THE UAS CONTROL SEGMENT ARCHITECTURE

Parag Batavia, Ph.D.[*], Rich Ernst[†], Kerry Fisherkeller[‡], Doug Gregory[§], Rob Hoffman[**], Ann Jennings[††], George Romanski[‡‡], Brian Schechter[§§], Gordon Hunt[***]

The Unmanned Aircraft System (UAS) Control Segment (UCS) Architecture is a software interface, data-model, and business system architecture, defining the rules and conventions for developing *interoperable* software components for UAS Ground Control Stations (GCS). The UCS Application Architecture, Platform Architecture, and Reference Architecture elements describe what services to build, how to build and host them, and how to assemble them into a functional GCS that facilitates subsequent certification of systems composed of these components.

In this context of architecture-enabled interoperability, the UCS Architecture supports these Office of Secretary of Defense (OSD)-stated high-level business objectives:

- Acquisition flexibility for control segment subsystems & components;
- Cost control of GCS development and maintenance;
- Innovation across and at all levels of industry;
- Reduced integration time for new capabilities;
- Reuse across Service and Joint UAS programs where appropriate.

The UCS Architecture may be employed alongside other Government Open Architecture initiatives that address Common Operating Environments, Networking/Communications and Human-Computer Interfaces, or it may be used to properly 'break open' previously closed and proprietary systems.

INTRODUCTION

Over the last decade, the US Department of Defense (DoD) has accrued great benefit from the use of UAS for both peacetime and wartime operations. This success has ignited a significant increase in the number of UAS being planned and procured. The future, independent of commercial applicability of these systems, looks to see an exponential increase in the quantity and diversity of UAS applications. Traditionally, each UAS was procured as a fully vertically integrated, vendor-specific solution, consisting of the air system, ground station, communications channels, encryption technologies, and payloads. These single-system variants are typically "closed" systems utilizing monolithic architectures or proprietary interfaces throughout

[*] Neya Systems, LLC, 12330 Perry Hwy, Suite 220, Wexford, PA 15090, paragb@neyasystems.com

[†] Office of Secretary of Defense, Pentagon, Washington DC, rich.ernst@osd.mil

[‡] Northrop Grumman, Inc., 15160 Innovation Dr, San Diego, CA, 92128, kerry.fisherkeller@ngc.com

[§] General Dynamics AIS, 8800 Queen Ave S., Bloomington, MN, 55431, douglas.gregory@gd-ais.com

[**] High Assurance Systems, 115 44th St, Manhattan Beach, CA 90266, rob.hoffman@highassure.com

[††] Dynetics, Inc., 1002 Explorer Blvd, Huntsville, AL 35806, ann.jennings@us.army.mil

[‡‡] Verocel, Inc., 234 Littleton Road, Westford, MA 01886, romanski@verocel.com

[§§] Raytheon, Inc., 22110 Pacific Blvd, Dulles, VA, 20166, bschechter@raytheon.com

[***] Real Time Innovation, Inc., 385 Moffett Park Dr, Sunnyvale, CA 94089, gordon@rti.com

the system architecture with tight interdependencies between the system software components. Single source procurement ensures these tightly coupled components work together, but guarantees expense when adding incremental capabilities, while limiting the source from which these enhancements can be procured. As the number of new UAS programmed in the Service budgets continues to increase, the magnitude of RDT&E requirements for development has skyrocketed.

For example, in order to improve situational awareness capabilities to take advantage of incremental improvements in technology, system acquisition program managers may have to recapitalize their entire inventory of GCS. This not only includes fielded assets, but also includes systems integration labs, flight test assets, backup inventory, initial qualification training assets and continuing training assets. Given the potentially large number of GCS', the cost of upgrading these closed systems to add new capabilities can quickly rise. Again, this is exacerbated given the variety of UAS systems being considered in future Defense planning

In addition to cost, the current acquisition and development approach has resulted in a number of unfavorable characteristics that impede progress and adoption:

- Reliance on large primes and vertical integrators who have little motivation for controlling costs and managing schedule.
- Lack of re-usability, resulting in RDT&E dollars being inefficiently utilized on repeated development of similar technology for different platforms.
- Difficulty of upgrading and enhancing capability, due to the proprietary nature of UAS.
- Inability to readily leverage research and development conducted at small businesses and non-traditional UAS system providers.

In order to enable future capability upgrades through competition without recapitalizing the fleet, the GCS must be "broken open" into a modular and scalable architecture that can be implemented according to individual program needs. The architecture needs to be defined to the level that allows for acquisition flexibility at system, subsystem, component, application, and service levels. It must enable innovation at all levels of industry--UAS & non-UAS domains, especially at the subsystem and component/application/service levels, allow for reuse across Service and Joint UAS programs where appropriate, and be scalable across systems such as a Scan Eagle laptop, a Gray Eagle UGCS, a Global Hawk fixed facility, and a ROVER.

STANAG 4586 took the first step towards interoperable control segments, documenting a standard for unmanned aircraft to GCS message level command and control interoperability. This is the first step towards enabling GCS to control and monitor multiple types of unmanned aircraft, improving overall cost by reusing GCS, and enabling competition at the system level for complete GCS solutions. It also allows for command and control to be instantiated separate from traditional "ground" control segments (e.g. Army Apache control of Medium Altitude Long Endurance UA).

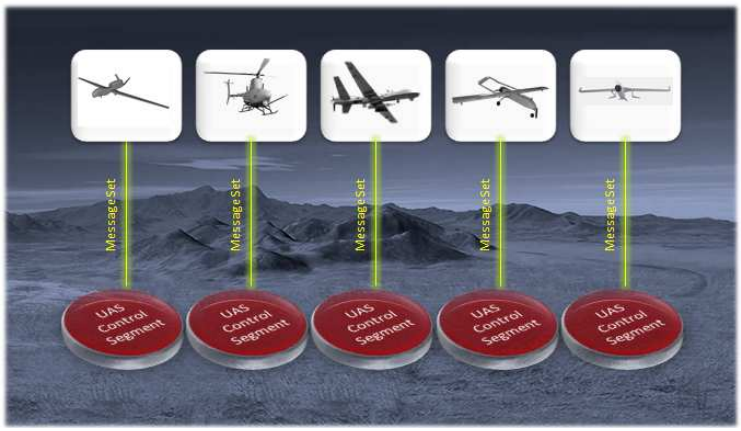


Figure 1: Standards-based interconnections but closed architecture and business models.

A single common control system, however, is not the answer to the RDT&E, interoperability, or production problems. This solution reduces competition and makes upgrades slower and more costly. One Service may have a particular need another does not, and in order to stay common, the sister Service would be forced to upgrade to a capability they do not want or need. In addition, each UAS has different missions, concepts of operation, physical requirements, deployment requirements, etc. Mission effectiveness mandates a tailored Human Machine Interface optimized through proper Human Systems Integration processes, considering the mission, information needs, and operator qualifications.

UCS OVERVIEW

Background and History

Through an acquisition decision memorandum signed 11 February 2009, OSD Acquisition Technology & Logistics (AT&L) directed the Services to develop an interoperable architecture for command and control of Unmanned Aircraft Systems Group 2 through Group 5. The architecture defines a common, open, and scalable infrastructure that supports flexible integration of disparate ground control system services across a variety of technology and topology deployment scenarios.

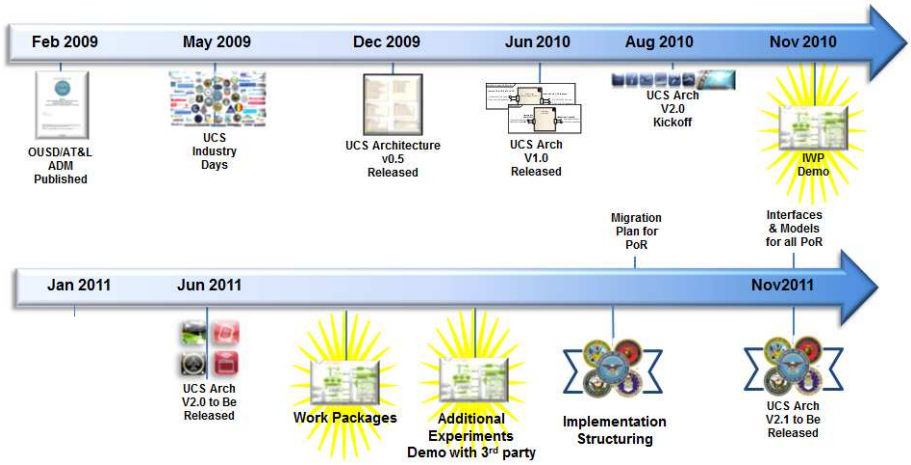


Figure 2: UCS Architecture Timeline

In response to this, Version 1.0 of the UCS Architecture was officially released in June 2010. The program of work for Version 1.0 developed the objectives for the architecture, determined the appropriate level of architecture required, defined what interoperability and methods to achieve it, and documented the architecture design.

The objectives for Version 2 and Version 2.1 of the UCS Architecture are:

1. Identify and define additional system and equipment use cases;
2. Complete definition and validation of the Application Architecture Domain Model;
3. Complete definition and validation of the Application Architecture Data Model;
4. Develop reference implementations of the UCS Architecture, perform and embody lessons learned in the architecture and architecture guidance;
5. Develop an Information Assurance Case and System Safety Case based on the Reference Architecture;
6. Update the UCS Architecture Quality Management System;
7. Update the requirements for the UCS Architecture tool environment;
8. Define mechanisms by which programs of record may demonstrate compliance with the requirements and intent of the UCS Architecture.

UCS Working Group Structure

The UCS Architecture is being developed by the UCS Working Group (UCS WG), a technical standards working group that operates in accordance with Public Law 104-113 (the National Technology Transfer and Advancement Act of 1995), and in accordance with the Executive Office of the President, Office of Management and Budget (OMB) Circular A-119. The UCS WG is organized into a set of subcommittees addressing functional areas, and is further decomposed into Task Groups that focus on specific technology areas. The UCS WG structure is shown in Figure 3.

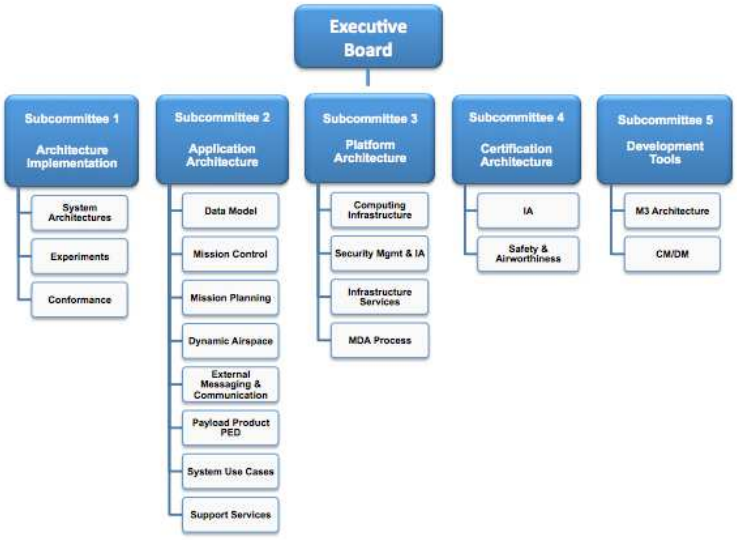


Figure 3: UCS WG Structure

The UCS WG has a structured, formal process for managing Configuration Items such as documents and models, along with a process for requesting and adjudicating engineering Change Requests. In addition, all Configuration Items go through a multi-step balloting process, beginning with balloting at the Task Group level, and culminating in a ballot at the Executive Board level. These processes are modeled after similar processes using in standards bodies such as the Society for Automotive Engineers (SAE), Object Management Group (OMG) and other similar groups. The UCS WG is designed to form the foundation of an enduring organization, to ensure continued support for the UCS Architecture.

UCS ARCHITECTURE OVERVIEW

Architecture Meta-Model and Ontology

The UCS approach for architectural governance is to utilize architecture best-practices while tailoring the governance for aspects of the system that are unique to the needs of unmanned systems. UCS governance is an evolving process, intended to address differences in control segment complexity, program phase, and respond to feedback from industry as the enduring UCS organization evolves with technology and acquisition models. Governance identifies key elements of the UCS reference architecture based upon DoD policy, open standards, and provides conformance guidance for users of the UCS architecture.

To meet the needs of a diverse UAS industry, development of a standard UCS reference architecture requires consensus on a common taxonomy and ontology of concepts for the Working Group and user community to normalize architecture artifacts and develop interoperable services for UAS. Ontologies are used to represent *semantics* and *knowledge* and makes explicit the implicit knowledge in a domain. There are two essential components of any ontology; a vocabulary of terms that refer to the objects of interest in a given domain and specification of meaning for the terms and concepts. The UCS ontology realizes common knowledge sharing amongst agents – in this case UAV Control Segments (UCS).

It is recognized that every UAS POR has the "best" architecture for its particular requirements. Each architecture uses similar business processes such as mission planning or payload control, but the different terms used and the functional organization of each UAS architecture leads to lack of interoperability. These "stovepipe" architectures limit exchange of battlespace information during operations and prevent code and component reuse across programs leading to duplicative development costs for dissimilar GCS. An example of this is the mission planning applications used for operating the USAF MQ-1/9 Predator/Reaper system are *semantically* incompatible with mission planning applications used by the USAF RQ-4B Global Hawk despite having several functional similarities. Joint mission planners such as the Joint Mission Planning System (JMPS), developed for manned aircraft address these issues but extension of JMPS to unmanned aircraft to date is still limited because of the need to specify all contingency actions for UAS and the process of funding upgrades to the existing UAS PORs.

An important development in architecture practice has been the emergence of standards for architecture description, principally through the adoption by ANSI and the IEEE of ANSI/IEEE Standards, Recommended Practice for Architectural Description of Software-Intensive Systems (ISO/IEC 42010:2007). One of the aims of this standard is to promote a consistent, systematic approach to the creation of architecture descriptions. UCS has chosen to adopt portions of the ISO/IEEE 42010 framework with an OASIS implementation of a SOA reference foundation. The current draft revision of ISO/IEC 42010:2007 introduces a formalization of architecture framework within the ontology underlying the earlier ANSI/IEEE Std 1471-2000 standard. Figure 4 depicts concepts based on best practices pertaining to architecture descriptions.

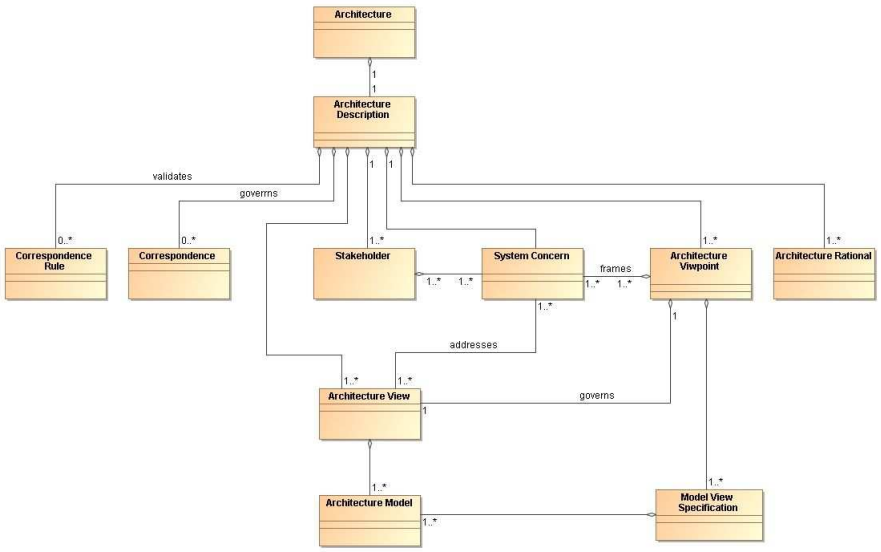


Figure 4: Architecture Description Meta-Model from ISO 42010

Architects participating in the UCS development effort are familiar with best practices from similar DOD UAS and commercial architecture software but the data names used and sometimes the overall approach varied based on past experience, organizational culture and customer convention. The meaning of "service" in a Service Oriented Architecture (SOA) had a different definition by each UCS architect depending on system viewpoint and areas of concern. This process led to definition of the initial domain model of the UCS enterprise and scope of the UCS effort. Addressing the multiple ontology situations led to the adoption by UCS of four industry standards. Ontology for the UCS SOA is now defined using UML® and UCS profiles of the Object Management Group (OMG®) SoaML® specification. The UCS Architecture, where practical, conforms to ISO/IEC 42010, the OASIS Reference Foundation for SOA and The Open Group SOA Ontology technical standard.

An additional benefit for the acquisition community is that use of a standard architecture description simplifies comparisons of control segment architectures based on the UCS Reference Architecture. The UCS process of reviewing existing architectures from the PORs illuminated difficulties in comparing and reuse of components from UAS architectures as the descriptions are provided in a different formats, types of

documents and models. Some programs have use cases and activity diagrams; others provide DoDAF views, and still others only provide of concept of operations (CONOPS) documents such as day-in-the-life descriptions. The use of a standard architecture description template will promote composability of capabilities and reduce overall DOD costs in procuring future UAS control segments.

Architecture Structure and Elements

The UCS Architecture represents an evolution from an Open System Interconnection (OSI) model of interoperability to an Open Architecture (OA) model based on an Open System Environment (OSE) in which applications may be acquired and integrated within a service-oriented architecture. An OSE requires an Open Infrastructure, an Open Acquisition Business Model, a Roadmap (Technical and Operational) and an Organization (the UCS Working Group) that can support and maintain these items.

An Open Infrastructure has several characteristics. It has an Open Data Model that is rigorously defined, rigorously described, and fully discoverable. The Data Model must be fully published, and based upon an abstraction that is broad enough to define the full domain of the system. An abstract Data Model enables one to model the full breadth of possible implementations, while also defining repeatable interoperable mappings between these possible implementations. First and foremost, the abstract Data Model is the foundation for Interoperability defining the semantics of all interoperable implementations.

The second characteristic of Open Infrastructures is that they are based solely on Open Standards, and they are flexible. Leveraging Open Standards does not limit differentiation, innovation, and competition and ensures a commodity infrastructure. Flexibility is as important as Open Standards because there is no one technology that is sufficient for the range of behaviors necessary for an enterprise level infrastructure. This flexibility is achieved through the use of Architectural "Super-Patterns" when designing and building the Infrastructure Architecture. The super-patterns for Organization, Topology, Interaction Paradigms, and Communication Patterns allow the construction of the other patterns. This enables flexibility as a base characteristic of Infrastructure, not as a "bolt-on" afterthought.

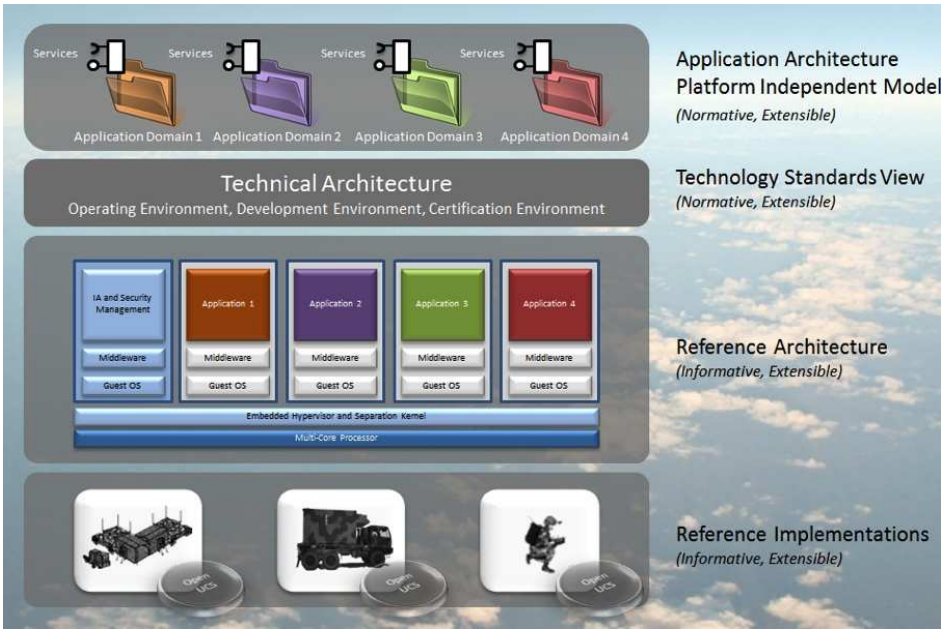


Figure 5: UCS Architecture Component View

The Technical Reference Model (TRM) for the Component Architecture is shown in Figure 6 and is based on the IEEE POSIX Open System Environment (OSE) Reference Model (IEEE-Std-1003.0-1995) and the SAE Aerospace Generic Open Architecture Framework Standard (SAE AS4893). The UCS Architecture comprises:

1. Application Software Entities within an Application Architecture (normative);
2. Application Platform Entities within a Platform Architecture (normative);
3. One or more Reference Architectures for system implementations, including External Environments (informative).

The Application Architecture is expressed as a Platform Independent Model (PIM) in a deployment-neutral development environment, such that it can be maintained independently of operating environment specifics. If required, this allows the Application Architecture to be deployed on different platforms, such as on legacy systems or systems that are highly optimized for performance or resource consumption. The deployment process and reuse process may therefore require the generation of platform-specific transforms and code for each UCS.

The Application Architecture PIM is organized into domains (subject matters) that are associated with services. It is anticipated that the services will be extended throughout the lifecycle of the UCS Architecture. An important aspect of the UCS Architecture is a common certification environment for airworthiness, system safety and Information Assurance (IA). To facilitate this, the UCS Architecture includes airworthiness, system safety and IA views.

The Platform Architecture PIM defines requirements for (1) the common System Services needed to support the Application Architecture PIM, (2) the mapping to supported underlying middleware and OS technologies, (3) the underlying User Partition, Separation Kernel, and System Administrator Partition Requirements (if an OSE Partitioning System is implemented), and (4) System Configuration Data. This is done to ensure semantic interoperability of the developed services.

The External Environment and External Environment Interface (IEE) are not defined in the normative architecture; however one or more informative Reference Architectures are defined, which include the External Environment. The External Environment includes Human-Computer Interfaces, Data Access, and External Communication Networks.

MDA Process and Architecture Transforms

The UCS Working Group has selected the Object Management Group (OMG) Model Driven Architecture ™ process as one important means to achieve and maintain interoperability of compliant UCS systems. MDA™ separates the UCS business logic and data contained in the application architecture from the underlying technology of the application platform and the software runtime architecture. In MDA, the application software is first expressed as a set of Platform Independent Models (PIMs), which are later transformed into Platform Specific Models (PSMs) incorporating platform choices and software runtime architecture choices.

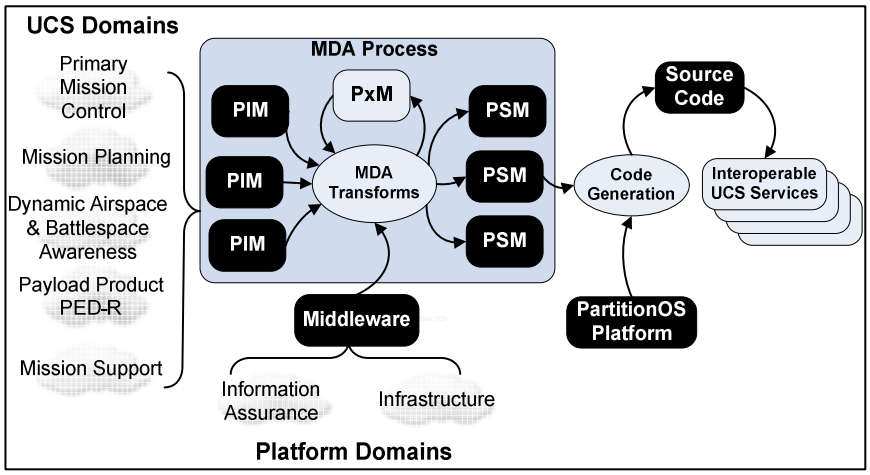


Figure 5: UCS Architecture MDA Process

Code generators and automatable processes transform PSMs into source code ready to compile, test, and deploy. Well-defined PIM-to-PSM transformations provide key benefits. First, the business logic formalized in the PIM is highly reusable from one execution environment to another. Second, once the PIM is instantiated as program source text through an automated and repeatable process, the code is highly maintainable and extensible through the PIM. Third, a system can be composed and integrated at the PIM level, resulting in early defect detection and resolution. Last, the PIM results in a more formal specification of the application business logic, data, service interfaces, and semantic behaviors that facilitate affordable certification of system safety and airworthiness. In the UCS Architecture, the MDA process ensures that ***interoperability is built-in*** to all of the PSMs derived from a common set of UCS service and data PIMs.

UCS ARCHITECTURE ELEMENTS

Application Architecture

The Application Architecture PIM is comprised of System Use Cases, Data Model, and Domain Model for UAS specific functions and capabilities. The software architecture is expressed as an open, vendor-neutral Platform Independent Model (PIM) using a Common Development Environment that complies with the OMG Meta-Object Facility (MOF™) and XML Metadata Interchange (XMI™) standards.

Use-Cases: Within the UCS Architecture, System Use Cases set the requirements and scope of the architecture and are being derived from Program of Record material from existing Army, Navy, and Air Force Programs. Use Cases represent the operational business processes required to perform UAS missions, as well as the business use cases needed for the UCS architecture to meet stakeholders' goals. They focus on describing how mission tasks are performed to help the system and software developers create applications and software services to achieve those tasks and also capture the quality attributes defined for the UCS architecture, such as interoperability, extensibility, and composability. The detailed description of the system is captured in the Use Cases Textual Descriptions and UML diagrams (Use Case, Activity, and Sequence). The stakeholders are represented as actors. For example, the Use Case diagram below provides the top-level (Level 0) System Use Cases.

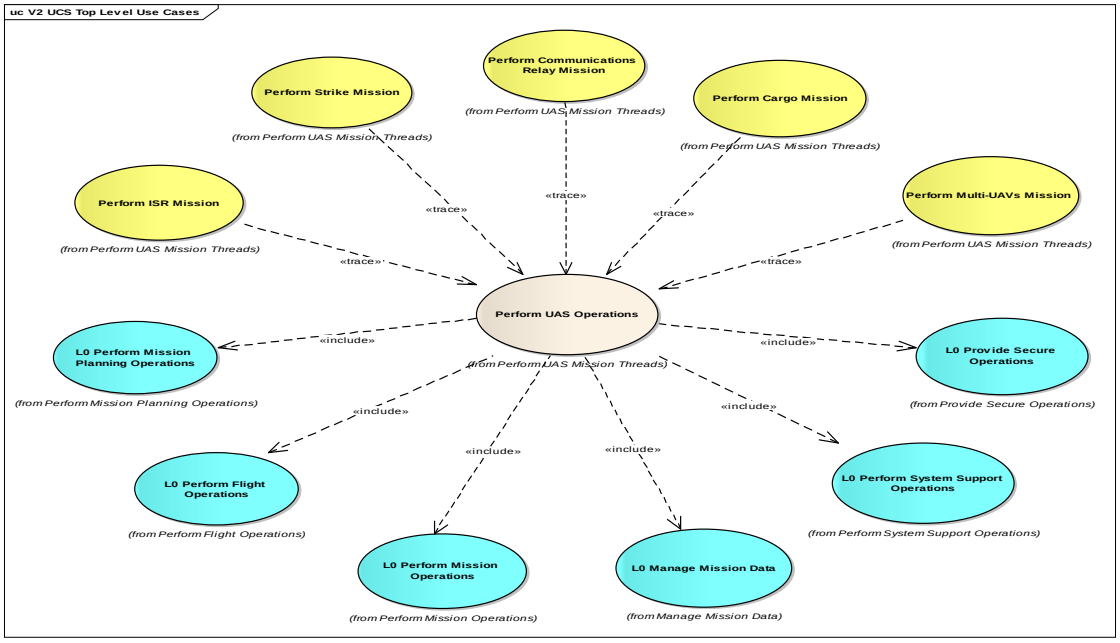


Figure 7: UCS System Use Cases (Level 0)

The UCS System Use Cases provide the system-level operational analysis that establishes the scope and defines the UCS system boundary through the operational activities and information exchanged between the system and associated actors (human and other external systems). Each System Use Case represents an

operational significant capability that the UCS system provides to its users. The UCS System Use Cases are further decomposed into lower-level Use Cases as defined in the following table.

Level	*Perspective*	*Description*
Level 0	System	Apply to the system as a whole, not just a single domain. Describe how the domains interact with each other. Each domain is represented in the sequence diagram as a swim lane. Each domain is considered a black box.
Level 1	Domain	Top-level Use Cases for a single domain that touch each sub-domain. In the sequence diagram for a Level 1 Use Case, each sub-domain would be represented as a swim lane.
Level 2	Sub-Domain	Use Case for each sub-domain. In the sequence diagrams for these Use Cases, the individual services will appear as the swim lanes in the diagram.
Level 3	Service	A single Use Case per service.

Data Model: By analyzing Use Cases, a Data Model is created based on real-world entities (such as air vehicles, payloads, communication links, etc). This approach provides a stable yet extensible architecture and contributes greatly to a number of System Quality-Attributes such as Interoperable, Portable, Efficient, Scalable, and Predictable. The UCS Data Model provides a platform independent representation of information shared between UCS Domains and Actors. This representation is comprised of a Logical Data Model, which is a self describing model of information required by all UCS Domains, and a set of Interface Classes, which represents collections of information as used by the Domains in their internal and external interactions.

Domain Model: The UCS Architecture is partitioned into Domains based on subject matter expertise to provide easier navigation and understanding. System-level Use Cases are decomposed into lower-level Domain-specific Use Cases. The Use Cases are analyzed to produce Service definitions with interfaces derived from the Data Model. The service interfaces are defined using the OMG Service Oriented Architecture Modeling Language (SoaML). There are currently six domains defined within the UCS Architecture: Primary Mission Control, Mission and Task Planning, Dynamic Airspace, External Messaging and Communications, Sensor Product Processing, Exploitation and Dissemination, and System Support.

Primary Mission Control Domain

The Primary Mission Control (PMC) domain is responsible for all aspects of receiving and accepting mission tasking from command authorities and successfully executing that tasking. The following sub-domains are included with the PMC domain:

Sub-domain	*Description*
Comms Relay Payload Management	Responsible for all aspects of configuring, controlling and providing the status of a generic communication relay payload that may be installed on an unmanned platform.
Cargo Payload Management	Responsible for all aspects of configuring, controlling and providing the status of a generic cargo payload that may be transported on an unmanned platform.
Communication Management	Responsible for all aspects of configuring, controlling and providing the status of all communication/data link systems utilized to communicate between the UAV Control System (UCS) and unmanned air vehicle(s).
Effects Management	Responsible for all aspects of configuring, controlling and providing the status of all payloads that result in some desired effect, e.g. putting a kinematic weapon on a specified target.
Mission Management	Responsible for receiving, evaluating, accepting and then executing a tasked UAS mission.

Sensor Management	Responsible for all aspects of sensor payload management to include the configuring, controlling and providing the status of all of the sensors that are part of the UAS.
Survivability Management	Responsible for all aspects of survivability of an air vehicle and its payloads in the course of executing its assigned mission
UAS Health & Status Management	Responsible for providing high level aggregation of UAS regular and periodic status collected from UAS components
UAS Resource Manager	Responsible for managing various resources required for the execution of a UAS mission
Vehicle Management	Responsible for all aspects of the configuring, controlling and providing the status of all of the unmanned air vehicles (UAV) that comprise a UAS
Vehicle Message Layer	Responsible for providing the services required to communicate with all elements of the unmanned platform to include the air vehicle and its subsystems, payloads, sensors, and effects

Mission and Task Planning (MTP) Domain

The Mission and Task Planning (MTP) domain is responsible for responsible for the management of tasking data as well as providing capabilities and services for the purpose of generating pre flight mission plans and supporting in flight dynamic mission planning activities. The MTP domain manages the receipt, processing, and routing of tasking data such as ATO, ACO, and ACTDF within the UCS. Pre flight mission plans are typically generated by mission planning programs of record including JMPS, MPS, AMPS, and PFPS or within the ground control station by operators. Dynamic mission planning is performed in response to receipt of new tasking, by operator request or through monitoring and detecting changes within the operational environment leading to re-calculation of active mission routes. The following sub-domains are included with the MTP domain:

Sub-domain	***Description***
Mission Planning Management	Manages incoming tasking requests and the subsequent parsing and routing of task messages to and from pre-flight mission planning systems and dynamic re-planning functions and services
Mission Information Management	Responsible for managing all Mission Information required for pre flight, dynamic mission planning, and mission report generation.
Route Planning	Responsible for generation of basic navigation plans, emergency and contingency plans, calculating route performance, and survivability
Payload Planning	Responsible for payload/sensor plan generation, dissemination of the payload data within the vehicle to a processor, recorder, and/or data link as well as communication plan generation.
Communications Planning	Responsible for the communications planning for the vehicle, payload, and weapons data links. This also includes communiations relay and voice relay to ground units and ATC
Effects Planning	Responsible for generating products in support of weapon releases, Electronic Warfare (jamming), and other types of counter measures
Airspace and Battlespace	Reserved for future scope but will support such products as de-confliction

Dynamic Airspace Domain

The Dynamic Airspace (DA) domain provides the situational awareness (SA) information needed to control, manage and protect the air vehicles. This SA information includes Battlespace information such as Blue Forces, Red Forces and other entity data, along with weather, terrain and relevant earth features. It also includes the air picture information to support collision avoidance and to maintain air vehicle separation. The following sub-domains are included with the DA domain:

Sub-domain	*Description*
Battlespace	Collects data on the entities in the Battlespace, including force positions, tracks, threats, and threat envelopes.
Air Traffic	Provides information on air entities, Required Navigational Performance (RNP), IFF information and ATC data.
Environment	Collects and publishes data on weather and other meteorological information, and this sub-domain also publishes requested terrain and map data
Management	Provides the DA workflow coordination. Provides the common interfaces to the infrastructure storage services. Manages the interfaces to the outside data.
Common Operational Picture	Placeholder for future development to meet the UAS Roadmap objectives of "autonomous and collaborative" Battlespace awareness

External Messaging and Communications Domain

The External Messaging and Communications (EMC) domain is responsible for managing the external interface between the Unmanned Aircraft System Control Segment (UCS) and the other Command, Control, Communications, Computers and Intelligence (C4I) systems. The EMC domain uses a prescribed set of messages to relay information to external systems, as the name implies, via native data paths. Examples of such data include Internet Relay Chat (IRC), United States Message Text Format (USMTF), Cursor on Target (CoT) messaging, etc. EMC does not manage the link between the Unmanned Air Vehicle (UAV) and the UCS, as that type of communication is handled by Primary Mission Control (PMC) domain. The following sub-domains are included with the EMC domain:

Sub-domain	*Description*
Tactical Messaging	Responsible for the short term tactical mission objectives or ad-hoc tasking such as fly the AV to a waypoint, slew the camera then report back the location of red forces to a mission commander.
Collaboration	Responsible for collaboration such as Text, Voice chat and White boarding. These messages are characterized by digital interactions between the UCS and other external systems – external or internal UCS(s) or any other number of systems.

Sensor Product Processing, Exploitation and Dissemination Domain

The Sensor Product Processing, Exploitation and Dissemination (PED) domain is responsible for receiving sensor output into the control segment, refining, enhancing or reformatting the data, then archiving and/or disseminating internally within the UCS and externally as appropriate. The following sub-domains are included with the PED domain:

Sub-domain	*Description*
Capture	Provides a short term cache to allow these data formats to be completely assembled
Dissemination	Processes the sensor products at a low level including format translation.
Exploitation	Modifies the sensor product as appropriate to meet mission requirements.
Archive	Provides persistent storage and retrieval of sensor products
Management	Allows automated processing to be configured for each subdomain service and ad hoc configuration changes during a mission.

System Support Domain

The System Support (SS) domain is responsible for providing Integrated Logistics Support (ILS) driven services that enable the operation and maintenance for current and future fielded UCS systems. The following sub-domains are included with the SS domain:

Sub-domain	*Description*
UCS Maintenance	Provides health and status monitoring and maintenance for the UCS software and hardware.
Logistics	Addresses the activities required to support a fielded product. This includes spares inventory, technical documentation, availability analysis, and in-service evaluation and certification.
Mission Support	Responsible for obtaining data from external sources that is needed in support of a given mission including pre and post mission activities
Training	Fulfills the need to provide training to the operators and other users of a UAV control segment
Administrative	Addresses pushing applications and services to the Ground Control Stations. The AF establishes the governing policies and pushes software updates to keep the system relevant and compliant. The Administrative Function subdomain also establishes failover and fault tolerance policies in the event of anomalous hardware/software behavior

Platform Architecture

The application domains and related services execute on a compute platform. That platform's potential capabilities are defined through a set of *infrastructure services* and a *system configuration management and governance structure*. Because the UCS Reference Architecture spans from Type 2 to Type 5 UAVs across fixed, transportable, and mounted environments, and to avoid constraining innovation, the definition of the compute platform and infrastructure services is critical to system interoperability goals. Furthermore, when required, the compute platform must support the safety and IA requirements as specified in the architecture guidance. For example, the Reference Architecture leaves to the implementer whether the control segment hardware consists of a small single board computer running multiple applications up through a large installation of multiple cooperating computers. Instead, and in support of the Quality Attributes, the infrastructure services conform to a few central principles: it is service-oriented, it enables sharing of compute resources across multiple applications across mixed safety and security domains (with safety and security requirements derived from the environment), the architecture and the safety and security cases are well-aligned to reduce evaluation and certification cost, and so on.

These infrastructure services are organized into six services areas per the following table:

Sub-domain	*Description*
Collaboration	Communication and messaging such as chat, email, voice, and whiteboarding.
Communication Services	Service abstraction of the information distribution infrastructure. Middleware, message and communication pattern quality of service
Data Management	Services that provide data-lifecycle management capabilities such as persistence, data mapping and transformation, data synchronization, and file transfer
Networking	Network management, monitoring, and QoS
Security, Safety and IA	Access control, identification / authentication / authorization, malware detection

SOA Runtime Infrastructure	Service registry, discovery, orchestration, application management
System Management	Configuration, capacity and availability management, health and incident management, time

Certification Architecture

In addition to defining the domains, use cases, service interfaces, and data model, development of a Certification Architecture is critical to successful deployment. This is similar to the FAA requirement that a passenger-carrying airplane flying in the National Air Space is linked to the Air Traffic Control system by a communications link. Federal Aviation Regulations Require that the software on the airplane satisfy the objectives of DO-178B. A system safety assessment process is carried out in accordance with ARP 4761 to assess the Design Assurance Level of the systems on the aircraft. These are determined by a combination of the severity of the failures (Catastrophic, Severe/Major, Minor, etc) and their probability. The design assurance levels A, B, C, and D are used to determine the objectives described in DO-178B that must be satisfied before system controlled by software is certified.

Ground based systems undergo a similar assessment, but the approval document used is DO-278. This has an Assurance Levels ranging from AL1 to AL5. Safety Critical airborne systems undergo very strict verification and validation processes because they often have a direct impact on the safety of the aircraft. They provide direct control over many of the aircraft functions. The ground-based systems, like Air Traffic Management (ATM) provide information to controllers, who can then make decisions using several sources of information, including direct communication with pilots. Before a new ATM installation is used in earnest, it may be run in parallel with an already approved system, and it's operational capabilities may be assessed for a long period before the decision is made to fully trust it.

An Unmanned Air System linked to its Control Segment is similar in many ways to the aircraft/ATM combination, but there are some significant differences.

- A UAS to UCS coupling is much "richer" in content, and a tighter coupling is maintained over secure communication links.
- In unplanned and extraordinary situations, the vehicle must accept new control decisions being made in the UCS or revert to preplanned reversionary controls.

This makes it impossible to have two UCS's providing concurrent control over a flying UAS, in the same way that and aircraft/ATM system can operate. The UAS/UCS combination needs have software certified using the same guidance document, and to make it easier to enter the NAS, DO-178B has been chosen. For any UAS/UCS combination an assessment would need to be made to determine the DAL of the systems responsible for continued safe flight and landing.

System and Component Integration

The UCS consists of many application programs that implement Components containing services, which are composed to provide all of the necessary GCS Capabilities. The application programs may run on one of several computing platforms. The computing platforms may be general-purpose operating system based systems for applications that are not safety critical, or they may be embedded style computers with real-time operating systems. Each of these computers may be able to host one or many applications, by providing a virtual processing environment (called a partition) which isolates each application by controlling it's use of shared resources. A partition may provide tasking and scheduling resources as well as standard programming libraries making easier to develop applications. The combination of partition and application software is combined and will have a DAL which is the lower of the two components. By using a Real Time Operating System (RTOS) that has certification evidence compliant with DAL A, and which is capable of providing robust partitioning, all applications hosted by the RTOS may be developed to any DAL that they have been assigned.

An application developer may provide a complete certification package together with the application software consisting of all of the lifecycle data and documentation required by DO-178B. Provided the inter-

faces and the underlying computing platform are common, such an application may be moved between different instantiations of the UCS architecture, carrying forward much of the certification credit.

Applications and Services

The UCS architecture provides a set of models consisting of a large number of services. The services have defined interfaces and specifications that describe what functionality they provide. The services may be developed by many different organizations and may be cataloged for easy reuse on many different programs of record. Services are then combined to form an application running in a partition. As the application has been assigned a DAL, all of the services that have been combined will have the DAL ascribed to the application.

Not all applications will be safety critical, so not all services will be safety critical, but for those that have a DAL, the DO-178B lifecycle data will be required. This means that development and verification lifecycle data will be required. This includes requirements, design, code, test, test results, coverage analysis, and all applicable reviews and analyses performed in accordance with a set of processes documented in an integrated set of development and verification plans. The the lifecycle data may be produced using a waterfall development model, spiral development model or even may be reverse engineered, providing that ultimately it satisfies the objectives of DO-178B. This data may be developed and delivered as a certification package, or set of packages, which may be assembled to provide evidence suitable for assessment and acceptance. Tests should be automated, test scripts should be provided and baseline results captured so that any service or set of services may be retested.

An application constructed from many services will need a reliable, repeatable and automated build process. Depending on the testing strategies and tools used, the services or groups of services will be tested before or after the application build. Testing after a build requires a more sophisticated approach, but provides credit towards application integration testing. Unit testing of services before application build may be easier as there are many tools that support this approach.

Depending on the integration approach adopted, the certification package for an application may be constructed by assembling all of the certification materials developed for the individual services. Hardware/Software integration testing is still required and system testing will be required at the application and integrated UCS level.

Presenting the certification arguments.

Subcommittee 4.2 within the UCS WG is developing certification cases for safety and security. For the safety case, sets of certification goals are being identified for a UCS platform independent model. The goals are decomposed using subgoals, strategies, and other components that are represented using the Graphical Structuring Notation (GSN). This notation is being used to present an argument that links a goal to a solution through several hierarchical elements. As the GSN diagrams reflect the composition and incremental certification strategies outlined above, it is necessary to introduce possible architectural compositional components that could be used to construct a specific UCS. It is not the aim of subcommittee 4 to provide a mapping to a platform specific model, but is is necessary to provide a mapping between the goals and solutions to an abstract architecture. This abstract architecture is being described using the Architecture Description Language (AADL).

AADL may be used to describe a system, with processing elements, subsystems, partitions, and application level tasks and task groups as components. These components may be interconnected using logical data busses. GSN diagrams may be linked to AADL diagrams to show how safety arguments may be mapped onto possible architectures. These diagrams do not describe the details necessary to complete a safety case, but they provide a framework a program of record could use to formulate a certification strategy.

Security Certification Architecture

The security certification process for a UCS conformant to the UCS WG reference architecture is traditional—meaning that the customary Certification & Accreditation (C&A) processes will be used at the system level (that is, at the level of at least the UCS and, depending on the certifier and authorizer, potentially at a larger system level that includes vehicles and communications). These customary processes are based on ICD 503, which replaces DCID 6/3 (elements of DCID 6/3 may remain in use for some programs), CNSS 1253, DIACAP, SABI, TSABI, program-specific C&A processes, etc.

As with safety where it is assumed that different applications may have different design assurance levels (i.e., Do-178B Level A through Level E), it is an objective of the UCS reference architecture that a conformant UCS accommodate applications processing data in different security domains, potentially ranging from TS/SCI to UNCLAS, called *two domain separation* (this standard term implies three domains, with, in the hierarchical case, A separated from B and B separated from C).

Further, as described in the section on safety certification [reference to George's section], a UCS consists of many application programs that may co-reside and run on one of several computers that may make up the entire UCS (a small UCS could run all applications on a single computer).

To support this execution of multiple applications in a single UCS with two domain separation (whether on one or several computers), the reference architecture mandates that a conformant UCS be strictly based on two core principles: *separation* and *least privilege*.

The separation principle requires:

- data separation of applications in different security domains
- strictly controlled information flows between applications in different security domains
- fault isolation so that a fault in an application in one partition (explained below) has no impact whatsoever on (independent) applications in any other partition

Fault isolation is useful independently (especially for safety), but is required for security. Consider a system where a fault in one application causes a delay of another independent application. A subversive application could cause a fault that is detected by another application due to the delay, thereby establishing a covert signaling channel.

The least privilege principle requires that a "subject" (an entity that consumes resources, such as an application or driver) must be able to access only such information and use such resources as are minimally necessary for its legitimate purpose.

The two core principles enable and imply several byproducts:

Security policy is distributed—each software component (application, driver, or middleware element) is responsible for only its own security, and in turn leaves the correct implementation of the security features of any other component on which it relies to the other component. This makes the system far more evaluatable than older, monolithic approaches, in which a large, supposedly secure operating system was responsible for all or most of the security elements of a system.

Security critical code should be isolated and minimized. This may require breaking up a larger component into, for example three (or more) components: a large component with no security critical code (and therefore having no or very low evaluation cost), a smaller component responsible for low security functions, and a very small component responsible for very high security functions (e.g., a downgrader). Doing so has the potential to dramatically reduce evaluation and certification costs.

The components and the system must be NEAT*:

Non-bypassable: there must be no information flow around a component responsible for some security feature (consider a bouncer at a club—there are no entries to the club other than through the door where the bouncer is stationed). This is a system-level requirement.

Evaluatable: security functions can be evaluated with high assurance. This is both a system-level and component-level requirement.

Always-invoked: all required security functions within a component are always invoked for every path through the component (bouncers inside the club cannot be distracted and are always checking for disturbances). This is a component level requirement.

Tamper-proof: subversive code cannot alter security data or functions. This is both a system-level and component-level requirement.

---

*. The *A*, *E*, and *T* attributes are from Anderson (1972); the *N* attribute was added in Alves-Foss, et al. (2004).

The two core principles do **not** imply implementation on a single computer; each application could run on its own computer with "air gap security" between them. However, it is expected that UCS will often run multiple applications—in different security domains—on a single computer. As with the multi-computer case, UCS reference architecture mandates that the single computer be capable of the separation and least privilege principles above.

A separation kernel (SK) is a type of specialized operating system that enables a computer to implement the two core principles. Separation kernels *partition* the resources—the processor, memory, and I/O of the computer—into *partitions* and execute each application in a partition configured so that the each application receives the resources it requires.

UCS WG guidance documents require that if a UCS uses a separation kernel, that SK must conform to the *U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness* (SKPP) (IAD 2007). The SKPP is a *protection profile*, in effect, a security requirements document spelling out a set of "security functional requirements" that an SK must meet and a set of "security assurance requirements" by which an SK can be evaluated. Protection profiles and the evaluation methodology are governed by the *Common Criteria for Information Technology Security Evaluation*, ISO/IEC 15408. Common Criteria is an international standard for defining an evaluating the security of any IT product.

(Note: the SKPP has recently been sunset by U.S. Government due to concerns about Common Criteria at high evaluation assurance levels in general and specific concerns about the SKPP, especially that security arguments must be made at the system level, not the component level (only). However, while the SKPP has been sunset, in its sunset transmittal, the Government noted that separation kernels, and the least privilege architecture that they support, continue to be sound design choices for security-critical systems and encouraged their continued use and development; it further noted that SKPP requirements may be used by Government programs seeking to obtain evidence (security arguments). There are three U.S. COTS separation kernel vendors; all remain committed to continued development, certification, and deployment of their separation kernels products.)

The above describes the security architecture required by the UCS reference architecture. As noted above, while the security certification arguments will be developed using traditional C&A processes, and must be at the system level, Common Criteria artifacts for a separation kernel can be submitted as part of that process.

The principles of separation and least privilege architecture enable a "divide and conquer" approach to system development for security. Independent components can be developed, integrated, evaluated, and certified; refreshed and re-evaluated and re-certified; and configured and operated independently and asynchronously (while also taking into account system-level concerns). The results of these choices for the security architecture of UCS will be lower original development and C&A cost and time, lower cost of change, including especially through preservation and reuse of artifacts for C&A, and lower operational cost through more secure and more easily configured data sharing exactly as required.

IMPLEMENTING THE UCS ARCHITECTURE

Architecture Implementation

The Architecture Implementation Subcommittee is concerned with specific implementation issues related to GCS that will be built using the UCS Architecture. It is composed of three Task Groups: 1) Reference Architecture, 2) Experimentation, and 3) Conformance. Each is described below.

Reference Architecture

The Reference Architecture of the UCS Architecture is represented by a series of Deployment Architectures. A Deployment Architecture is a canonical example of a system that is compliant with the UCS Architecture. It is not a fully fleshed out architecture as would be done on a Program of Record (POR), instead it is intended to document the views of an architecture to the extent necessary to reveal how the quality attributes are addressed, and to provide guidance to a reference implementation. The Deployment Architectures will adhere to conformance measures developed by the UCSWG. The Deployment Architecture is not mandated, but does suggest best technical approaches and key architectural decisions that would typically be embodied in a real system. Version 2.0 of the UCS Architecture will document three such Dep-

loyment Architectures, attempting to capture practical architectural styles that meet the quality attributes of the UCS Architecture.

The UCS Working Group activities have resulted in a domain model that describes the areas of concern for a control segment designed for unmanned aerial vehicles and how those domains relate and interact. It contains a data model that describes the data that is shared by the domains. The model describes services that are exposed at the lowest domain level to allow data to flow between domains. These constituents form a reference model that is the basis for a particular Deployment Architecture. The following diagram shows the relationships between these items.

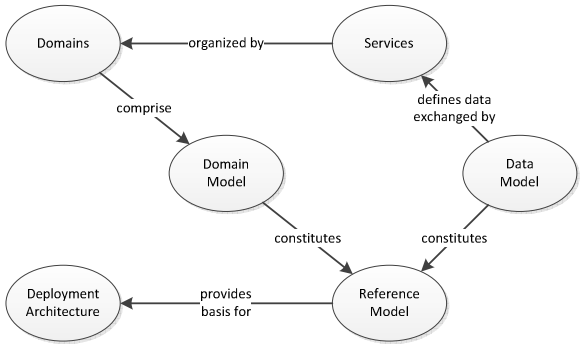


Figure 6 - Reference Model and Deployment Architecture

Version 2.0 of the UCS architecture will document three specific Deployment Architectures. These will document architectural patterns that will be applied to the reference model defined by the UCS Architecture. The reference model will then be mapped onto software elements along with the externally visible properties of those elements to describe the Deployment Architectures. Deployment Architectures selected for modeling in the UCSWG need to be congruent with the overall UCS Architecture. The Deployment Architecture needs to be consistent with the quality attributes defined in the UCS Architecture and be documented to the extent necessary to judge how well various quality attributes are fulfilled.

The various Deployment Architectures will examine differences in size-weight-and-power (SWAP), network connectivity, safety, and threat environments. This will reveal how well the UCS Architecture addresses the wide variety of possible control segment systems. Different deployment views will require different certification approaches that should be compatible with the affordable airworthiness and information assurance approach defined by the UCS Architecture. The Deployment Architectures for version 2.0 are but a few of the many possible systems that could be constructed using the UCS Architecture.

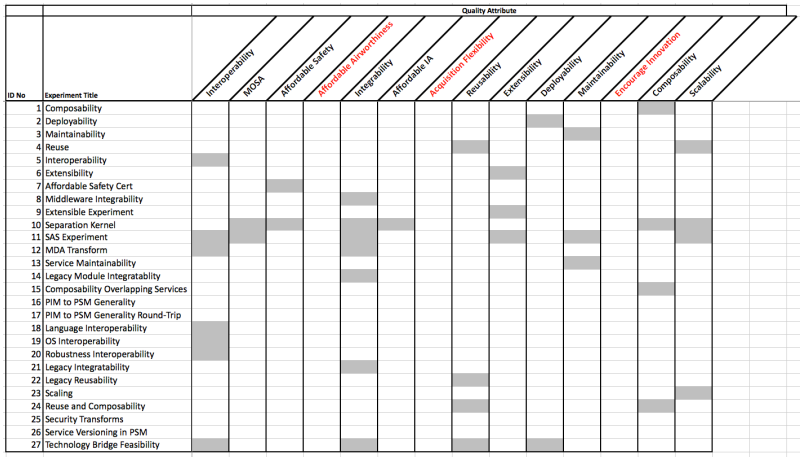
Deployment Architecture 1 - Dismounted
This system is based loosely on a two-way Remote Optical Video Receiver (ROVER) capability such as the US Army's One System Remote Video Terminal. It can receive payload products, control payloads, and exploit the data therein. It is capable of allowing the operator to designate a target with the UAV's laser designator. For purposes of this Deployment Architecture, the system is being limited to an EO/IR control capability.

Deployment Architecture 2 - Mobile
This system is transportable in something the size of a C-130 down to a mobile army shelter. It is capable of performing all UAV missions and can be completely stand-alone or connected to the GIG. This system is based on the USAF forward deployment GCSs and the Army's One System GCSs. For purposes of this Deployment Architecture, it has access to line of sight and beyond line of sight communications equipment.

Deployment Architecture 3 - Fixed Facility
This system is installed in a fixed facility with abundant power, space, and IT resources. It can be distributed across the network and is based both on the UCI CMCC model, and the USAF CONUS fixed facilities capabilities. For purposes of this Deployment Architecture it is limited to beyond line of sight communications which includes direct satellite communications and network bridged satellite communications.

Experimentation

The Experimentation Task Group is responsible for developing and executing experiments on the UCS Architecture to confirm and demonstrate adherence to a set of quality attributes, such as interoperability, scalability, deployability, maintainability, etc. To date, 27 experiments have been proposed, and down-selected to a set of 9 experiments, some of which will be performed over the second half of 2011***Figure 9*** shows a depiction of all the developed experiments, along with which of the 14 identified UCS Quality Attributes they address.



ID No	Experiment Title	Interoperability	MOSA	Affordable Safety	Affordable Airworthiness	Integrability	Affordable IA	Acquisition Flexibility	Reusability	Extensibility	Deployability	Maintainability	Encourage Innovation	Composability	Scalability
	Quality Attribute														
1	Composability													■	
2	Deployability										■				
3	Maintainability											■			
4	Reuse								■						■
5	Interoperability	■													
6	Extensibility									■					
7	Affordable Safety Cert			■											
8	Middleware Integrability					■									
9	Extensible Experiment									■					
10	Separation Kernel		■	■		■	■							■	■
11	SAS Experiment	■				■				■		■			
12	MDA Transform	■													
13	Service Maintainability											■			
14	Legacy Module Integratablity					■									
15	Composability Overlapping Services													■	
16	PIM to PSM Generality														
17	PIM to PSM Generality Round-Trip														
18	Language Interoperability	■													
19	OS Interoperability														
20	Robustness Interoperability	■													
21	Legacy Integratability					■									
22	Legacy Reusability								■						
23	Scaling														■
24	Reuse and Composability								■					■	
25	Security Transforms														
26	Service Versioning in PSM														
27	Technology Bridge Feasibility	■				■			■		■				

Figure 7: Table of Defined Experiments and Addressed Quality Attributes

We used a selection process to down-select and combine the initial list of 27 experiments into a final set of nine. The process we used is outlined below.

1) The 27 experiments were analyzed and grouped into 12 core categories, recognizing that there was some overlap and redundancy between experiments.
2) The total number of QAs addressed by each grouped experiment was noted.
3) The importance ranking of the major QA addressed by each grouped experiment was noted.
4) A subjective estimate of complexity was noted, from a rank of 1-4, with 4 being highest.
5) A subjective estimate of 'utility' was noted, which estimated the usefulness of a specific experiment in furthering the QA goals of the UCS Architecture. Utility took a range of 1-4, with 4 being highest.
6) The experiments were ranked based on utility, and a cut-off of '2' or above was used to determine which experiments would be included in this document.

The final selected set of experiments is being performed by members of the UCS WG along with the Joint System Integration Laboratory (JSIL) in Huntsville, AL. The UCS WG will evaluate the results of these experiments, and recommendations will be made to the UCS Executive Board regarding any potential modifications.

Conformance

The Conformance Task Group is responsible for defining the rules used to determine if a system or its components conform to the UCS WG Architecture. There are three different types of conformance pertinent to the USCWG:

- Development Environment (M2/M3) – conformance of the development environment to specifications defined by the UCS WG
- Certification – the system must be certified for both safety and security by an accredited organization
- System/Interface Conformance – conformance of the end products

For the purposes of version 2.0, only system/component conformance will be addressed.

System Conformance ensures that entire system is compliant to UCS Architecture and consists of conformance of the following: PIM/PSM Transformation, Service Interfaces, Infrastructure, Use Cases, Safety, and Security

System conformance goes beyond just the PIM/PSM Transformation and the service interfaces. The infrastructure must meet the requirements of the UCS Core API Standards (CAPIS) document. Safety and security must be implemented to UCS WG requirements. Finally use cases will be used to measure the level of conformance. System conformance provides a scale of conformance to allow a transition to the UCS WG, which will in part be measured by L0 use cases.

The system integrator will create a PIM which will include a marking model which is needed for the transformation. The transformation will translate the SI PIM into the PSM. This transformation will be evaluated for PIM/PSM Transformation conformance. For the PSM, the components will be implemented. Each component will have one or more service interfaces. Service Interface Conformance evaluates whether these service interfaces are implemented as defined in the PSM. System conformance includes all of the components; the transformation, the PSM and the implementation.

System and components conformance is measured against a PSM. The transformation of the PIM into a PSM affects the individual service interfaces. The transformation may result in the addition or the deletion of operations for a service. For example, the capabilities of the middleware may implement specific types of operations (e.g. Data Distribution Service (DDS) implements the delete operation). Also capabilities such as safety, security and monitoring may require new operations for each service. Since the PIM and PSM interfaces may be different, and the component's interfaces are based on the PSM, it makes sense to measure conformance against a PSM.

PIM/PSM Transformation Conformance measures the conformance of a transformation to the UCS WG MDA Process. In order to ensure that the PSM traces correctly to the UCS WG PIM, the transformation must be defined and implemented correctly. The UCS WG MDA Process defines the rules for creating a transformation. If the rules are follow in creating the system integrator specific transformation then the transformation will be conformant.

Service Interface Conformance measures the conformance of an individual service interface, which has been implemented as a component, to UCS WG Architecture. A component may implement one or more service interfaces. Each service interface on the component will be validated for conformance. Conformance for a component is defined as meeting the service interface as defined in a PIM or PSM. This includes the messages/operations and the quality of service. All messages in the PIM/PSM must be accepted by the service. However, not all the operations have to be implemented. Conformance may be to the PIM or a specific PSM. While in order to implement a service in a system, it must conform to a PSM, it is possible for a service to only conform to the PIM.

System Conformance measures the entire system against the USCWG Architecture which goes beyond just the PSM. The individual services in the system must conform to the PSM. However other requirements are defined by the USCWG including safety, security, middleware, and supported standards. System conformance will be designed to have a scale that would allow programs to have an initial score based on partial conformance and would increase over time as the design migrates to be more closely in line with the UCS Architecture. UCS use cases will be evaluated to determine system conformance because they provide a segmentation of the system that is meaningful to the end user and the acquisition community. The

majority of UCS systems will implement most of the use cases but the score can be adjusted based on the requirements of the system. Systems that are not design to be a fully UCS systems (e.g. payload receipt only systems) would also be measured on a reduced scale.

Initial Work Packages

In addition to ongoing efforts in developing the architecture, a set of experiments have been conducted to bring together the DoD Services in developing an early set of UCS Architecture compliant software services. In this multi-staged effort, UCS Architecture Version 1.0 was used as the foundation for creation of a set of services: Weather Reporting, Vehicle Flight Status, Payload Command and Control, Video Archive, Video Stream Service, Blue Force Situational Awareness, and Cursor on Target, METOC Weather, and Airfield Management. These services represent relevant and existing capabilities within currently deployed GCS. Four UAV GCS vendors, representing programs of record (PoR) from the Army, Navy, and Air Force each developed 2-3 of these services, and demonstrated integration of all of the services into Systems Integration Lab (SIL) versions of their PoR Ground Control Station. This experiment demonstrated two key results: 1) It is possible to integrate UCS-compliant software services within legacy GCS, and 2) it is possible to share GCS software services between the Joint DoD Services.

A common data bus, based on the OMG DDS standard, was used as the middleware layer. The service creation process started with the UCS Architecture Version 1.0 PIM, and a manual transformation and mapping process was used to develop a set of DDS Interface Description Language (IDL) files representing a DDS PSM. The IDL files were then run through a commercial DDS Code Generator, to build software service skeletons that were populated with business logic. Once each vendor finished its services, it handed them off to the other participating vendors for integration within their GCS.

CONCLUSION

This UCS Architecture effort incorporates the best practices of current Army, Air Force and Navy development efforts to include: definition of a common functional architecture, interface standards, business rules, use of open-source and government-owned software as appropriate, competitive acquisition options, and refinement of message sets to support all operational requirements current and planned programs of record.

The UCS Architecture is supported by the UCS WG, which is an open technical standards group consisting of over 100+ organizations from the major UAV primes, the Services, small business, and additional subject matter experts. The UCS WG is bringing together all the Services to work with industry to define open standards for GCS components.