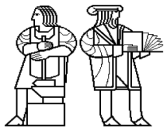

Massachusetts Institute of Technology
Engineering Systems Division

Working Paper Series

ESD-WP-2007-09

MANAGING COMPLEXITY WITH THE DEPARTMENT OF DEFENSE ARCHITECTURE FRAMEWORK: DEVELOPMENT OF A DYNAMIC SYSTEM ARCHITECTURE MODEL

Presented at the *Conference on Systems Engineering Research*, Los Angeles, CA, April 2006

Matthew G. Richards[1], Nirav B. Shah[2], Daniel E. Hastings[3], and Donna H. Rhodes[4]

[1]Massachusetts Institute of Technology
mgr@mit.edu

[2]Massachusetts Institute of Technology
nbshah@mit.edu

[3]Massachusetts Institute of Technology
hastings@mit.edu

[4]Massachusetts Institute of Technology
rhodes@mit.edu

February 2007

Paper #177

Managing Complexity with the Department of Defense Architecture Framework: Development of a Dynamic System Architecture Model

Matthew G. Richards
Massachusetts Institute of Technology
77 Massachusetts Ave., 41-205
Cambridge, MA 02139
mgr@mit.edu

Nirav B. Shah
Massachusetts Institute of Technology
77 Massachusetts Ave., 41-205
Cambridge, MA 02139
nbshah@mit.edu

Daniel E. Hastings
Massachusetts Institute of Technology
77 Massachusetts Ave., 4-110
Cambridge, MA 02139
hastings@mit.edu

Donna H. Rhodes
Massachusetts Institute of Technology
77 Massachusetts Ave., E40-215A
Cambridge, MA 02139
rhodes@mit.edu

Abstract

Architecture frameworks are tools for managing system complexity by structuring data in a common language and format. By characterizing the form, function, and rules governing systems, architecture frameworks serve as a communication tool to stakeholder communities with different views of the system and facilitate comparative evaluation across architectures. The goal of this research is to explore the applicability of architecture frameworks to the study of emergent properties of satellites. The U.S. Department of Defense Architecture Framework was selected to achieve this goal given its orientation towards technical systems in contrast to the majority of architecture frameworks focused on business enterprises. Although developed by military planners in the 1990's to support the acquisition of interoperable information systems, the Department of Defense Architecture Framework can be used to connect operational concepts and capabilities to the technical architecture of any system. While the views of the Department of Defense Architecture Framework are well-defined, little guidance is provided on how the views are to be constructed. Vitech Corporation's software program CORE,® a systems engineering modeling tool with the ability rapidly to produce architecture views from a common data repository, was employed to complete Department of Defense Architecture Frameworks for the Hubble Space Telescope.

Upon characterizing Hubble within this common structure, the value of the Department of Defense Architecture Framework for conducting dynamic quantitative analyses of system architectures was explored. A methodology is proposed and tested for evaluating human and robotic architectures for on-orbit servicing—the extension of the useful life of spacecraft through refueling, upgrading, repair, relocation, *et al*. In particular, a multi-year servicing campaign is modeled for Hubble including behavioral threads that characterize the Orbiting Observatory, servicing architecture, and science customers. Preliminary results indicate that, when coupled

with an executable model, the Department of Defense Architecture Framework can be utilized for dynamic quantitative evaluation of space system architectures. The paper concludes with lessons learned from using the Department of Defense Architecture Framework and proposes improvements for the application of its static views to model-based systems engineering.

Introduction

The goal of this research is to explore the applicability of architecture frameworks to the study of emergent properties of satellites. In the first section, architecture frameworks are introduced and illustrated. Upon selecting the U.S. Department of Defense Architecture Framework (DoDAF) for describing space systems, the paper identifies desirable attributes of architecture framework development tools and surveys industry-leading software applications. A systems engineering modeling language called CORE is then applied to the construction of a high-level DoDAF representation of the Hubble Space Telescope. Next, the value of the DoDAF and CORE software for conducting serviceability assessments is explored through the development of a discrete event simulation of Hubble servicing missions. Lessons learned from the Hubble DoDAF and executable model are offered for improving the DoDAF as are prescriptive considerations for users and developers of architecture frameworks.

Overview of Architecture Frameworks

Architecture frameworks are tools for managing complexity by establishing standards for the description of architectures. These standards define the system to be characterized as well as how the system is to be constructed and operated. Architecture frameworks serve as a communication tool by presenting a common set of information with multiple views (Figure 1). Each view reflects the perspective of a unique stakeholder (*e.g.*, customer, designer, user). Maier and Rechtin (2002) identify five goals of architecture frameworks: (1) institutionalize best practices for architectural description, (2) ensure system sponsors receive information in the format they desire, (3) facilitate comparative evaluation of architectures, (4) improve the productivity of development teams, and (5) improve interoperability of information systems by

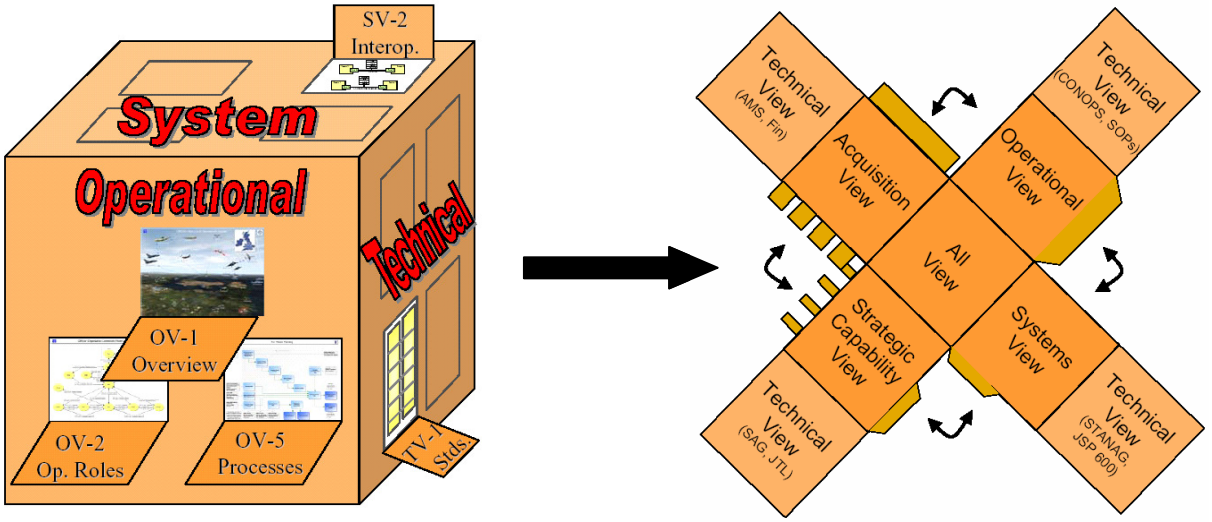

Figure 1. Architecture Framework "Unwrapped" (U.K. Ministry of Defence 2005)

requiring that critical interfaces are described. The third goal (*i.e*., comparative evaluation of architectures) motivates the application of architecture frameworks to this research.

Several architecture frameworks have been developed for enterprises, systems, and software since the late 1980's. This section provides a brief overview of three enterprise architectures critical in the evolution of these tools: the Zachman Framework, The Open Group Architecture Framework (TOGAF), and Federal Enterprise Architecture Framework (FEAF). Two architecture frameworks for communicating system design concepts—the DoDAF and U.K. Ministry of Defence Architecture Framework (MoDAF)—are then evaluated for application to space systems. Major architecture frameworks excluded in this overview include the Computer Integrated Manufacturing Open Systems Architecture (CIMOSA), Integrated Architecture Framework (IAF), Architectural Descriptions of Software Intensive Systems (IEEE 1471), and International Standards Organization Reference Model for Open Distributed Processing (ISO RM-ODP). A detailed analysis of architecture frameworks is outside the scope of this paper and has been conducted elsewhere (Tang *et al*. 2004).

The **Zachman Framework** was released in 1987 by IBM to provide a blueprint for an organization's information infrastructure. Embraced by the architecture community, the Zachman Framework has been incorporated into the four other architecture frameworks discussed in this section. The framework consists of populating a 6x6 matrix—establishing relationships of six elements of information systems (*i.e.* data, function, network, people, time, motivation) across six perspectives (*i.e*., planner, owner, designer, builder, subcontractor, and working system). Unlike the TOGAF, FEAF, DoDAF, and MoDAF, design tradeoffs are not captured (Tang *et al.* 2004). Since the Zachman Framework was not developed by a professional organization, no explicit compliance rules have been published. No architectural development process is documented in publications and most prescriptive guidance is only offered through consulting services by the Zachman Institute (Schekkerman 2004).

The Open Group Architecture Framework is a freely-available industry standard for designing, evaluating, and building enterprise architectures. Although it does include documentation on architecture framework development and views for design rationale, TOGAF is principally a tool for business organization.

The **Federal Enterprise Architecture Framework** was first published in 1999 and represents the realization of the 1996 Clinger-Cohen Act, which requires federal agencies to develop, maintain, and facilitate integrated systems architectures. The FEAF structure borrows heavily from Zachman (Couretas 2003) and is optimized for enterprise engineering and program and capital management.

In contrast to enterprise architectures which connect organizational goals to business activities, system architectures relate operational concepts and capabilities to technical architectures. The **Department of Defense Architecture Framework** Version 1.0, released in 2003, defines a common approach for describing and comparing DoD architectures. The DoDAF evolved from the 1996 Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance (C4ISR) Architecture Framework which was developed following lessons learned from the Persian Gulf War of 1991. A host of integration problems occurred during Desert Shield and Desert Storm as C4ISR systems were deployed for the first time in support of tactical operations for a large-scale conflict. Older platforms were used for missions for which they were not designed (*e.g.,* Defense Support Program satellites for Scud detection), new technologies were applied piecemeal, and interoperability problems hindered full exploitation of information technology (Spires 2001). Some of these integration problems were

solved during the six month build-up to war (*e.g.*, early warning satellites were used successfully for detection of tactical ballistic missiles) while others were not (*e.g.*, paper copies of air tasking orders had to be flown from the command center in Riyadh to the decks of aircraft carriers) (Zinn 2004).

In developing the C4ISR/DoD Architecture Framework to aid interoperability and system-of-systems integration, the Department of Defense selected three views (composed of multiple work products) to characterize major systems: Operational, Systems, and Technical. To first order, the Operational View may be thought of as a functional decomposition of the system, specifying mission-critical activities and information exchanges. The Systems View constitutes the form decomposition of the system, tracing the needs identified in the Operational View to resources and capabilities of the technical architecture. Taken together, the Systems View and the Operational View fully describe the system and how it will operate. Since the two views are built simultaneously, the system architect allocates operational tasks to particular system components, whether physical or organizational. Conversely, knowledge of the system behavior can inform operational design. Finally, the Technical View captures standards and conventions for the architecture, prescribing the minimal set of rules governing the arrangement, interaction, and interdependence of system components (DoDAF Working Group 2003).

Although developed for acquisition supervisors concerned with interoperability, the DoDAF in practice is primarily used to produce architecture descriptions during the early-stages of system development (Maier *et al*. 2004). Maier further argues that the DoDAF is not necessarily well-suited for this application. Another criticism of DoDAF is that it does not provide analytical techniques or mechanisms for synthesizing the architecture information into "cogent, compelling conclusions" (French 2005). No formal DoDAF development process is prescribed. A variety of tools, discussed in the following section, have been developed to aid in the construction of DoDAF work products.

The UK **Ministry of Defence Architecture Framework**, released in 2005, is an extension of the DoDAF with identical Operational, Systems, and Technical views to facilitate information exchange for interoperability analyses across US-UK systems. The MoDAF formalizes two perspectives not explicitly addressed in the DoDAF by adding two views: Strategic and Acquisitions. Both are aimed at improving portfolio management across MoD programs. The Strategic Viewpoint translates MoD policies into appropriate measures of effectiveness that can be used for capability audit and gap/overlap analysis. The Acquisitions Viewpoint incorporates programmatic details such as dependencies across development efforts (Ministry of Defence 2005). Through these new views, the MoDAF intends to capture the perspectives of all MoD system stakeholders throughout the acquisitions process. This is consistent with the principles of enterprise architecture but at odds with the primarily technical approach prescribed in the DoDAF (Barrett 2004).

For the application of comparative evaluation of architectures (*i.e.*, serviceability assessments of space systems), the DoDAF and MoDAF offer similar qualities. Both are oriented towards technical architecture with Operational and Systems views to enable structured analyses of satellite functions, physical attributes, and servicing activities. However, the DoDAF was selected for this research due to a variety factors exogenous to the frameworks themselves: industry experience with DoD architecture frameworks over the last decade, availability of literature and research on the DoDAF, and the existence of several tools supporting the DoDAF development process.

Tools Available for Constructing Architecture Frameworks

While the views of the DoDAF are well-defined, little documentation is provided on how the views are to be constructed. This lack of documentation, coupled with a focus on final view outputs in early user training, led to a work product-centric approach to DoDAF development. As a result, many early DoDAF work products were pictures (many done in PowerPoint) that were neither internally consistent nor complete in capturing relevant data. In order to analyze the behavior of a system, it is essential to capture dependencies and parallelisms among activities, processes, and supporting technologies. However, these abilities are lacking in standard office automation programs that are often used to develop DoD architecture frameworks (Troche *et al.* 2004). To fix this problem, DoD has made a significant push towards data-centric architecture development with the implementation of the DoD Architecture Repository System (DARS) for certified formal methods and modeling languages.

Ideally, a common process for constructing DoDAF views is followed to maintain consistency and enable comparisons across architectures. Several companies offer enterprise

Table 1: Tools Supporting DoDAF Development (as of February 2006)

Product	Company	Key Features
Core Workstation	Vitech Corporation	• Modeling language with modifiable database schema • Executable behavior models with discrete event simulator • Automatically export DoDAF views from central data repository
DoDAF*Live!*	Wizdom Systems	• Niche DoDAF project management tool • Provides online data repository for all information and models
EA WebModeler	Agilense	• Central repository of data accessed via standard web browser • Supports Zachman, TOGAF, FEAF, and DoDAF development
Elements Repository	Enterprise Elements	• Web-based data management tool • Integrates DoDAF views from multiple modeling tools
Metis Desktop	Troux Technologies (acquired from Computas)	• "Living Timeline" support for system evolution • Operational capabilities as objects • Architecture reuse via broad support for DoD reference models
netViz Enterprise	netViz	• Generic enterprise architecture tool with relationship modeling • Documents all 27 DoDAF work products
ProVision Modeling Suite	Proforma Corporation	• Only DARS-certified tool (July 2005) • Web-based repository for enterprise and system architectures • Discrete event simulator coupled with Monte Carlo analysis
Rhapsody	I-Logix	• UML/SysML modeling and simulation tool • Includes “DoDAF pack” for outputting DoDAF views
System Architect	Telelogic (acquired from Popkin Software)	• First tool to offer a DoDAF extension (industry leader) • Integrated with Telelogic DOORS for requirements traceability • Supports Zachman, TOGAF, and DoDAF development
TAU	Telelogic	• UML/SysML modeling and simulation tool • Includes “Enterprise Architect for DoDAF” extension • Integrated with Telelogic DOORS for requirements traceability

architecture tools with templates to construct DoDAF work products. In general, each tool offers relatively complete DoDAF support with certain tools offering unique capabilities (Table 1).

One of the shortcomings of architecture frameworks is that they rely on static pictures, diagrams, and textual descriptions—not necessarily adequate for conveying the logical, behavioral, and performance properties of the architecture (Levis and Wagenhals 2000). To capture the dynamic properties of a system, an executable model is necessary to carry out simulations (Wagenhals *et al*. 2000). Therefore, it is important to select a DoDAF development tool that can input the information contained in the static views to an executable model.

In selecting a tool for constructing architecture frameworks for space systems, five criteria were deemed essential: (1) a hierarchical structure to enable high-level representations, (2) support for exporting operational and systems views, (3) modeling and simulation capabilities for dynamic performance analyses, (4) a learning curve consistent with an eight-month research project, and (5) affordability. Vitech Corporation's CORE® Workstation meets these criteria and was used for this research.

CORE is a systems engineering tool that couples requirements management with functional analysis and simulation. The tool accomplishes this by representing these domains with a common modeling language. Leveraging a central system design repository for rapid population of DoDAF views, CORE traces originating requirements to functions in the behavior domain. Behaviors are represented with Enhanced Functional Flow Block Diagrams (EFFBD) and then allocated to physical components in the architecture domain. One of the most interesting elements of CORE is a discrete event simulator that allows an EFFBD to be executed. Ascent Logic Corporation's RDD-100 tool offered a similar suite of capabilities but has been discontinued.

Sample DoDAF: Hubble Space Telescope

Given the large quantity of open-source data available (National Research Council 2005, Nelson *et al*. 2002) and its status as the only uninhabited space platform that is currently serviced, Hubble was a natural choice for exploring the value of the DoDAF and CORE. In scoping the problem, populating all 27 DoDAF work products in full detail was found to be unnecessary and unrealistic. Completion of a DoDAF for a small uninhabited air vehicle with only 150 components took two person-years (Cooper and Ewoldt 2005)—Hubble has 400,000 parts. Therefore, each DoDAF work product was studied and seven were found to be applicable to the problem of conducting serviceability assessments: (1) Overview and Summary Information,

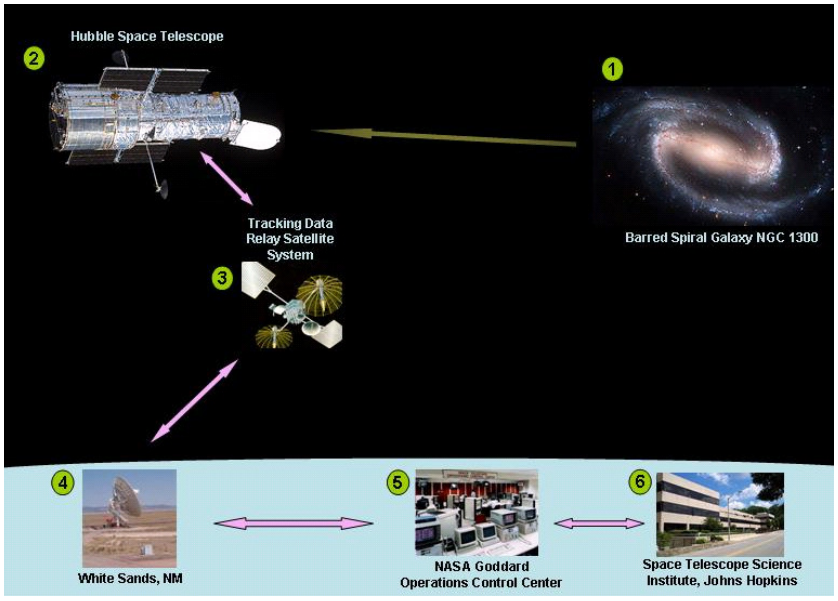

Figure 2. High-Level Operational Concept Graphic (OV-1)

(2) High-Level Operational Concept Graphic, (3) Operational Node Connectivity Description, (4) Operational Activity Model, (5) Systems Interface Description, (6) Systems Evolution Description, and (7) Systems Technology Forecast. In constructing high-level views, a data-centric build sequence was followed whereby several DoDAF relationship and attribute classes for subsequent work products were automatically generated from core entities constructed in earlier work products.

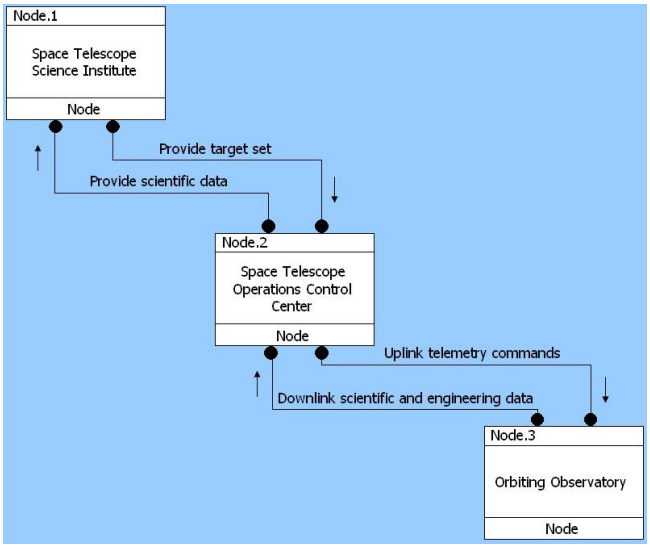

Figure 3. Operational Node Connectivity (OV-2)

Overview and Summary Information (AV-1) provides an executive-level summary of Hubble including scope, assumptions, constraints, and limitations of the architecture description. High-level features of the Orbiting Observatory are described (*e.g.,* 2.4 meter primary mirror, 11,110 kilogram weight).

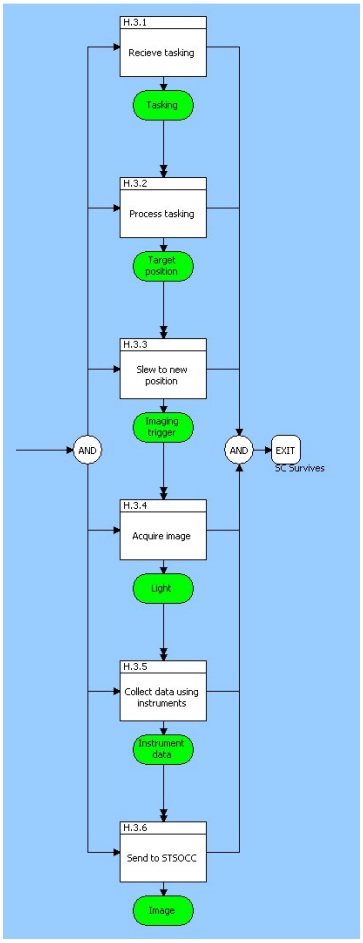

Figure 4. Operational Activity Model (OV-5)

High Level Operational Concept Graphic (OV-1) depicts Hubble's interaction with its environment as well as with external systems (Figure 2). For the purposes of this research, OV-1 was deemed useful for rapidly communicating the missions of various space systems and their operational context—both of which might elicit constraints on servicing operations (*e.g.,* imaging payload sensitivity to thruster plume impingement).

Operational Node Connectivity Description (OV-2) tracks the need to exchange information across nodes. This includes internal operational nodes as well as external nodes. OV-2 does not depict the connectivity between nodes. For example, Figure 3 shows that the Orbiting Observatory depends on the Space Telescope Operations Control Center (STOCC) for command and control, which in turn needs to downlink data to STOCC. The Tracking Data Relay Satellite System (TDRSS)—the communications pipeline between these two operational nodes—is not depicted. Understanding communication needlines for satellites is necessary for eliciting constraints on servicing operations (*e.g.,* aversion of fixed satellite service providers to transponder downtime).

Operational Activity Model (OV-5) describes the operations that are normally conducted in the course of achieving a mission. It specifies activities and inputs and outputs between activities. OV-5 delineates lines of responsibility when coupled with OV-2 and is a necessary foundation for depicting activity

sequencing and timing. Figure 4 is a high-level EFFBD representation of OV-5 for a typical Hubble science mission. Similar diagrams were constructed for other Hubble activities including monitoring spacecraft health, attitude determination and control, and Space Shuttle servicing operations. In the next section, two Hubble servicing methods are simulated using an executable EEFBD—enabling comparison of two architectures in the behavioral domain.

Systems Interface Description (SV-1) identifies the systems nodes that support operational nodes. Detailed SV-1 work products may be used for specifying requirements and for interoperability assessments. For the Hubble architecture, SV-1 was used to show the physical decomposition of 42 components, including the TDRSS communications pipeline excluded in OV-2. Five levels are present in the constructed hierarchy. Figure 5 displays a sample of this decomposition—level IV-V components of the Telescope Instrument Assembly.

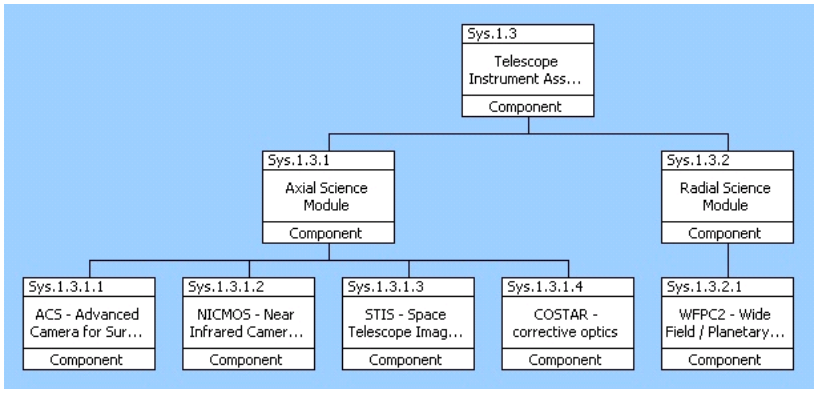

Figure 5. Systems Interface (SV-1) of Instrument Assembly

Systems Evolution Description (SV-8) captures how the system, or the architecture in which the system is embedded, will evolve over time. Applied to Hubble, SV-8 is used to record completed and planned maintenance and upgrades of the Orbiting Observatory during Space Shuttle servicing missions. Since its launch in 1990, Hubble has been serviced four times, enabling NASA to equip Hubble with state-of-the-art science instruments every few years and replace limited-life components.

Systems Technology Forecast (SV-9) provides a summary of expected improvements in technology that affect the capabilities of the architecture or its systems. For Hubble, SV-9 principally involves a survey of emerging instruments. If there is a fifth servicing mission, the Wide Field Camera 3 and Cosmic Origins Spectrograph will be installed to allow Hubble to continue its high level of scientific return (National Research Council 2005).

Hubble Servicing Simulation

Upon developing static DoDAF work products, the value of architecture frameworks for conducting serviceability assessments was explored through the development of an executable model that captures the dynamic properties of Hubble at a level of detail consistent with conceptual design. In particular, the performance of two servicing architectures was compared using the discrete event simulator packaged with CORE. Figure 6 is a top-level view of the multi-layered behavioral model developed of Hubble performance in the context of a multi-year servicing campaign. The behavioral model is depicted as an EFFBD and shows the interactions of three key actors in the simulation: the Orbiting Observatory, servicing architecture, and science customers. System behaviors are represented using sequential, parallel, repetitive, and decision logic. Five parallel threads are modeled: (1) Hubble Component Failure and Degradation, (2) Hubble Health, (3) Hubble Imaging, (4) Shuttle/Robotic Servicing, (5) Science Dissemination.

Hubble Component Failure and Degradation Thread models the wear-out of various critical Hubble components (*i.e.*, batteries, avionics, gyroscopes, reactions wheels, and fine-guidance sensors). Hubble has a battery capacity of 540 ampere-hours (Ah) with energy storage requirements of 160 Ah to support science operations and 40 Ah to maintain thermal stability of the optical assembly. Gradual loss of charge capacity may be projected and is modeled

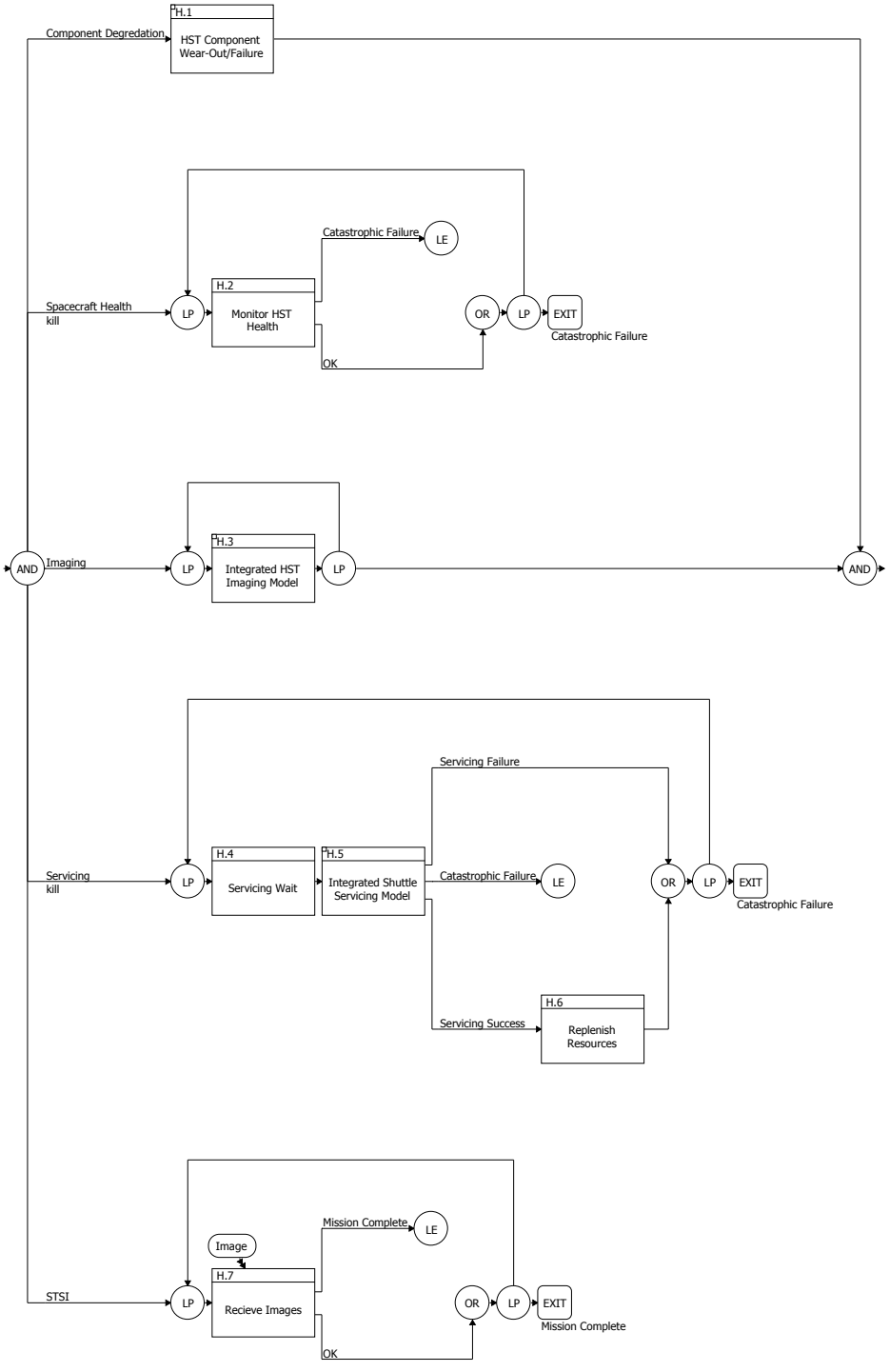

Figure 6. Integrated Hubble Servicing Model

Table 2: Key Assumptions in Servicing Simulation

Component Wear-Out	
Probabilistic Failures	Monthly Rate
gyroscopes	0.036
reaction wheels	0.022
fine-guidance sensors	0.025
avionics system	0.006
Deterministic Degradation	Monthly Rate
solar panels	0
battery capacity	5 Ah

Servicing Missions		
Probabilistic Failures	Space Shuttle	Robotic Vehicle
launch and rendezvous	0.02	0.10
dock (catastrophic)	0.01	0.05
dock (non-catastrophic)	0.01	0.05
access	0.05	0.15
service	0.02	0.10
mission frequency	36 months	36 months
services avionics?	yes	yes

deterministically at 5 Ah each month. The state of the avionics system is modeled as a binary whereby it is either functioning or broken. If broken, no science operations are conducted and Hubble waits for the next successful servicing mission to restore the avionics system.

Gyroscopes, reaction wheels, and fine-guidance sensors failure is both probabilistic and deterministic. For simplicity, these three components are assumed to fail probabilistically (Table 2) with half-lives mapped to most-likely failure projections (NRC 2005). Three gyroscopes are required to sense drift rates during normal pointing and slewing operations. At launch and following successful servicing operations, six healthy gyroscopes are on the telescope—offering "three for six" redundancy. Fine-guidance sensors, used for precision pointing of the observatory, are modeled similarly and have "two for three" redundancy. Reaction wheels

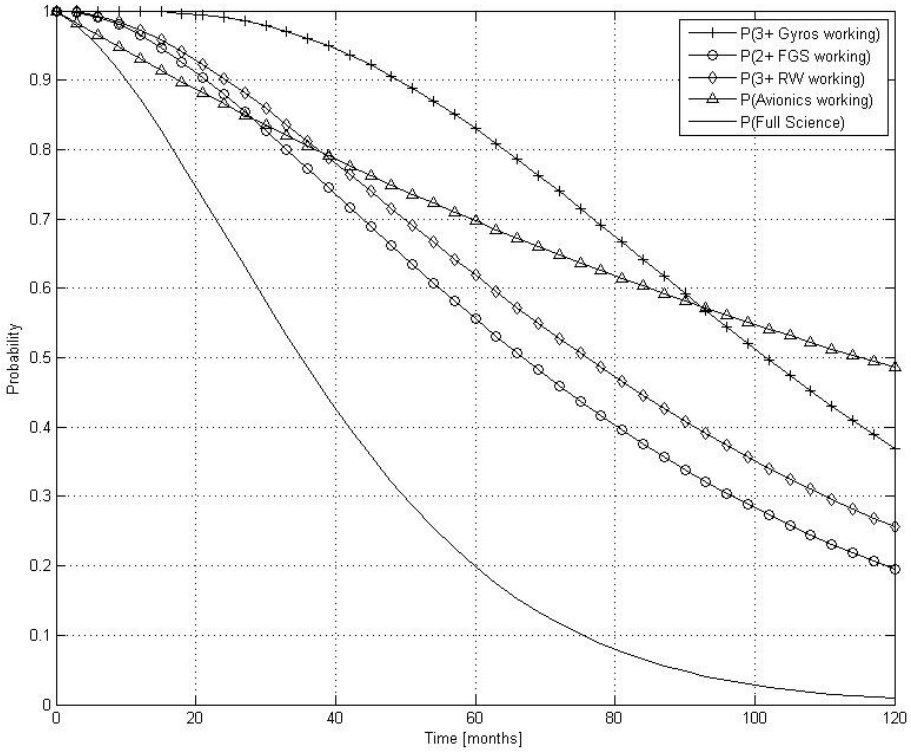

Figure 7. Assumed Reliability of Critical Subsystems over Time (no servicing)

provide three-axis control of the telescope and incorporate "three for four" redundancy.

Hubble Health Thread monitors the overall state of Hubble. Three states are possible: fully functional for conducting science operations, survival mode during which no science is conducted, and dead. To be fully functional, Hubble must possess a working avionics system and at least 160 Ah of battery capacity, three healthy gyroscopes, two healthy fine-guidance sensors, and three healthy reaction wheels. Figure 7 plots the probability of each of these conditions being met (in isolation) as a function of time (if no servicing missions were carried out). Once one of these conditions is not met *(e.g.,* four of the six gyroscopes fail), Hubble enters into a survival mode, pausing science operations until the next successful servicing or the occurrence of a catastrophic failure. Catastrophic failures may be caused by docking collisions during attempted servicing operations or degradation of battery capacity below 40 Ah (minimum energy required to prevent irreversible structural deformation of the optical assembly).

Hubble Imaging Thread tracks science operations (Figure 4). The imaging thread is triggered by the sequence of images sent by a fully functioning Hubble. Science operations are conducted in one-month increments. In the simulation, Hubble's target science goal is 120 months of successful imaging operations. In its first ten years of operation, the actual Hubble system took approximately 350,000 exposures of 14,000 astronomical targets (Nelson et. al. 2002).

Shuttle/Robotic Servicing Thread tracks the implementation of the servicing architecture. Two servicing threads were created—one representative of the Space Shuttle and the other of a robotic servicing vehicle. Four servicing activities are included in each model: (1) Launch and Rendezvous, (2) Dock, (3) Access, and (4) Service. Launch is defined as the movement of a servicing vehicle from a starting position (*i.e.,* launch pad) to a position where relative navigation is possible with laser ranging, radar, and cameras (< 500 meters). Rendezvous positions the servicing vehicle for docking (< 3 meters). Docking is defined as the mating of the servicing vehicle to Hubble. In the case of a robotic servicing vehicle, autonomous execution is required for proximity operations because of a two-second communications delay in routing signals through TDRSS (National Research Council 2005). Access constitutes all activities required to deploy the stowed tools, upgrades, and replacement parts of the servicing vehicle to the Hubble components which require servicing. Finally, Service entails operation of the servicing vehicle to improve Hubble and return it to full operation. All four servicing activities must be completed for servicing to be successful. Differences between Space Shuttle and robotic vehicle servicing activity success probabilities are outlined in Table 2. If servicing is successful, all subsystems are replenished to beginning-of-life levels, restoring Hubble to a "like new" condition.

Science Dissemination Thread tracks the transfer of images to the Space Telescope Science Institute (STScI) and terminates the simulation upon STScI receipt of 120 months of science data. The simulation terminates earlier if a catastrophic failure occurs.

Upon developing the multi-layered Integrated Servicing Model, the dynamic performance and functional behavior of Hubble was analyzed using CORE's discrete event simulator. The simulator outputs a timeline of functional activation, execution, and duration. Wait states, resource inventory history, and queuing triggers (items waiting to be processed by functions) are all depicted. Colored duration bars are used to represent different types of events. Grey specifies the amount of resources available, teal indicates the execution of a function, yellow indicates that a function is enabled but waiting for a trigger, and magenta indicates that a function is enabled but waiting for resources.

Figure 8 shows a sample run of the Integrated Hubble Servicing Model for a Space Shuttle servicing campaign. Rows depict resource levels and function states with a horizontal axis of time (measured in months). It can be observed that in the 20th month the reaction wheel resource had fallen from four to two (below the required level of three). This resource change initiated a state change in Hubble from fully functional to survival mode, pausing science operations until successful Shuttle servicing during the 37th month. The "ImageMonths" resource continued to count down successfully from 120 until the 85th month when the fine-guidance sensor resource fell to one (below the required level of two), followed by a failure of the avionics system a month later. Science operations were put on hold again until the third servicing mission succeeded in replenishing all of the resources. In total, five servicing missions were attempted, and all were successful. Once 120 months worth of science had been collected and disseminated, the simulation terminated during Hubble's 17th year of operation.

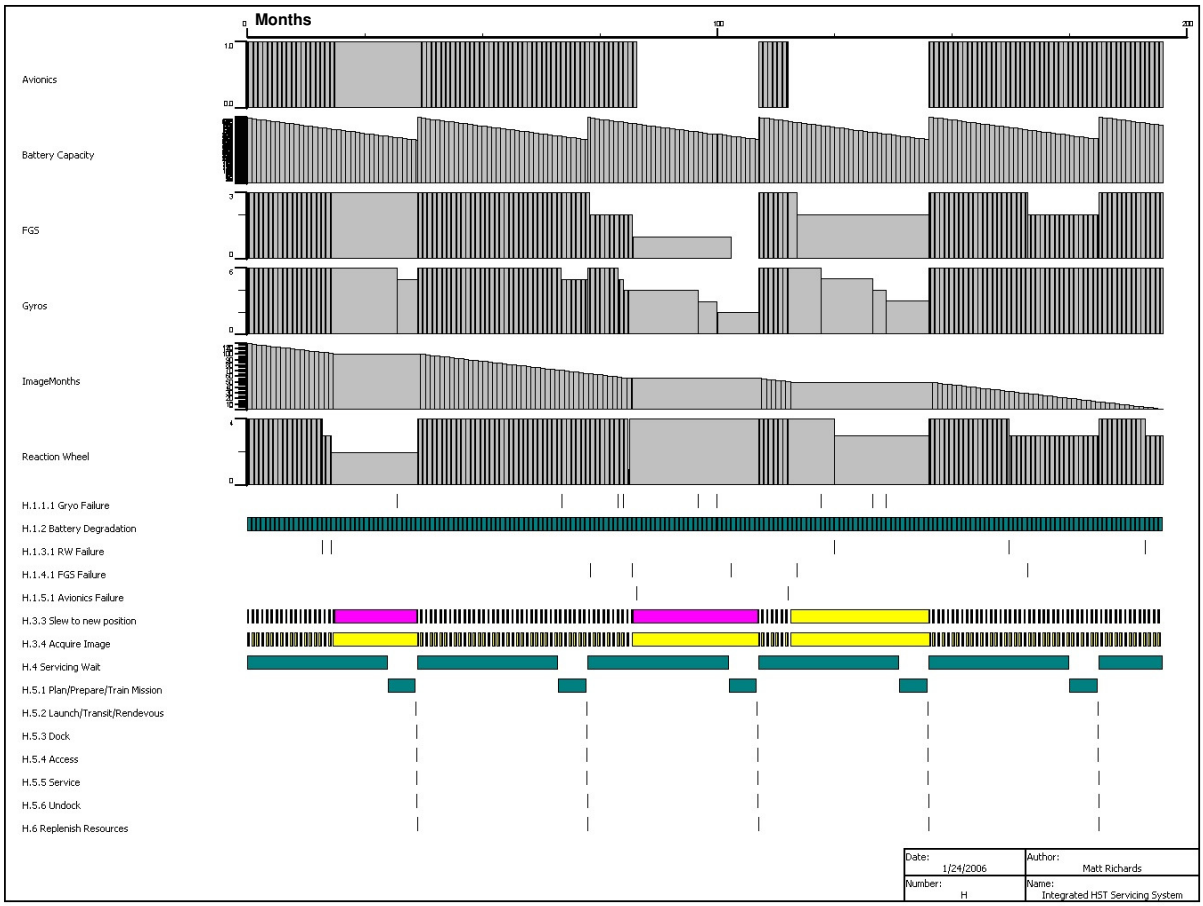

Figure 8. Hubble Servicing Simulation – Space Shuttle

Figure 9 shows a sample run of the Integrated Hubble Servicing Model with a robotic servicing vehicle. In this particular run, an initial two years of science operations were followed by nearly four years of survival mode due two events: an early failure of the avionics system and a failure of the Access servicing activity during the first attempted servicing mission. The second attempted servicing mission was successful in restoring, among other things, energy

storage capacity before battery degradation caused a catastrophic loss.[1] Of the five robotic servicing missions that were attempted, three were successful. The simulation terminated during the 16th year after 120 months of science data had been returned.

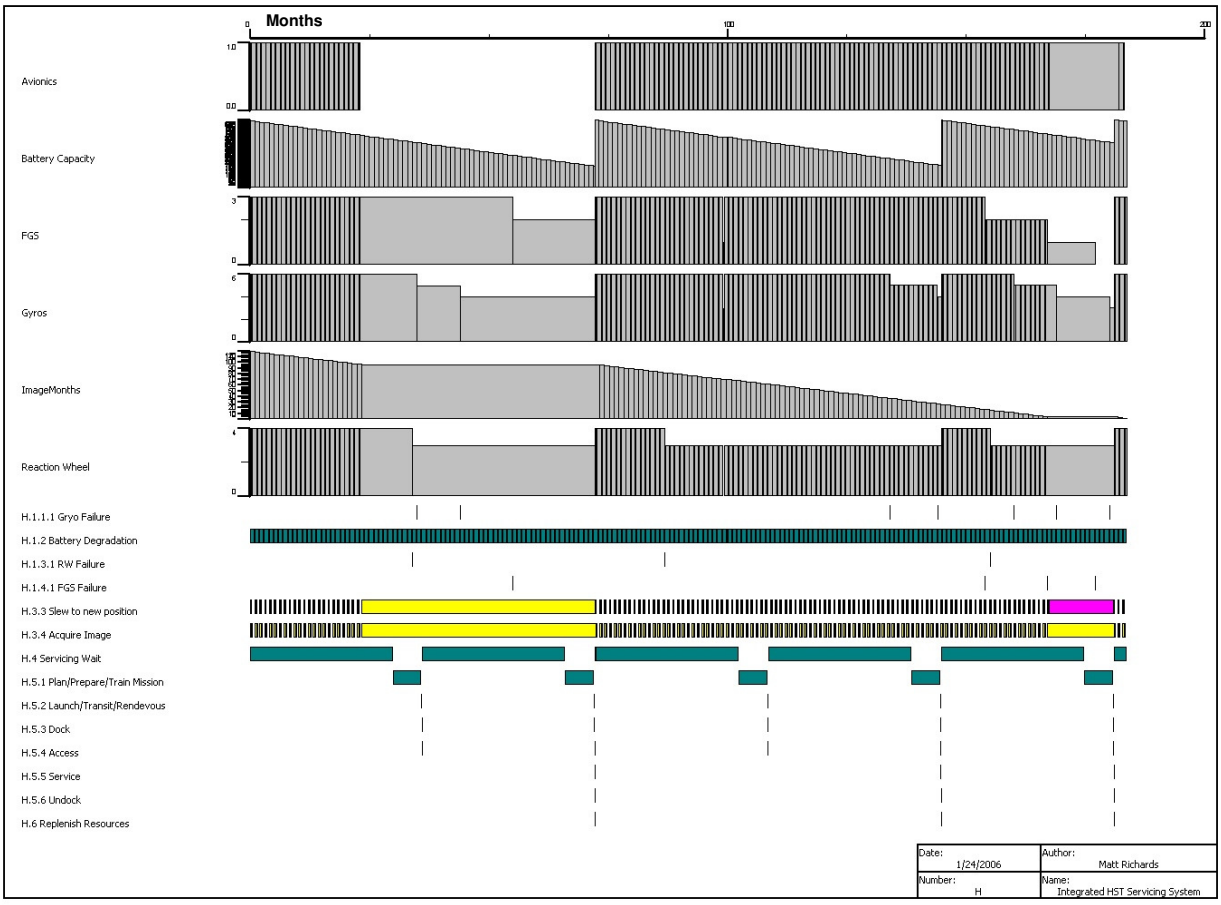

Figure 9. Hubble Servicing Simulation – Robotic Vehicle

One metric for comparing the relative performance of the two servicing architectures is availability, the percentage of time in the simulation that Hubble was able to perform its science mission. (If a catastrophic failure occurs during the first 120 months of operation, 120 months is used in the denominator of the availability calculation.) Given that each run of the discrete event simulator produces a unique outcome, a Monte Carlo analysis was performed to calculate availability across multiple Space Shuttle and robotic vehicle servicing campaigns. Table 3 shows the results of this preliminary analysis. The higher probability of success assumed for Shuttle in all four servicing activities is evident with the 17% average availability advantage.

Although these preliminary results imply the superiority of a Shuttle servicing architecture, it is important to keep in mind the impact of assumptions, the simplicity of the model, and the limitations of the availability metric. Aggregating the assumed probabilities of servicing success across the four servicing activities, the Space Shuttle only fails one out of ten missions while the robotic vehicle fails nearly four out of ten missions. These assumptions were not subjected to

[1] With servicing scheduled once every three years, Hubble will not survive if there are two consecutive servicing failures. After six year without servicing, the battery capacity will have degraded below the necessary level to maintain the optical assembly.

Table 3: Monte Carlo Comparison of Two Servicing Architectures

	Number of Simulations	Average # of Successful Servicing Missions	Average Availability
Space Shuttle	25	4.12	72.8%
Robotic Vehicle	25	2.24	56.0%

sensitivity analysis and also may need to be modified as teleoperated and autonomously controlled vehicle technology improves. Regarding simplicity of the model, the executable portion only describes the functional aspects of Hubble's operation and servicing as the physical architecture of Hubble does not inform the success or failure of servicing. With another layer of detail, the linkages in CORE between the physical architecture and functional behavior domains can be leveraged to model the impact of physical design choices on serviceability. The model also does not allow on-demand servicing to supplement the shortcomings of the scheduled servicing campaign (*e.g.,* four-year pause in science operations observed in Figure 9). Most importantly, the availability metric captures only one of many attributes of a servicing architecture. The added utility of instrument upgrades, servicing cost trades, and the risk to astronaut life all need to be incorporated into the value proposition.

Throughout the process of constructing the static work products and building an executable model, the DoDAF allowed the authors quickly to understand the structure and operation of the larger Hubble system (*i.e*., Orbiting Observatory, Space Telescope Operations Control Center, and Space Telescope Science Institute). On its own, the model of Hubble does not provide insights into whether Hubble is more or less physically amenable to servicing than other space systems. However, other space systems can rapidly be incorporated into the same overall servicing architecture to allow for comparison. Once these models for candidate target satellites are in place, judgments will become possible about the relative amenability of spacecraft to on-orbit servicing.

Conclusion

Architecture frameworks bring structure to describing complex systems. The DoDAF views alone are insufficient for characterizing the dynamic behavior inherent in a satellite servicing architecture. However, when such views are constructed using a system engineering modeling tool such as CORE, both the DoDAF work products and an executable behavior model are created simultaneously. The executable model can then be used for quantitative evaluation of the dynamic system behavior.

Over the course of this research, lessons emerged regarding the DoDAF and its development process. Emphasis is placed on final architecture products rather than process. Work products are frequently too complex to present to senior leadership without modification. Most fundamentally, weaknesses in the DoDAF have been identified as it undergoes transition from a static, descriptive tool to a tool that attempts to characterize dynamic system properties. Little guidance is provided on how to translate requirements into the design of the work products. As promulgated, the DoDAF does not have a companion architecture development process to take advantage of its interconnected views. As a result, many developers of DoDAF have treated it as a contract deliverable as opposed to a central communications tool in the design process. While it is not the business of DoD to stipulate how contractors conduct system design,

it is in the interest of DoD to require architectures that are internally consistent and support dynamic performance analysis. Architecture development software with DoDAF extensions and integrated modeling and simulation capabilities is available to fill this void. In practice, however, 70% of DoDAF developers are not building executable architectures (Office of the Assistant Secretary of Defense for Networks and Information Integration 2005).

Finally, the existence of a clear purpose for building the high-level Hubble architecture framework (*i.e.*, serviceability assessment) was a critical element in the construction of views that were both compliant with the static DoDAF taxonomy and useful for understanding dynamic system properties. For the value of the DoDAF to be fully realized, its construction must be mission-driven, focused on providing information that supports decision-making processes.

Acknowledgements

Funding for this work was provided by the Lean Aerospace Initiative at the Massachusetts Institute of Technology (MIT) and the Defense Advanced Research Projects Agency. The authors would like to thank Tim Tritsch of Vitech Corporation, Ian Komorowski of EA Frameworks LLC, Dennis Connolly of Lockheed Martin Corporation, and former Astronaut Jeffrey Hoffman for their assistance over the course of this study.

References

Barrett., J., Hartkey, K., et. Al., "Improving requirement modelling and traceability within an enterprise architecture framework." *Command and Control Research and Technology Symposium* (San Diego, CA, 2004).

Cooper, C., and Ewoldt, M., *A Systems Architectural Model for Man-Packable/Operable Intelligence, Surveillance, and Reconnaissance Mini/Micro Aerial Vehicles*. Department of Aeronautics and Astronautics, Master's thesis. Air Force Institute of Technology, Wright-Patterson Air Force Base, OH, 2005.

Couretas, J.M., Adrounie, V.P., and Francis, M.S., "Enterprise Architectures as a Catalyst for System-of-Systems Development: A Global ISR Perspective." *Proceedings of the AIAA/ICAS International Air and Space Symposium* (Dayton, OH, 2003).

DoDAF Working Group, *DoD Architecture Framework*. Version 1.0, June 2003.

French, S.S., "DoDAF Lessons Learned." *Conference of Systems Engineering Research.* (Hoboken, NJ, March 23-25, 2005), Booz Allen Hamilton, Inc., San Antonio, TX.

Levis, A.H., and Wagenhals, L.W., "C4ISR Architectures: I. Developing a Process for C4ISR Architecture Design." *Systems Engineering*, 3(4): 225-247, 2000.

Maier, M.W., Emery, D., and Hilliard, R., "ANSI/IEEE 1471 and Systems Engineering." *Systems Engineering*, 7(3): 257-270, 2004.

Maier, M.W., and Rechtin, E., *The Art of Systems Architecting*. CRC Press, Boca Raton, 2002.

Ministry of Defence. *MOD Architectural Framework*. Version 1.0, 31 August 2005.

National Research Council, *Assessment of Options for Extending the Life of the Hubble Space Telescope: Final Report*. National Academies Press, 2005.

Nelson, B., et al., *Hubble Space Telescope: Servicing Mission 3B Media Reference Guide*. Lockheed Martin, 2002.

Office of the Assistant Secretary of Defense for Networks and Information Integration, *The State of DoD Architecting*. Architecture Development and Analysis Survey, Winter 2005.

Schekkerman, Jaap, *How to survive in the jungle of Enterprise Architecture Frameworks*. Trafford, Victoria, 2004.

Spires, David, *Beyond Horizons: A Half Century of Air Force Space Leadership*. Maxwell Air Force Base, Air University Press, 2001.

Tang, A., Han, J., and Chen, P., "A Comparative Analysis of Architecture Frameworks." *Proceeding of the 11th IEEE Asia-Pacific Software Engineering Conference* (Busan, Korea, 2004).

Troche, C., Eiden Jr., G.F., Potts, F.C., *Architecture Development Lessons-Learned: A Three-Year Retrospective*. MITRE Technical Report, January 2004.

Wagenhals, L.W., Shin, I., et. al., "C4ISR Archit ectures: II. A Structured Analysis Approach for Architecture Design." *Systems Engineering*, 3(4): 248-286, 2000.

Zinn, A., *The Use of Integrated Architectures to Support Agent Based Simulation.* Department of Aeronautics and Astronautics, Master's thesis. Air Force Institute of Technology, Wright-Patterson Air Force Base, OH, 2004.

Biographies

Matthew Richards is a graduate student at MIT pursuing Master of Science degrees in Aeronautics and Astronautics and Technology and Policy. Supported by MIT's Lean Aerospace Initiative, Matt's research focuses on systems engineering, space systems architecture and design, emergent behavior, and innovation management. His work experience includes Mars rover mission design at the Jet Propulsion Laboratory and systems engineering support on two autonomous vehicle programs for the Defense Advanced Research Projects Agency. Matt received an S.B. degree in Aeronautics and Astronautics from MIT in 2004.

Nirav Shah is a graduate student at MIT pursuing a Ph.D in Aeronautics and Astronautics. Supported by MIT's Lean Aerospace Initiative, his research focuses on architecture of systems of systems. In particular, he is studying the impact of coupling in the physical, functional, and organizational spaces on system evolution. His work experiences include positions at Los Alamos National Laboratory and with Booz Allen Hamilton. Nirav received an S.B. (2001) degree and an S.M. (2004), both in Aeronautics and Astronautics, from MIT.

Daniel Hastings is a Professor of Aeronautics and Astronautics and Engineering Systems at MIT. Dr. Hastings has taught courses and seminars in plasma physics, rocket propulsion, advanced space power and propulsion systems, aerospace policy, technology and policy, and space systems engineering. He served as chief scientist to the U.S. Air Force from 1997 to 1999 and as director of MIT's Engineering Systems Division from 2004 to 2005. He is a member of the National Science Board, the International Academy of Astronautics, the Applied Physics Lab Science and Technology Advisory Panel, and the Air Force Scientific Advisory Board.

Donna Rhodes holds a Ph.D. in Systems Science from the T.J. Watson School of Engineering at SUNY Binghamton. Her research interests are focused on systems engineering, systems management, and enterprise architecting. Dr. Rhodes has 20 years of experience in the aerospace, defense systems, systems integration, and commercial product industries. Prior to joining MIT, she held senior level management positions at IBM Federal Systems, Lockheed Martin, and Lucent Technologies in the areas of systems engineering and enterprise transformation. Dr. Rhodes is a Past-President and Fellow of the International Council on Systems Engineering (INCOSE).