

A Multifaceted Approach to Modernizing NASA's Advanced Multi-Mission Operations System (AMMOS) System Architecture

Jeff A. Estefan¹ and Brian J. Giovannoni²

Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California 91109

The [Advanced Multi-Mission Operations Systems \(AMMOS\)](#) is NASA's premier space mission operations product line offering for use in deep-space robotic and astrophysics missions. The general approach to AMMOS modernization over the course of its 29-year history exemplifies a continual, evolutionary approach with periods of sponsor investment peaks and valleys in between. Today, the Multimission Ground Systems and Services (MGSS) office—the program office that manages the AMMOS for NASA—actively pursues modernization initiatives and continues to evolve the AMMOS by incorporating enhanced capabilities and newer technologies into its end-user tool and service offerings. Despite the myriad of modernization investments that have been made over the evolutionary course of the AMMOS, pain points remain. These pain points, based on interviews with numerous flight project mission operations personnel, can be classified principally into two major categories: 1) information-related issues, and 2) process-related issues. By information-related issues, we mean pain points associated with the management and flow of MOS data across the various system interfaces. By process-related issues, we mean pain points associated with the MOS activities performed by mission operators (i.e., humans) and supporting software infrastructure used in support of those activities. In this paper, three foundational concepts—*Timeline*, *Closed Loop Control*, and *Separation of Concerns*—collectively form the basis for expressing a set of core architectural tenets that provides a multifaceted approach to AMMOS system architecture modernization intended to address the information- and process-related issues. Each of these architectural tenets will be further explored in this paper. Ultimately, we envision the application of these core tenets resulting in a unified vision of a future-state architecture for the AMMOS—one that is intended to result in a highly adaptable, highly efficient, and highly cost-effective set of multimission MOS products and services.

I. Introduction

THIS paper describes a set of core architectural tenets that are intended to drive a future-state transformational system architecture for the AMMOS. Work has already initiated on a number of AMMOS modernization tasks that embody some of the core tenets described herein, and in fact serve as the primary source of input to this paper. The work products and artifacts associated with those modernizations initiatives are scattered in many different resources and workspaces and not easily accessible to the interested stakeholder. These core tenets are intended to provide a unifying message for the MGSS program office to help articulate its vision of a future-state architecture for the AMMOS.

II. The Evolvable AMMOS

In this Section, we review describe the evolutionary approach to AMMOS modernization followed by identification of mission system customer pain points that remain. These pain points are categorized as either 1) *information-related* issues or 2) *process-related* issues.

¹ MGSS Chief Architect, Mission Systems Engineering Section, M/S 264-255.

² MGSS Chief Engineer, Mission Systems Engineering Section, M/S 264-255.

A. AMMOS Modernization to Date

The general approach to AMMOS modernization over the course of its 29-year history exemplifies a continual, evolutionary approach with periods of sponsor investment peaks and valleys in between. In more recent past—starting with the formation of the Multimission Ground Systems and Services (MGSS) program office in FY2005—an even more proactive approach to modernization has ensued under a programmatic mandate to “reinvigorate the AMMOS” following a period of a drop in modernization investment in the late 1990s and early 2000s^{1,2} Today, MGSS actively pursues modernization initiatives and continues to evolve the AMMOS by incorporating enhanced capabilities and newer technologies into its end-user tool and service offerings.

To the credit of recent NASA program executive sponsors and MGSS program office management, the evolutionary approach to modernization has addressed a number of pain points typically experienced by missions as part of their Project Mission Operations System (MOS). It is doing so by targeting investments in a number of key areas. These include, for example, replacing obsolete software-intensive systems (some with a heritage from the 1960s) by using modern programming languages and standards in the software applications and improving their operability, testability, and maintainability.* Remediation of dated hardware has occurred replacing aging and unreliable systems (workstations, routers, switches, etc.). Infusion of advanced technologies such as new instrument operations data processing techniques, opportunistic science capture, core and distributed automated planning, and advanced navigation and mission design techniques has greatly expanded the portfolio of fundamental capabilities the AMMOS can offer missions. More recently, modernization investments include revitalization of multimission operations teams and adaptable processes using a systems architecting and model-based engineering approach that is part of an initiative known as Operations Revitalization or “Ops Revitalization” for short. There are also efforts to pro-actively seek opportunities to extend the use and implementation of the AMMOS to a wider NASA community beyond its traditional charter of supporting robotic deep space and astrophysics missions to potentially include support for human-robotic exploration missions as well as a broader set of Earth science missions all resulting in greater collaboration with other NASA Centers and industry partners^{2,4-7}

B. Pain Points Remain

Despite the myriad of modernization investments that have been made over the evolutionary course of the AMMOS, pain points remain. These can be classified principally into two major categories: 1) *information-related issues*, and 2) *process-related issues*. By information-related issues, we mean pain points associated with the management and flow of MOS data across the various system interfaces. By process-related issues, we mean pain points associated with the MOS activities performed by mission operators (i.e., humans) and supporting software infrastructure used in support of those activities.

Starting with process-related issues, historically, many AMMOS multimission operations processes have been optimized for a particular class of missions and spacecraft heritage (e.g., planetary orbiters and fly-bys using a single spacecraft manufacturer). With the possible exception of the functions provided by the Navigation and Mission Design element, this limits the potential to garner other so-called “high-usage” mission customers that are not in this class.³ Organization and function of the AMMOS are convolved, which confuses responsibility with capability or function. The Ground Data System (GDS) functional software applications impose limiting constraints on MOS processes and operations responsiveness such that with the current system, it is difficult to reconcile plans and predictions with observed data (aka “closing the loop”); again, with the exception of functions provided by the Navigation and Mission Design element in which closing the loop is a routine part of the flight mechanics process. More generally, and not necessarily unique to the AMMOS, there seems to be an overall lack of multimission analysis support to address the difficulty and complexity of closing the loop to reconcile plans with observed execution.^{10,11} Today, multimission processes are weak on analysis leaving missions to define their own processes for analysis and closing the loop. In other words, multimission MOS architectures often represent data and disciplines rather than closed-loop system level functionality.

As illustrated in Fig. 1, the fundamental Project MOS architecture lacks evidence of the principle of “separation of concerns” in that behavior, information types and flow, organization, functionality, and interfaces are all comingled. The functional pieces performed by specific teams all tie to most other functions. This leads to a quadratic (the so-called “N-squared”) integration problem and exists for all missions large and small.¹¹ Adding

* An interesting new area of academic research related to this topic is the field of Software Architecture Evolution.^{8,9}

Table 1. Leading information- and process-related issues associated with today’s AMMOS.

Information-Related Issues
Large number of point-to-point interfaces
Integration is costly, laborious, and not easily “trusted”
Large number of file types
Many versions of files and different file types duplicate information
Process-Related Issues
Hard to reconcile plans and predictions with observed data in many cases
Traditional Project and MOS software impose limiting constraints on MOS processes and operations responsiveness
Organization and function are tied together
Lack of multimission analysis support to address the difficulty and complexity of closing the loop to reconcile plans with observed execution

C. Opportunities for Improvement

Each of the information- and process-related issues summarized in Table 1 present opportunities for modernization of the underlying AMMOS architecture to help mitigate their impact on the overall cost, risk, and technical delivery of a Project MOS to a mission. A brief statement of these opportunities and their potential impact against each information and process-related issue is captured in Table 2 and Table 3.

Table 2. Information-related opportunities to modernize the AMMOS architecture against known information-related issues and their potential impact on Project MOS cost, risk, and/or technical delivery.

Information-Related Issue	Information-Related Opportunity	Potential Impact/Benefit
Large number of point-to-point interfaces	Reduce the numbers of point-to-point interfaces	➤ Reduce cost to update, re-engineer, and maintain interfaces
Integration is costly, laborious and not easily “trusted”	Improve the integration and test approach	➤ Reduce cost of testing, deployment, and simplify updates for minor changes
Large number of file types	Reduce the number of file types	➤ Reduce cost to develop and maintain software and simplify translations
Many versions of files and different file types duplicate information	Define a definitive source of information	➤ Reduce cost to develop, maintain, and operate file version management functions and procedures ➤ Eliminate ambiguity

Table 3. Process-related opportunities to modernize the AMMOS architecture against known process-related issues and their potential impact on Project MOS cost, risk, and/or technical delivery.

Process-Related Issue	Process-Related Opportunity	Potential Impact/Benefit
Hard to reconcile plans and predictions with observed data in many cases	Improve the system’s ability to reconcile plans and predictions with observed data	➤ Reduce cost associated with effort to “close the loop”
Traditional Project and MOS software impose limiting constraints on MOS processes and operations responsiveness	Design software to be directly responsive to operational processes (“operationally-responsive software”)	➤ Reduce the cost associated with number of needed “Workarounds” (e.g., custom scripts, “glueware,” etc.) ➤ Reduce risk of command-related errors ➤ Allow for parallel and collaborative processes unconstrained by serialized

Process-Related Issue	Process-Related Opportunity	Potential Impact/Benefit
		software applications in critical path
Organization and function are tied together	Improve integration of process flows across the MOS	➤ Tasks and roles (expertise) needed to perform them are considered first, then assigned to Teams (i.e., missions can organize as needed, without affecting Ops processes)
Lack of multimission analysis support to address the difficulty and complexity of closing the loop to reconcile plans with observed execution	Closing the loop becomes one of the unifying tenets of the MOS	➤ Paradigm shift (<i>in terms of AMMOS support perspective, not projects/missions</i>): MOS performs more than “Uplink” and “Downlink,” in other words, an essential focus of the MOS function is to fly a spacecraft under positive (closed-loop) control

While some of these opportunities to modernize the AMMOS architecture seem relatively straightforward and evolutionary in nature, many are not. Many are quite transformational and require a significant paradigm shift both in thinking about the AMMOS in light of supporting a future-state vision for the MOS as well as requiring novel technical solutions to address the challenging problems at hand.

This forms the basis and motivation for what is described next in Section III where we introduce the concept of the “Transformational AMMOS.” We fully recognize the fact that any AMMOS architecture modernization effort of any substantial scope must be balanced against available programmatic and sponsor resources and the need to support legacy ground systems and mission customers currently using the AMMOS as part of their Project MOS, both today and for the future. We are also not naïve to the fact that there will never be enough resources in terms of people, time, and sponsor funding to support a “big bang” approach to architecture modernization. That would be more akin to a *revolutionary* approach to modernization versus the *transformational* approach that we are proposing.[‡]

Despite these challenges and constraints, it is felt that a new approach to architecture modernization is needed to truly address the core information- and process-related issues associated with today’s Evolvable AMMOS. Fortunately, we are well on our way by virtue of a tremendous amount of collaboration across the programmatic elements within the MGSS program office and novel solution approaches being offered by the technical community supporting currently funded AMMOS modernization initiatives.

III. The Transformational AMMOS

In this Section, we introduce the foundational concepts of *Timeline*, *Closed-Loop Control*, and *Separation of Concerns (SoC)* that collectively serve as the basis for expressing three core architectural tenets for the future-state Transformational AMMOS. Each of these core tenets is designed to address the major information- and process-related issues associated with the current Evolvable AMMOS as summarized in Table 1 of Section II.B and is described and detailed in Sections III.A through III.C.

A. Timeline as the Foundational Data Structure of our Domain

It is well recognized that the “lifeblood” of an MOS is time-varying information.¹⁷ For example, activities have start times and durations, sequences have to be developed by a certain time, spacecraft have to arrive at a target by some specific time, Principal Investigators (PIs) expect their data after a certain time. In today’s AMMOS, however, this time-varying information for the MOS is scattered among various non-standard file formats as defined by a myriad of AMMOS Software Interface Specifications (SISs).

[‡]*Transformational* as used in this context is intended to be characterized as implementing and operating with carefully chosen paradigm changes that can be incrementally incorporated into the AMMOS over time. These include information-related changes as well as process-related changes.

What is proposed for the Transformational AMMOS is the introduction of the concept of “Timeline” as the unifying canonical (common/standardized) information model for the storage and communication of MOS time-varying information. The perceived benefit is that through use of a unifying representation of MOS data as timelines, the method of integration between the functional software applications will decrease adaption cost. In addition, operations efficiency will increase because historically segregated elements will be more easily integrated so that there will be fewer gaps in the operations processes that must currently be closed (if they are closed at all) by costly and inefficient means.^{18,19} This will also serve as a basis to help address end-to-end data accountability throughout the MOS. Of course, it is recognized that some data types will remain outside of the timeline domain for a considerable period of time.

1. Introduction to the Concept of Timeline

The concept of “Timeline” has long lineage to the advanced planning and scheduling community, particularly with respect to automation, and has a formal mathematical basis in the fields of temporal constraint networks, constraint programming, and predicate logic.^{21,22} Generally speaking, a *timeline* can be defined (informally) as some representation of time-varying information; more specifically, a representation of a set of values with associated times. A *value* can be a numeric or a non-numeric quantity. The time domain of a timeline may be discrete or continuous. A *discrete timeline* represents a set of values of discrete instances of time while a *continuous timeline* represents a set of values over a continuous interval of time.

As suggested earlier, for purposes of our domain, most MOS data can be well-represented as a time-ordered sequence of events (i.e., timelines). For example, a planned activity or sequence over time, a planned instrument command over time, a predicted instrument state over time, actual science data or power usage over time as captured in various telemetry channels, and estimated instrument or heater states over time. Examples of such timeline representations of time-varying data in the MOS domain are illustrated in Fig. 2.²⁴⁻²⁶ Although not the case for today’s Evolutionary AMMOS in which time-varying information is scattered in various file-based SISs, it is natural to use timelines as a basis for a canonical (common/standardized) information model for our domain of MOS although this is not the case today.

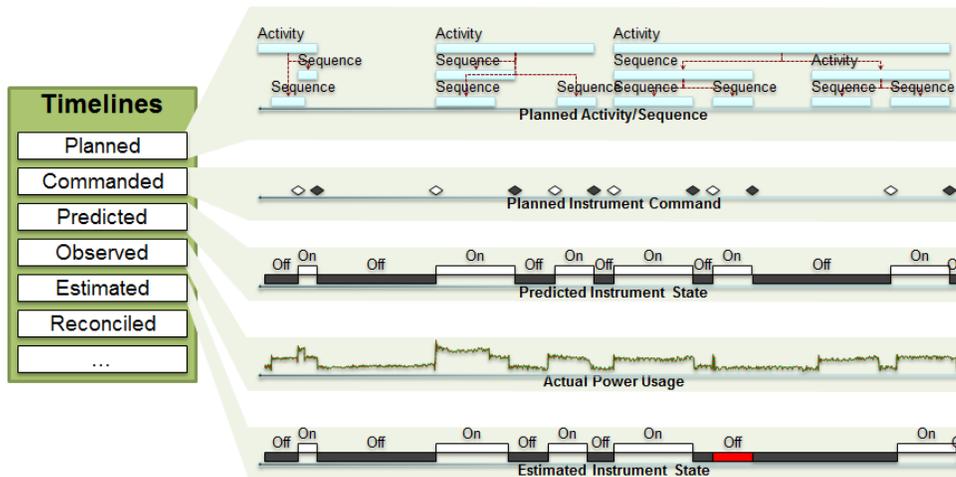


Figure 2. Example timelines for representing time-varying MOS data.

More formally, timelines are aggregations of events (and temporal constraints among those events) that are ordered by a specific native temporal reference. In fact, a formal mathematical basis for timeline is described in an AIAA SpaceOps 2012 paper by S. Chung and D. Bindschadler, which characterizes a timeline as a triple comprised of *variables*, *temporal constraints*, and *events*.²⁷ It is in fact this formal mathematical representation on which the unified timeline information model for the Transformational AMMOS is based and is currently being captured in both a formal system-level object model as well as a formal ontology.[§]

Where formalism is required for precise modeling of timeline information semantics, we refer to the unified timeline information model using the mathematical basis cited in Ref. 27 as our foundation. For purposes of general discussion, we utilize the informal notion of timeline as a representation of a set of values with associated times.

[§]An *ontology* is a set of unifying concepts, axioms, and relationships within a particular problem domain.

This distinction between a formal or informal definition of timeline is not particularly important to dwell upon for purposes of this introduction to the concept. What is important is that timelines provide a powerful way to model the temporal evolution of a system as they provide an abstraction of the changing state of that system, which can be manipulated and reasoned about.

2. Practical Implementation of Timelines

In order to realize the potential benefit of using timelines as the foundational data structure for the future-state Transformational AMMOS, a practical means of defining the syntax (structure) and semantics (meaning) of timelines in underlying infrastructure support software such as a relational database is needed. Timelines need to be rigorously versioned and each version needs to be immutable (i.e., absolute and irreversible) such that a versioned timeline name forever represents exactly the same contents.¹⁸⁻²⁰ Consequently, the name is as good as the contents. This alleviates the need to keep files of contents for communicating between functional software applications as well as operations processes (or for associating several timelines or even values on those timelines, or for keeping a record of past values).¹⁸

The key concepts used to support the practical implementation of timelines to be stored and managed in application platform infrastructure software such as a Relational Database Management System (RDBMS) and are characterized in greater detail in Refs. 18-20 and thus will not be repeated here.

3. Timeline Integration Patterns and Components

The key architectural concept that we are proposing for the Transformational AMMOS is that timelines become the common representation of time-varying MOS information, and future-state AMMOS components read and write timelines from a centrally accessible timeline information store (see Fig. 3).²⁶

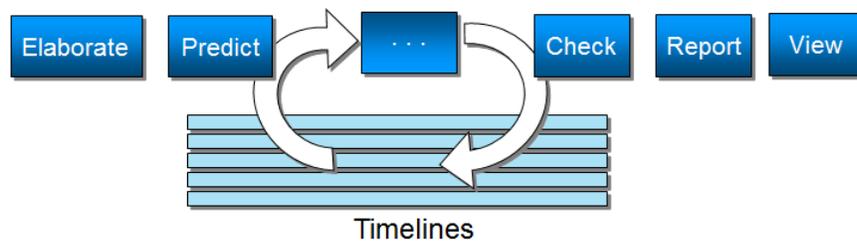


Figure 3. Notional depiction of components that read and write timelines.

A *component* in this context is a piece of software that read and writes timelines.[†] The power of this model is in having a common, centrally accessible representation for all MOS temporal information instead of having MOS information scattered across several applications and in many different, non-standard file formats.²⁶ Components can focus on one set of tightly focused concerns, making them easier to develop and maintain than large monolithic systems.

In this new architecture, the components themselves will not be permitted to make direct access to the timeline store (e.g., timeline database (TLDB)) but rather access will be provided through the TLDB published interface that will be offered as a common (shared) software service.¹⁸ In addition to simply reading and writing timelines, this service will provide versioning and querying capability (e.g., “give me the heater timeline between 5pm and 10pm on Sol 27”).²⁶ A companion utility service is also envisioned for the new architecture that will be used to provide additional functions such as discovery (“what timelines are there, what you can tell about them”), determining relationships between timelines, and management of timeline metadata.

Components can be organized into a few categories such as *elaborators*, *predictors*, *checkers*, *viewers/reporters*, and *converters*. A few of these component types were illustrated in Fig. 3 but are depicted in Fig. 4 in a conceptual context relative to the notion of a central timeline service with an *orchestrator* component suggested as the primary “controller” component that could coordinate the execution of the other components.

[†] Such components can be thought of as candidate mathematical operations that can be performed on a set of time-dependent functions, in this case, timelines.

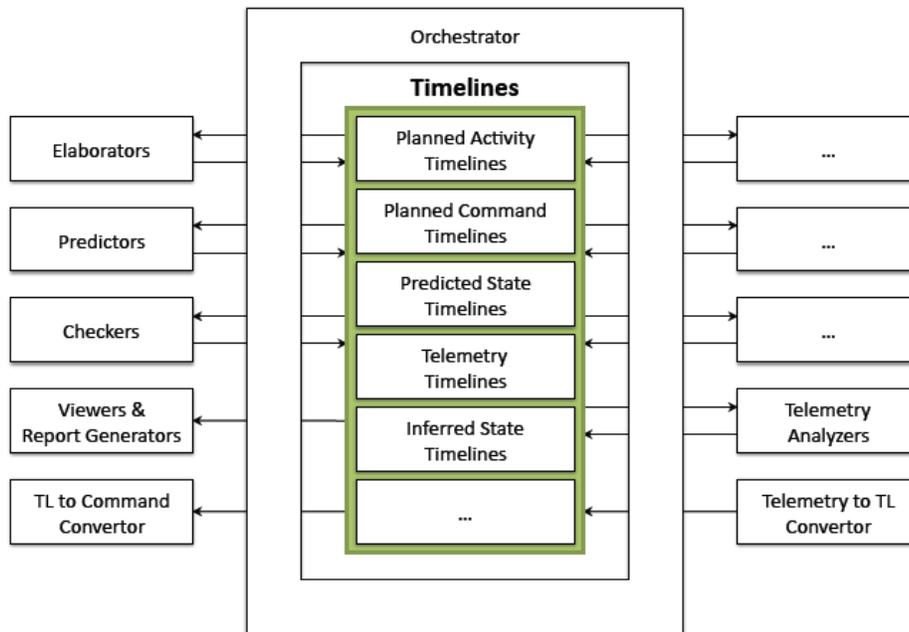


Figure 4. Conceptual depiction of components and their interactions with a central timeline service.

A brief description of each of the primary components for operating on timelines and coordinating their interaction is summarized in Table 4.

Table 4. Description of the basic software component types that interact with a central timeline service.

Component Type	Component Type Description
Elaborator	An elaborator component expands higher-level activity timelines into lower-level activity or command timelines with more details.
Predictor	A predictor component predicts values on a set of timelines based on activities or commands on other timelines.
Checker	A checker component checks a set of timelines for undesired or missing values.
Converter	A convertor component converts information from a timeline format to another format or from another format to a timeline format.
Derivator	A derivator component mathematically derives a new timeline from one or more existing timelines.
Orchestrator	An orchestrator component provides an interface to a user for invoking elaborators, predictors, checkers, convertors, and derivators.

These basic components can be abstracted to a set of timeline integration patterns for the Transformational AMMOS.[#] In fact, each of these components could be describe as integration component patterns in an analogous manner to the set of Enterprise Integration Patterns from G. Hohpe and B. Woolf that are widely cited in industry for application integration (see <http://www.eaipatterns.com/>).¹⁴ Additional patterns and supporting components can be specified as needed.

The core architecture tenet of *Timeline as the Foundational Data Structure of our Domain* serves to address three of the information-related issues and two of the process-related issues of the current Evolvable AMMOS as noted in Table 5.

[#]A *pattern* is essentially a description to a known recurring problem and its solution in a particular context, and to communicate this knowledge to others.^{29,30} Each pattern represents a decision that must be made and the decisions that go into that decision. A *pattern language* then is a web of related patterns where each pattern relates to others, guiding one through the decision-making process.^{14,30}

Table 5. Information-and process-related issues addressed by the core architectural tenet of “Timeline as the Foundational Data Structure of our Domain.”

	Information-Related Issues		Process-Related Issues
	Large number of point-to-point interfaces	✓	Hard to reconcile plans and predictions with observed data in many cases
✓	Integration is costly, laborious, and not easily “trusted”		Traditional Project and MOS software impose limiting constraints on MOS processes and operations responsiveness
✓	Large number of file types		Organization and function are tied together
✓	Many versions of files and different file types duplicate information	✓	Lack of multimission analysis support to address the difficulty and complexity of closing the loop to reconcile plans with observed execution

B. MOS as a Closed-Loop Control System

Deep space mission operations require precision command and control capabilities on the ground. If the MOS fails to properly control its flight system, the mission can be lost. In the Transformational AMMOS, the MOS is envisioned as a *Closed-Loop Control* system tasked with achievement of mission and science goals, managing mission resources, and being capable of closing the loop on goals and resource management.^{23,28} For an MOS to function as a control system it must be capable of closing the loop on specified objectives and states, that is, providing analysis to enable reconciliation between predicted and observed states and objectives.

A key aspect of the MOS is that it also includes the human element as part of the system definition. This means that humans are at some level always in the control loop.^{23,28} Some control loops can be closed automatically in terms of lower-level state, but understanding the achievement of higher level goals where that must address performance and/or knowledge about off-nominal conditions, typically need to be capable of being analyzed and reported by the human operator.

Two key architectural patterns emerge from the core architectural tenet of MOS as a closed-loop control system.^{23,28} The first pattern, which is referred to as the *external interface pattern*—a behavioral pattern—identifies the fact that, as a control system that must serve or collaborate with other systems, the MOS must interact with external systems in an accountable way. These are typically external exchanges of planning products and measurements with systems such as the Deep Space Network (DSN) or the actual Flight System (see Fig. 5). There are three aspects of this behavioral pattern: 1) where the MOS is interfacing with a system under control, this pattern is required if the MOS is to perform its essential function, 2) if it is merely “collaborating” with a peer system, the pattern is useful for ensuring that the external system can close any control loops for which it is responsible and that the MOS is being true to its own “close the loop” principle, and 3) interacting with customer input such as science or project needs. While interactions with external entities have always been a part of the MOS, this level of formality and accountability has not always been present.



Figure 5. External interaction [behavioral] pattern as envisioned for a future-state MOS. This generalized pattern is specialized for each interface with an external entity (e.g., Flight System, DSN, science data archive, etc.).

The second pattern provides clarification of the fundamental concept for any MOS. The MOS *control loop pattern* illustrated in Fig. 6—also a behavioral pattern—represents the central control concept within the MOS as a closed-loop control system that includes the primary common MOS functions of Planning, Execution, and Analysis (PEA) and the flow of planned, approved, observed, and analyzed information.

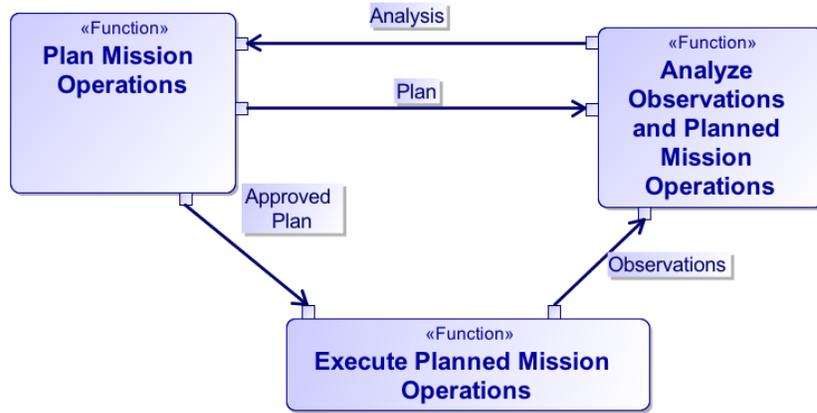


Figure 6. Idealized MOS closed-loop control [behavioral] pattern.

This fundamental MOS control loop pattern provides an explicit functional specification for how any deep space MOS ought to behave.^{23,28} It effectively clarifies and formalizes a unifying concept of operations in which all elements “know” their parts in achieving overall goals. Models that follow this pattern still permit views that show other important aspects of the system such as uplink or downlink; the key addition is the (closed) prediction-reconciliation loop. This loop explicitly requires that any of the three functions of PEA of an MOS be supported by the other two, given their input-output dependencies and its facilitation of the key tasks of reconciliation (e.g., plans against results, predicts against actuals). Again, while these common control functions have always been a part of the MOS, the level of formality has varied and closed loop reconciliation has been difficult to achieve across the full scope of the AMMOS.

1. Timeline-Based Closed Loop Architecture

By adopting timelines as unifying information model as was described in Section III.A, we capture the necessary behaviors, states, and constraints needed for the MOS to simplify and unify generation of products to command and control mission assets. This basic framework provides specialization options that can span the transition from the current file-based AMMOS information products to sophisticated information products that fully support a fully reconciled MOS.^{23,28}

As an example, we can overlay the primary functions of the MOS PEA control loop pattern (Plan, Execute, Analyze) with a set of timelines for the mission as illustrated in Fig. 7. The Planning function produces a collection of timelines that capture all of the intended states for a future uplink opportunity. The Execution function records the results and observations as timelines, and the Analysis function retrieves them along with the predicted timelines (from the Plan function). The Analysis function reconciles the predicted timelines with the observed timelines and updates the known states of the spacecraft for a future planning cycle.

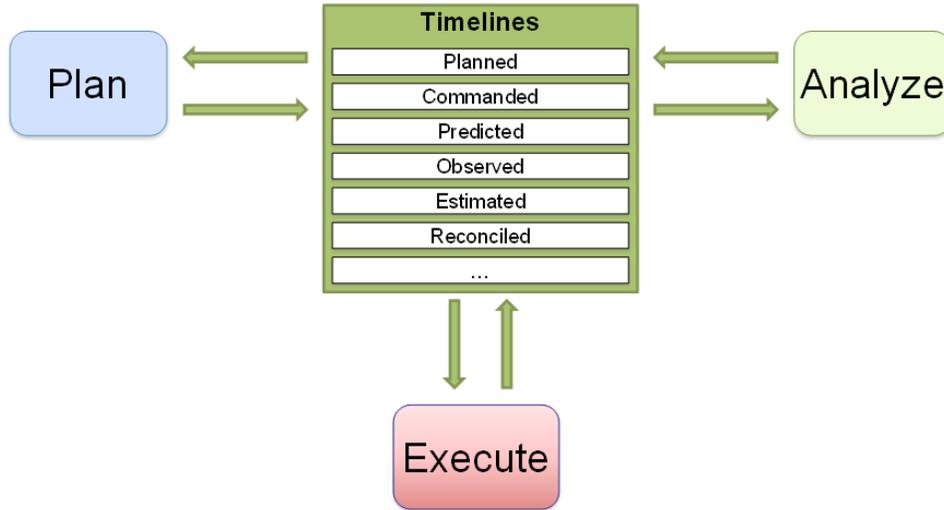


Figure 7. Timeline-based closed loop control pattern.

This approach to a timeline-based closed loop control pattern greatly simplifies the ability to reconcile plans and predictions with observed data, which today—again, with the possible exception of navigation system operations—is a labor-intensive set of ad hoc or implicit processes necessary to “close the loop.”

2. Toward Goal-Based/Goal-Directed Operations Engineering

As part of the Ops Revitalization initiative, a formal information modeling activity is underway that synthesizes concepts from the AMMOS, State Analysis/Mission Data System (MDS), and the Automated Scheduling and Planning ENvironment (ASPEN) Modeling Language to introduce information concepts that facilitate the achievement of mission and science goals and management of mission resources.^{23,28,31,32} These concepts center on temporally constrained behavior models and timeline models, which consist of the states and events of interest along with the temporal and value constraints placed upon those states and events. While the basis for these concepts is relatively straightforward, the details of its application and the interrelationships between state, event, and constraints (both value and temporal) are inherently complex.

Considering the fact that the MOS must command and control the mission assets, these information models become the medium of exchange between the services and associated processes that permit the effective operation of the mission. Since we wish to use our goals (e.g., the successful collection of scientific observations) as the means by which the system is directed and controlled, the information must account for dynamics and must be structured to easily support comparison of predicted to measured or observed outcomes.^{23,28} With such feedback, we are able to meet our goals of ensuring and improving upon performance and reliability.

Under the aegis of the Ops Revitalization initiative, a future state AMMOS is being architected that will be consistent with a core concept adopted from the State Analysis methodology of differentiating the “Control System” from the “System Under Control” and one that supports a methodological and rigorous approach to goal-based/goal-directed operations engineering.³¹

The core architectural tenet of *MOS as a Closed-Loop Control System* as described herein serves to address two of the four the underlying process-related issues of the current Evolvable AMMOS as noted in Table 6.

Table 6. Information-and process-related issues addressed by the core architectural tenet of “MOS as a Closed-Loop Control System.”

	Information-Related Issues		Process-Related Issues
	Large number of point-to-point interfaces	✓	Hard to reconcile plans and predictions with observed data in many cases
	Integration is costly, laborious, and not easily “trusted”		Traditional Project and MOS software impose limiting constraints on MOS processes and operations responsiveness
	Large number of file types		Organization and function are tied together

	Many versions of files and different file types duplicate information	✓	Lack of multimission analysis support to address the difficulty and complexity of closing the loop to reconcile plans with observed execution
--	---	---	---

C. Institutionalizing the Practice of Separation of Concerns (SoC)

We recognize from Table 5 and Table 6 that while the two foundational concepts of *Timeline* and *Closed-Loop Control* address some of the major information-related issues and some of the process-related issues, they alone do not address all of the information- and process-related issues such as large number of point-to-point interfaces and software imposing on process and mission timelines, and organization and function being tied together. For this, we leverage the industry best practice of *Separation of Concerns (SoC)*. The purpose of a strict SoC is to keep independent things independent, so that a change in one part of the system does not adversely affect other parts.**

One manner in which to achieve a SoC is application of the basic architectural pattern of “layering.” *Architectural Layering* describes a logical separation of functions such that each layer has a specified set of roles and responsibilities.¹³ A very powerful advantage to a layered approach to architecture is that it not only facilitates a SoC but also a separation of development and administrative roles, meaning that the right mix of expertise and skills can focus one particular layer without having to have deep knowledge of the others. What is critically important for the Transformational AMMOS is that we not only develop a set of agreed-upon architectural layers to notionally depict a logical separation of functional roles and responsibilities, but to also institutionalize the practice of SoC in order to fully realize its benefits.

The logical separation for architectural layers being put forward as candidates for the Transformational AMMOS is based on the desire to separate multimission operations processes (i.e., business processes) from the underlying Ground Data System (GDS) capabilities that support those processes, and to further layer the GDS capabilities. In the case of the GDS capabilities, for example, we logically separate functional (subsystem) software applications from common cross-cutting utility functions that can be used across the functional applications such as application-level security. And these from the underlying application platform infrastructure supported by third-party software (TPS) and operating system (OS) functionality as well as Virtual Machine (VM) and hardware (HW) resources provided by the hosting institution. Other provided infrastructure resources by the hosting institution would include the communication networks and facilities but could also be extended to include local IT security, enterprise systems management, and backup & recovery including disaster recovery. Collectively these form what is usually referred to in industry as a “shared infrastructure.” A notional depiction of this layered approach is illustrated in Fig. 8.

** Source: http://en.wikipedia.org/wiki/Separation_of_concerns.

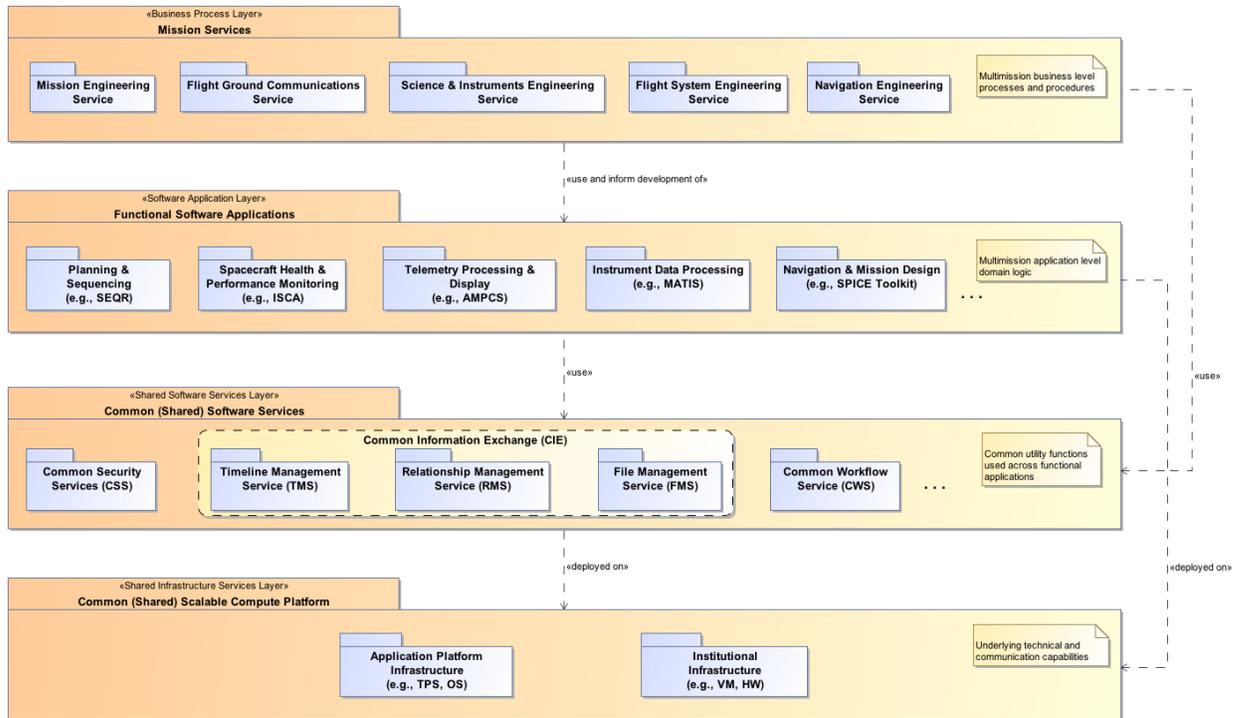


Figure 8. Architectural layering depicting a logical separation of concerns (SoC) between major Operations and GDS elements of the Transformational AMMOS. Note that this is not intended to impose strict hierarchy between layers as some lower layers may be used by multiple upper layers as seen through the dependency relationships (labeled dashed arrows).

The **Business Process Layer** depicted at the top of the layer diagram in Fig. 8 correspond to future-state, multimission Mission Services, which are being specified in a formal model-based context as part of the Ops Revitalization initiative described in Ref. 28.^{††} This layer provides the primary interface to Project elements that are external to the MOS. It also provides the interfaces between MOS internal functions and software, and between individual services.

Each discipline-based Mission Service has responsibility for managing sets of mission information. These services provide the capabilities (or functions) need to operate a mission.^{3,17,23,28,31}

Note: “Services” as defined here represent a slice through the full stack of architectural layers as we are defining here and as depicted in Fig. 8. We also recognize the fact that the generalized concept a service is inherently hierarchical and fractal in nature and thus we see still see architectural layering a very relevant pattern and extremely important in our efforts to institutionalize the practice of SoC. One can think of it as applying the metaphor of “peeling the onion.”

The **Software Application Layer** represents AMMOS functional software applications for various subsystems and/or assemblies that correspond to the classical AMMOS functional areas of Planning & Sequencing, Downlink, Navigation & Mission Design, GDS Integration, Test, Deployment, and Support, and Operations Engineering as traditionally depicted in Fig. 9. These functional software applications are available to prospective mission customers as either AMMOS “Tool” or “Service” offerings, which are described in the online AMMOS Tools and Services catalog and available to missions via coordination with the MGSS Mission Interface Office.^{‡‡} Functional software applications offered as tools can be adapted to a project per the mission’s specific requirements.

Some of the functional applications shown as examples for various functional areas and subsystems within those areas as depicted in the architecture layer diagram of Fig. 8 correspond to newer applications that have been developed either in recent past such as the AMMOS Mission data Processing & Control System (AMPCS) or are

^{††}Generally speaking, a *service* can be thought of as a capability offered according to an agreement, where *capability* is the ability to do something (perform a task, activity, or function or set of tasks, activities, or functions) based on expertise and capacity.

^{‡‡}AMMOS Catalog (see https://ammos.jpl.nasa.gov/AMMOS_Catalog/index.cfm).

currently under development such as Sequence Revitalization (SEQR). Some applications are being proposed for future development such as Integrated Spacecraft Analysis (ISCA). Still others represent existing functional software applications available to missions today such as the instrument data processing Automated, Multimission Instrument Task Invocation (MATIS) tool and the Navigation and Ancillary Information (NAIF) SPICE Toolkit. It should be noted that the functional software applications shown in Fig. 8 are representative of only a small subset of the available software capabilities from the AMMOS Tools and Services catalog or that will be available as future capabilities.

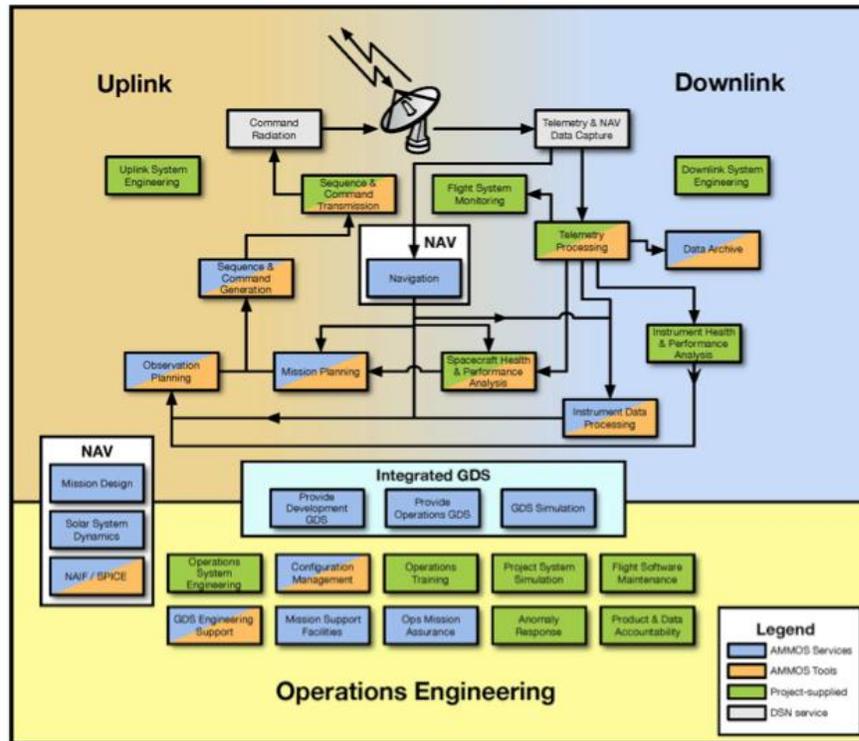


Figure 9. Classical representation of AMMOS functional capabilities.

Where we envision a transformational shift that is different from traditional evolutionary modernization efforts of the AMMOS in the past is that newly minted and/or modernized functional software capabilities exhibit the characteristic that they are responsive to multimission MOS business processes—what we like to refer to as “operationally-responsive software.” This is similar in vision to industry trends in the aerospace industry such as the U.S. Defense Department’s Operationally Responsive Space (ORS) and 2008 Ground System Architecture Workshop (GSAW) theme of “Operationally Responsive Ground Systems.”^{§§,¶¶} What this requires is the decomposition of monolithic software applications into distributed modular software components or “agents” that map to multimission MOS operations processes as well as the potential automation of those business processes.

One example of a functional application within the Planning & Sequencing area that is being architected as an operationally-responsive software offering is the Sequence Revitalization (SEQR) application, currently being developed as part of the SEQR initiative within the Mission Planning and Sequencing (MPS) program element. As of the time of this writing, there is active dialog between key SEQR and Ops Revitalization initiative management and engineering staff to help ensure the new SEQR functional software application is readily adaptable (i.e., responsive) to mission operations processes and any changes in those processes. This collaborative process is intended to serve as a model for modernization of existing software applications as well as development of new applications across the MGSS program elements and functional areas and subsystems within those elements.

The **Shared Software Services Layer** represents the common software services that are to be “shared” for use by any or all of the functional software applications in the Software Application Layer. In some cases, the

^{§§} Operationally Responsive Space (see <http://ors.csd.disa.mil/>).

^{¶¶} GSAW2008 (see <http://sunset.usc.edu/gsaw/gsaw2008/agenda08.html>).

multimission Mission Services in the Business Process Layer may use these shared software offerings; for example, a Common Business Process Management (BPM)/Workflow Service (CWS) in support of automated multimission operations business processes. These shared software offerings provide common utility functions that are highly cross-cutting in nature and by which all AMMOS applications at the functional level will be encouraged to use rather than the highly cost ineffective approach of standing up or provisioning these separately and independently for each functional area.³⁴ The objective of utilizing shared software offerings that provide common utility functions is to reap the benefit of economies of scale that can be achieved when using a common set of application programming interfaces (APIs), training, industry standards, best practices, and system administration.

Often these shared software offerings will require the provision of a set of (preferably) industry-standard capabilities that are implemented using Off-The-Shelf (OTS) components, whether Commercial-Off-The-Shelf (COTS), Government-Off-The-Shelf (GOTS), Modified-Off-The-Shelf (MOTS), or Open Source Software (OSS). For solutions where high reliability, availability, and scalability are needed, a COTS offering will most likely need to be provisioned to support such capability. One example would be a robust, commercial-grade Relational Database Management System (RDBMS) to support ultra-high transaction rates that need the assurance of referential integrity and concurrency control. The assessment and provision of a candidate COTS offering can be an expensive undertaking. This, together with the previously cited opportunity for gaining economies of scale, are some of the key reasons why in this new era of the Transformational AMMOS we cannot permit each functional application area to choose its own set of cross-cutting utility software without first addressing AMMOS system-wide concerns.

Another example of such a shared software capability that provides cross-cutting utility functions and identified in the third layer of Fig. 8 includes the Common Security Services (CSS) (e.g., access management, key management & cryptography), currently being developed and offered within the Computing, Communications, and Configuration (CCC) program element. This offering is in the process of providing application-level security functions of authentication, authorization, and auditing.^{35,36} With respect to scope, the authorization function offered by the CSS is also intended to provide support for common management of security policy.

Other shared software capabilities that are currently being architected and developed to support the Transformational AMMOS includes a set of so-called “Common Information Exchange (CIE)” services, which comprises the Timeline Management Service (TMS), the Relationship Management Service (RMS), and the File Management Service (FMS). The TMS is a new, shared software capability that provides a means to store and manage definitive sources of operations data that is based on timelines. (Recall *timeline* is one of the three foundational concepts for the Transformational AMMOS that we described in Section III.A.) The RMS is also a new service, which is being architected and designed as a means to store and manage definitive sources of operations metadata and data relationships.³⁷ Discovery services such as registry-repository capabilities are example utility functions that will be supported by the RMS. The FMS, also a new service, provides a means to store and manage definitive sources of non-timeline operations data (e.g., command and telemetry dictionaries, data products, etc.).^{##}

By no means an exhaustive list but additional shared software offerings such as the common notification services, enterprise systems management services, logging services, etc. are being considered and will be prioritized based on anticipated future need to support the Transformational AMMOS for future infusion targets (e.g., mission deliveries, engineering releases, etc.).

Finally, the **Shared Infrastructure Services Layer** represents a common scalable compute platform that is to be “shared” for the deployment of the functional software applications in the Functional Software Application Layer and the common software services in the Shared Software Services Layer. This layer includes Application Platform Infrastructure and Institutional Infrastructure. Application Platform Infrastructure is intended to represent the collection of all third-party application platform support software such as application server middleware on which distributed components of the functional software applications and the shared software services are deployed as well as other third-party support software such as Access Managers (AMs), Database Management Systems (DBMSs), and Business Process Management Suites (BPMSs) to name a few. Some of these offerings such as enterprise-class application servers provide native support for additional runtime qualities of service including workload management (i.e., load balancing and failover management), native security capabilities, auditing and logging, and systems management. Ideally, these capabilities would be capable of transparent integration with support infrastructure capabilities offered by a hosting institution.

Within the AMMOS context, the core collection of Off-The-Shelf (OTS) support software together the approved set of baseline Operating System (OS) software is officially referred to “Third-Party Software (TPS).”³⁸ It is

^{##} The term “File” in File Management Service (FMS) is a bit of a misnomer in that it does not necessarily imply that non-time ordered (i.e., non-timeline) operations will be managed by a file system. It just means MOS information that has been traditionally file-based.

important to note, however, that additional OTS support software may be included as part of the overall Application Platform Infrastructure. These may be candidate OTS products that are being used to support engineering deliveries of functional applications and/or shared software services that require further additional systems engineering investigation prior to being included as part of an ‘official’ list of TPS components.

Just as it is prudent to gain economies of scale in the cross-cutting utility functions provided by the core set of shared software services, similar economies of scale can be gained by utilizing a fixed set of approved OTS third-party software for use by the functional software applications and shared software services.

Institutional Infrastructure is intended to represent the actual hardware-oriented runtime environment that runs the TPS components, OS, and other support software that make up the Application Platform Infrastructure; for example, virtual machines (VMs), and server hardware (HW). In addition, it is expected that institutional infrastructure would also be comprised of communication networks and facilities such as Mission Support Areas (MSAs) and Science Operations Centers (SOCs). Additional capabilities that are typically offered as part of Institutional Infrastructure include IT security such as providing institutional management and stores of identity and credential information, storage and backup management, enterprise systems management, and disaster recovery.

Some Institutional Infrastructure may actually be provisioned that is physically external to the institution, yet managed and offered by the institution. The most notable examples envisioned for the Transformational AMMOS would be VM and Cloud Computing*** services from an elastic cloud services provider such as Amazon Web Services (AWS), AWS GovCloud for ITAR-sensitive data, Google Apps for Business, or Microsoft Azure and Cloud Services environments.⁴⁹

In summary, the core architectural tenet of *Institutionalizing the Practice of Separation of Concerns (SoC)* as described herein serves to address the one remaining information-related issue and two remaining process-related issues as shown in Table 7.

Table 7. Information-and process-related issues addressed by the core architectural tenet of “Institutionalizing the Practice of Separation of Concerns (SoC).”

	Information-Related Issues		Process-Related Issues
✓	Large number of point-to-point interfaces		Hard to reconcile plans and predictions with observed data in many cases
	Integration is costly, laborious, and not easily “trusted”	✓	Traditional Project and MOS software impose limiting constraints on MOS processes and operations responsiveness
	Large number of file types	✓	Organization and function are tied together
	Many versions of files and different file types duplicate information		Lack of multimission analysis support to address the difficulty and complexity of closing the loop to reconcile plans with observed execution

Taken collectively, all three core tenets serve to address the major information- and process-related issues that were described and summarized in Section II.B for the Evolvable AMMOS.

IV. Conclusion

In this paper, three foundational concepts were elaborated—*Timeline*, *Closed Loop Control*, and *Separation of Concerns*. These three concepts collectively form the basis for expressing a set of core architectural tenets that

*** Burton Group (now part of Gartner, Inc.) defines “cloud computing” as “*The set of disciplines, technologies, and business models used to deliver IT capabilities (software, platforms, hardware) as an on-demand, scalable, elastic service.*”⁴⁰ According to the cited Burton report, the cloud computing can be characterized by five essential characteristics: 1) it uses *shared infrastructure*, 2) it provides *on-demand self-service*, 3) it is *elastic and scalable*, 4) it is *priced by consumption*, and 5) it is *dynamic and virtualized*. The cloud offers four categories of service: 1) Software as a Service (SaaS), 2) Platform as a Service (PaaS), 3) Software infrastructure as a Service (IaaS), and 4) Hardware infrastructure as a Service (HaaS). Cloud computing is deployed using one or more of five models: 1) a *public cloud* offers IT capabilities as a service to any consumer over the public Internet, 2) a *private cloud* offers IT capabilities as a service to a select group of consumers, 3) an *internal cloud* is a private cloud by which an IT organization offers an IT capability as a service to its own business, 4) an *external cloud* is an IT capability offered by a service provider to a third-party business, and 5) a *hybrid cloud* is an IT capability offered as a service using both internal and external IT resources.

address a multifaceted approach to AMMOS system architecture modernization intended to address information- and process-related issues as expressed by flight project mission customers. These foundational concepts and associated tenets form a basis for articulating a unified vision for a future-state AMMOS system architecture—one that is intended to result in a highly adaptable, highly efficient, and highly cost-effective set of multimission MOS products and services.

Acronyms and Abbreviations

AM	Access Manager
AMMOS	Advanced Multi-Mission Operations System
AMPCS	Advanced Mission Data Processing and Control System
API	Application Programming Interface
BPMS	Business Process Management Service (or System or Suite)
CCC	Computing, Communications, and Configuration
CSS	Common Security Service
COTS	Commercial-Off-The-Shelf
DISA	Deep Space Information Systems Architecture
DM&A	Data Management & Accountability
DSN	Deep Space Network
GDS	Ground Data System
GOTS	Government-Off-The-Shelf
GSAW	Ground System Architecture Workshop
HW	Hardware
IMS	Information Management Service
ISCA	Integrated Spacecraft Analysis
ITAR	International Traffic in Arms
M2D2	MOS 2.0 Design and Development
MATIS	Automated, Multimission Instrument Task Invocation
MDAS	Mission Control, Data Management, and Spacecraft Analysis
MDS	Mission Data System
MGSS	Multimission Ground System and Services
MOS	Mission Operations System
MOTS	Modified-Off-The-Shelf
MPS	Mission Planning and Sequencing
NAIF	Navigation Ancillary Information Facility
NCI	Network Communications and Infrastructure
NS	Notification Service
ORS	Operationally Responsive Space
OS	Operating System
OSS	Open Source Software
OTS	Off-The-Shelf
PEA	Planning, Execution, and Analysis
PI	Principal Investigator
RDBMS	Relational Database Management System
SEQR	Sequence Revitalization
SoC	Separation of Concerns
SIS	Software Interface Specification
TL	Timeline
TMS	Timeline Management Service
TPS	Third Party Software
VM	Virtual Machine
WG	Working Group
XML	Extensible Markup Language

Acknowledgements

In addition to MGSS program and element managers, Jeff Estefan and Brian Giovannoni would like to thank the following members of the early MOS 2.0 Design and Development Working Group (M2D2 WG) and Operations Revitalization Team for providing valuable input and feedback to this deliverable (listed alphabetically by last name): Louise Anderson (formerly at JPL), Bob Barry (retired), Duane Bindschadler, Carlos Carrion (formerly at JPL), Seung Cheung, Chris Delp, Elyse Fosse, Brian Giovannoni, Daniel Hurley, Adans Ko, Doris Lam, Scott Lewicki, Michelle McCullar (formerly at JPL), John McKinney (retired), Kenny Meyer (retired), Dave Noble (formerly at JPL), Mike Pajevski, Kirk Reinholtz, George Rinker, Dave Santo, Marc Sarrel, Ben Smith, and Rob Smith. Each of these individuals has contributed significantly to the content contained in this architecture vision document and many are actively working to help realize such a vision for the future-state Transformational AMMOS.

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

- ¹Green, W. B., "Multimission Ground Data System Support of NASA'S Planetary Program," *Acta Astronautica*, vol. 27, pp. 407-415, 1995.
- ²Gunn, J. and E. Basilio, "Strategic Context," presentation slides (internal document), NASA AMMOS Working Group Meeting, Jet Propulsion Laboratory, California Institute of Technology, Nov. 15, 2011.
- ³Bindschadler, D. L., Boyles, C. A., Carrion, C., and C. L. Delp, "MOS 2.0: The Next Generation in Mission Operations Systems," Paper AIAA 2010-1953, SpaceOps 2010, Huntsville, Alabama, Apr. 25-30, 2010.
- ⁴Meyer, K. "MDAS Introduction," presentation slides (internal document), NASA AMMOS Working Group Meeting, Jet Propulsion Laboratory, California Institute of Technology, Nov. 15, 2011.
- ⁵Smith, D. "MDAS-0: Information Architecture Standards," presentation slides (internal document), NASA AMMOS Working Group Meeting, Jet Propulsion Laboratory, California Institute of Technology, Nov. 15, 2011.
- ⁶Best, S. "MDAS-2: HOSC Interoperability Prototype," presentation slides (internal document), NASA AMMOS Working Group Meeting, Jet Propulsion Laboratory, California Institute of Technology, Nov. 15, 2011.
- ⁷Trimble, J. "MDAS-4: Telemetry & Command Display," presentation slides (internal document), NASA AMMOS Working Group Meeting, Jet Propulsion Laboratory, California Institute of Technology, Nov. 15, 2011.
- ⁸Garlan, D., Barnes, J. M., Schmerl B., and O. Celiku, "Evolution Styles: Foundations and Tool Support for Software Architecture Evolution," in *Proc. WICSA/ECSA'09*, pp. 131-140, 2009.
- ⁹Barnes, J., M., "NASA's Advanced Multimission Operations System: A Case Study in Software Architecture Evolution" (internal report), Jet Propulsion Laboratory, California Institute of Technology, Oct. 24, 2011.
- ¹⁰Giovannoni, B. "NASA AMMOS System Context: A primer for the NASA AMMOS WG," presentation slides (internal document), NASA AMMOS Working Group, Jet Propulsion Laboratory, California Institute of Technology, Nov. 15, 2011.
- ¹¹Bindschadler, D. "Operations Revitalization Update," presentation slides (internal document), NASA AMMOS Working Group, Jet Propulsion Laboratory, California Institute of Technology, Nov. 15, 2011.
- ¹²Bindschadler, D. "Architectural Considerations for Next-Gen Mission Operations System," (presentation slides), Fourth IEEE International Conference on Space Mission Challenges for Information Technology 2011 (SMC-IT 2011) Conference, Aug. 3, 2011.
- ¹³Garlan, D. and M. Shaw, "An Introduction to Software Architecture," CMU-CS-94-166 Report, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, Jan. 1994.
- ¹⁴Hohpe, G. and B. Woolf, *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*, Addison-Wesley Professional, Pearson Education, Inc., Boston, MA, 2003.
- ¹⁵Ko, A. Y., Maldague, P. F., Bui, T., Lam, D. T., and J. C. McKinney, "The Evolvable Advanced Multi-Mission Operations System: (AMMOS): Making Systems Interoperable," paper AIAA 2010-2303, SpaceOps 2010 Conference, Huntsville, Alabama, American Institute of Aeronautics and Astronautics, Inc., Apr. 25-30, 2010.
- ¹⁶Attiyah, A. A., Berry, D. S., and V. N. Legerton, "Conversion from Text to XML Format: One-Way Light Time File (LTF) SIS NAV-003," presentation slides (internal document), Jet Propulsion Laboratory, California Institute of Technology, Jul. 14, 2011.
- ¹⁷Carrion, C., Sarrel, M., Smith, R., and M. McCullar, "OpsRev Sect 318 Briefing," presentation slides (internal document), Jet Propulsion Laboratory, California Institute of Technology, Dec. 7, 2011.
- ¹⁸Reinholtz, K., "Timeline Central Concepts," JPL D-71055 (internal document), Jet Propulsion Laboratory, California Institute of Technology, Aug. 24, 2011.
- ¹⁹Reinholtz, K., "Time-Synchronized Display of Timelines on Multiple Display Terminals," Working Draft (internal document), Jet Propulsion Laboratory, California Institute of Technology, Aug. 2011.
- ²⁰Reinholtz, K., "Timelines as Unifying Concept for Spacecraft Operations," Paper No. 1274906, SpaceOps 2012, Stockholm, Sweden, Jun. 11-15, 2012

- ²¹Dechter, R., Meiri, I., and J. Pearl, “Temporal Constraint Networks,” *Artificial Intelligence*, vol. 49, no. 1, pp. 61-95, Sep. 1991.
- ²²Knight, R. L., G. Rabideau, and S. Chien, “Extending the Representational Power of Model-Based Systems Using Generalized Timelines,” in *Proc. Of Sixth International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, Montreal, Canada, Jun. 18-22, 2001.
- ²³Delp, C. L., Bindschadler, D., Wollaeger, R., Carrion, C., McCullar, M., Jackson, M., Sarrel, M., Anderson, L., and Lam, D., “MOS 2.0 – Modeling the Next Revolutionary Mission Operations System,” IEEEAC paper #1512, Ver. 2, IEEE/AIAA Aerospace Conference, Big Sky, MT, Institute of Electrical and Electronics Engineers (IEEE)/American Institute for Aeronautics and Astronautics (AIAA), Jan. 10, 2011.
- ²⁴Cheung, S., Smith, B., Bui, T., Maldauge, P., O’Reily, T., and K. Reinholtz, “Sequence Revitalization Concept for the Proposed Systems,” presentation slides (internal document), Jet Propulsion Laboratory, California Institute of Technology, Jan. 13, 2011.
- ²⁵Cheung, S., Smith, B., Bui, T., Maldauge, P., O’Reily, T., and K. Reinholtz, “Sequence Revitalization Operational Scenarios,” presentation slides (internal document), Jet Propulsion Laboratory, California Institute of Technology, Jan. 13, 2011.
- ²⁶Cheung, S., “Sequence Revitalization Concept of Operations,” MGSS Doc. No. DOC-000615 (internal document), Multimission Ground System and Services (MGSS) Office, Jet Propulsion Laboratory, California Institute of Technology, Mar. 18, 2011.
- ²⁷Cheung, S. and D. Bindschadler, “Timeline-based Mission Operations Architecture: An Overview,” Paper No. 1269750, SpaceOps 2012, American Institute of Aeronautics and Astronautics, Stockholm, Sweden, Jun. 11-15, 2012.
- ²⁸Bindschadler, D., Delp, C. and M. McCullar, “Principles to Products: Toward Realizing MOS 2.0,” Paper No. 1261336, SpaceOps 2012, American Institute of Aeronautics and Astronautics, Stockholm, Sweden, Jun. 11-15, 2012.
- ²⁹Alexander, C., *The Timeless Way of Building*, Oxford University Press: New York, NY, 1979.
- ³⁰Alexander, C., Ishikawa, S., Silverstein, M. and M. Jacobson, *A Pattern Language*, Oxford University Press: New York, NY, 1977.
- ³¹Ingham, M. D., Rasmussen, R. D., Bennett, M. B., and A. C. Moncada, “Generating Requirements for Complex Embedded Systems Using State Analysis,” *Acta Astronautica*, **58**, Iss. 12, pp. 648-661, Jun. 2006.
- ³²Chien, S. Rabideau, G., Knight, R., Sherwood, R., Engelhardt, B., Mutz, D., Estlin, T., Smith, B., Fisher, F., Barrett, T., Stebbins, G., and D. Tran, “ASPEN – Automating Space Mission Operations using Automated Planning and Scheduling,” International Conference on Space Operations 2000 (SpaceOps 2000), Toulouse, France. Jun. 2000.
- ³³Carlos C., Delp, C. L., Illsley, J., and O. Liepack, “Use of Operational Scenarios in Architecting MOS 2.0,” International Conference on Space Operations 2010 (SpaceOps 2010), Huntsville, AL, Apr. 2010.
- ³⁴McVittie, T., “DSMS Software Architecture Overview: Web-based GDS,” Working Draft (internal document), Jet Propulsion Laboratory, California Institute of Technology, Sep. 14, 2004.
- ³⁵Pajevski, M., “DISA Security Service Software Interface Specification,” MGSS Doc. No. DOC-000645 DRAFT (internal document), Multimission Ground System and Services (MGSS) Office, Jet Propulsion Laboratory, California Institute of Technology, Nov. 9, 2011.
- ³⁶Pajevski, M., “DISA Security Service Software Description Document,” MGSS Doc. No. DOC-000023 DRAFT (internal document), Multimission Ground System and Services (MGSS) Office, Jet Propulsion Laboratory, California Institute of Technology, Nov. 11, 2011.
- ³⁷Santo, D., “Revised DM&A Requirements,” Working Draft Spreadsheet (internal document), Jet Propulsion Laboratory, California Institute of Technology, Oct. 14, 2011.
- ³⁸Monson, E., “MGSS Common Software Environment, Third Party Software (TPS) V18.0.1, Work Implementation Plan (WIP),” MGSS Doc. No. DOC-000770 (internal document), Multimission Ground System and Services (MGSS) Office, Jet Propulsion Laboratory, California Institute of Technology, Jan. 3, 2012.
- ³⁹Soderstrom/Shams SMC-IT 2012 talk, “Beyond the Pervasive Cloud: Lessons and the Future for Space Organizations,” (presentation slides and panel discussion), Fourth IEEE International Conference on Space Mission Challenges for Information Technology 2011 (SMC-IT 2011) Conference, Aug. 4, 2011.
- ⁴⁰Blakley, B., Reeves, F., and C. Howard, “Defining Cloud Computing,” Burton In-Depth Research Management Briefing, Ver. 1.0, Burton Group (now Gartner, Inc.), Mar. 25, 2010.