of the more challenging refinements in the CAS process. Several decisions must be made as part of the derivation of the ObjArch FA.

The first decision is identifying the complete list of functions that the product line will provide. The RefArch RFA provides documentation of the functions that may be used in a derived ObjArch. The refinement of the RefArch RFA into the ObjArch FA involves the down-select of functions that will be part of the ObjArch FA. This refinement and down-select can be accomplished using several different processes. Regardless of the specific process used, the ObjArch FA should be complete, comprehensive, and selected from the RefArch RFA.

Once this refinement is complete, if the RefArch RFA specifies any structural organization (e.g., separation of concerns) requirements, the full list of ObjArch FA functions is organized into groups that align to this organization. For example, if the RefArch RFA contains a requirement to separate business logic from infrastructure functions, the ObjArch FA may additionally be divided into groups:

- Infrastructure Functions
    - The basic resources necessary for the operation of a system; includes, but is not limited to, data storage, data exchange, data mediation, logging, reporting, alerting/warnings
- Management Functions
    - The functions responsible for workflow, execution, and management of the mission functions
- Mission Functions
    - Those functions that support the execution of the Mission or set of related Missions.
    - Core Mission Functions
        - The subset of Mission Functions that are common across related Missions
        - Examples include communications and navigation
        - Candidate commodity functions across multiple product lines
    - Mission Specific Functions (Modifiable)
        - The subset of Mission Functions that are specific to a Mission, or that may be modified to be specific to a particular Mission (including platform specific modifications)

The ObjArch FA would then consider any further organization that should be imposed upon the ObjArch FA as a result of factors affecting the ObjArch FA as illustrated in Figure 4. Since the ObjArch is primarily concerned with product lines, the ObjArch will likely have a derived requirement to separate the Mission

Functions into 2 (or more) groups that represent functions that are common across the product line and those that are not.

If it is determined that common or non-variable (Infrastructure Functions and Core Mission Functions) portions of the ObjArch are reusable assets, it is possible that they can form independent product lines managed by use of specialized ObjArchs. In this case, the "Infrastructure ObjArch" or "Core Mission ObjArch" become assets that can be managed and reused in the same manner as a reusable function. The derivation of the ObjArch FA may include a choice of an Infrastructure Services (or Core Mission Services) ObjArch FA if a suitable one already exists.

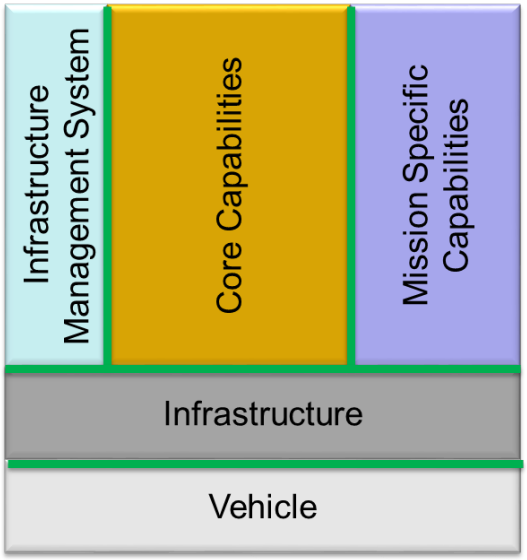

Figure 4 – Possible ObjArch FA Organization

The next area of concern in the derivation of the ObjArch FA from the RefArch RFA is the type and location of the variation points within the ObjArch FA. This is a critical decision in the derivation of the ObjArch FA. It is the primary discriminator between the RefArch RFA and the ObjArch FA. The first part of this choice is the identification of which functions will be variation points for subsequent SysArchs. This choice limits the variation points to the functions chosen as part of this group. In addition to identifying which functions provide variation within the product line, the manner of variation is also specified as part of the ObjArch FA derivation. There are many tactics, patterns and methods (Architectural, Design, and Implementation) that provide suitable variation mechanisms [Cohen 2010]. The mechanisms chosen should apply to all products developed from the ObjArch specification. Examples of variation mechanisms at the FA include:

- Configurable functions
- Extensible functions
    - Pluggable modules
    - Linked modules
- Design patterns for variation

- Implementation patterns for variation

The final set of decisions to be made for the ObjArch FA is the initial choice of deployment options for the functions selected functions. Functions are allocated into logical groupings that support procurement, qualification and performance decisions. At this level, there are several ways to allocate and deploy a function:

- Hardware
    - Application-specific integrated circuit (ASIC)
    - Field Programmable Gate Array (FPGA)
    - Custom hardware
- Firmware
- Software

While the deployment choices for the functions may be deferred, the ObjArch is the first opportunity to declare a deployment requirement for a function. If a function is intended to be deployed in the same manner in all products of the ObjArch product line (i.e., core asset), then it is likely that that decision will be made at the ObjArch level.

8.5. Hardware Architecture Specification

The ObjArch hardware specification is derived from the RefArch hardware specification and includes the hardware requirements that support the creation of a product line. While the RefArch hardware architecture will require general features of the hardware to meet the KADs, the ObjArch hardware specification will also incorporate specific hardware requirements based upon design and implementation decisions driven by the type and performance characteristics of the product line.

Some of the decisions that will drive ObjArch hardware architecture requirements include:

- Choice of hardware “category” (e.g., embedded, IT)
- Choice of hardware standards to meet the RefArch hardware specification
- Local Bus to meet required cyber-security and computing bandwidths and latency
- Network Bus to meet required cyber-security and distributed bandwidth and latency
- Graphics processors
- Specialty processors (array, vector, FP, etc.)
- Specialty hardware (ASICs, FPGAs)

8.6. Software Architecture Specification

The specification of the Software Architecture within the ObjArch begins with the derivation of requirements from the RefArch and continues with the addition of requirements necessary to create a suitable software environment to support a product line. This process includes the specification of one or more appropriate SW TRFs (such as Future Airborne Capability Environment (FACE), Joint Tactical Networking Center (JTNC) Software Communication Architecture (SCA), Java Enterprise Edition (Java EE),

European Component Oriented Architecture (ECOA), Oasis Service Component Architecture (SCA), CORBA Component Model (CCM), Service Oriented Architecture (SOA), etc.).

The SW TRF must meet the RefArch and derived ObjArch requirements, and be suitable for the product line. For example, a web services framework would be an inappropriate choice as the framework for a Command and Control product line. The SW TRF may be specified by requirement to use a documented framework, or as a complete set of requirements that specify the intended framework. Using a software framework as a mechanism is a universally applicable approach to address cost and time-based business drivers. It is not applicable at the RefArch level because of the variability of potential products that the RefArch specifies. However, at the ObjArch level, where the focus of the ObjArch is a single product line, a SW TRF can be chosen to meet the specific needs of the product line. The use of a SW TRF will assure that the quality attributes of the framework are enforced in all the products developed for the product line. If it is a requirement that components developed for one ObjArch product line be reusable (or interoperable) in another ObjArch product line, the RefArch managing both ObjArchs must contain the appropriate requirements on software implementation to accommodate the possible differences in frameworks. In addition to the SW TRF, the product line requires one or more variation mechanisms, and one or more variation points.

Software variation mechanisms are ways to implement the functional variation points specified in the ObjArch FA. There are many patterns, methods and tactics to implement variation points in software. Each pattern, method or tactic will have impacts to one or more quality attributes. It is the responsibility of the ObjArch architect to evaluate the variation mechanisms against the ObjArch KADs and supporting QAs to determine the appropriate variation mechanisms to specify in the ObjArch Software Architecture.

The software variation points are the locations within the software where variation is allowed. The variation points are affected by the choice of variation mechanisms. Because of the dependency of variation points on variation mechanisms, the choice of variation mechanisms and variation points is challenging. It may be necessary to trade off variation mechanisms to choose a specific variation point. Careful analysis of the impact of variation points on variation mechanisms, and the relevant QAs must be conducted to determine that a variation point is suitable and meets the ObjArch QAs.

Some of the decisions that will drive ObjArch software architecture requirements include:

- Choice of software “category” (e.g., embedded, IT)
- Choice of Software Standards to meet the RefArch software specification
- Choice of Frameworks
- Choice of Variation Mechanisms
- Choice of Variation Points

8.7. Data Architecture Specification

The Data Architecture (DA) requirements of the ObjArch are derived from the RefArch DA requirements, and further informed by the operational requirements of the product line. In addition to the DA

requirements specified by the RefArch, the ObjArch DA requirements will specify mechanisms to ensure that the RefArch requirements for interoperability (if any) can be met across the many products developed to a SysArch developed from the ObjArch. The use of a DSDM is one way to ensure semantic consistency across all the products specified by the ObjArch. If the RefArch specified a DSDM, the ObjArch specification can extend or elaborate the DSDM through the addition of Logical or Platform elements if needed to support the product line. If the RefArch did not specify a DSDM, it can be specified at the ObjArch level as a refinement of the RefArch DA requirements.

8.8. Governance

The governance of the ObjArch is similar if not identical to the governance of the RefArch. It follows the governance plan for the organizations implementation of CAS.

8.8.1. Guidance

As with the RefArch, the ObjArch should provide guidance on use of the architecture, best practices to derive or develop SysArchs from the ObjArch, acquisition strategy recommendations, and contract language guidance. Guidance will also include the recommended processes, standards and tools that assist in meeting requirements. Each requirement offered should come with guidance as to why it was selected so that when decisions are made during development of a SysArch it can direct users to a common solution.

9. Purpose of a System Architecture

A System Architecture (SysArch) is the architecture for a specific product. It is developed from an ObjArch and informed by the product performance requirements. The performance specification, together with the SysArch, specifies the overarching system requirements needed for procurement of a system (the product). It is also known as “Design-to Architecture.”

The SysArch represents the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution. A SysArch is focused on the architectural elements specific to a product. These include a specific set of functions from the ObjArch FA, detailed specification of Software and Hardware Architectures, and a Data Architecture for the product. In addition to the product specific refinement of the ObjArch requirements, the SysArch introduces additional decisions that must be addressed at this stage of development. These additional concerns include the system context, initial deployment decisions, concurrency management, orchestration of functions, and others.

The majority of design mechanism requirements are introduced at the SysArch level. While the RefArch and ObjArch mechanism requirements are primarily architecture mechanisms, the SysArch captures implementation and technology agnostic design mechanisms. The SysArch design mechanisms are introduced either as refinements of ObjArch requirements, or as architecturally significant requirements derived from performance, policy or regulatory requirements of the program.

10. What is Documented in a System Architecture

The scope of a SysArch is first determined by the ObjArch from which it is derived and then further constrained to the product that the SysArch documents. Whereas the ObjArch defined the product line architecture, the SysArch captures the additional information needed to procure components or systems and includes technology decisions to enable subsystem and component-level procurement decisions.

If developed by a government organization, the government furnished SysArch would provide only the details necessary to enable government procurement goals. The remaining architecture decisions will be specified by the integrator or developer of the system and captured in a refined SysArch and other system design documents.

A SysArch contains the same kind of elements as its parent ObjArch (and therefore, RefArch, as well) but with greater specificity directed at the system specific information necessary to ensure the product meets the business goals of the organization. Each of the SysArch elements is derived from the corresponding ObjArch element through refinement and the addition of detail. As with the RefArch and ObjArch, each requirement offered within a SysArch should come with guidance as to why it was selected as part of the specification so that when decisions are made during derivation of a system design it can guide users to a consistent solution. The rationale for each selection should be well documented and traceable to the parent ObjArch, which allows the owner to know what it affects upwards and downwards in the architectural levels.

10.1. Key Business Drivers

The KBDs refined in the ObjArch are the foundation for the SysArch KBDs. However, because the SysArch organization may have more specific business concerns, the KBDs of the SysArch may need to be tailored to accommodate the specific concerns of the SysArch organization. It is possible that a QA identified at the SysArch level will result in the selection of a specific architecture or design mechanism to meet the objectives of the SysArch organization or ultimate end user. Any QAs identified at the SysArch level cannot contradict RefArch or ObjArch KBDs, KADs or QAs. Any additional business concerns or QAs, and their impact on architectural decisions, should be captured in the documentation for the SysArch level, so they can flow down as guidance during the derivation of the system design.

Refinement of the KBDs at the SysArch level should be carefully considered. While refinement of the RefArch KBDs at the ObjArch level is likely because of the breadth of product lines within the RefArch organization, the products within a product line are more cohesive. Any refinement of the SysArch KBDs will likely be updates to the precedence of KBDs, or similar permutations, rather than the introduction of new business drivers, or changes to the value proposition of existing business drivers.

10.2. Key Architecture Drivers (KAD)

Similar to the KBDs, the KADs need to be reviewed, and if necessary, refined to capture the specific architecturally significant qualities, most likely driven by performance requirements, while still meeting the organization's business goals for the product specified by the SysArch.

10.3. Policy and Regulatory Constraints

The Policy and Regulatory requirements for the SysArch are flowed down from the ObjArch Policy and Regulatory requirements, with additional consideration given to any policies and mandates of the SysArch organization. For example, organizational mandates to use “common” equipment or to adopt standards that are driven by non-technical needs or organizational policy are considered in this phase of architectural analysis. These policy and regulatory constraints are one way that specification of “common” equipment or use of specific architectural mechanisms becomes part of the SysArch.

Once the Policy and Regulatory requirements of the ObjArch have been addressed, any additional Policy or regulatory concerns of the SysArch organization are addressed by creation of appropriate SysArch requirements.

10.4. Functional Architecture

At the SysArch level, the Functional Architecture (FA) specification defined in the ObjArch will be further refined by making additional architecture decisions. FA decisions include deployment choices, informed by specific system or product performance requirements. Initially, the FA choices which may be deferred until the system design are determined. After this determination has been made, then the SysArch requirements for the necessary topics may be developed. The initial assessment and the resulting requirements derivation may be accomplished using a variety of processes and procedures. Regardless of the specific process used, the SysArch FA should be complete, comprehensive, and selected from the ObjArch FA, which in turn conforms to the RefArch RFA.

Because the SysArch is the last architecture specification prior to design and implementation, the SysArch refinement must include requirement specifications for all architecture decisions where the architecture analysis indicates that there is only one suitable design alternative. If this analysis is incomplete, inconclusive, or incorrect, it is likely that some of the KBDs, KADs, and resulting QAs of the organization will not be met by the system. During the refinement of the ObjArch FA into the SysArch FA, there are several specific areas to consider. Among the refinement considerations are:

- The selection of product specific functions that will be implemented in the product
- The aggregation of function into components
- Choice of specific FA specified variation points
- The deployment of function in hardware, firmware, or software

The selection of functions to be implemented in the product may not be deferred until design. It must be a complete specification of the function necessary to achieve the functional purpose of the product. If functions necessary to the operation of the product are not present in the ObjArch FA, they must be included in the SysArch FA. For functions that have not reached technical maturity, the externally visible aspects of the function (the interfaces and required behavior) may be deferred until system design to encourage innovation. For technically mature functions, the externally visible aspect requirements should be captured in the SysArch FA and contributed to the ObjArch Governance process.

After the required functions have been identified, the aggregation of functions into components may be considered. In some cases, there will be existing component definitions that have already been chosen such as previously discussed management, core, commodity or infrastructure functions. In these cases, there are likely component specifications that are available for reuse. In the case of highly specific mission functionality, the aggregation of functions into components is likely to be deferred until the SysArch. Because the aggregation of functions into components is primarily a business decision, it should not be deferred until system design.

For ObjArch FA functions that have been identified as having multiple variation point mechanisms (e.g., configuration, extension modules, multiple implementation, etc.) in the ObjArch, it is necessary at the SysArch level to choose the variation mechanism for each affected component. Because this decision is architectural in nature (again, primarily a business decision) it should not be deferred until design. This decision is recorded in the SysArch as a requirement.

Deployment choices for components that were deferred at the ObjArch level will be addressed at the SysArch level. If a component deployment is architecturally significant, the appropriate decision will be made at the SysArch level and will restrict the System Design. For example, due to the architectural significance of safety and security concerns, these deployment decisions should not be deferred until system design.

10.5. Hardware Architecture Specification

The SysArch hardware specification is developed from the ObjArch hardware specification. The SysArch hardware specification incorporates specific hardware requirements based upon design and implementation decisions driven by the type and performance characteristics of the product.

Because the SysArch is the last architecture specification prior to design and implementation, the SysArch refinement must include requirement specifications for all architecture decisions where the architecture analysis indicates that there is only one suitable design alternative. If this analysis is incomplete, inconclusive, or incorrect, it is likely that some of the KBDs, KADs, and resulting QAs of the organization will not be met by the system. Some of the decisions that will drive SysArch hardware architecture requirements include:

- Choice of hardware “categories” (e.g., embedded, IT)
- Choice of CPU Architectures (Including Instruction Set Architectures)
- Choice of Memory Architecture
- Choice of Backplane Architectures
- Total hardware processing capacity
- Total hardware processing rate
- Choice of Distribution Architecture
- Local Bus to meet required cyber-security and computing bandwidths and latency
- Network Bus to meet required cyber-security and distributed bandwidth and latency
- Graphics processors
- Specialty processors (array, vector, FP, etc.)

- Specialty hardware (ASICs, FPGAs)
- Infrastructure Peripheral hardware devices (Storage, display, output devices, input devices, etc.)
- Choice of Hardware Standards to meet the SysArch hardware specification

At this point in the development of the SysArch, the traditional considerations of the hardware system architecture are addressed. The "category" (embedded, control, IT, etc.) of system has direct impact on the fundamental Hardware Architecture decisions that are documented in the SysArch. The type of processing required by the functions is a major driver of the choice of CPU architecture(s) that will be required by the product. This is true whether the function is deployed in hardware or software. The choice of CPU Architecture(s) will have major impact on the choice of memory architecture, which is driven by the SysArch FA requirements. Once the CPU and Memory Architecture needs have been identified, the Backplane Bus Architecture can be considered.

After the basic hardware processing decisions have been made, the total processing capacity and rate(s) should be evaluated based upon initial estimates of the processing requirements of the SysArch FA. Considerations of redundancy for safety, security (e.g., red/black separation requirements) and growth margin should be factored into the estimate of hardware capacity and rate. Once the total capacity is estimated, and based upon the CPU and Memory decisions made earlier, an initial assessment of the number and types of processors can be made. Options for distributed deployment of the required CPU and Memory can be considered once the initial CPU count is available.

After the local (to the backplane bus) decisions have been considered, the choices for Local Bus, and Network Bus can be evaluated. Considerations at this point include the probability of distributed processing capabilities, the need to connect to external components, the expected data rates and volumes, the expected growth needs over the life of the hardware, among many others.

Finally, hardware deployment decisions made in the SysArch FA will drive the need to consider specialty processors such as Array, MPSP, ASIC, FPGA, and others (such as graphics processors used for massive array processing). At the same time as specialty processors are considered, the other hardware peripherals necessary to meet SysArch FA requirements, and hardware infrastructure requirements will need to be evaluated.

The standards and frameworks that have been specified in the ObjArch will be carried forward to the SysArch. For design choices that have not been specified by an ObjArch requirement for a standard, consideration should be given to including the applicable standards in the SysArch. Use of standards to achieve architecture requirements provides a shortcut for the designer and guides the implementation to reduce the design effort to meet the SysArch requirements.

10.6. Software Architecture Specification

The specification of the Software Architecture within the SysArch begins with the derivation of requirements from the ObjArch. Additional software requirements necessary for the product design are pulled from the product performance specification. Except where design deployment decisions impact

the software architecture choice, all software architecture decisions should be made and documented in the SysArch.

As mentioned previously, because the SysArch is the last architecture specification prior to design and implementation, the SysArch refinement must include requirement specifications for all architecture decisions where the architecture analysis indicates that there is only one suitable design alternative. The areas of refinement of the ObjArch software architecture that are significant to development of a suitable SysArch are:

- Software Organization
- Software Topology
- Software deployment
- Software Communication Styles
- Software Invocation Pattern
- Software Interface Styles
- Software State Maintenance
- Software Framework
- Software Variation points
- Software Variation Mechanisms

While the ObjArch software architecture specifies the software architecture decisions that apply to a product line, the SysArch is specific to one product. Because there is no further product variability that must be maintained, it is important that the SysArch be complete. After the SysArch FA is finalized; and the hardware/software deployment decisions have been made; and in conjunction with the computing hardware decisions; the software organization, topology, and deployment decisions should be made. These choices must be consistent with the ObjArch requirements, and will most likely involve complex trade analysis of multiple solution alternatives. It is unlikely that there will be a clear best option, and there will probably be the need to sacrifice some qualities to achieve others.

Once the initial deployment choices have been made, the software communication patterns that are necessary to connect the distributed components must be considered. As with all other refinements, the ObjArch will establish the boundary conditions for the SysArch communications requirements. It is at this point, that separation of management, command, control, and data should be considered. The invocation pattern for the various executable components should be considered as part of the communication decisions. As part of the communication pattern analysis, the interface style should be established. After the communication, and interface specifications are established, the software state maintenance mechanisms should be specified.

Most often, the ObjArch will specify a set of one or more SW TRFs that will achieve the RefArch QAs and the KBDs and KADs that drive them. Many of the decisions mentioned above will be addressed by the ObjArch frameworks. Choosing from the available ObjArch frameworks based upon the decisions mentioned above will often reduce the effort to meet all of the SysArch software requirements by reuse of the analysis, and trades that were conducted to produce the framework.

In addition to the software requirements and the SW TRF, the SysArch will choose the applicable variation mechanisms and utilize the available variation points from the ObjArch to meet the performance and operational requirements of the product under consideration.

10.7. Data Architecture Specification

The Data Architecture (DA) requirements and selected mechanisms for the SysArch are developed from the ObjArch DA requirements, and further informed by the operational requirements for the system under consideration. If a DSDM has already been selected in the ObjArch, there may not be any additional SysArch requirements. If the ObjArch is silent on a DSDM, one should be considered for inclusion in the SysArch. If a DSDM is specified, the SysArch specification can extend or elaborate the DSDM through the addition of Logical or Platform elements if needed to support the product. The Data Architecture, and DSDM if specified, at the SysArch level, defines the specific method for describing the data exchanged by the product.

10.8. Governance

The governance of the SysArch is similar if not identical to the governance of the ObjArch and RefArch. It follows the governance plan for the organization's implementation of CAS.

10.8.1. Guidance

Just like the ObjArch, the SysArch specification should include guidance on use of the architecture, best practices, recommended architectural requirements, and mechanisms available for use (based on user needs). To the maximum extent possible, each option offered within the SysArch specification should include the rationale as to why it was selected as part of the architecture. Suppliers and integrators of the software and hardware for a system or capability will utilize the SysArch guidance to define detailed solution requirements, interfaces, data models and designs.

11. Summary

A comprehensive architecture approach is needed to ensure the acquisition of systems that achieve the acquiring organization's business and technical goals while allowing for elaboration of business and technical drivers at each level of an organization. A RefArch provides a standardized structure and guidance to capture architectural decisions during the systems engineering process. The elements and guidance in the RefArch should be of sufficient detail and specification to enable users to derive an ObjArch tailored for a product line based on the acquisition organization requirements (stakeholder concerns) or the users to limit or eliminate divergence from the enterprise goals. Subsequently, the ObjArch will provide a baseline for SysArch developers to pair architectural and performance requirements into a System Architecture Specification that enables industry to provide competitive innovative System Designs that meet an acquiring organization's overarching KBDs. Finally, the traceability between each level of architectural definition can be captured in a consistent set of model based views using an architecture framework such as DoDAF. The CAS approach allows an organization to procure systems that meet business and technical characteristics such as affordability, interoperability, robustness and consistency.

12. Dictionary

A common lexicon is necessary to understand the architectural strategy. Common terms used throughout this document include:

Term	Definition
Architectural Requirement	Describes a key system property (e.g., quality attribute) which has an important role in determining the architecture of a system and may include an appropriate pattern, process, standard, tool or guidance to meet the desired attribute
Capability Set	A related grouping of capabilities that are needed to support mission-level scenarios
CORBA Component Model (CCM)	The CCM is the OMG's extension to the original CORBA specification. CCM is designed to allow greater software reuse for servers, and provide greater flexibility for dynamic configuration of CORBA applications. The CCM extends the CORBA object model by defining features and services that enable application developers to implement, manage, configure, and deploy components that integrate commonly used CORBA services, such as transaction, security, persistent state, and event notification services, in a standard environment
Data Architecture	A Data Architecture is a set of related models, specifications, and governance policies with the primary purpose of supporting an interoperable means of data exchange among software components by unambiguously documenting the semantics of data
Data Model	A common model based upon a defined Data Architecture that captures the logical inter-relationships and data exchanged between different elements in the system. The Data Model contains common definition of conceptual and logical terms and documents the way data is passed between software components. Data models help represent what data is required and what format is to be used for different processes.
Data Rights	An aggregate term covering license, title or ownership of technical data and software. Technical data consists of information of a technical or scientific nature and includes reports, design information and software documentation. Software includes executable, object and source code as well as related tools, files, processes, and configuration data
Domain Specific Data Model	A data model, designed to the FACE Data Architecture requirements, which captures domain specific semantics. A DSDM generally does not contain Unit of Portability (UoP) Models. DSDMs typically contain Conceptual Data Model (CDM), Logical Data Model (LDM) and potentially Platform Data Model (PDM) elements. The creator of the DSDM has control over which types of modeled elements are included to best meet interoperability goals of their domain. All DSDMs must meet all requirements of a UoP Supplied Model (USM), except for those related to the specification of a UoP Model. DSDMs must still conform to the FACE Data Architecture meta-model requirements, the Object Constraint Language (OCL) constraints and all requirements from the FACE Shared

Term	Definition
	Data Model Governance Plan.
Enterprise	The high level “domain” or “sphere of influence” for the Reference Architecture levels under consideration. Enterprise level can range from a singular family of systems to a grouping of families of systems.
European Component Oriented Architecture (ECOA)	An open specification for a software framework for mission system software comprising components that are both real-time and service-oriented
Family of Systems (FoS)	A set of systems that provide similar capabilities through different approaches to achieve similar or complementary effects (IEEE definition)
Functional Decomposition	Functional decomposition is the breaking down of a complex system into smaller pieces which can be more easily managed. In its most basic form, functional decomposition is a simple hierarchical decomposition of the functions with associated performance requirements. (Ref: System Engineering Fundamentals Chapter 5).
Future Airborne Capability Environment (FACE)	A government-industry software standard and business strategy for acquisition of affordable software systems that promotes innovation and rapid integration of portable capabilities across global defense programs
Implementation Independent	An architecture or design that can be met via multiple implementations. Innovation can occur and still meet the requirements
Java Enterprise Edition (Java EE)	Created to simplify application development in a thin-client-tiered environment. Java EE simplifies app development and decreases the need for programming by creating standardized, reusable modular components and by enabling the tier to handle many aspects of programming automatically
Joint Tactical Networking Center (JTNC) Software Communication Architecture (SCA)	An open architecture framework that standardizes development of “Software Defined” systems. It aims to promote software reuse by enhancing portability, interoperability and configurability of software applications that run on embedded systems.
Key Business Drivers (KBD)	A resource, process or condition that has major impact on the business and is vital for the continued success and growth of an organization
Oasis Service Component Architecture (SCA)	Set of specifications which describe a model for building applications and systems using a Service-Oriented Architecture (SOA). SCA extends and complements prior approaches to implementing services, and SCA builds on open standards such as Web services.
Objective Architecture (ObjArch)	An ObjArch is an authoritative source of governing information derived from the Reference Architecture (RefArch), incorporating additional key business drivers to tailor the RefArch to a more specific set of stakeholder and operational platform or FoS level requirements. The ObjArch guides and constrains the instantiations of subsequent System Architectures and designs. The ObjArch reflects tailoring based upon capability documents, Concept of Operations (CONOPS), performance specifications, as well as specific approaches to commonality and reuse of hardware and software assets to satisfy high level requirements, capabilities, and objectives.

Term	Definition
Open Architecture (OA)	An OA is a hardware (i.e. computer) or software architecture specifically intended to allow efficient addition, modification and upgrade of system capability. OAs are characterized by defined and disclosed interfaces, relevant design disclosure and the use of publicly or officially supported standards. OA can promote vendor independent acquisition as well as intentional reuse of previously developed and interoperable components.
Patterns	Architectural patterns exist to describe generalized architectural experience, and they are possible because many systems have similar structure. Patterns address recurring design problems, and their usage enables solution reuse because they do not require understanding the particular solution that employs them. Patterns may be conveyed through a variety of methods (tables, text, structure, schemas, graphics, behavior, etc.).
Pedigree of effectiveness	Demonstrated, objective measure of meeting requirements
Quality Attributes	A property of a work product or goods by which its quality will be judged by some stakeholder or stakeholders. Quality attribute requirements such as those for performance, security, modifiability, reliability, and usability have a significant influence on the software architecture of a system (SEI definition). These attributes can be selected and documented in a manner that is independent of implementation and based on pedigree of effectiveness.
Reference Architecture (RefArch)	A RefArch provides a standardized structure and guidance to capture architectural decisions during the systems engineering process. It is an authoritative source of governing information about a specific subject area that guides and constrains the instantiations of all subsequent architectures and solutions. It provides a common lexicon and taxonomy based upon stakeholder mission, vision, and strategy to facilitate communication and alignment efforts of the current and future architectures of the system and/or family of systems. Since it is intended to be a high-level baseline architecture defined to incorporate a large sphere of influence or domain, it provides no performance capabilities or implementation details, leaving these decisions and constraints to subsequent architecture analysis.
Reference Functional Architecture (RFA)	A RFA is the product of recombining capability functions from the functional decompositions into discrete core functions, and the information provided and required by those functions. The RFA describes, in terms of structure, the desired modularity of a superset of functions at a conceptual level capable of satisfying the various operational activities of a given mission system within the system or Family of Systems (FoS).
Service Oriented Architecture (SOA)	A collection of services. These services communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some activity. Some means of connecting services to each other is needed.

Term	Definition
System Architecture (SysArch)	The SysArch is an architecture product developed from the Objective Architecture that applies system level performance requirements. It represents the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution. The SysArch determines mission system allocation of requirements and constrains system designs to Reference and Objective Architecture principles resulting in delivered capabilities that exhibit desired common characteristics (e.g. affordability, interoperability). It is also known as 'Design-to Architecture.'
Technical Reference Frameworks (TRFs)	Enterprise-level templates and patterns for classes of computing environments. They guide the development of architectures for a series of products. They provide a common vocabulary to promote consistency and commonality of interfaces and interactions across architectural boundaries such as layers and partitions. They separate requirements from the implementation details of systems, thereby enabling more rapid and cost-effective customization or reapplication of mission capability across platforms. TRFs clearly convey requirements for conformance, testing, and acceptance. A TRF enables realization of the key business drivers of the enterprise such as portability, interoperability, security, performance, or reusability.
Technology Independent	An architecture or design that can be met using different technologies. Multiple technologies can be employed to meet the requirements. This allows for innovation to occur.

13. References and Citations

Note – this list contains documents directly referenced in the body of the document as well as documents which influenced the decisions and recommendations of the authors.

[ANSI/IEEE 2000] Recommended Practice for Architectural Description for Software-Intensive Systems. New York, NY, USA: American National Standards Institute (ANSI)/Institute of Electrical and Electronics Engineers (IEEE), ANSI/IEEE 1471-2000. http://cabibbo.dia.uniroma3.it/ids/altrui/ieee1471.pdf

[Bachmann 2007] Bachmann, Felix; Bass, Len; & Nord, Robert. Modifiability Tactics. CMU/SEI-2007-TR-002. Software Engineering Institute, Carnegie Mellon University. 2007. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8299

[Baldwin 2016] Baldwin, Kristen and Lucero, D. Scott, Defense System Complexity: Engineering Challenges and Opportunities, The ITEA Journal of Test and Evaluation, March 2016: 10-16

[Bass 1999] Bass, Len; & Kazman, Rick. Architecture-Based Development. CMU/SEI-99-TR-007. Software Engineering Institute, Carnegie Mellon University. 1999. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=13385

[Bergey 2010] Bergey, John; Chastek, Gary; Cohen, Sholom; Donohoe, Patrick; Jones, Lawrence; & Northrop, Linda. Software Product Lines: Report of the 2010 U.S. Army Software Product Line Workshop. CMU/SEI-2010-TR-014. Software Engineering Institute, Carnegie Mellon University. 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9495

[Bergey 2005] Bergey, John; & Clements, Paul. Software Architecture in DoD Acquisition: A Reference Standard for a Software Architecture Document. CMU/SEI-2005-TN-020. Software Engineering Institute, Carnegie Mellon University. 2005. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=7437

[Bergey and Jones 2010] Bergey, John K & Jones, Lawrence G. Exploring Acquisition Strategies for Adopting a Software Product Line, August 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=28879

[SEBoK] The Guide to the Systems Engineering Body of Knowledge (SEBoK), SEBoK v. 1.9 released 17 November 2017. R.D. Adcock (EIC). Hoboken, NJ: The Trustees of the Stevens Institute of Technology. www.sebokwiki.org. BKCASE is managed and maintained by the Stevens Institute of Technology Systems Engineering Research Center, the International Council on Systems Engineering, and the Institute of Electrical and Electronics Engineers Computer Society.

[Buschman 2007] Buschmann, Frank, Henney, Kevlin and Schmidt, Douglas C. Pattern-Oriented Software Architecture, Volume 5: On Patterns and Pattern Languages. Chichester, UK: Wiley, 2007.

[Clements 2005] Clements, Paul; & Bergey, John. The U.S. Army's Common Avionics Architecture System (CAAS) Product Line: A Case Study. CMU/SEI-2005-TR-019. Software Engineering Institute, Carnegie Mellon University. 2005. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=7707

[Clements and Northrop] Clements, Paul & Northrop, Linda. A Framework for Software Product Line Practice, Version 5.0, Product Line Practice Initiative. http://www.sei.cmu.edu/productlines/frame_report/index.html

[Cohen 2001] Cohen, Sholom. Case Study: Building and Communicating a Business Case for a DoD Product Line. April 2001, Product Line Practice Initiative. ftp://ftp.sei.cmu.edu/pub/documents/01.reports/pdf/01tn020.pdf

[Cohen 2010] Cohen, Sholom; & Krut, Jr., Robert. Managing Variation in Services in a Software Product Line Context. CMU/SEI-2010-TN-007. Software Engineering Institute, Carnegie Mellon University. 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9293

[DAG] Defense Acquisition Guidebook, https://www.dau.mil/tools/dag

[DSB 2018] Department of Defense Defense Science Board, Design and Acquisition of Software for Defense Systems, February 2018. https://www.acq.osd.mil/dsb/reports/2010s/DSB_SWA_Report_FINALdelivered2-21-2018.pdf

[DoDD 5000.1] DoD Directive 5000.1, "The Defense Acquisition System," November 2, 2017 https://www.dau.mil/guidebooks/Shared%20Documents%20HTML/DoDD%205000.01.aspx#toc4

[DoDI 5000.2] DoD Instruction 5000.2, "Operation of the Defense Acquisition System," Change 3 August 10, 2107. https://www.dau.mil/guidebooks/Shared%20Documents%20HTML/DoDI%205000.02.aspx

[DoD 2008] DoD System Engineering Guide for System of Systems, 2008 https://www.acq.osd.mil/se/docs/SE-Guide-for-SoS.pdf

[Dvorak 2009] Dvorak, Daniel . NASA Study on Flight Software Complexity, AIAA Infotech@Aerospace Conference, Infotech@Aerospace Conferences, 2009. https://www.nasa.gov/pdf/418878main_FSWC_Final_Report.pdf

[GAO 2011] Military Departments Can Improve Their Enterprise Architecture Programs, GAO-11-902: Published September 2011.

[GAO 2013] United States Government Accountability Office, Defense Acquisitions: DOD Efforts to Adopt Open Systems for Its Unmanned Aircraft Systems Have Progressed Slowly, GAO-13-651: Published July 2013. https://www.gao.gov/assets/660/656419.pdf

[Hohpe 2005] Hohpe, Gregor and Woolf, Bobby. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. 6. Boston: Addison-Wesley, 2005.

[INCOSE 2015] International Council on Systems Engineering (INCOSE), Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, version 4.0. San Diego, CA. 2015 https://sepnobrasil.yolasite.com/resources/INCOSE%20Systems%20Engineering%20Handbook%204e2015.pdf

Innovative Integration Approaches, Version 2.0 presented at the NATO Science & Technology Organization's Specialists' Meeting on "Future Rotorcraft Requirements" in October 2015.

[ISO/IEC 2007] Systems Engineering – Application and Management of The Systems Engineering Process. International Organization for Standards (ISO)/International Electrotechnical Commission (IEC), Geneva, Switzerland, ISO/IEC 26702:2007.

[ISO/IEC/IEEE 2011] Systems and Software Engineering - Architecture Description. International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), Geneva, Switzerland, ISO/IEC/IEEE 42010.

[ISO/IEC/IEEE 2015] Systems and Software Engineering - System Life Cycle Processes, International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) / Institute of Electrical and Electronics Engineers.. Geneva, Switzerland, ISO/IEC/IEEE 15288:2015.

[Jones 1999] Jones, Lawrence G. Product Line Acquisition in the DoD: The Promise, The Challenges, Software Engineering Institute. Carnegie Mellon University, Pittsburgh, PA. November 1999

[Kendall 2014] Kendall, F., Office of the Under Secretary of Defense, Acquisition, Technology and Logistics (2014). Better Buying Power 3.0. White Paper.

[Kazman 2000] Kazman, Rick; Klein, Mark; & Clements, Paul. ATAM: Method for Architecture Evaluation. CMU/SEI-2000-TR-004. Software Engineering Institute, Carnegie Mellon University. 2000. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5177

[Moore 2016] Moore, Michael S. and Saylor, Kase J. (MOSA) for Military Systems: Addressing Challenges of Complex Systems, Interoperable Open Architecture Conference, 27 April 2016. https://www.dau.mil/cop/mosa/DAU%20Sponsored%20Documents/MOSA%20Military%20Systems%20IOA%20final.pdf

[Muller 2007] Muller, Gerrit and Hole, Eirik, ed. Reference Architectures; Why, What and How, White Paper Resulting from Architecture Forum Meeting, March 12 & 13, 2007 (Hoboken NJ, USA), Embedded Systems Institute and Stevens Institute of Technology. http://www.architectingforum.org/whitepapers/SAF_WhitePaper_2007_4.pdf

[Northrop 2008] Northrop, Linda. Software Product Line Essentials, Software Engineering Institute. Carnegie Mellon University, Pittsburgh, PA. 2008 https://resources.sei.cmu.edu/asset_files/Presentation/2008_017_001_24246.pdf

[OASD/NII 2010] Office of the Assistant Secretary of Defense, Networks and Information Integration (OASD/NII) DoD Reference Architecture Description, Office of the Assistant Secretary of Defense, OSAD/NII, June 2010. https://dodcio.defense.gov/Portals/0/Documents/DIEA/Ref_Archi_Description_Final_v1_18Jun10.pdf

[ODUSD 2008] Office of the Deputy Under Secretary of Defense for Acquisition and Technology, Systems and Software Engineering, Systems Engineering Guide for Systems of Systems, Version 1.0. Washington, DC: ODUSD(A&T)SSE, 2008.

[Patel 2013] Patel, P., and Fischerkeller, M. Prepare to be Wrong: Assessing and Designing for Adaptability, Flexibility, and Responsiveness (IDA Paper P-5005). Institute for Defense Analyses, Alexandria, VA 2013. https://www.ida.org/idamedia/Corporate/Files/.../IDA.../CARD/ida-document-p-5005.pdf

[Tate 2016] Tate, David M. Acquisition Cycle Time: Defining the Problem (Revised), Institute for Defense Analyses (CARD). Alexandria, VA. 2016

[Tate 2017] Tate, David M. Software Productivity Trends and Issues, Institute for Defense Analyses (CARD). Alexandria, VA. 2017 https://www.ida.org/idamedia/Corporate/Files/Publications/IDA.../2017/D-8367.pdf

[Wigginton] Wigginton, Scott. Joint Common Architecture Demonstration (JCA Demo) Final Report, TR-RDMR-AD-16-01, July 2016, http://www.dtic.mil/get-tr-doc/pdf?AD=AD1012511

[FAA] Federal Aviation Administration Advisory Circular for Reusable Software Components FAA AC 20-148, December 7, 2004

Joint Common System Function List: JCSFL Portal on NIPRNET at https://www.intelink.gov/wiki/JCSFL and on SIPRNET at http://www.intelink.sgov.gov/wiki/JointCommon_Systems_Function_List1 (last accessed 28 January 2015)

[NASA 2007] Systems Engineering Handbook. Washington, D.C.: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105.

The Open Group, Future Airborne Capability Environment (FACE™) Technical Standard, Edition 3.0, November 2017

The Open Group, Future Airborne Capability Environment (FACETM)Technical Standard, Edition 2.1, May 2014