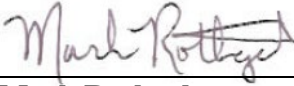


Unmanned Maritime Autonomy Architecture  
(UMAA)  
Architecture Design Description  
(ADD)




Version 1.1a\*  
(UMAA-INF-ADD)  
December 19, 2019

### Signature Page

Submitted by:  Date Signed: 19 Dec 2019  


---

Mark Rothgeb  
Unmanned Maritime Autonomy  
Architecture Standards Board Chair

PMS 406 Approvals:  Date Signed: 9 Mar 2020  

---

CDR Jeremiah Anderson  
PMS 406 Advanced Autonomous  
Capabilities, Principal Assistant  
Program Manager

 Date Signed: 3/9/20  

---

CAPT Pete Small  
Program Manager  
PMS 406 Unmanned Maritime Systems,  
PEO Unmanned and Small Combatants

**TABLE OF CONTENTS**

<u>SECTION</u>	<u>PAGE</u>
TABLE OF CONTENTS .....	iii
FIGURES .....	iv
1.0 INTRODUCTION.....	1
1.1 UMAA SCOPE .....	1
1.2 RELATED STANDARDS .....	2
1.3 DOCUMENT CONTEXT.....	4
1.4 DOCUMENT ORGANIZATION .....	5
2.0 ARCHITECTURE GOVERNANCE PROCESS .....	5
2.1 UNMANNED MARITIME AUTONOMY ARCHITECTURE BOARD (UMAAB).....	5
2.2 ADOPTION OF UMAA .....	6
2.3 REFERENCE IMPLEMENTATION.....	6
3.0 AUTONOMY ARCHITECTURE DESIGN PRINCIPLES.....	6
3.1 MODULAR OPEN ARCHITECTURE .....	6
3.2 QUALITY ATTRIBUTES.....	6
3.2.1 ENABLE LOOSE COUPLING.....	6
3.2.2 ALLOW PAIR-WISE CHANGES .....	7
3.2.3 IMPROVE SUSTAINABILITY .....	7
3.2.4 ENABLE NEW CAPABILITIES AND MISSIONS.....	7
3.2.5 ENABLE NET-CENTRICITY .....	7
3.2.6 ENABLE CROSS-PLATFORM DOMAIN FUNCTIONAL COMMONALITY .....	7
3.2.7 ENABLE COMPETITION.....	7
3.2.8 ENABLE FEDERATED ACQUISITION .....	8
3.2.9 SUPPORT VERIFICATION .....	8
3.2.10 SUPPORT SOFTWARE REUSE .....	8
3.2.11 SUPPORT DATA QUALITY .....	8
3.2.12 ENABLE SYSTEM SECURITY.....	8
3.2.13 MANAGE OBSOLESCENCE .....	8
3.2.14 SUPPORT SAFETY CRITICAL SOFTWARE .....	9
4.0 ARCHITECTURE GUIDELINES.....	9
4.1 INTERNAL COMMUNICATION MECHANISMS.....	9
4.2 TRANSPORTS .....	9
4.3 UTILIZE OPEN SOURCE SOFTWARE .....	9
4.4 INFRASTRUCTURE FUNCTIONS SUPPORT .....	10
5.0 ARCHITECTURE DESCRIPTION.....	10
5.1 FUNCTIONAL VIEW .....	11
5.1.1 MISSION MANAGEMENT .....	12
5.1.2 MANEUVER OPERATIONS .....	13
5.1.3 ENGINEERING OPERATIONS.....	14
5.1.4 SENSOR AND EFFECTOR MANAGEMENT .....	15
5.1.5 COMMUNICATIONS OPERATIONS.....	16
5.1.6 SITUATIONAL AWARENESS.....	17
5.1.7 PROCESSING OPERATIONS .....	18

5.1.8 SUPPORT OPERATIONS ..... 19

5.2 INTERFACE VIEW..... 19

5.3 DATA VIEW ..... 21

6.0 SUMMARY ..... 21

7.0 REFERENCES..... 23

8.0 ACRONYMS AND ABBREVIATIONS..... 24

**FIGURES**

<u>FIGURE</u>	<u>PAGE</u>
FIGURE 1: UNMANNED MARITIME SYSTEMS OPERATIONAL VIEW .....	2
FIGURE 2: FUNCTIONAL ALLOCATION OF UUV AUTONOMY - ASTM (WITHDRAWN IN 2015) .....	3
FIGURE 3: UMAA PRODUCT HIERARCHY .....	4
FIGURE 4: UMAA HIGH-LEVEL FUNCTIONS .....	11
FIGURE 5: MISSION MANAGEMENT FUNCTIONS .....	12
FIGURE 6: MANEUVER OPERATIONS FUNCTIONS .....	14
FIGURE 7: ENGINEERING OPERATIONS FUNCTIONS.....	15
FIGURE 8: SENSOR AND EFFECTOR MANAGEMENT FUNCTIONS.....	16
FIGURE 9: COMMUNICATIONS OPERATIONS FUNCTIONS.....	17
FIGURE 10: SITUATIONAL AWARENESS FUNCTIONS.....	18
FIGURE 11: PROCESSING OPERATIONS FUNCTIONS .....	19
FIGURE 12: SUPPORT OPERATION FUNCTIONS .....	19
FIGURE 13: PUBLISH/SUBSCRIBE .....	20
FIGURE 14: REQUEST/REPLY .....	20
FIGURE 15: COMMAND/RESPONSE .....	20

\* Version 1.1a represents a non-content administrative change to version 1.1 consisting of only corrected figure numbers and a title change on the approvals page with no additional signoff required by PMS 406.

## 1.0 INTRODUCTION

This document describes the Unmanned Maritime Autonomy Architecture (UMAA). The fundamental purpose of this architecture is to promote the development of common, modular, and scalable software for Unmanned Maritime Vehicles (UMVs) that is independent of a particular autonomy implementation. This document consists of guidelines that enable development, evolution, and innovation of autonomy on-board a UMV without requiring a re-design of the system. In order to accomplish this, the UMAA will define the:

- Architecture framework – defines high-level architecture and the guidelines for implementing the functionality associated with UMV autonomy
- Key interfaces – identify the interfaces of common capabilities or functions associated with UMV autonomy as well as any external interfaces that support UMV autonomy
- Data model – describes the data associated with UMV autonomy key interfaces
- Governance – defines the roles and responsibilities associated with the development of UMAA, the process for extending/updating UMAA, and the compliance requirements for UMAA

The UMAA adopts the Institute of Electrical and Electronics Engineers (IEEE) definition of Architecture as “Architecture is the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.”<sup>[1]</sup> Consequently, the UMAA will define the services and associated interfaces used to support autonomous operations on a UMV. The degree of autonomy may vary across UMVs as well as within a UMV over time where autonomy is defined as “...an unmanned system’s own ability of sensing, perceiving, analyzing, communicating, planning, decision-making, and acting/executing, to achieve its goals as assigned by its human operator(s) through designed Human/Computer Interaction or assigned through another system that the Unmanned System interacts with.”<sup>[2]</sup>

This document introduces the starting point of the initial architecture. Separate documentation will be used to expand the definition of the key interfaces, the associated data model for UMV autonomy, compliance, and governance. The anticipated consumers of this architecture include, but are not limited to, lead system integrators (government and industry), autonomy developers, autonomy test and evaluation (T&E) engineers, and in-service engineering support.

### 1.1 UMAA SCOPE

Unmanned Maritime Systems (UMSs) consist of command and control (C2), one or more UMVs, and support equipment and software (e.g., recovery system, Post Mission Analysis applications, etc.). The scope of the UMAA is focused on the autonomy that resides on-board the UMV. This includes the autonomy for all classes of UMVs and must support varying levels of communication in mission (i.e., constant, intermittent, or none) with its C2 System.

UMVs conforming to UMAA must be complementary with their C2 System, such as the Navy’s Common Control System (CCS), TOPSIDE, etc. Whereas the UMAA supports fully autonomous decision-making on-board the UMV, CCS supports human-on-the-loop decision-making through an operator interface. The UMAA will consider standards utilized by CCS

where possible in order to maintain a maximum level of continuity between the efforts. CCS and UMAA are complementary as outlined in Figure 1.

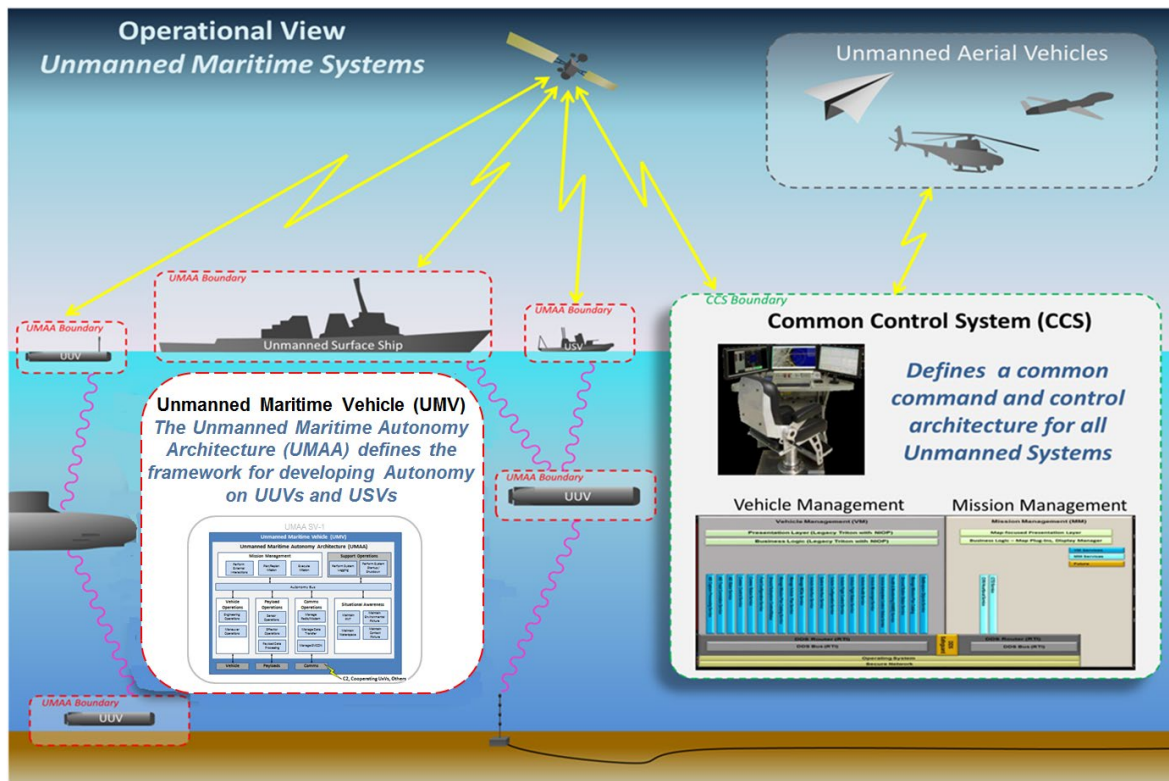


FIGURE 1: UNMANNED MARITIME SYSTEMS OPERATIONAL VIEW

## 1.2 RELATED STANDARDS

A number of autonomy architectures and standards have been adopted by different programs and projects within the Navy’s unmanned underwater and surface vehicles community. Given the relatively new state of autonomous technology when compared to other enabling unmanned system technologies, most resulting system architectures are born out of prototype innovations and layering autonomy solutions on existing systems within the architectural constraints of those systems. These approaches have driven forward the development of specific autonomous technologies in the maritime domains at a rapid pace but have not resulted in a modular and extensible, open-architecture standard that enables the Navy to procure truly modular and affordable UMS solutions.

Lacking any true central governance within the community, an autonomy architecture standard that some development efforts have loosely followed is the American Society for Testing and Materials (ASTM) F2541 Unmanned Undersea Vehicles (UUVs) Autonomy and Control<sup>[3]</sup>. This standard has served as an initial guide for the unmanned systems development community since 2006. It provided the first attempt at a governing framework for autonomous systems in Navy unmanned vehicles. The functional architecture of this ASTM UUV standard is shown in Figure 2.

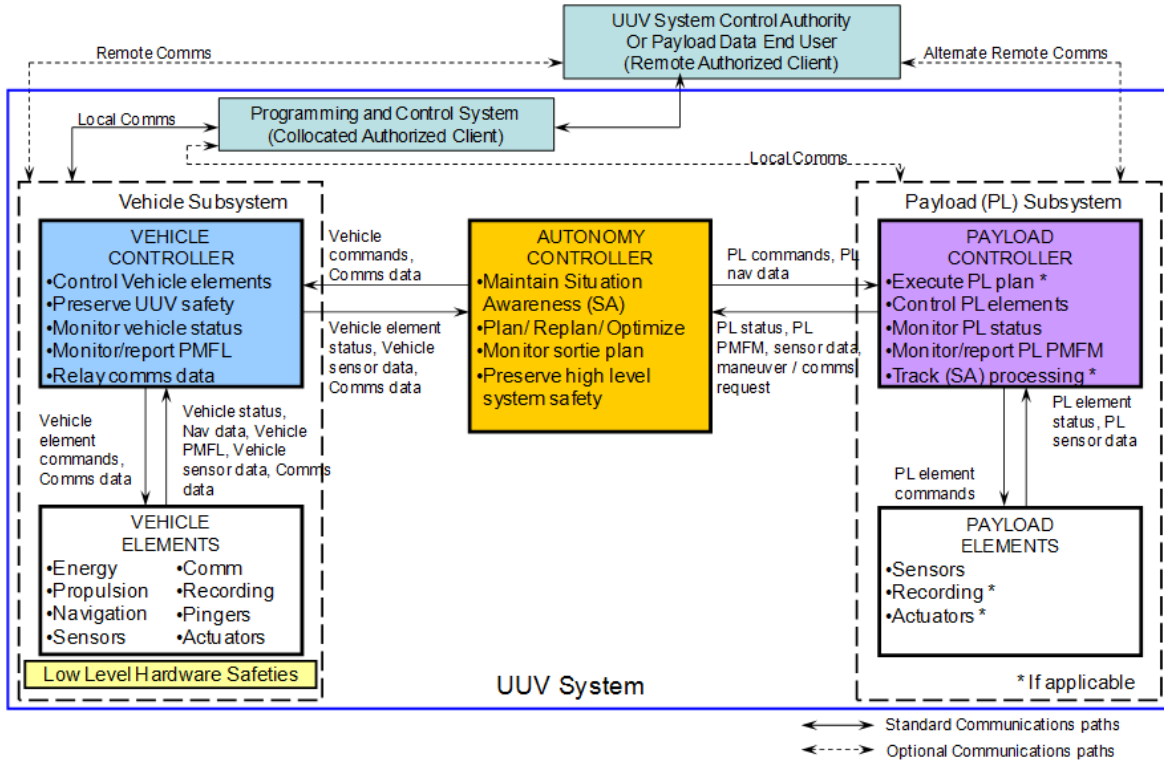


FIGURE 2: FUNCTIONAL ALLOCATION OF UUV AUTONOMY - ASTM (WITHDRAWN IN 2015)

The terminology and definitions defined by this architecture are commonly used in the UUV autonomy community. However, this standard has been withdrawn and is not used in Program of Record (PoR) procurement activities. There is no active governance of this standard. Additionally, there are no reference implementations, verification processes, or formal software definitions to ensure architecture compliance. *As an architecture document without any supporting interface definitions or verification procedures, the ASTM standard lacked any ability to support software reuse, modular software components, or upgrades to key technologies.*

Two standards with substantial effort behind them that have been used within Navy S&T include the Joint Architecture for Unmanned Systems (JAUS)<sup>[4]</sup> and the Unmanned System (UxS) Control Segment (UCS) Architecture<sup>[5]</sup>. Both JAUS and UCS have been developed with the goal of promoting modularity and interoperability by defining a set of common unmanned system capabilities and modeling the data associated with their interfaces. JAUS was initially developed by the JAUS Working Group (WG). The JAUS WG was chartered by the Deputy Director, Office of the Undersecretary of Defense, Acquisition, Technology, and Logistics (OUSD (AT&L)), Strategic & Tactical Systems/Land Warfare to develop an architecture that facilitates the development of modular unmanned systems with increased interoperability. In 2005, the JAUS WG began its transition to Society of Automotive Engineers (SAE) with the initial publications under SAE being based on the documents published under the WG. JAUS employs a service-oriented architecture approach, representing unmanned system capabilities as defined services with defined message-based interfaces. JAUS currently defines approximately 75 services that address common unmanned system capabilities including vehicle mobility, environment sensing, and serial manipulators. These

services and interface definitions are used extensively by unmanned ground vehicle programs. The use of JAUS in UMSs occurs on a case-by-case basis, typically on prototype UMVs. Use in the maritime domain is not widespread and varying additional services are typically defined to completely address on-board UMV autonomy.

The UCS Architecture began as the Unmanned Aircraft System (UAS) Control Segment (UCS) Architecture and was initially developed by the UCS WG of the OUSD (AT&L). OUSD (AT&L) transitioned this effort to the AS-4 UCS Technical Committee in April 2015 with the publication of Release 3.4. With the transition, the scope of UCS was increased to include all vehicle domains. The UCS Architecture provides a Service Oriented Architecture (SOA) and modeling framework for the specification, integration, implementation, and deployment of control segment software. The architecture is centered on a service package Platform Independent Model (PIM) and associated foundation models. The UCS Architecture defines platform independence as the independence of the software operating environment, which allows it to be implemented on different computing infrastructures and with different communication protocols. The UCS Architecture is extensible and describes approximately 150 application software services to support the current capabilities of the Department of Defense (DoD) UxS portfolio. The UCS architecture is not focused on on-board UMV autonomy.

### 1.3 DOCUMENT CONTEXT

The UMAA documents' hierarchy is structured into three tiers to delineate and guide the level of detail and purpose of the documents within each tier as shown in Figure 3. The tiers are described as follows:

1. Governing Documents

This tier includes the Architecture Design Description (ADD), governance, and compliance documents. The architecture design is described at the functional level and provides the overall technical guidance for the lower tiers.

2. Modeling Documents

This tier defines the architecture data model and services at an abstract level that is independent of the implementation and deployment details.

3. Interface Documents

This tier defines the detailed interfaces for the services defined by the architecture. Implementation details such as middleware and software development are defined in this tier to support interoperability between software services.

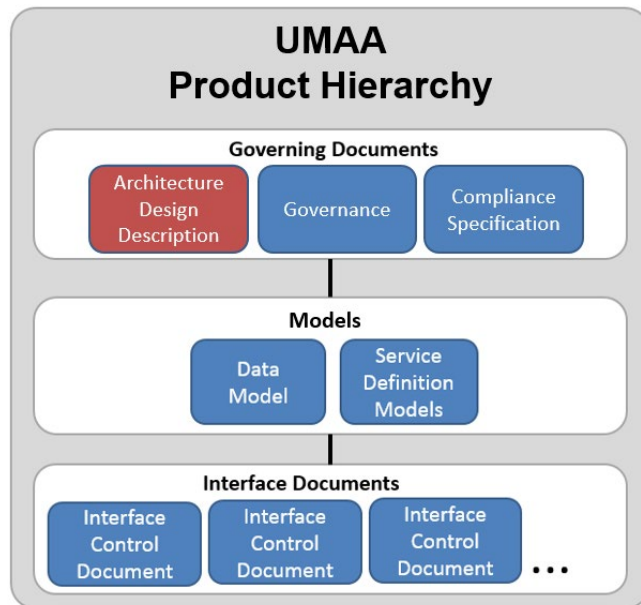


FIGURE 3: UMAA PRODUCT HIERARCHY



## 1.4 DOCUMENT ORGANIZATION

*Section 1: Introduction and Context* – Identifies the applicability of this document, its purpose, its scope, and its intended audience.

*Section 2: Architecture Governance Process*– Describes the supporting processes related to the architecture.

*Section 3: Autonomy Architecture Design Principles* – Describes the non-functional requirements the architecture has been designed to address.

*Section 4: Architecture Guidelines* – Defines the constraints on the architecture and how they affect the architecture decisions made within this document.

*Section 5: Architecture Description* – Provides a description of the UMAA by summarizing the key functions of the system and how those aspects are significant to the architecture, presenting the structure of the system through its services and their interactions, and describing the management of data and documenting the data flows.

*Section 6: Summary*

*Section 7: References*

*Section 8: Acronyms and Abbreviations*

## 2.0 ARCHITECTURE GOVERNANCE PROCESS

The UMAA must evolve to incorporate new services, re-design of interfaces, introduction of new core capabilities, or re-arrangement of architectural services. These modifications and additions will be reviewed for incorporation into UMAA according to the UMAA governance process. When found not to be beneficial for the UMAA community, the proposed changes will remain as custom definitions for a specific program when no other standardized service or interface is incorporated into the UMAA. The governance process is defined external to this document.

### 2.1 UNMANNED MARITIME AUTONOMY ARCHITECTURE BOARD (UMAAB)

The UMAAB's membership is comprised of board members and organizations in accordance with a separately maintained charter<sup>[6]</sup> instituted by PMS 406 in coordination with key stakeholder leadership (resource sponsors, Program Executive Offices (PEOs), etc.). The UMAAB will coordinate efforts with all applicable stakeholders to provide a process and forum for discussion and resolution of issues in developing the architecture, work with UxS programs, and ensure architecture recommendations are coordinated and transparent.

The UMAAB will develop and maintain this ADD and associated documentation. The UMAAB will establish a process by which modifications and additions can be introduced into the architecture as well as defining conformance with UMAA. The ADD will be updated in accordance with the defined governance process by the UMAAB, adding guidance to the ADD to document required architectural changes.

The UMAAB shall recommend policies to the PMS 406 PM who will coordinate with Navy's Digital Warfare Office (DWO), maritime system resource sponsors (N95, N96, and N97),

Navy acquisition leadership, and NAVSEA leadership to support decisions and implementation for UMV development and acquisition programs with a significant on-board autonomy capability.

## **2.2 ADOPTION OF UMAA**

PMS 406 is coordinating with multiple stakeholders to establish the UMAA as a Navy standard that can be used in future autonomous vehicle acquisitions. The goal is for existing programs to produce either a transition plan or a request for exemption by FY22 for approval.

PMS 406 intends to promulgate the UMAA ADD as the umbrella document under which standard UMAA Interface Control Documents (ICDs) will be created to establish standards-based autonomy software interfaces across the family of Navy UMVs.

## **2.3 REFERENCE IMPLEMENTATION**

PMS 406 will maintain and own reference implementations available for use and review by the community as Government Open Source Software (GOSS) as a starting point for UMV programs making it available for use by the broader UMV community. The reference implementations will provide exemplar interfaces and software tools for compliance verification, and enable a service-based acquisition approach allowing autonomy services to be leveraged, replaced, or augmented through the lifecycle of an UMV program.

## **3.0 AUTONOMY ARCHITECTURE DESIGN PRINCIPLES**

This section defines quality attributes and non-functional requirements that inform the design of UMAA. These design principles are manifested in the development of open interfaces and services.

### **3.1 MODULAR OPEN ARCHITECTURE**

The architecture must adhere to best practices for open architecture systems engineering. This means the architecture will support software that is modular, decomposable, replaceable, and interchangeable, so that functional services may be used in a variety of applications through well-defined open interfaces.

The openness of the autonomy architecture can be evaluated using the U.S. Navy's Open Architecture Assessment Tool (OAAT), Version 3.0. OAAT and its supporting documents are available on the Naval Open Architecture website<sup>[7]</sup>.

### **3.2 QUALITY ATTRIBUTES**

The UMAA is intended to be built to satisfy a set of common principles or attributes. These attributes will be used to evolve the UMAA as new requirements and technologies emerge.

#### **3.2.1 Enable Loose Coupling**

The architecture must enable UMVs to be composed of a set of separate loosely coupled services resulting in a modular system. These services will include (but not limited to) vehicle control, sensors, payloads, and situational awareness.

### **3.2.2 Allow Pair-Wise Changes**

The architecture must enable pair-wise changes without unnecessarily impacting other services. If two services require an external interface change, other services using that data shall not be required to recompile and redeliver their software if they do not use the new or modified interface.

### **3.2.3 Improve Sustainability**

The architecture must support modular services that will enable replacement of one of the services without impacting any of the others. This requirement will result in improved sustainability of the UMaVs employing the architecture by enabling targeted updates due to capability improvements or obsolescence.

### **3.2.4 Enable New Capabilities and Missions**

The architecture must support upgrading to new capabilities including those that support (but not limited to) long duration autonomous operation, multi-vehicle missions, and manned-unmanned teaming. The architecture must support this ability to integrate new capabilities into the autonomy with minimal changes allowing for a very high degree of reuse and minimal rework of services. The update schedules of each of the services may differ depending on technology insertion rates. The aggregation of all old and new services must result in a seamlessly integrated operational system.

### **3.2.5 Enable Net-Centricity**

The DoD has mandated net-centric requirements on all platforms and programs. The UMaA needs to be interoperable with net-centric services and provide useful services for others to utilize. Services provided by the UMaA to external platforms could include (but are not limited to) position data, environmental data, contact data, and health status. UMaA will leverage existing standards from the Navy enterprise to the maximum extent possible in support of net-centricity.

### **3.2.6 Enable Cross-Platform Domain Functional Commonality**

The UMaA should capitalize on the commonality between autonomy capabilities and architecture regardless of the UMaV domain. The UMaA will support cross-platform domain service definitions. Cross-platform domain for UMaA will explicitly include surface and underwater UMaVs, but will include air and ground domains as a goal. UMaA is expected to share common services between platform domains and support the development of common software implementations. UMaA modularity will enable domain-specific configurations with maximum reuse of services and provide modular service interfaces to support platform domain-specific extensions when needed.

### **3.2.7 Enable Competition**

The desire to incorporate innovative capabilities requires that the UMaA enables competition for software services. The UMaA and derived documents will provide well-documented, modular interfaces that are based upon standards to support increased service-level competition to enable innovation and reduce cost. The architecture will be under configuration management control to ensure a library of software services is created over multiple programs. The goal is

for programs to have the option to integrate UMAA compliant services from multiple performers to produce an autonomy solution faster, better, and with reduced risk of vendor-lock.

### **3.2.8 Enable Federated Acquisition**

The architecture aims to allow programs to spread development responsibility among performers potentially across multiple PoRs. UMAA will support upgrades for new capability when required by incorporating feedback from programs into the UMAA as determined by its governance.

### **3.2.9 Support Verification**

The UMAA will provide properly structured services, interfaces, and capabilities that allow for verification of the system. All architecture requirements and all implemented interfaces will be verifiable.

### **3.2.10 Support Software Reuse**

The architecture must define software services such that multiple programs can leverage the software implementations. Programs incur costs to design, develop, verify, and validate software. Programs may reduce cost by reusing or extending pre-existing software services when functionality is deemed appropriate by the program.

### **3.2.11 Support Data Quality**

The UMAA must support data quality attributes including validity, completeness, consistency, and timeliness for information shared among services. The use of well-defined ICDs will ensure data are complete and enable testing for validity. Consistency and timeliness are implementation and integration concerns (e.g., designating a single, authoritative data source; Quality of Service (QoS) transport guarantees), but loosely coupled, transport-independent services will allow service developers and system integrators the flexibility needed to achieve those goals.

### **3.2.12 Enable System Security**

Cyber threats have dramatically increased and will continue to pose a significant threat to all U.S. Navy platforms. UUVs represent a more dramatic threat to cyber vulnerabilities due to the fact that there would be no manned intervention onboard the platform to identify, remedy, or address a cyber-attack. While these considerations are primarily program-, mission-, and vehicle-specific, the UMAA will not preclude incorporation of cybersecurity measures addressing vehicle software, communications, multi-level security, and anti-tamper features.

### **3.2.13 Manage Obsolescence**

Managing obsolescence is required for system sustainability and cost avoidance, and to support functional improvements. The UMAA must allow autonomous systems to evolve as new technologies replace older technologies. It is anticipated that frequent updates of the autonomy technologies will occur on unmanned platforms. As standards evolve, and hardware and software technologies mature, it is anticipated that the infrastructure technologies on which this architecture is built will require upgrades (such as computing hardware, operating systems,

etc.). The set of UMAA services will change or evolve as new technologies become available and more affordable. Backward compatibility becomes a key factor in choosing new technologies. The architecture must embrace modularity and open system engineering, and provide strong interface management. The use of virtualization and/or containerization to support obsolescence assurance may be applicable in specific implementations.

#### **3.2.14 Support Safety Critical Software**

Safety critical software includes any software whose inadvertent response to stimuli, failure to respond when required, out of sequence response, or other type of response failure can result in a safety or significant material loss failure. The UMAA architecture will facilitate separation of “Safety Critical” and “Non-Safety Critical” software modules. Separate modules designed in accordance with the modular and open interfaces will facilitate flexibility and reduce the burden of testing when software updates do not impact the safety critical autonomy software. This separation addresses the high level of effort and time associated with the testing and validation of safety critical autonomy software.

### **4.0 ARCHITECTURE GUIDELINES**

This section defines the architecture guidelines for the UMAA.

#### **4.1 INTERNAL COMMUNICATION MECHANISMS**

Communication between services in the UMAA should utilize publish/subscribe, request/reply, or command/response mechanisms. Other mechanisms (e.g., database sharing) should be limited to cases when these mechanisms are not feasible. In addition, the publish/subscribe, request/reply, or command/response mechanisms should follow the patterns described in section 5.25.1.7. Deviations may be needed depending on program requirements.

#### **4.2 TRANSPORTS**

The functionality provided by UMAA services should be independent of the transport used to provide the underlying data link. Service functionality should not rely on any specific transport but instead should handle any specific data link requirements by identifying QoS parameters such as the prioritization of message traffic. This also allows for interface compliance verification independent of the transport technology.

#### **4.3 UTILIZE OPEN SOURCE SOFTWARE**

Open Source Software (OSS) should be used within the implementation unless there is insufficient life-cycle cost Return on Investment (ROI). In addition, published industry standards should also be used where applicable. Examples of rationale for using a non-OSS solution (i.e., a commercial or proprietary product) could include:

- Reduced development costs over the life of the interface/service
- Needed performance improvements (as identified by system key requirements)
- Reduced deployment costs
- Reduced maintenance costs over the life of the interface/service
- No viable OSS solution available that incorporates cybersecurity requirements

OSS licensing restrictions should also be considered when it restricts government use and distribution. When considering a non-OSS product, extreme care must be taken to consider the control that the commercial business has and will have in supporting the product over the life of the system. For all acquired software, data rights will be a significant contractual consideration.

#### 4.4 INFRASTRUCTURE FUNCTIONS SUPPORT

Intrinsic infrastructure functions are external to the UMAA and will be provided by the host computing environment. Some examples of host functions include:

##### **Directory/Lookup Services**

- Domain Name System/Service (DNS)
- Lightweight Directory Access Protocol (LDAP)
- Cybersecurity Services
- Access Control
- Auditing
- Key Management
- Enclave Guard
- Intrusion Detection
- Virus Scan

##### **Synchronized Time Service**

- Network Time Protocol (NTP)
- Precision Time Protocol (PTP)

### 5.0 ARCHITECTURE DESCRIPTION

The UMAA is composed of a set of service definitions that define a foundational basis for an autonomous UMV. The service definitions include the expected functional capability and interfaces for the service. A system integrator builds the architecture by developing interoperable software components compliant with UMAA to meet specific mission system requirements. By adhering to the service definitions and standard interfaces, the integrator is able to select the performers who are best equipped to deliver the required functionality for each service. Other systems may use, extend, or scale their implementation by reusing, adding, or removing services available in the library of available software services.

The architecture assumes a common autonomy data bus between all software components with few exceptions (see Section 4.1). Specific program requirements may drive segmentation of the bus for security or other design reasons. Accommodation will be made to share data across boundaries as needed as determined by specific program requirements. Although the service definitions include both inputs and outputs of the service, the service implementation and interface can easily be modified to access any information that is published to the common autonomy data bus to produce the expected output of the service. The architecture aims to create decoupled services, which means the producer of the information is not required to know the consumers. This allows new consumers to be added to the system without any impact to the producer. The architecture will allow more than one instance of a service to be instantiated and services to be distributed on networked computing platforms.

### 5.1 FUNCTIONAL VIEW

To enable modular development and upgrade of the functional capabilities of the on-board autonomy, UMAA defines eight high-level functions. These core functions include: Mission Management, Engineering Operations, Maneuver Operations, Processing Operations, Sensor and Effector Management, Communications Operations, Support Operations, and Situational Awareness. In each of these areas, it is anticipated that new capabilities will be required to satisfy evolving Navy missions over time. UMAA seeks to define standard interfaces for services for these functions, so that individual programs can leverage capabilities developed to these standard interfaces thus allowing software libraries to be developed and shared across programs that meet the standard interface specifications. Individual programs may group services and interfaces into components in different ways; however, the interfaces and services defined by UMAA will be required as defined in the ICDs. Figure 4 depicts a functional decomposition of these eight core functions and representative capabilities within them.

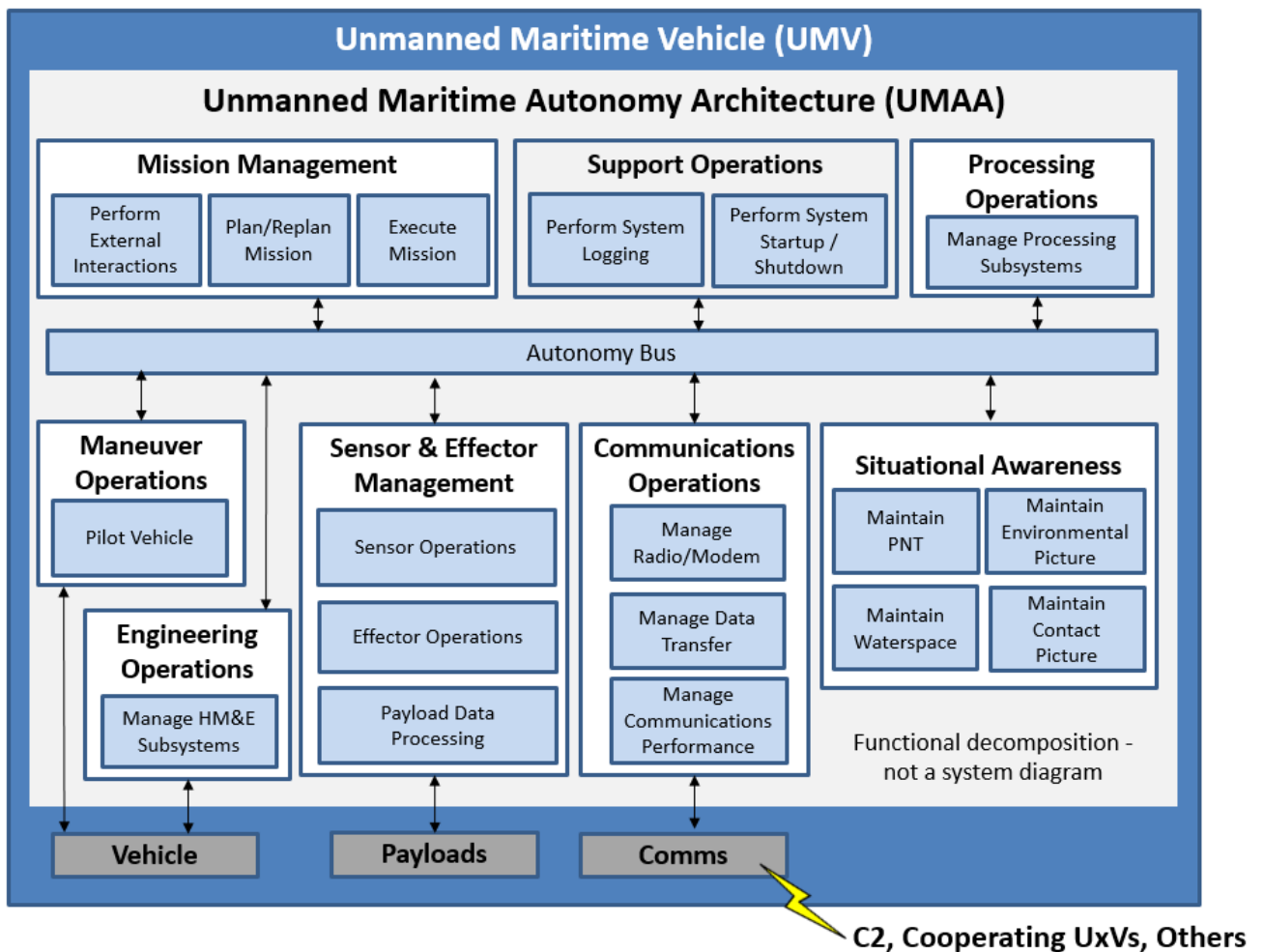
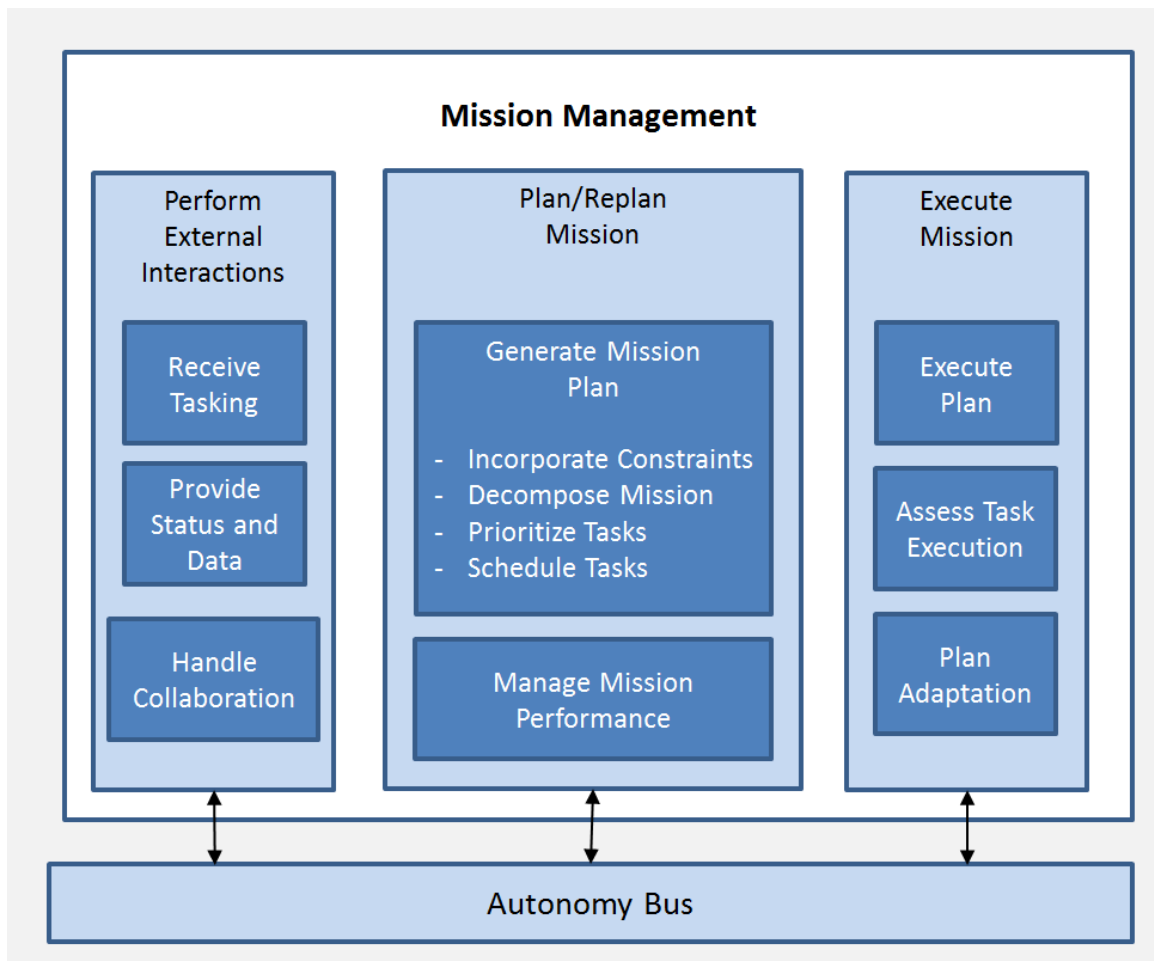


FIGURE 4: UMAA HIGH-LEVEL FUNCTIONS

### 5.1.1 Mission Management

The Mission Management provides the management and execution of the overall mission and governs the overall operation of the system. Figure 5 shows the functions of the Mission Management. It provides the reasoning and planning that executes the mission on-board the vehicle. The Mission Management will contain elements that are specific to a particular program and mission. It will be responsible for decomposing the user-provided mission objectives into executable tasks for services in the lower-level functional areas. It will communicate over a common bus with the lower-level autonomy services using common interface standards defined by UMAA derived documents. The Mission Management will aggregate individual status of services to produce an overall mission status. Where applicable, Mission Management will delegate self-contained operations to lower-level services such as letting Maneuver Operations run a loiter pattern. Similarly, lower-level services will inform Mission Management of operational state and constraints, so that these may be considered in concert for overall mission planning and execution.



**FIGURE 5: MISSION MANAGEMENT FUNCTIONS**

Mission Management provides an UMV with the self-governing ability to perform its tasked mission objectives using sensing to perceive the external environment (Sensor and Effector Management and Situational Awareness), self-sensing to maintain its operational state



(Engineering Operations), vehicle maneuvering to proactively control vehicle dynamics (Maneuver Operations), and communications management (Communications Operations) to coordinate its objectives with external entities.

Mission Management can include the capability to perform dynamic planning and re-planning in support of the tasked objectives, apply constraints to its operations, schedule activities, manage its resources, manage what and when to communicate (status, tasking, data exchange), operate in collaboration with other manned and unmanned assets, manage execution of the mission, monitor progress, and perform the decision-making required to accomplish its mission objectives.

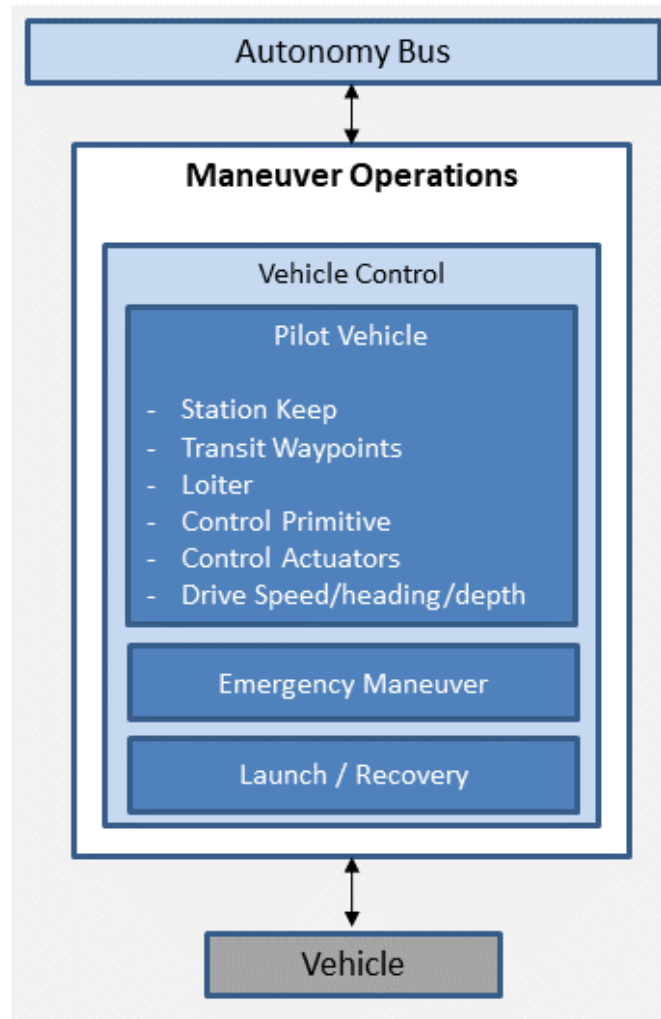
One of the key functional differences between external C2 and the Mission Management is which system is making the maneuvering decisions. For systems that require “operator-in-the-loop” control mode, the C2 system will provide control of the path planning and maneuvering (often provided as a script or manually directed). For systems that enable a fully autonomous control mode, the Mission Management within the platform will provide path planning and maneuvering executed via the Maneuver Operations. In the case of "operator-on-the-loop", the mission autonomy can reach back to the operator for intervention and verification when required. In autonomous operations, the Mission Management path-planning component will generate a path plan for the UMV to meet its mission objectives. The path planner will also be provided vehicle and payload constraints to enable it to choose a sufficient path to satisfy the objectives within the constraint bounds. The path-planning component must be able to ingest varying types of constraints expected in the implementation of the system. Constraints can include such things as available power, risk of detection based on location and any transmissions from vehicle or payload, payload effectiveness for its mission based on location and vehicle dynamics, etc.

Multiple constraints could be fed into the path-planning component in Mission Management. These multi-constraint inputs are defined independently from the path planner and allow for an extensible architecture and implementation. Each constraint provider will use a priori or field data along with algorithms to produce constraints that are provided to the path-planning component. The constraints may have multiple formats such as weighted trajectories, heat maps in local or geographic coordinates, or defined keep-out or keep-in areas.

### **5.1.2 Maneuver Operations**

This function manages and controls UMV movement including movements due to mission re-planning and regulating safe movements as shown in Figure 6. Maneuvering must take into account vehicle constraints (e.g., maximum/minimum vehicle speed and accelerations, minimum turning radius, maximum depth, fuel load, etc.) and current status (position, speed, battery level, leak detections, etc.) as well as any configuration parameters (minimum distance to obstacle, minimum waypoint closest point of approach (CPA), maximum cross track error, etc.) associated with a particular maneuver. Regulating movements includes both the execution control as well as status feedback for each specific maneuver. Maneuver commands associated with general vehicle movement include waypoints, station keep, vector, etc. Maneuver commands may be executed to ensure vehicle safety including reactive obstacle avoidance and emergency vehicle procedures, etc. Maneuver constraints may be received from

other services within UMAA (e.g., Communications Operations, Mission Management), which may further bound maneuver control.



**FIGURE 6: MANEUVER OPERATIONS FUNCTIONS**

The Maneuver Operations function exposes the vehicle's navigation control and status for use by the other UMAA services. The Maneuver Operations control interface receives the control commands from the common autonomy data bus and translates commands to the vehicle-specific protocol for execution of the command. Maneuver Operations acts to balance the capabilities of the platform and the higher-level autonomy. It is expected that platforms will expose interfaces at multiple levels of control including actuator level (e.g., throttle, rudder), set point level (e.g., speed, heading, depth, turn rate), simple behaviors (e.g., waypoints, loiter pattern) and potentially higher-level behaviors.

### 5.1.3 Engineering Operations

This function manages decision-making associated with the vehicle hardware and software to include optimizing use of the vehicle resources as shown in Figure 7. It is responsible for maintaining all of the HM&E systems on the vessel. Reconfiguration of the power plant,

electrical, software, and mechanical configurations are included. Management of these services would include managing their configuration, maintaining and reporting health and status, and providing control. Decision-making associated with both long-term maintenance and short-term emergency recovery procedures is included. Status is reported to Mission Management that would allow higher-level re-planning to take place when necessary.

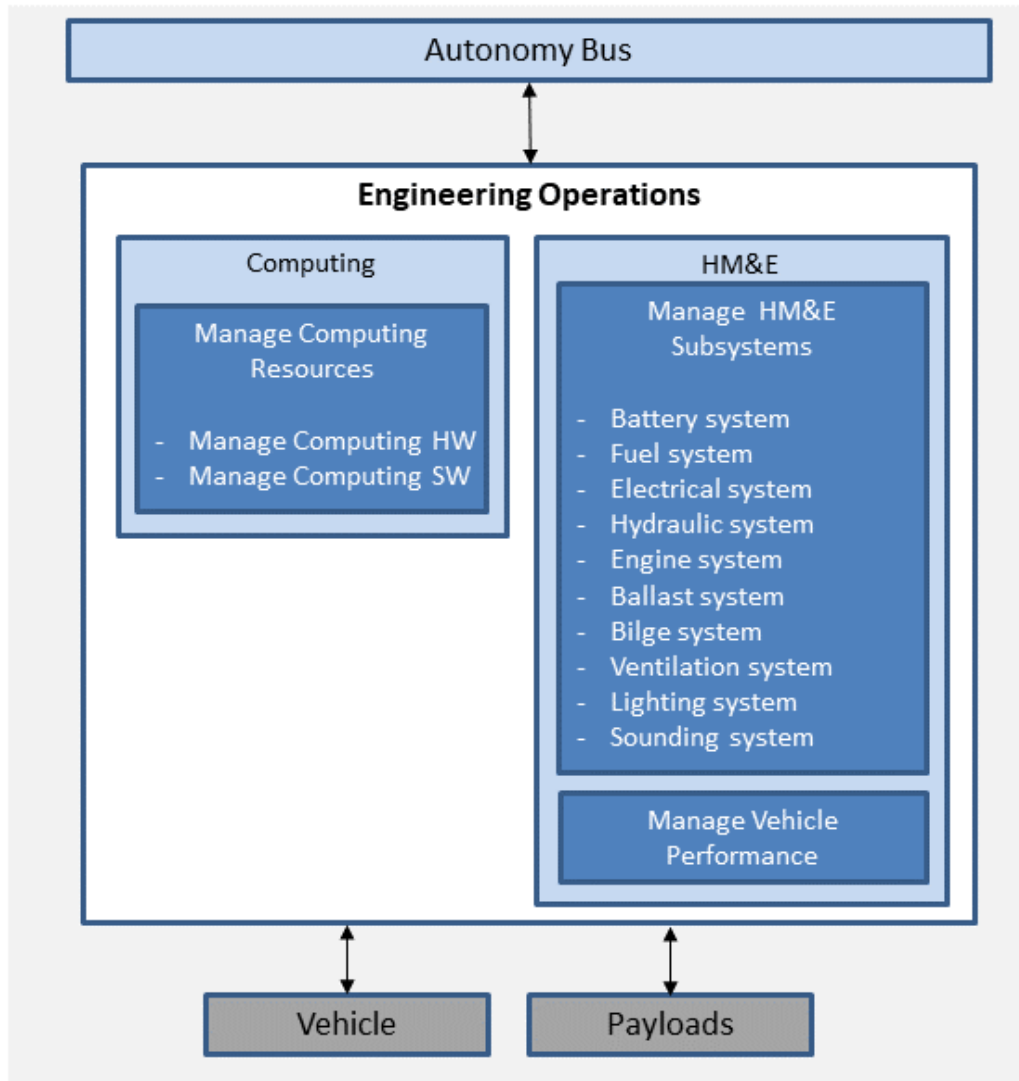
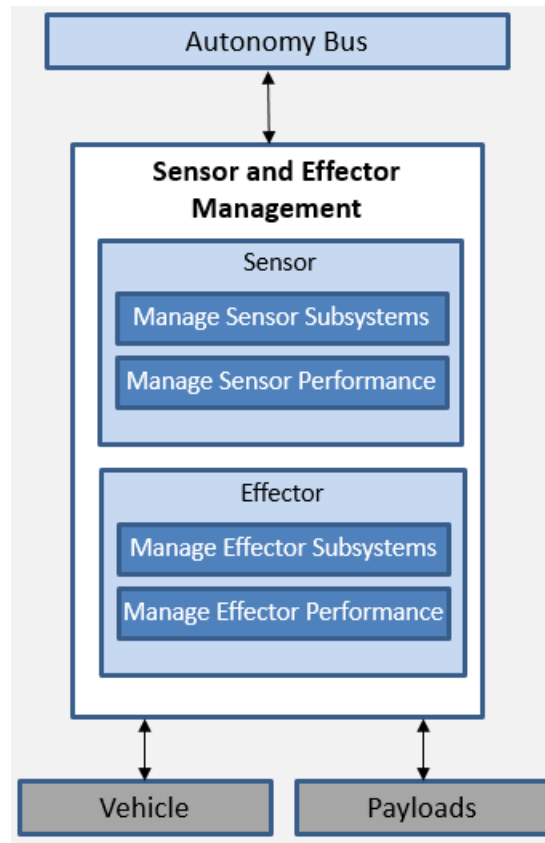


FIGURE 7: ENGINEERING OPERATIONS FUNCTIONS

#### 5.1.4 Sensor and Effector Management

This function manages and controls the payload suite, which may consist of sensor(s) and/or effector(s) on-board the unmanned system as shown in Figure 8. Managing the payloads will be based on mission objectives and current situational awareness. Management functions should employ necessary payloads as the mission progresses and adapts payload configuration as the environment changes.

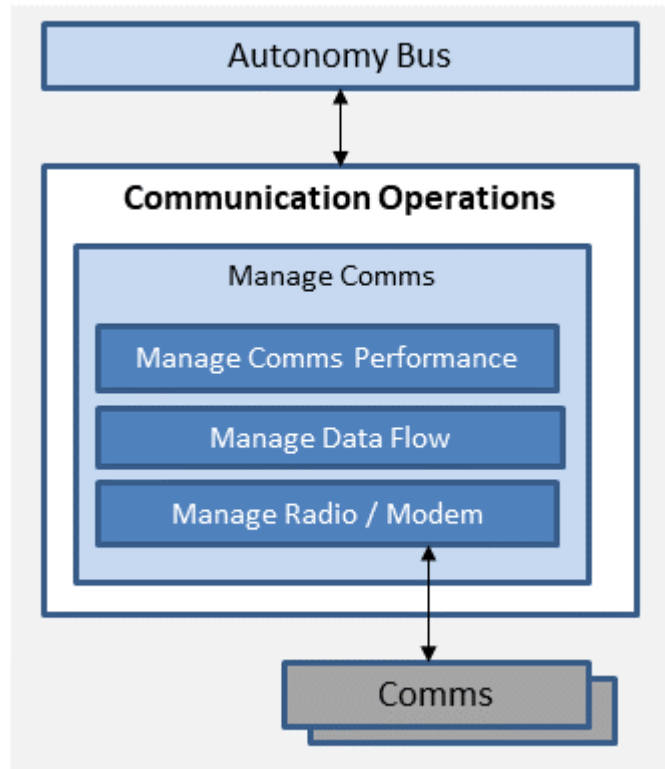


**FIGURE 8: SENSOR AND EFFECTOR MANAGEMENT FUNCTIONS**

Sensor and Effector Management will contain interfaces for payload control. There will be a close tie-in with any payload control services on the remote C2 system with the Sensor and Effector Management. One of the key functional differences between the C2 and on-board autonomy is which system is making the payload control decisions. For systems that require “operator-in-the-loop” control mode, the C2 system will be the primary driver for payload operations. For systems that enable “operator-on-the-loop” or fully autonomous control mode, the platform will be the primary driver for payload operations. Arming and firing sequence autonomy functions are managed by the Sensor and Effector Management. The Sensor and Effector Management may influence or direct other vehicle operations (e.g., maneuvering) but only in coordination with Mission Management, which has the holistic view of vehicle and payload operations.

### 5.1.5 Communications Operations

This function manages the bandwidth and packet routing to optimize the use of multiple communication links based on factors including priority, compression, network availability, and QoS as shown in Figure 9. Communications Operations interfaces with all communication devices onboard the UUV to manage emissions during emissions control (EMCON) and other modes as directed by the Mission Management. Note that in most cases, the very nature of communications for UUVs will be very intermittent due to physics.



**FIGURE 9: COMMUNICATIONS OPERATIONS FUNCTIONS**

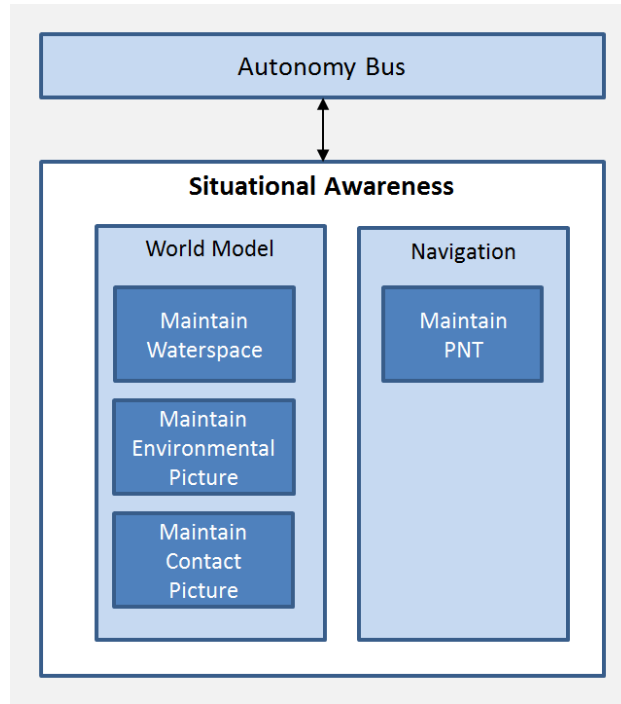
Like the Maneuver Operations and Sensor and Effector Management, Communications Operations relies on external constraints to guide the behavior of the platform and/or communications systems. Communications Operations may produce maneuver constraints based on information and algorithms specific to communications (see Maneuver Operations in Section 5.1.2).

In addition to maneuver constraints that might limit where communications systems can be active, there can also be schedule constraints that define when communications equipment can operate. All of the communications control (location based and time based) is part of Communications Operations. Multiple constraints, independent from the scheduler, create an extensible architecture and implementation. Each constraint provider will use a priori or field data along with algorithms to produce constraints that are provided to the scheduler service in Mission Management. Communications Operations will also be responsible for reporting status to the common autonomy data bus. In the event of communications faults, Mission Management would be notified through that status and will adjust the plan appropriately.

### 5.1.6 Situational Awareness

This function maintains the Situational Awareness also referred to as the world model for decision-making by Mission Management. This includes knowledge of allocated waterspace (constraints on operational area for blue de-confliction in all four dimensions), the environmental picture (bathymetry, nautical charts, and meteorological and ocean data), contact picture (red, blue, and gray contacts, targets, obstacles, objects of interest), and position, navigation, and timing (PNT) as shown in Figure 10. Situational awareness of the

environment in proximity to an unmanned vehicle is critical for navigation and mission success. This includes both real-time perception of the environment through sensors and also use of a priori data (current and historical) either loaded pre-mission or updated from external sources in-stride. It encompasses using the processed payload and organic sensor outputs for higher levels of knowledge inferencing such as what is done by target motion analysis and data fusion of multiple sensors.



**FIGURE 10: SITUATIONAL AWARENESS FUNCTIONS**

The ability to perceive the surrounding environment is what enables safe pilotage including operating in accordance with Collision Regulation (COLREGS). Situational awareness is enabled by sensors such as sonars, stereo cameras, radar, etc., which detect, identify, and track objects or other hazards in the surrounding environment. Situational awareness capability is core to the execution of the autonomy decision-making and a critical part of this architecture. Modular payloads can be added or interchanged without changes to Situational Awareness, so that Situational Awareness is not coupled to a specific payload.

The architecture will enable Situational Awareness data to be published over the common autonomy data bus in order to share information with the payload and other services of the vehicle. The architecture functions that use this data may be required to locally transform or store (cache) portions of this data to optimize their use of the data, but it is the intent of Situational Awareness to be the single authoritative source of the data for the system.

### 5.1.7 Processing Operations

This functional area manages software services that refine data received from sensing and other systems into higher-level information constructs as shown in Figure 11. Examples include sonar beam forming, image processing algorithms, wake detection algorithms, and other algorithmic type processing.

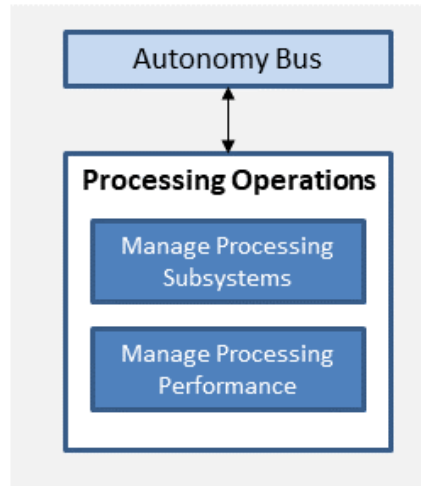


FIGURE 11: PROCESSING OPERATIONS FUNCTIONS

### 5.1.8 Support Operations

This function provides support capabilities for services that can be shared across all of the other functional areas within UMAA (Figure 12). These operations provide common interfaces for infrastructure services.

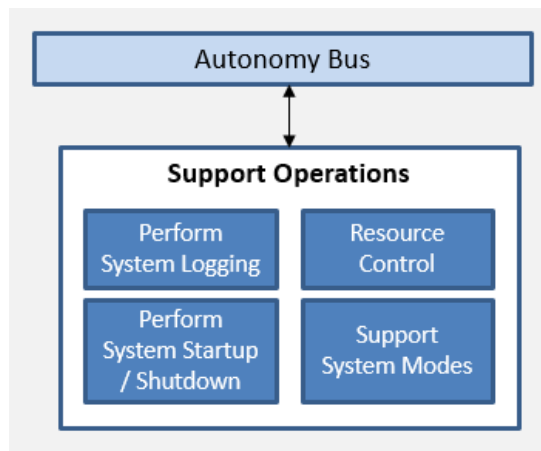


FIGURE 12: SUPPORT OPERATIONS FUNCTIONS

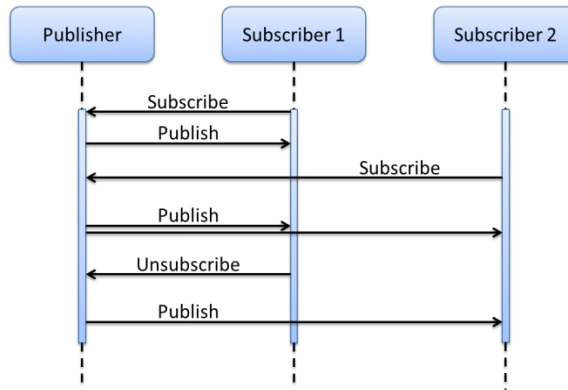
## 5.2 INTERFACE VIEW

The UMAA utilizes network-based communications for the on-board exchange of data and information between services. This transfer is critical to the objective of the UMAA and follows a defined set of interface patterns that generalize the types of data exchanges being made as well as the performance and failover characteristics. The UMAA primarily utilizes three high-level interface patterns for inter-service communications:

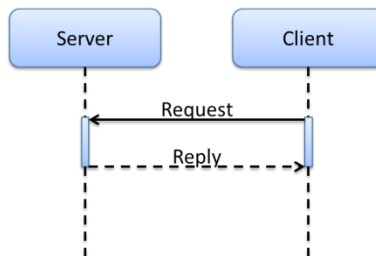
1. Publish/Subscribe with a single data source and multiple consumers, e.g., navigation data being pushed to multiple recipients (See Figure 13)
2. Request/Reply with data being provided from fixed sources to consumers, e.g., environmental data being requested by a consumer (See Figure 14)

3. Command/Response with data being provided from a single source to a single fixed consumer, e.g., waypoint data being commanded to Maneuver Operations and a response from the consumer (See Figure 15)

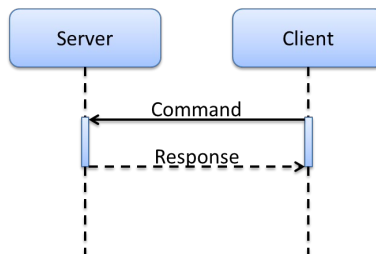
While it is recognized that all three mechanisms are useful for different purposes, the Publish/Subscribe metaphor should be the primary pattern used to maximize de-coupling of the system. In addition, different interfaces types such as streaming data that may be required and would be outside these interface patterns. These other interfaces should follow relevant standards as determined by the specific program. See section 4.0 for more on the architecture guidelines.



**FIGURE 13: PUBLISH/SUBSCRIBE**



**FIGURE 14: REQUEST/REPLY**



**FIGURE 15: COMMAND/RESPONSE**



### 5.3 DATA VIEW

This section examines the UMAA primarily from the view of the data that is exchanged between its components and services. It does not intend to delve into a definition of data, but rather it focuses on data whose scope transcends internal boundaries and is required to be distributed and/or archived. Specific data definitions will be documented in service-based ICDs.

The UMAA has a few defining characteristics that affect the architecture with respect to data management. First, services are not developed by a single provider or procured by a single contracting entity, and their development and delivery schedules are not aligned. Therefore, it is imperative that the specification of the data and its attributes must be properly managed, and updates to the definition must be controlled and coordinated by a governing body that is not associated with any particular service but responsible for the overall reference architecture (UMAA). Second, the UMAA needs to consider processing power and hardware form factors required for data processing (e.g., total available processing power) across the range of UMVs that it will support. Therefore, some of the functions provided in the reference implementation may not be available for some of the smaller, less capable platforms.

It should also be acknowledged that many of the processing resources in the UMVs may be diskless, while other UMVs may have permanent data storage but may not be writable for intermediate storage of data during mission. As such, it is sometimes necessary to develop strategies for designing around these storage constraints.

Third, data within the UMAA deployed UMV is likely classified, and not all data is at the same classification level. The UMAA must be carefully architected to enable separation of data at different classification levels, and care must be taken to manage how (and under what control) data will be taken off of the system. The UMAAB is working with cyber-security subject matter experts as the architecture evolves to ensure the UMAA will support data across multiple security levels when required.

Fourth, much of the UMV data is temporal. It needs to traverse the system within aggressive latency constraints, and it has a fixed expiration time based on storage capacity and data recall requirements.

Fifth, an UMV may employ data reduction and data compression techniques to reduce storage capacity requirements or increase endurance. The UMAA interface definitions must account for these latency and retention requirements, and handle them appropriately.

### 6.0 SUMMARY

The UMAA defines an architecture that includes service definitions, interface definitions, and a reference implementation with the goal to create commonality across the Navy's family of UMVs. Unmanned Surface Vehicle (USV) and UUV programs that leverage UMAA as their foundation for their program-specific, on-board autonomy will benefit from a core set of capabilities that enable the programs to focus on their specific mission requirements and not reinvent common core capabilities, algorithms, and infrastructure. UMAA governance is responsible to coordinate with programs to ensure interfaces and architecture definitions are changed only as required, so that interoperability with other development efforts is maximized,

and critical interface compatibility with the UMV family is maintained. UMAA will evolve over time as new capabilities, sensing systems, and vehicle systems evolve. It is anticipated that there will be feedback to UMAA to incorporate program-developed services and update the requisite interfaces.

## 7.0 REFERENCES

1. IEEE 1471-2000 – “IEEE Recommended Practice for Architectural Description for Software-Intensive Systems”, 2000.
2. SAE AIR5665B – “Architecture Framework for Unmanned Systems”, 2013.
3. Standard Guide for Unmanned Undersea Vehicles (UUV) Autonomy and Control (Withdrawn 2015), <https://www.astm.org/Standards/F2541.htm>
4. AS-4 JAUS Committee Fact Sheet,  
<https://www.sae.org/works/committeeResources.do?resourceID=592774>
5. AS-4 UCS Fact Sheet May 2016,  
<https://www.sae.org/works/committeeResources.do?resourceID=493337>
6. NAVSEA PMS 406 Unmanned Maritime Autonomy Architecture Board Charter Sept 2017
7. Open Architecture Assessment Tool (OAAT) Version 3.0  
<https://www.dau.edu/cop/mosa/Lists/Tools/AllItems.aspx>.

## 8.0 ACRONYMS AND ABBREVIATIONS

ADD	Architecture Design Description
ASTM	American Society for Testing and Materials
C2	Command and Control
CCS	Common Control System
COLREGS	Collision Regulations
CPA	Closest Point of Approach
DNS	Domain Name Service
DoD	Department of Defense
DWO	Digital Warfare Office
EMCON	Emissions Control
GOSS	Government Open Source Software
HM&E	Hull, Mechanical, and Electrical
ICD	Interface Control Document
IEEE	Institute of Electrical and Electronics Engineers
JAUS	Joint Architecture for Unmanned Systems
LDAP	Lightweight Directory Access Protocol
NTP	Network Time Protocol
OAAT	Open Architecture Assessment Tool
OSS	Open Source Software
OUSD (AT&L)	Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics
PEO	Program Executive Office
PIM	Platform Independent Model
PNT	Positioning, Navigation, Timing
PoR	Program of Record
PTP	Precision Time Protocol
QoS	Quality of Service
ROI	Return on Investment
SAE	Society of Automotive Engineers (archaic-no longer an acronym)
SOA	Service Oriented Architecture
T&E	Test and Evaluation
UAS	Unmanned Aircraft System
UCS	Unmanned Control Segment
UMAA	Unmanned Maritime Autonomy Architecture
UMAAB	Unmanned Maritime Autonomy Architecture Board
UMS	Unmanned Maritime System
UMV	Unmanned Maritime Vehicle

USV	Unmanned Surface Vehicle
UUV	Unmanned Undersea Vehicle
UxS	Unmanned Vehicle
WG	Working Group