Architectural Modeling Considerations For an Autonomous Air Traffic Control System

By Lanny Fields

Senior Engineer
Honeywell International, Phoenix, AZ

For SAE 574:
Net-Centric Systems Architecting and Engineering
Prof. Ken Cureton
Fall 2008
Section #32344

Table of Contents

1. Abstract 4
2. Background 5
3. AATCS System Description 6
3.1. AATCS System Overview 6
3.2. AATCS System Operation 6
3.3. AATCS System Elements 8
3.3.1. Ground Station Network 8
3.3.2. Ground Stations 8
3.3.3. External Data Sources (SWIM System) 9
3.3.4. Air-to-ground network 10
3.3.5. Airplanes and the Flight Control System (FCS) Network 10
3.3.6. Airplane to Airplane Network 11
4. AATCS System Architecture Modeling and Analysis 12
4.1. Spiral Development Model Stages 12
4.2. UML System Diagrams 16
4.2.1. Airplane Approach and Landing Description 16
4.2.2. Use Case Diagram 17
4.2.3. Class Diagram 22
4.2.4. Sequence Diagram 23
4.3. DoDAF Architectural Diagrams 26
4.3.1. Top-Level Operational View (OV-1) 26
4.3.2. Operational Node Connectivity (OV-2) 27
4.3.3. Information Exchange Requirements (OV-3) 28
4.3.4. Top-Level System View (SV-1) 32
4.3.5. System Interface Details (SV-3) 33
4.4. AATCS in the Enterprise Architecture Framework 35
4.4.1. Zachman Framework 35
4.4.2. Federal Enterprise Architecture System Component Reference Model 37
4.5. Ontologies and Semantic Models 39
4.5.1. AATCS Ontology Examples 39
5. Summary/Conclusions 43

6. Acknowledgements 44
7. References 44

Table of Figures

Figure 1 – Automated Air Traffic Control System 7
Figure 2 – Regional Ground Station Network 9
Figure 3 – Airplane Flight Control System Network (simplified) 11
Figure 4 – AATCS Spiral Development Model 13
Figure 5 – AATCS Fault-Free Airplane Arrival Use Case Diagram 19
Figure 6 – AATCS Top-Level Class Diagram Example 23
Figure 7 – AATCS Airplane Arrival Sequence Diagram Example 25
Figure 8 – AATCS High-Level Operational Concept Graphic Diagram (OV-1) 27
Figure 9 – AATCS Operational Node Connectivity Diagram (OV-2) 28
Figure 10 – AATCS Operational Information Exchange Matrix (OV-3) 31
Figure 11 – AATCS System Information Description (SV-1) 32
Figure 12 – AATCS System Interface Characteristics Matrix (SV-3) 34
Figure 13 – AATCS Zachman Framework Diagram 36
Figure 14 – AATCS FEA SRM Example 38

Table of Tables

Table 1 – Description Of Enumerated Values in Spiral Development Model 14
Table 2 – Airplane Approach and Landing Sequence 16
Table 3 – Airplane Approach and Landing Sequence with Use Case Correlation 20
Table 4 – Sequence Diagram Key 24
Table 5 – AATCS Semantic Model Behavior Characterization, Flight Control Computer 39
Table 6 – AATCS Semantic Model Behavior Characterization, Air-to-Ground Network Transceiver 40
Table 7 – AATCS Semantic Model Behavior Characterization, Hydraulic Actuator 42

1. Abstract

The current trend toward the use of digital systems in airplane and air traffic control has been made possible with the exponential increase in the computational capabilities of processor-based systems. Another well-known trend in the aerospace industry is the ever-increasing amount of air and ground traffic resulting from airlines and airports attempting to accommodate the needs of the flying public – a growing population in and of itself. The intersection of these suggests that one possible solution to alleviate air travel congestion could be the automation of air traffic control and allowing it to have direct control over airplane flight paths. Such a system would, in theory, reduce the workload of the flight crew and the air traffic controllers, as well as increase traffic flow. The system would also be very complex even by modern air traffic and data processing standards; great care would need to be taken in developing its architecture in order to properly design the system.

This paper is the second in this series which presents several analyses of such a conceptual system from a "net-centric" perspective. First, the system's operation is described from the context of a flight, in order to provide a basis for the discussion of various system models and views. Spiral development model stages as well as related events which occur during system design give an idea of how the system would be developed incrementally. Architectural analyses in the form of Unified Modeling Language (UML) models and Department of Defense Architecture Framework (DoDAF) operational and system views are presented in order to characterize and model various aspects of the system. Other analyses include looking at the system from a business context: formulation as an enterprise architecture (EA) and what services are provided. Finally, semantic models and ontological considerations are discussed.

The concept of an automated air traffic control system which controls airplanes requires a high degree of operational integrity and availability. The overall benefit to the flying public is to allow more airplanes to be in the sky and on the ground – an obvious necessity as air traffic continues to increase. However, with increased complexity comes increased risk and it is important to fully understand the issues in order to mitigate those risks. By applying techniques for architecting the system from many different viewpoints, including stakeholder viewpoints, the problems can be reduced to something more manageable and the system architecture can be communicated in unambiguous terms, thereby lessening the risk associated with developing the system.

About the author: Lanny Fields has worked as a systems engineer in aerospace for 15 years on numerous fly-by-wire flight control systems. His master's degree coursework at USC in the area of systems architecting and engineering has engendered a keen interest in architecting, modeling and analysis of complex systems, including net-centric architectures and systems. He is currently working on the Boeing 787 Dreamliner program.

2. Background

Air traffic congestion is rapidly becoming one of the major commercial transportation challenges at the start of the 21st century as more people take to the skies for their travel needs. "Forecasts indicate a significant increase in demand, ranging from a factor of two to three by 2025…. In short, U.S. competitiveness depends upon an air transportation system that can significantly expand capacity and flexibility, in the presence of weather and other uncertainties, while maintaining safety and protecting the environment. "[1] The FAA's current plan to comprehensively upgrade the existing air traffic control system to meet this projected demand is collectively called the Next Generation Air Transportation System, or 'NextGen'.

This paper will describe the concept of an Automated Air Traffic Control System (AATCS) which is designed to enhance the existing capabilities of NextGen in and around airports. Airplanes normally receive much of their real-time in-flight data from air traffic control over voice communication with the pilot, particularly during take-off and landing. Pilots then act upon the instructions that they received verbally, however, this system does not efficiently allow pilots to use all of the available airspace and it is prone to error. The AATCS seeks to improve upon this by allowing the flight path of the airplane to be controlled by commands from stations on the ground, with the pilot still able to assume control during degraded, reversionary and emergency modes of operation. One aspect of NextGen called "Free Flight" moves away from point-to-point travel in unrestricted airspace and allows pilots more discretion in flight planning to avoid zig-zagging between air traffic control stations, however, this freedom is curtailed while in restricted airspace.[2] This renders the AATCS less suitable for use during an airplane's cruise mode, but more desirable in the vicinity of an airport.

The potential benefits from using a system like the AATCS stem from its net-centric system characteristics, which are distinguished from network-centric characteristics by a key concept: an "enhanced ability to operate and use a system that has been enabled by network technology" by the primary enabling characteristics of a net-centric system: time, location independence and collaboration.[3] The combination of the enabling characteristics allows for rapid access and comprehension of time-critical information, rapid decision-making and action, the ability for nodes to communicate without knowledge of physical location, and permitting information exchange between nodes to facilitate the decision-making process.[4] The net-centric nature of the AATCS would effectively permit airplanes to fly more closely spaced apart while maintaining safe navigation throughout the departure and arrival patterns. This type of system requires a high level of security and a robust architecture as well as bandwidth and computational power in order to manage the complexity of the data being processed, transmitted and acted upon in real-time, such as a net-centric system is capable of providing.

The AATCS system's operation will be presented along with a spiral development model showing the project's development. The architecture will then modeled from several

perspectives using UML and DoDAF diagrams, EA techniques, and semantic models. Key examples of these modeling languages will be provided; analyses which exhaustively cover the entire suite of models within each language or framework are outside the scope of this paper.

3. AATCS System Description

Although the system description of the AATCS was presented previously in a different paper, the information is included here in order to provide clarity and context for the analyses discussed in this paper.

3.1. AATCS System Overview

The Automated Air Traffic Control System consists of two primary system element types: ground stations and airplanes. The two types are connected via an air-to-ground wireless network and are in constant communication with the other nodes in the network. Each system element type also communicates with other network members of its own type: ground stations within the vicinity of an airport are linked to each other and airplanes communicate with other airplanes within range. Ground stations have additional interfaces with secondary system elements such as external data sources. Airplanes possess their own internal networks which connect on-board subsystems to flight control computers. Each element and its architecture and interfaces are described in further detail in this section.

A top-level diagram of the system is shown in Figure 1.

3.2. AATCS System Operation

The operation of the system is best described in the context of a flight.

When an airplane has taxied onto the runway and is ready to depart, it connects to the air-to-ground network and, after authentication, begins processing the flight commands it receives from the ground stations. At this point, the flight commands are nothing more than instructions to hold for take-off. The airplane also connects to and similarly authenticates with the air-to-air network. After the airplane verifies that the data received is valid, the pilot engages the automatic control system and allows the flight commands from the ground stations to have full authority. The airplane then accelerates through take off and rotation into the air. As the airplane follows the computed trajectory during climb to its cruise altitude, it eventually loses contact with the ground stations at the point of departure. If the commands become invalid or communication is lost at any point during these maneuvers, the system automatically disengages and the pilot takes over.

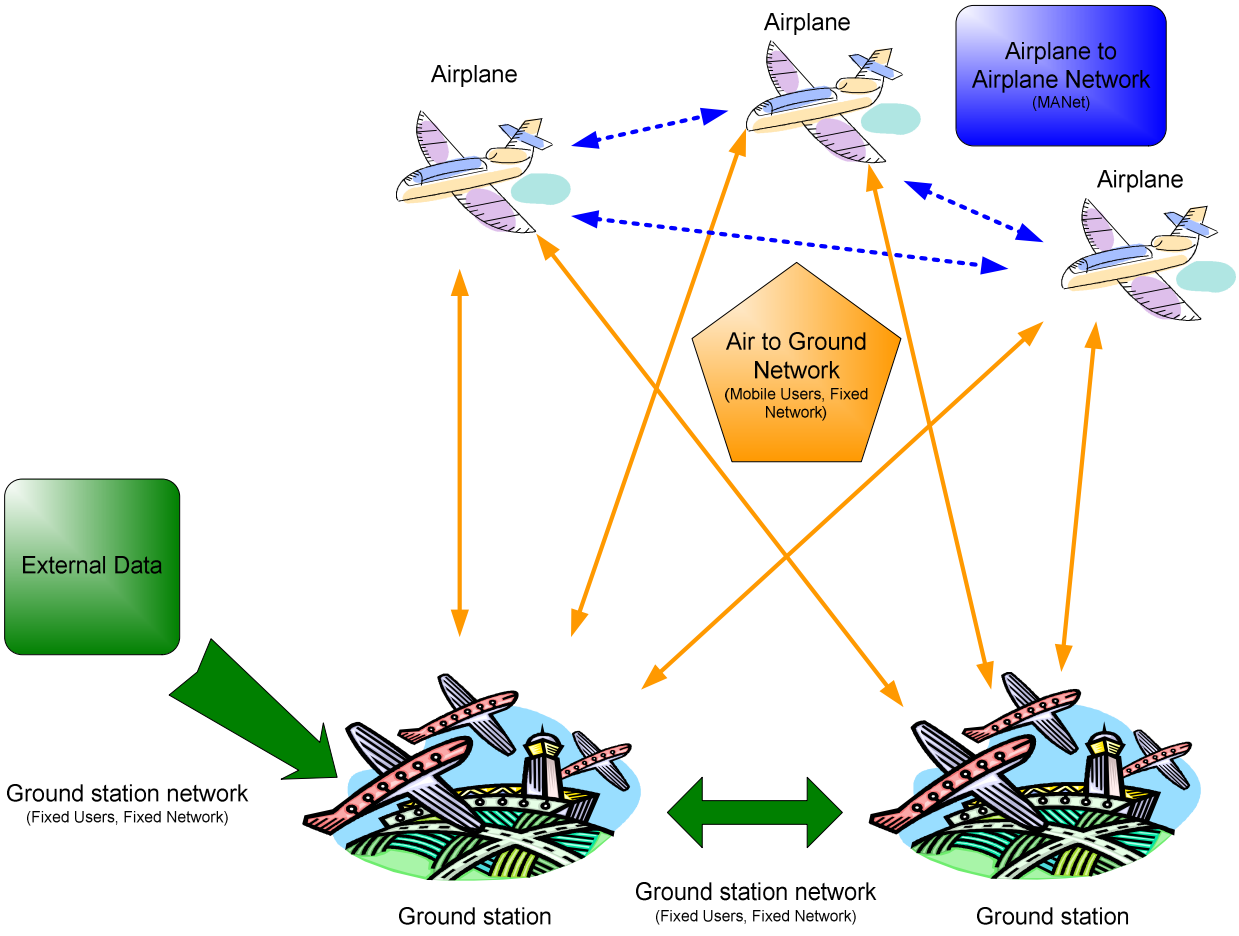

Figure 1 – Automated Air Traffic Control System

The pilot is assumed to take control at this point for Free Flight during cruise for the reasons previously mentioned in the Background section. However, the pilot could decide to allow the automatic mode to continue computing the flight vector and fly the plane based on the last valid commands received and its current position, with updates provided by any "waypoint" ground stations it connects to and authenticates with en route. The airplane does not attempt to connect with another airplane in an ad-hoc air-to-air network until it reaches its destination.

As the airplane enters the airspace of the destination airport, it once again connects to and authenticates with the local air-to-ground and air-to-air networks. The system performs the same actions as during take-off, though in reverse. The pilot, if in command, relinquishes control of the airplane after the data from the local networks has been validated. The airplane then automatically slots itself for approach and landing, in accordance with the ground station's instructions. After landing, the airplane taxies off the runway and transitions back to pilot

control before reaching the gate. One possible improvement might be to include automated maneuvers to guide the airplane all the way back to the gate, however, the discussion and analysis of that part of the system is beyond the scope of this paper.

3.3. AATCS System Elements

3.3.1. Ground Station Network

The ground station network is comprised of ground stations which continually receive and process data from external sources, which are described in further detail below. The ground stations also communicate with each other and verify their results against the results received from other ground stations in the airport's network. The processed results, which are the flight commands for airplanes in the network, are broadcast wirelessly while the stations and the external data sources are all connected via a fiber optic backbone.

A notional diagram of the regional ground station network is shown in Figure 2.

3.3.2. Ground Stations

The ground stations are located in close proximity to airports and can be co-located with air traffic control. Each station receives flight-related data on the ground station network from the new NextGen System-Wide Information Management (SWIM) system, which is a service of the National Airspace System (NAS) that provides "surveillance, weather, and flight data, aeronautical and NAS status information"[5]

The ground station processes the data continually and transmits flight path corrections on a real-time basis (once per second) to the airplanes within its airspace. These corrections are commands which are received and interpreted by the flight control computers on an airplane. The ground stations also receive flight vector and status information from airplanes, as well as data from other ground stations to cross-check the flight commands that were transmitted to the airplanes.

The stations' processors and data storage for non-repudiation are located on-site in separate server rooms. Station personnel monitor the data in real-time on displays and consoles, advising pilots and remote ground station personnel as needed during normal operations and continuously during an emergency situation.

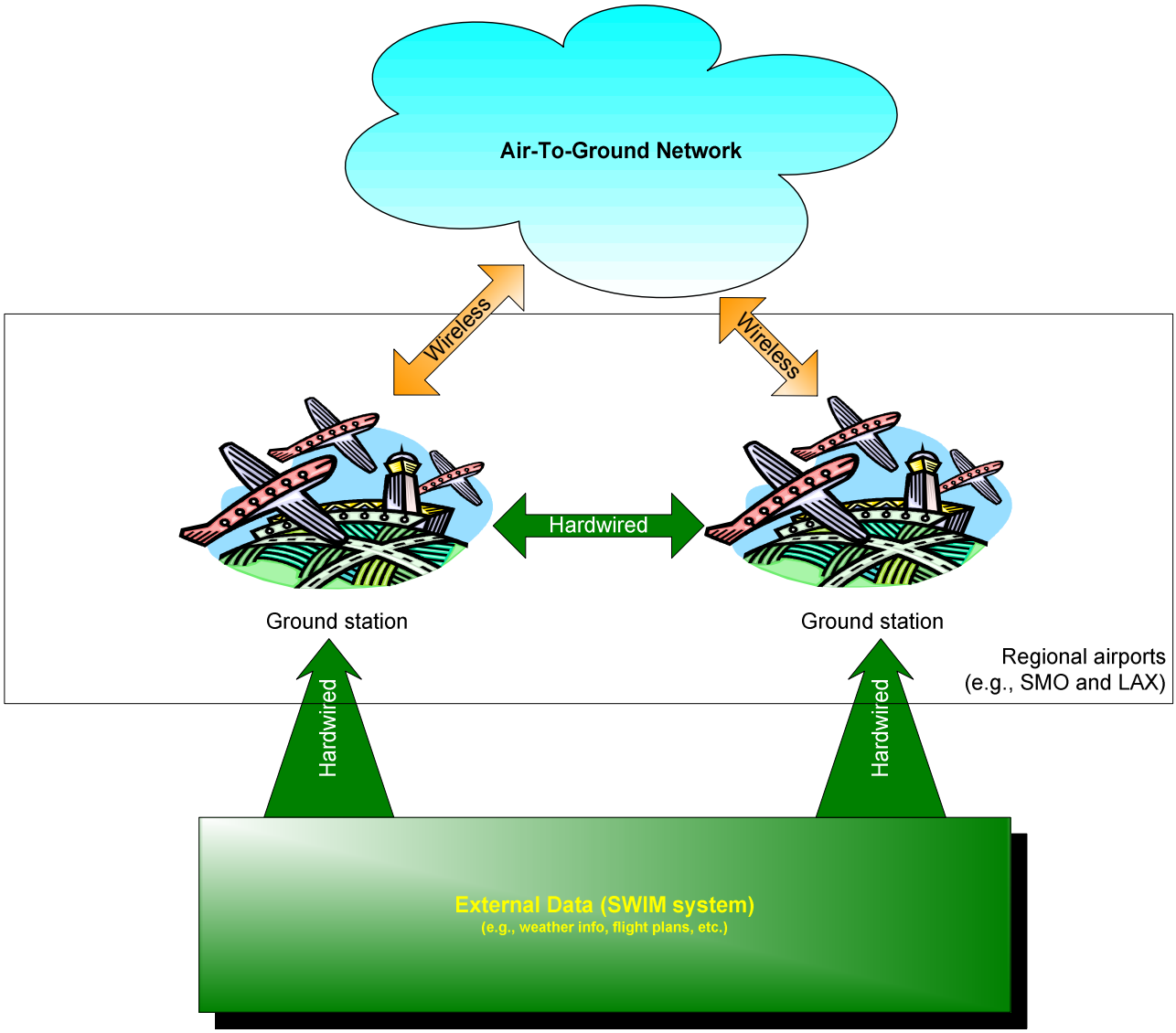

Figure 2 – Regional Ground Station Network

3.3.3. External Data Sources (SWIM System)

External data from a variety of sources is required in order to analyze and understand the impact of effects in real-time which could affect air travel:

- Local weather data from the immediate airfield and surrounding airports
- A new national weather data "enterprise service dissemination of common weather observations and forecasts to enable collaborative and dynamic NAS decision making" known as the NextGen Network Enabled Weather (NNEW)[6]
- Electronically filed flight plan information
- Coordinates and vectors of the local airspace traffic

- Reports of turbulence and flight path deviations from other airplanes
- Redundant data from stations within the regional ground station network

This data is concentrated using the NextGen SWIM system service and is broadcast to ground stations over the ground station network.

3.3.4. Air-to-ground network

The air-to-ground network is the wireless communication link between the ground stations that are within the range of airplanes in the local airspace. The ground stations provide flight commands to the airplanes based on the data received from the external sources and from flight vector information from the airplanes. Due to the critical nature of the data transmitted on the network, two transceiver channels are required to satisfy the corresponding FMECA events. Airplanes connect to the network and authenticate before receiving and accepting airplane-specific flight commands from the ground stations. An airplane will disconnect from the network when out of range from the ground stations for departures, or when taxiing from the runway for arrivals.

3.3.5. Airplanes and the Flight Control System (FCS) Network

Airplanes in the automated air traffic control system process the flight path commands received from the ground stations using a specific type of electronic unit which will be referred to as a flight control computer (FCC) for the purposes of this paper. FCCs are connected to the airplane's dual-redundant transceiver units over similarly redundant high-speed data bus links. The FCCs receive the flight path commands and data from other on-board sensors which relay inertial and local air data in order to provide commands to fly the airplane. The gust suppression system in the Boeing 777, which senses aerodynamic impulses and provides commands to actuators to counteract the gust and smooth out the ride for passengers, is one example of an on-board sensor suite[7].

Control laws resident in the FCC software then compute commands to drive actuators, which in turn move the flight surfaces and allow the airplane to maneuver as commanded. The FCCs also annunciate status messages on the displays and back-drive the cockpit controls in order to enhance the pilot's situational awareness. They provide status and actual flight path data to the ground stations, and provide certain redundant SWIM system data to other airplanes. The flight control system receives the data and uses it only as a monitor to indicate potentially corrupted ground station data; in this case, an alert is provided to the pilot, who can then decide whether or not to disconnect from the AATCS and pilot the airplane manually. The level of redundancy of the flight control system's elements serves to mitigate many of the failure modes associated with the flight control system, due to the flight critical nature of the FCS.

A top-level diagram of the airplane flight control system is shown in Figure 3.

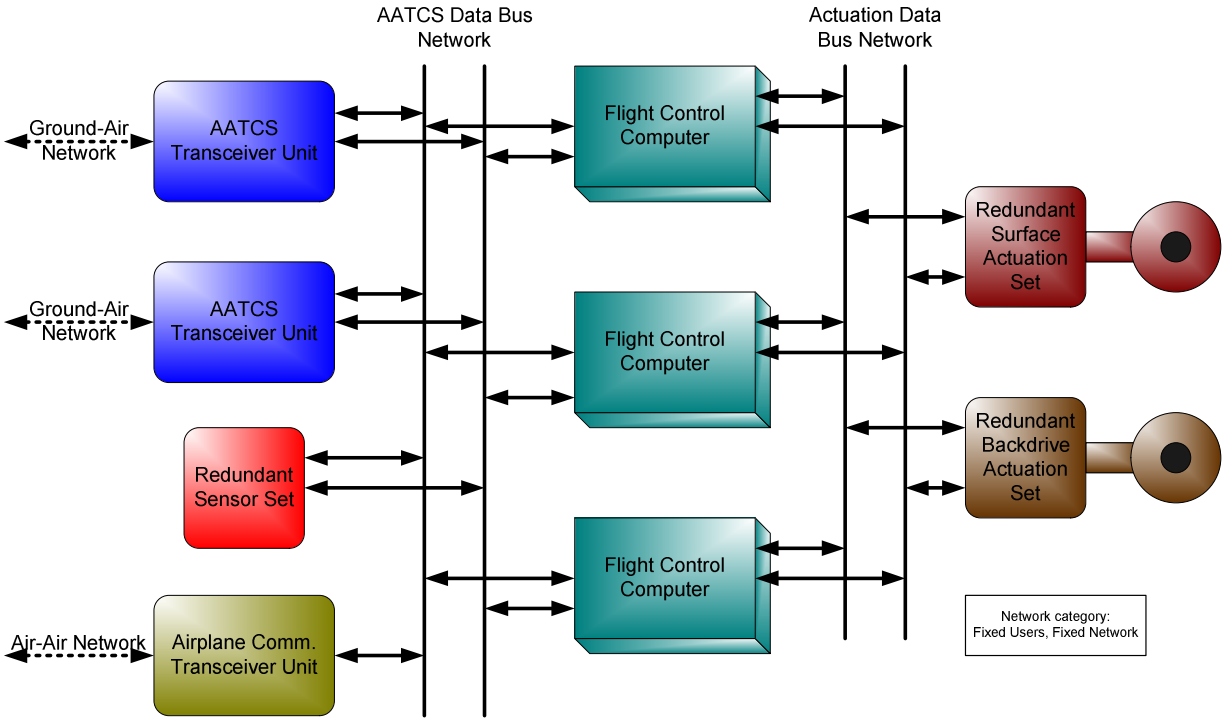

Figure 3 – Airplane Flight Control System Network (simplified)

3.3.6. Airplane to Airplane Network

The airplane-to-airplane network is a wireless network that connects the various airplanes within the vicinity of the airport. The data communicated on this network consists of redundant SWIM system data received from the ground stations that is rebroadcast to other airplanes for the purpose of monitoring data received from ground stations. Because this data is non-critical (i.e., not directly used in calculating commands to fly the airplane), only one transceiver channel is required to satisfy the corresponding FMECA events.

4. AATCS System Architecture Modeling and Analysis

Architectural modeling is an important enabler for the understanding and comprehension of a complex system because it can provide unambiguous representations or views of the system's architecture and behavior. One definition of a model is "a virtual or physical representation of an entity for purposes of presenting, studying and analyzing its characteristics such as appearance, behavior or performance for a prescribed set of operating environment conditions and scenarios."[8] Any system can be modeled from numerous points of view which are essentially projections of the system onto one or more operational domains. It should be noted, however, that although models may adequately represent the system for the purpose of further design and implementation, they are still a finite set of projections which limit their ability to exhaustively describe the system and its behavior because of the heuristic which states that "a model is not reality."[9] It is just as important to be aware of the models' limitations as the information contained in the models themselves.

Irrespective of their limitations, it is important to develop system models early in the development phase of a program in order for stakeholders – people with an interest in the development or outcome of the design – to develop a common understanding of what the system will look like and how it will operate. Without this, errors from misinterpreting or misunderstanding the system's characteristics creep into the design and create nontrivial problems (often very big problems) in terms of schedule and cost when the errors are discovered and need to be fixed. In fact, poor communications has been cited as the number one cause of project failure.[10] 'A picture is worth a thousand words' is a classic heuristic and a good set of system models can be worth their development cost by preventing errors which, if undiscovered, can propagate to later design, implementation and verification phases.

To that end, the spiral development model for the AATCS will be introduced first in this section, followed by a presentation of AATCS system architecture models. The models are diagrammed using the Unified Modeling Language (UML) and the Department of Defense Architectural Framework (DoDAF) specifications. Examples of Enterprise Architecture modeling will also be discussed, along with semantic modeling.

4.1. Spiral Development Model Stages

"The spiral model is a software development process combining elements of both design and prototyping-in-stages, in an effort to combine advantages of top-down and bottom-up concepts."[11] This approach allows for the iterative risk assessment of the design at various stages along the development path and "promotes quality assurance through prototyping at each stage in systems development." [12] Each loop of the spiral represents a single iteration and each quadrant represents one of four stages of design: determining objectives, alternatives and constraints; identifying and resolving risks; development and testing; and planning the next iteration.

As the spiral progresses outward from the origin, each successive loop builds on the previous iteration and provides incremental functionality and risk reduction prior to the next loop. The horizontal axis is labeled 'review' to indicate the point in the spiral where a review is required before proceeding into the next loop and the vertical axis is labeled 'cumulative cost' to show the accumulating cost per loop.

Figure 4 shows the spiral development model for the AATCS. The spiral does not start at the origin but instead starts already established in quadrant 1. This is to indicate that the first task to be done in the development of the system is an initial review and determination of objectives, alternatives and constraints at the very top level. The dashed radial line in quadrant 2 is the dividing line between risk (left) and prototype development (right) for a given loop. This shows that the risks must be assessed and a "phase gate" type of evaluation must be passed in order to allow development to continue for that iteration or phase. If the evaluation does not meet its pre-determined criteria, development can be halted or terminated.

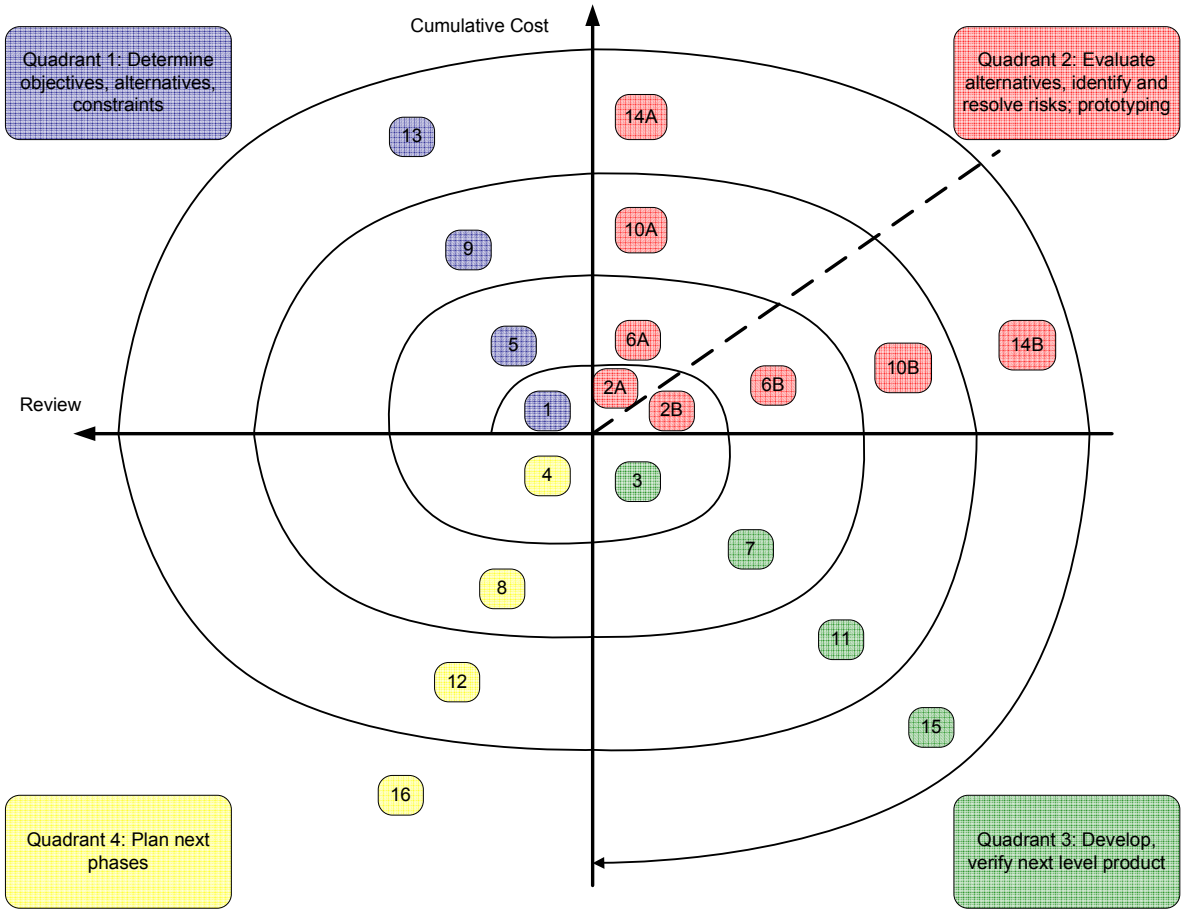

Figure 4 – AATCS Spiral Development Model

Figure 4 depicts enumerated indications of activities in each quadrant in each loop of the spiral, presented in a compact format for clarity. The corresponding textual descriptions of the enumerated indications are presented in Table 1. Colors in Table 1 provide a visual indication of different loops in the spiral and are not representative of any quadrant.

Table 1 – Description Of Enumerated Values in Spiral Development Model	
Number	Description
1	Mission objectives definition and concept operation diagrams from alternatives
2	A. Risk assessment: Is the development of a solution which meets the mission objectives feasible? Buy-in from regulatory agencies and airlines to ensure understanding of problem statement.
	B. Evaluate: Decide on best solution
3	Develop and evaluate mission requirements, use case diagrams, EA diagrams, DoDAF operational view (OV-x) diagrams and system view (SV-x) diagrams for all AATCS elements
4	Develop top-level program plan, SOW and schedule. Define phases of incremental functionality, develop engineering planning documentation
5	Define system requirements and allocate functions to AATCS elements, develop high-level hardware and software requirements
6	A. Phase gate review: Assess risks associated with mission requirements, use cases and plans
	B. Prototype: Develop simulations based on architectural models, use cases and requirements, initial hardware and software prototypes
7	Evaluate performance of architectural models and use cases, requirements validation
8	Develop framework for hardware and software development, integration and testing
9	Develop lower-level requirements and detailed design
10	A. Phase gate review: Evaluate results of architectural model simulations and use case performance and requirements validation, assess risks associated with hardware/software development
	B. Prototype: Develop lab equipment, simulation and test capability with real hardware and software

Table 1 – Description Of Enumerated Values in Spiral Development Model	
Number	Description
11	Develop initial implementation of system, initial integration and dry-run testing
12	Plan for formal qualification of system
13	Review results of dry-run testing, update requirements, design and implementation as needed, perform regression tests
14	A. Phase gate review: Formal test readiness review
	B. Prototype: N/A
15	Final testing and certification of system
16	Deploy system

The first loop of the spiral (numbered items 1 - 4) describes the mission-level objectives and requirements development. The risk assessment is made against the mission-level problem statement and if the authorization is given then mission requirements and use cases are developed and planning activities are started.

The objectives to be determined for the second loop (items 5 - 8) involve determining the system requirements, functional allocations between AATCS elements and high-level software and hardware requirements that will be used in the development of the design. The risk assessment is again performed – this time, against the finalized mission requirements, AATCS use cases, DoDAF diagrams, architectural models and planning documents from the previous loop. If the program is still deemed feasible, simulations based on the architectural models as well as 'proof-of-concept' types of elements of the hardware and software are prototyped and tested.

The planning for the next iterative loop (loop 3) involves developing the framework for the next stage of development: lower-level requirements, design and implementation of the AATCS. The risk assessment here is the evaluation of the architectural model simulation testing as well as looking ahead to potential implementation issues. If this phase gate is passed, lab equipment, simulations and other test capabilities are prototyped to support integration and testing of the implemented AATCS hardware and software. In this case, some of the development work can utilize the outputs (i.e., the architectural model simulations) of the previous loop. Informal "dry run" testing of the overall system is performed and plans for formal qualification testing of the AATCS are developed in preparation for the final iteration.

The start of loop 4 involves feeding back issues with dry run testing back into the requirements, design and implementation to help prepare for the formal test readiness review – the final risk assessment. No prototyping is included following the review; the AATCS is ready to be deployed

after a successful formal qualification test. Certification of the system with regulatory agencies follows qualification testing and is the final development stage prior to delivery.

It should be noted that, although not indicated on the diagram, each development stage is likely to have "mini-loops" iterating within its main loop. The entire AATCS is developed simultaneously rather than by one element at a time, to minimize re-work of finalized subsystems based on 'gotchas' or errors found with other subsystems. This also minimizes unanticipated emergent behavior that can arise when subsystems are integrated into successively higher-level systems which are not developed holistically.

4.2. UML System Diagrams

UML is perhaps the most well-known commercial industry modeling language today. The Unified Modeling Language is a method by which one can "describe a complex system rigorously and unambiguously…such that the integrated system design can be tested and verified to meet requirements before generating any code or designing any hardware,"[13] for the reasons mentioned previously. System modeling takes place in task 3 of the spiral development model, which is early enough to provide assurance that the mission requirements, the overall system architecture, and the subsequent hierarchical decomposition are communicated among and understood by the program stakeholders.

The UML system architecture diagrams presented in this section are described from the use case perspective of an airplane's approach and landing. However, in order to model the system correctly, a brief discussion to provide understanding of the mission-level operations for this use case shall first be presented.

4.2.1. Airplane Approach and Landing Description

Some of the high-level description of an airplane's fault-free approach and landing has been previously mentioned in the discussion of the AATCS system's operation. Additional details which describe the order of events (shown in Table 2) can also be used to help establish the context for modeling the system properly using UML. It should be noted that understanding the system's fault response, which can be based on a system-level FMECA, is also required to generate a more complete model of the system. Such an analysis is beyond the scope of this paper.

Table 2 – Airplane Approach and Landing Sequence

Step	AATCS Operations
0	Airplane approaching destination airport, no communication with air-to-ground or air-to-air network. This is the initial condition.
1	Airplane acquires air-to-ground network and sends request to connect. Airplane ignores air-to-air network traffic. Connection request stored as non-repudiation data.
2	Ground station uses security functions to decrypt data, uses SWIM data to authenticate,

Table 2 – Airplane Approach and Landing Sequence	
Step	AATCS Operations
	authenticates airplane. Authentication stored as non-repudiation data.
3	Airplane receives authentication verification, transmits flight vector data and status; pilot manually engages 'AATCS armed' mode to allow commands to be accepted from the ground station. Pilot and FCS monitor the system. Authentication verification, flight vector data and status stored as non-repudiation data.
4	Ground station begins calculating and transmitting flight commands once per second using SWIM data and received flight trajectory information from airplane. Flight commands stored as non-repudiation data.
5	Pilot and FCS monitor the system. Airplane receives flight commands, computes actuation and engine commands for flight vectoring, transmits status back to ground station. Connects to air-to-air network. Flight vector data and status stored as non-repudiation data. Repeat steps 4 and 5 until on ground.
6	Pilot and FCS monitor the system. Airplane lands, taxis off runway, disconnects from air-to-air network on taxiway. Sends request to disconnect message to ground station. Disconnect request stored as non-repudiation data.
7	Ground station receives request disconnect message, terminates connection. Disconnect request stored as non-repudiation data.
8	Pilot and FCS monitor the system. Airplane disconnects from networks and disengages AATCS mode and reverts to pilot control. Disconnect message stored as non-repudiation data.
9	Ground station logs successful flight completion (non-repudiation), general clean-up.

Additionally, "global" events which are not step-dependent and are not mentioned here, such as updating the airport environmental (weather) status, should be similarly taken into account prior to system modeling.

4.2.2. Use Case Diagram

The use case diagram "provides a tool for organizing system requirements in order to understand interactions between:

- "Actors" that make a request, and
- "Activities" made in response by the system"[14]

Use cases define the operation of the system (the activities) from the perspective of one or more users of the system (the actors). In a use case diagram, actors are represented by stick figures and the system is represented by a block with a clearly defined boundary of what is internal and external to the system. Activities in the system are modeled as ellipses; the use of the word "node" is avoided here to prevent confusion with activity nodes in the DoDAF OV-2

diagram. Actors are connected to one or more activity. External systems are also represented as blocks with lines that connect to activity ellipse in the system. Actors inside a block represent actor-controlled systems which connect to one or more activities. An example of this is when a pilot arms the AATCS mode in step 3 of the airplane approach and landing sequence – although the action initiated by a user, the connection to the activity is via the airplane manual control 'system'.

The AATCS use case diagram in Figure 5 shows the "fault-free arrival" use case for the AATCS. Table 3 correlates the actors and activities in the use case diagram back to the steps in the airplane landing sequence. Note that the Ground Station Operator actor, External Data Source external system and the Update Environmental Data and Provide Information Assurance activities are considered "global" as previously defined and are not mentioned in the table.

Figure 5 – AATCS Fault-Free Airplane Arrival Use Case Diagram

Table 3 – Airplane Approach and Landing Sequence with Use Case Correlation			
Step	AATCS Operations	Actor(s)	Activities
0	Airplane approaching destination airport, no communication with air-to-ground or air-to-air network. This is the initial condition.	N/A	N/A
1	Airplane acquires air-to-ground network and sends request to connect. Airplane ignores air-to-air network traffic. Connection request stored as non-repudiation data.	Airplane Flight Control System	Connect to air-to-ground network, store non-repudiation data
2	Ground station uses security functions to decrypt data, uses SWIM data to authenticate, authenticates airplane. Authentication stored as non-repudiation data.	Ground Station Computer System	Authenticate users, store non-repudiation data
3	Airplane receives authentication verification, transmits flight vector data and status; pilot manually engages 'AATCS armed' mode to allow commands to be accepted from the ground station. Pilot and FCS monitor the system. Authentication verification, flight vector data and status stored as non-repudiation data.	Pilot, airplane flight control system, airplane mode control	Arm AATCS mode, monitor system status, provide flight status, store non-repudiation data
4	Ground station begins calculating and transmitting flight commands once per second using SWIM data and received flight trajectory information from airplane. Flight commands stored as non-repudiation data.	Ground Station Computer System	Calculate flight commands, transmit flight commands, monitor system status, store non-repudiation data

Table 3 – Airplane Approach and Landing Sequence with Use Case Correlation			
Step	AATCS Operations	Actor(s)	Activities
5	Pilot and FCS monitor the system. Airplane receives flight commands, computes actuation and engine commands for flight vectoring, transmits status back to ground station. Connects to air-to-air network. Flight vector data and status stored as non-repudiation data. Repeat steps 4 and 5 until on ground.	Pilot, airplane flight control system	Connect to air-to-air network, fly airplane, provide flight status, monitor system status, transmit received commands, receive redundant command set from airplanes, store non-repudiation data
6	Pilot and FCS monitor the system. Airplane lands, taxis off runway, disconnects from air-to-air network on taxiway. Sends request to disconnect message to ground station. Disconnect request stored as non-repudiation data	Pilot, airplane flight control system	Fly airplane, provide flight status, monitor system status, transmit received commands, disconnect from network, store non-repudiation data
7	Ground station receives request disconnect message, terminates connection. Disconnect request stored as non-repudiation data	Ground Station Computer System	Disconnect from network, store non-repudiation data
8	Pilot and FCS monitor the system. Airplane disconnects from networks and disengages AATCS mode and reverts to pilot control. Disconnect message stored as non-repudiation data.	Pilot, airplane flight control system	Disconnect from network, disconnect AATCS mode, store non-repudiation data
9	Ground station logs successful flight completion (non-repudiation), general clean-up.	Ground Station Computer System	Store non-repudiation data

4.2.3. Class Diagram

UML class diagrams "show the static structure of the system at an abstract level."[15] In object-oriented programming, classes are abstract representations of software objects, which are, in turn, instantiations of the class. Classes contain two types of information: attributes (data), and methods (functions) which operate on the attributes. A graphical representation of classes and their instantiated objects form a hierarchy by which instantiated objects inherit the characteristics of the class, though the data contained within the attributes of the class may be different. For example, a Tom object instantiation of a class Person might have a height attribute of 6.2 feet, while a Harry object might have a height attribute of 5.75 feet. Continuing the example, each would inherit a "how_high_can_I_jump()" method which could operate on their height attribute, using a predetermined formula to return a "maximum jump height" parameter. The results would be different (unless other non-height factors conspired to make the results the same) even though the attributes were the same. The parentheses after the name of the method indicates that it is essentially equivalent to a function call in software.

The class diagram in Figure 6 shows representations of the two primary object templates in the AATCS, the AATCS subsystem and the network. The highest, most abstracted level of the class hierarchy for net-centric systems would generally depict only a generic object type in the system and possible connection methods (i.e., the network(s)). Each class in the diagram has three fields; from top to bottom, they are the name of the class, the attributes associated with the class, and the methods associated with the class. Additionally, the connections in the hierarchy depict aggregation, inheritance and multiplicity. A closed (filled) diamond indicates that the parent class is comprised of N child level items, with N being a specific or unspecific number or range of numbers such as 1, 1..3 (1 to 3), N, or 1..* (1 or more). The example shows that the AATCS is comprised of 1 to 5 network types (air-to-air, air-to-ground, ground station network, AATCS data bus network, or actuation data bus network).

There are three types of lower-level classes called ground control system, SWIM system and airplane connected to the subsystem class. These inherit the attributes and methods listed in the respective fields in their parent class; the inheritance is shown by the open (unfilled) triangular arrows pointing up from the child classes to the parent class. Each of the three inherited classes has continuing levels of decomposition which are left out of the diagram for clarity except for a few key examples. Similarly, the network class has the five child classes which inherit characteristics from it, as previously described. The empty fields of the child classes indicate that they are not instantiatable – they are the equivalent of abstract classes in object-oriented programming.

The SWIM system class is the only class which is decomposed in greater detail in this example; the other classes at this level all decompose to one or more levels further down in the overall hierarchy. The SWIM system class is shown to be comprised of SWIM flight data and system status. The flight data class contains attributes of weather data, airplane flight plan data and pilot authentication data, which are all "inputs" to the class and are annotated as private data

(the minus sign preceding the name). Private attributes are not exposed to other classes. The fourth attribute, SWIM data, represents the outgoing message to the ground station and is considered public data (the plus sign preceding the name) because it would be exposed to other objects as part of the transmission process allocated to the public method "provideFlightPlanData()". In the system status class, the internal status methods and the data attributes are private and the message attributes, along with the display message method, are public. By continuing this process for all objects, we can derive a hierarchical representation of each object in the system which describes not only the data but also the actions performed on that data.

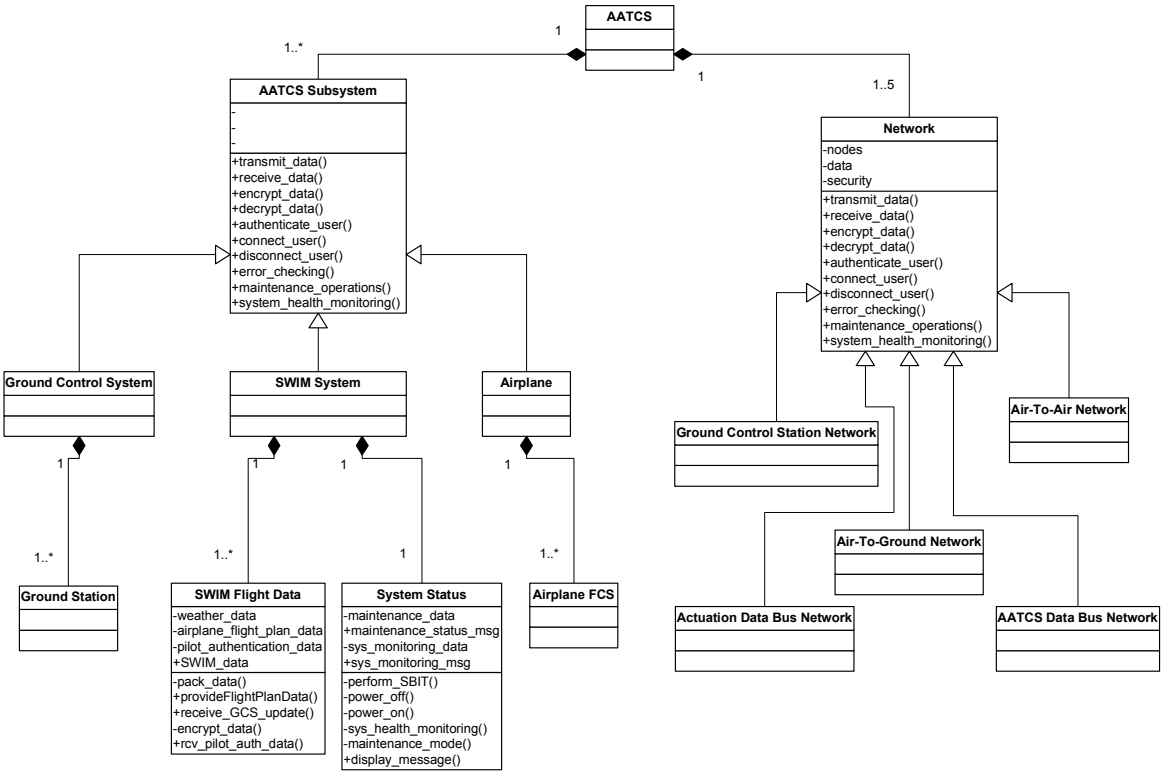

Figure 6 – AATCS Top-Level Class Diagram Example

4.2.4. Sequence Diagram

A UML sequence diagram will "model logic flow within a system in a visual manner, especially dynamic modeling of system behavior."[16] The sequence diagram does this by showing interactions between objects at various points in time, in sequential order. Time is represented increasing from top to bottom and each entity (e.g., object, actor, etc.) will have a dashed vertical line beneath it which indicates its lifetime within the sequence. Objects in particular are

shown as instantiations of their class; they are specified as OBJECT:CLASS. Object lifelines turn into a wider bar (an 'activation') upon instantiation and return to the dashed lifeline when the object has been removed from the sequence (i.e., destroyed or de-allocated). "The activation represents an execution of an operation the object carries out."[17] Each activity line contains the name of the message associated with the source object, along with data that is passed to the destination object. Other definitions of graphics in the sequence diagram are shown in Table 4. These definitions – which were not in the tabular format presented below – were obtained from "SAMS Teach Yourself UML in 24 Hours, Third Edition." This book is referenced in greater detail in the References section.

Table 4 – Sequence Diagram Key	
Entity	Meaning
line with closed (filled) arrowhead	synchronous message (calling object waits for response)
line with open arrowhead	asynchronous message (calling object does not wait for response)
solid line with arrowhead	call message (from originator)
dashed line (NOT the needline!)	return message (to originator)
filled dot at start of line	initial/entry state
line which loops back to originator	object performs action internal to itself
[condition in rectangular brackets]	perform action if condition resolves to true
(attribute in parentheses)	attribute passed to destination object
<<text in double brackets>>	final message for scenario
text in box with dog-eared page	comment or note

Figure 7 shows an example of the AATCS system during a plane's approach and landing sequence. The yellow circles represent steps in the Airplane Approach and Landing Sequence, as detailed in Table 2 and Table 3; these allow correlation back to the 'mission requirements' which drove the development of the diagram. By proceeding through the sequence diagram from left to right, top to bottom, the steps of the airplane approach and landing sequence and the interactions between objects can be followed.

There are some error conditions and system responses which are provided as examples in the sequence diagram. This was done in order to illustrate the usage of conditional messages in the diagram.

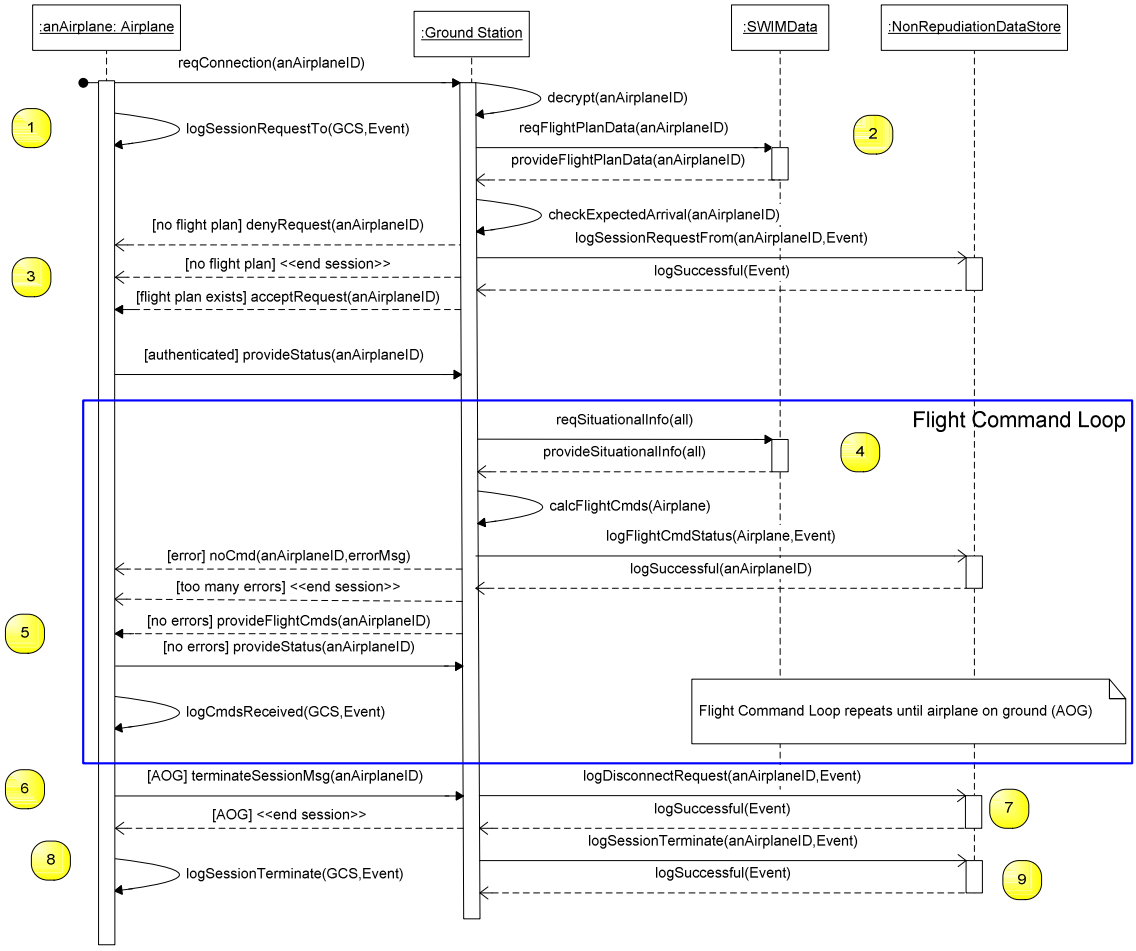

Figure 7 – AATCS Airplane Arrival Sequence Diagram Example

4.3. DoDAF Architectural Diagrams

"From a practical perspective, experience has demonstrated that the management of large organizations employing sophisticated systems and technologies in pursuit of joint missions demands a structured, repeatable method for evaluating investments and investment alternatives, implementing organizational change, creating new systems, and deploying new technologies. Towards this end, the DoD Architecture Framework (DoDAF) was established as a guide for the development of architectures."[18] The Department of Defense Architectural Framework (DoDAF) provides another means of capturing different views of the system architecture. The architecture framework is a "roadmap for the development process,"[19] necessitating its early position at task 3 on the spiral development model.

DoDAF architectural diagrams in version 1.5 of the framework are comprised of four types: operational view (OV) diagrams, system view (SV) diagrams, technical standards view (TV) diagrams and all view (AV) diagrams. This paper presents selected examples of the operational and system view diagram types.

4.3.1. Top-Level Operational View (OV-1)

The OV-1 diagram is a "high-level graphical/textual description of operational concept"[20] of the system whose purpose "is to provide a quick, high-level description of what the architecture is supposed to do, and how it is supposed to do it."[21] It is intended to show the idealized operation of the system at the mission-level with little to no description of implementation-level details.

Figure 8 shows the top-level operational view of the AATCS. The lower left of the diagram shows the pilot entering in the flight plan and authentication key which are used when attempting to connect to the air-to-ground networks at the departure and destination airports. The SWIM system is shown as an informational hub to which pilot information and both local and national weather data is disseminated to ground stations nationwide. Using this information, the ground stations transmit flight commands to airplanes based on arrival and departure queuing schedules, which is designed to implement the main AATCS goal of increased air traffic (represented by the stacked airplanes near the top of the figure). Finally, the diagram also shows the pilot engaging the AATCS on the ground at the departing airport and in the air upon entering the destination airport's airspace. Overall, the diagram visually depicts the major operations of the AATCS without providing implementation details.

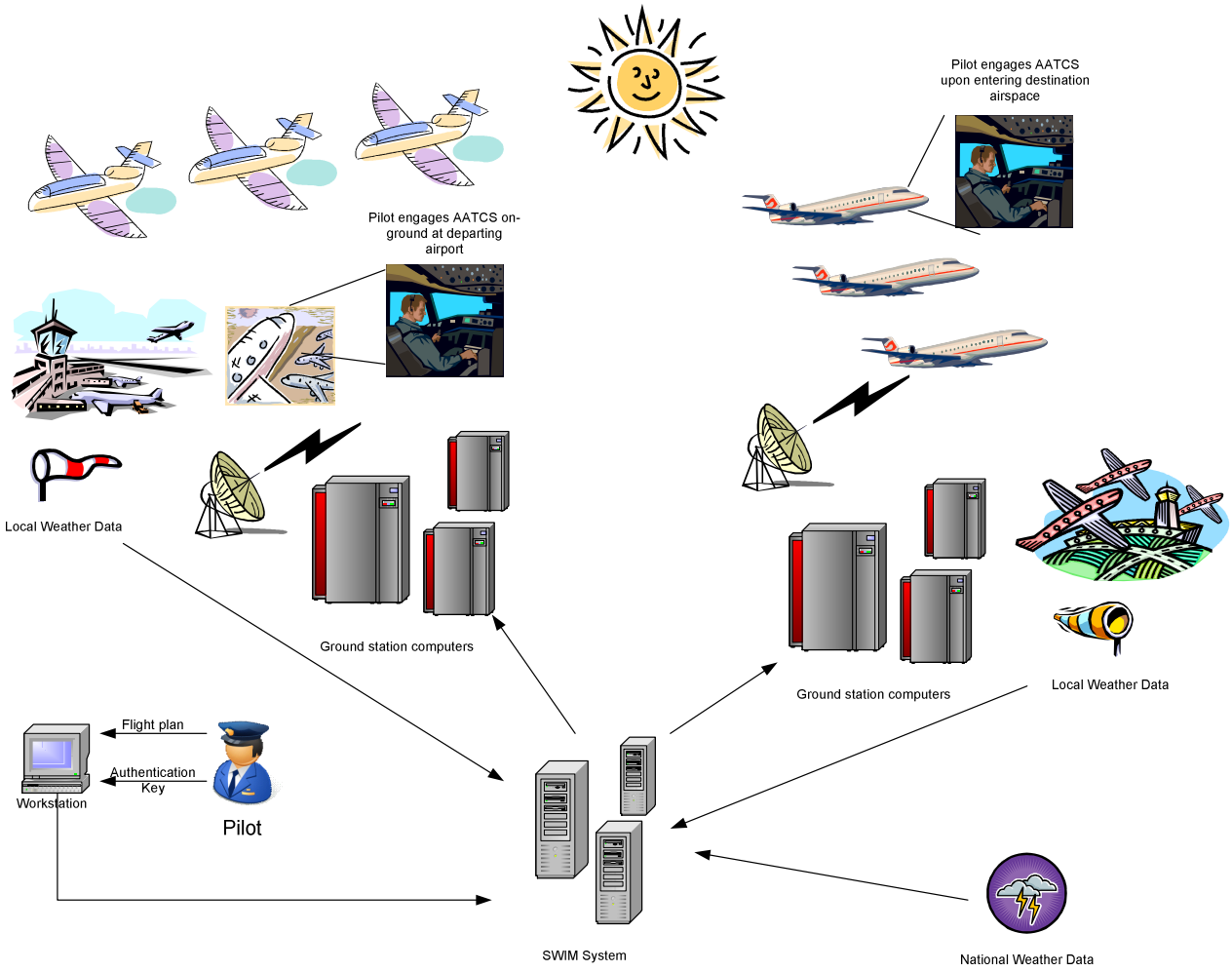

Figure 8 – AATCS High-Level Operational Concept Graphic Diagram (OV-1)

4.3.2. Operational Node Connectivity (OV-2)

The OV-2, or Operational Node Connectivity, diagram "graphically depicts the operational nodes (or organizations) with needlines between those nodes that indicate a need to exchange information…OV-2 is intended to track the need to exchange information from specific operational nodes (that play a key role in the architecture) to others. OV-2 does not depict the connectivity between the nodes."[22] In other words, operational node connectivity diagrams are independent of how the connections or the exchanges are implemented; implementation details are shown in system view diagrams. Finally, OV-2 diagrams also show connections to external sources and destinations.

Figure 9 shows the operational node connectivity of the AATCS. Some necessary details of the system data and activities to be accomplished are shown both in the needline information exchange description and in the node activity description. Needlines connect nodes to external

systems or to other nodes. Text in the 3-D boxes which point to a needline indicates what information is being exchanged and colored text near a processing node of the same color describes what activities are performed on the information being exchanged at that node. The complete set of needlines are shown for only one set of primary elements in the AATCS with the exception of the cross-checking data needed by other processing nodes for monitoring. Redundant descriptions and needlines for those processing nodes are not shown in order to preserve clarity.

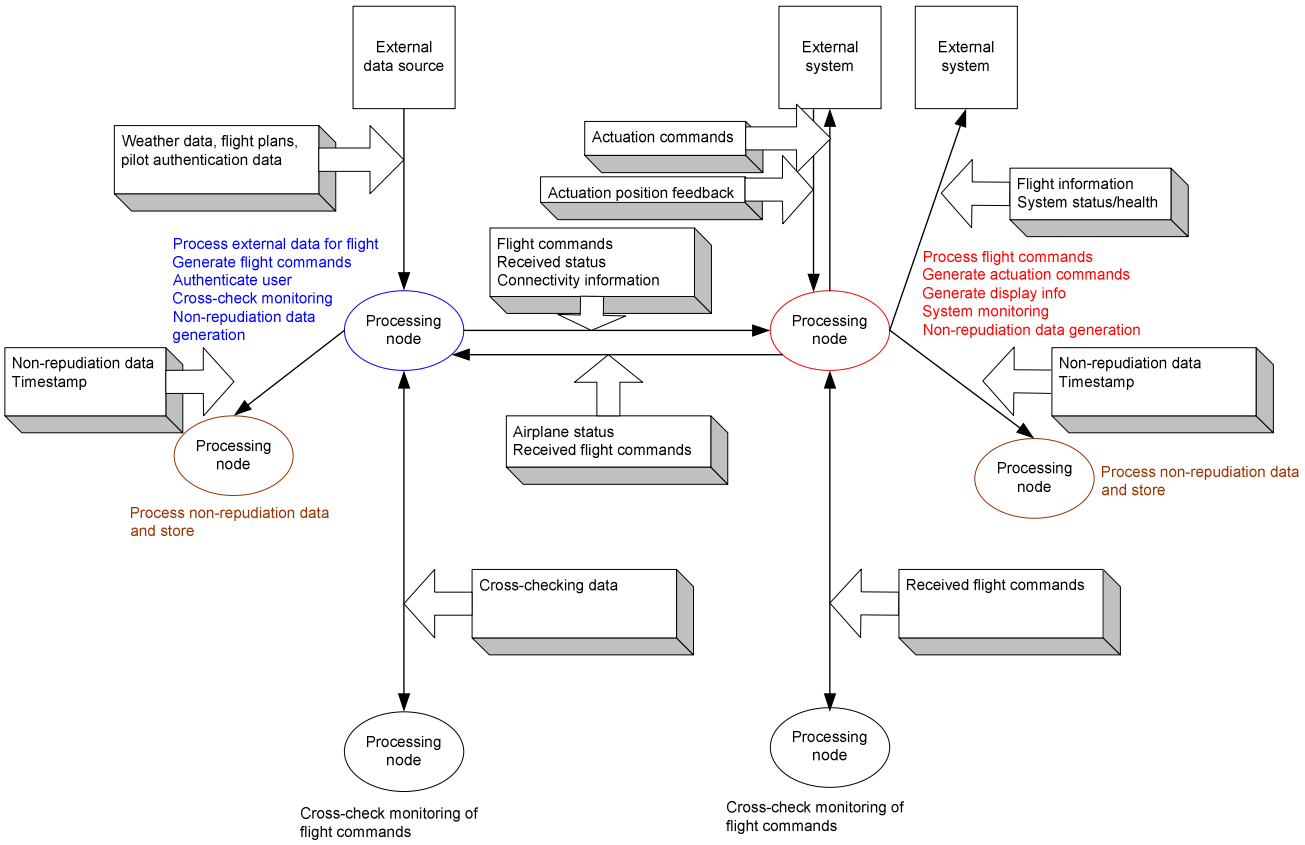

Figure 9 – AATCS Operational Node Connectivity Diagram (OV-2)

4.3.3. Information Exchange Requirements (OV-3)

The OV-3 diagram is a continuation of the OV-2 diagram and expands it by providing further details regarding the information exchanged between nodes: "The OV-3 details information exchanges and identifies "who exchanges what information, with whom, why the information is necessary, and how the information exchange must occur"…There is not a one-to-one mapping of OV-3 information exchanges to OV-2 needlines; rather, many individual information exchanges may be associated with one needline."[23]

In one sense, the OV-3 diagram serves as a notional high-level interface description, from which an interface control document (ICD) or an interface requirement specification (IRS) can be derived. In the spiral development model, the OV-3 diagram is developed in loop 1, as part of task 3. Developing the OV-3 diagram early in the program ensures that the architectural information flows down into the system interface requirements and design in the second loop.

The OV-3 is shown below in Figure 10 in tabular format. This diagram describes some of the key information exchanges in the AATCS between the ground station and the airplane. The diagram is divided into four color-coded areas: information description (green), information source (red), information destination (blue) and information exchange attributes (orange). The amount of information in the figure requires dividing the information in the entire OV-3 diagram into three smaller tables which are made consistent using the PUID parameter.

The information description area contains several columns which describes the characteristics of the data. The identifier column shows the project-unique identifier (PUID), which serves to uniquely label each element to ensure consistency and prevent miscommunication about data with similar names or other characteristics. The next two columns link back to the OV-2 diagram by specifying which operational element is being described along with the sub-element, to distinguish between different parameters in the same OV-2 information element. A description of the data is provided next, along with the media by which the data is exchanged. Finally, characteristics of the data are described as they are used by the system such as the data type, range, units and/or polarity.

The information source and destination areas are simple descriptions of the operational elements and producing or consuming activity associated with each data element. These also link back to the OV-2 diagram. The information exchange attributes describe, as the name suggests, the characteristics of the information exchanges rather than the data. The frequency describes how often the data is transmitted. The timeliness parameter indicates at what point within the transmitting frequency the data must be exchanged – for example, the 1 Hz transmission 'schedule' is conceptually divided into fixed slots called timeslices in which an element is expected to be transmitted. The element must be transmitted within this time slot in order for it to be 'timely'. The information assurance column describes what measures are applied to the data in order to provide information assurance – reference the previous paper in this series for IA details. The interoperability and usability requirement column defines what is needed in order to properly perform the information exchange.

Information Description								
Identifier (PUID)	OV-2 Operational Information Element	Operational Information Sub-element	Description	Media	Size	Data Type	Range	Units, Polarity
AG-0001	Flight commands	Heading	Flight correction information (heading) calculated by the ground station	Air-to-ground network	1	IEEE-32	0-359.99	Degrees
AG-0002	Flight commands	Altitude	Flight correction information (altitude) calculated by the ground	Air-to-ground	1	IEEE-32	0-30,000	Feet

Information Description								
Identifier (PUID)	OV-2 Operational Information Element	Operational Information Sub-element	Description	Media	Size	Data Type	Range	Units, Polarity
			station	network				
AG-0003	Received status	N/A	Status word received from airplane wrapped back to airplane with receiveOK bit set	Air-to-ground network	1	Packed Boolean	N/A	N/A
AG-0004	Connectivity information	Heartbeat	Incrementing activity counter (heartbeat) to indicate data is not stale	Air-to-ground network	1	Packed Boolean	N/A	N/A
AG-0005	Airplane status	AATCS disconnected	Status indicating that the pilot has assumed control of the airplane	Air-to-ground network	1	Boolean	N/A	1 = AATCS disconnected
AG-0006	Airplane status	Operational Mode	Enumerated indication of the mode that the airplane is currently flying in: normal, degraded or emergency	Air-to-ground network	1	32-bit Unsigned integer	1-3	1=Normal 2=Degraded 3=Emergency
AG-0007	Received flight commands	Heading	Flight correction information (heading) wrapped back to ground station to indicate proper receipt of command	Air-to-ground network	1	IEEE-32	0-359.99	Degrees
SW-0001	Pilot authentication data	Password	Password entered with flight plan before boarding airplane, used for authentication	Ground station network	16	Unicode encoded bytes	N/A	N/A
GN-0001	Ground station non-repudiation data	Timestamp	Indicate what events happened at a certain time	Ground station network	3	Time	ISO-8601 format	ISO-8601 format

Information Description	Information Source		Information Destination	
Identifier (PUID)	Operational Element	Producing Activity	Operational Element	Consuming Activity
AG-0001	Ground Station flight command processing node	Calculate and transmit flight commands to inbound airplane	Airplane FCS processing node	Receive and process flight commands from ground station
AG-0002	Ground Station flight command processing node	Calculate and transmit flight commands to inbound airplane	Airplane FCS processing node	Receive and process flight commands from ground station
AG-0003	Ground Station flight command processing node	Calculate and transmit flight commands to inbound airplane	Airplane FCS processing node	Receive and process status from ground station
AG-0004	Ground Station flight command processing node	Calculate and transmit flight commands to inbound airplane	Airplane FCS processing node	Receive and process status from ground station
AG-0005	Airplane FCS processing node	Transmit airplane status to ground station	Ground Station flight command processing node	Receive and process status from airplane
AG-0006	Airplane FCS processing node	Transmit airplane status to ground station	Ground Station flight command processing node	Receive and process status from airplane
AG-0007	Airplane FCS processing node	Transmit airplane status to ground station	Ground Station flight command processing node	Receive and process status from airplane

Information Description	Information Source		Information Destination	
Identifier (PUID)	Operational Element	Producing Activity	Operational Element	Consuming Activity
SW-0001	SWIM System	Transmit SWIM data to ground station	Ground Station flight command processing node	Receive and store pilot password for authentication
GN-0001	Ground Station flight command processing node	Store non-repudiation data	Non-repudiation processing node	Receive and store non-repudiation data

Information Description	Information Exchange Attributes			
Identifier (PUID)	Frequency	Timeliness	Information assurance	Interoperability and Usability Requirements
AG-0001	1 Hz	Fixed timeslice in major (1 Hz) frame	VPN, CRC, non-repudiation, authentication, checksum	AATCS system on inbound airplane; predefined message format
AG-0002	1 Hz	Fixed timeslice in major (1 Hz) frame	VPN, CRC, non-repudiation, authentication, checksum	AATCS system on inbound airplane; predefined message format
AG-0003	1 Hz	Fixed timeslice in major (1 Hz) frame	VPN, CRC, non-repudiation, authentication, checksum	AATCS system on inbound airplane; predefined message format
AG-0004	1 Hz	Fixed timeslice in major (1 Hz) frame	VPN, CRC, non-repudiation, authentication, checksum	AATCS system on inbound airplane; predefined message format
AG-0005	1 Hz	Fixed timeslice in major (1 Hz) frame	VPN, CRC, non-repudiation, authentication, checksum	AATCS system on inbound airplane; predefined message format
AG-0006	1 Hz	Fixed timeslice in major (1 Hz) frame	VPN, CRC, non-repudiation, authentication, checksum	AATCS system on inbound airplane; predefined message format
AG-0007	1 Hz	Fixed timeslice in major (1 Hz) frame	VPN, CRC, non-repudiation, authentication, checksum	AATCS system on inbound airplane; predefined message format
SW-0001	Once per flight	Must be done before attempting to connect to air-to-ground network	VPN, CRC, non-repudiation, authentication, checksum	Data terminal and screen; user interface; predefined message format
GN-0001	1 Hz	Fixed timeslice in major (1 Hz) frame	VPN, CRC, non-repudiation, authentication, checksum	Sufficient storage space; predefined message format

Figure 10 – AATCS Operational Information Exchange Matrix (OV-3)

4.3.4. Top-Level System View (SV-1)

"The SV-1 depicts systems nodes and the systems resident at these nodes to support organizations/human roles represented by operational nodes of the OV-2. SV-1 also identifies the interfaces between systems and systems nodes….SV-1 links together the OV and SV by depicting the assignments of systems and systems nodes (and their associated interfaces) to the operational nodes (and their associated needlines) described in OV-2. OV-2 depicts the operational nodes representing organizations, organization types, and/or human roles, while SV-1 depicts the systems nodes that house operational nodes (e.g., platforms, units, facilities, and locations) and the corresponding systems resident at these systems nodes that support the operational nodes."[24]

Figure 11 shows the AATCS system information description diagram. As might be expected, its appearance is structurally similar to the OV-2 diagram, however, the SV-1 diagram shows the allocation of major system functions and interconnections to specific subsystem and network elements along the needlines presented in the OV-2. The SV-1 also links back to the OV-3 diagram by representing the additional detail from that diagram (such as the media and interoperability/usability requirements columns) in the system view. The interfaces for ground station beta and airplane N+1 are shown only to highlight their interface to ground station alpha and airplane N, respectively. Other interfaces are redundant and not shown.

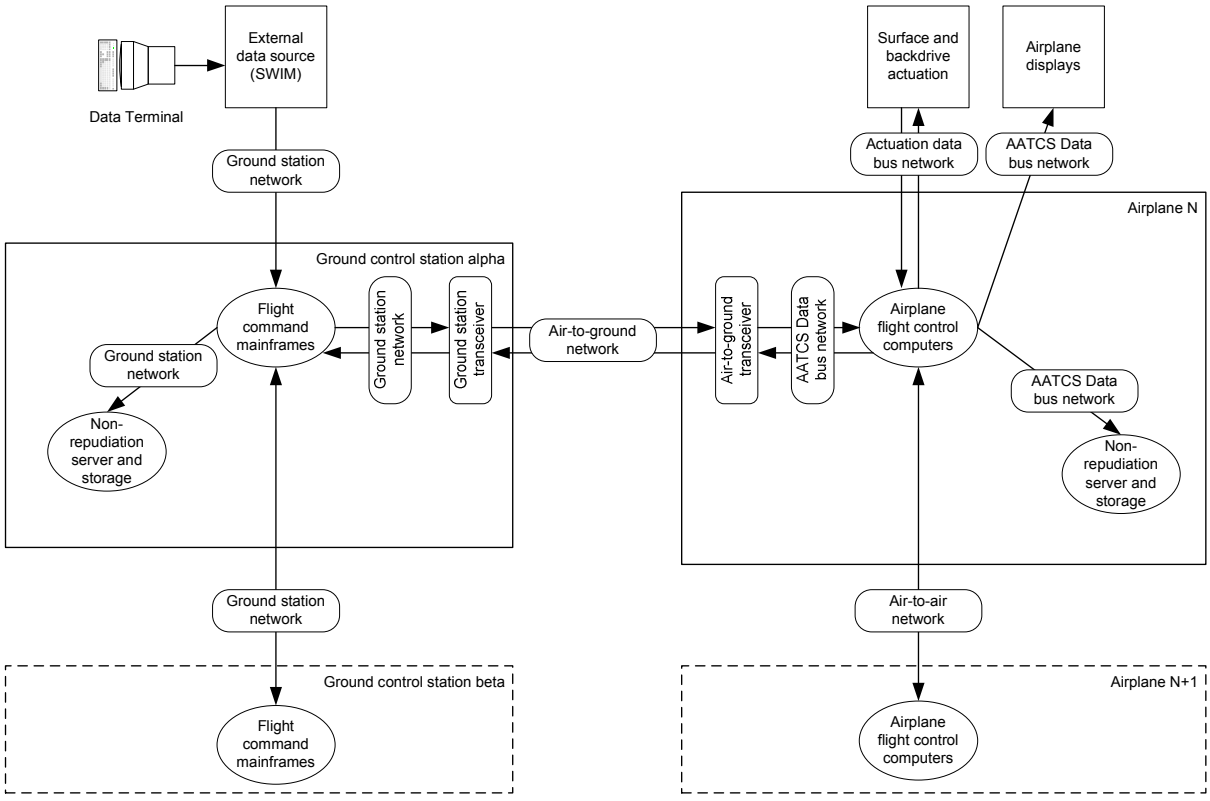

Figure 11 – AATCS System Information Description (SV-1)

4.3.5. System Interface Details (SV-3)

"The SV-3 provides detail on the interface characteristics described in SV-1 for the architecture, arranged in matrix form….SV-3 is a summary description of the system-system interfaces identified in SV-1. SV-3 is similar to an N^2-type matrix, where the systems are listed in the rows and columns of the matrix, and each cell indicates a system pair interface, if one exists."[25]

Figure 12 below shows the AATCS system interface characteristics matrix. As is typical for an N^2 matrix, the row and column headers are the different subsystems that comprise the AATCS, which are depicted in the SV-1 diagram. The SWIM system interfaces to the data terminal by providing the user interface and electronic forms to the pilot to enter his authentication password. The SWIM system then takes the authentication data and, along with the weather and flight plan data, transmits them to the ground station flight command mainframes. The SWIM system can also receive any updated flight plan status, such as actual departure times of airplanes.

The main function of the ground station is to provide connectivity information, flight commands to airplanes, as well as previously received (last available) airplane status information as a 'wrapback'. Ground stations also receive 'wrapback' versions of the flight commands that were transmitted in order to indicate proper receipt of the data. Ground stations also transmit data to other ground stations as a cross-check monitoring of their calculations. Non-repudiation data is timestamped and stored off on separate servers.

In addition to previously mentioned information exchanges between an airplane and a ground station, airplane flight control computers process the received flight commands through the control laws and generate actuation commands to surface actuators (for flight) and to backdrive actuators (for situational awareness). Flight information and airplane status is transmitted to the displays to enhance the pilot's situational awareness. Airplanes transmit their received flight commands to other airplanes as a cross-check similar to ground stations. They also generate non-repudiation data which is timestamped and stored down to a separate electronic unit such as a flight data recorder.

It should be noted that the physical transceivers are not listed in the SV-3 diagram in order to give the diagram a higher-level system functional perspective. Interface implementation details can be shown on an SV-2 diagram, which "depicts pertinent information about communication systems, communication links and communication networks…and documents the kinds of communication media that support the systems and implement their interfaces as described in SV-1."[26]

To⇒ / ⇓ From	Data Terminal	SWIM system	Flight Command Mainframes (ground station alpha)	Non-repudiation server and storage (ground station)	Flight Command Mainframes (ground station beta)	Airplane flight control computers (Airplane N)	Non-repudiation server and storage (airplane)	Airplane flight control computers (Airplane N+1)	Surface and backdrive actuation	Airplane displays
Data Terminal		Pilot authentication and flight plan data	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
SWIM system	User interface and electronic forms		Weather data, flight plan data, pilot authentication	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Flight Command Mainframes (ground station alpha)	N/A	Flight plan updates		Non-repudiation data and timestamp	Cross-check monitor data	Flight commands, received status, connectivity information	N/A	N/A	N/A	N/A
Non-repudiation server and storage (ground station)	N/A	N/A	N/A		N/A	N/A	N/A	N/A	N/A	N/A
Flight Command Mainframes (ground station beta)	N/A	N/A	Cross-check monitor data	N/A		N/A	N/A	N/A	N/A	N/A
Airplane flight control computers (Airplane N)	N/A	N/A	Airplane status, received flight commands	N/A	N/A		Non-repudiation data and timestamp	Flight commands received from ground station	Actuation commands	Flight information, System status/health
Non-repudiation server and storage (airplane)	N/A	N/A	N/A	N/A	N/A	N/A		N/A	N/A	N/A
Airplane flight control computers (Airplane N+1)	N/A	N/A	N/A	N/A	N/A	Flight commands received from ground station	N/A		N/A	N/A
Surface and backdrive actuation	N/A	N/A	N/A	N/A	N/A	Actuation position feedback	N/A	N/A		N/A
Airplane displays	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Figure 12 – AATCS System Interface Characteristics Matrix (SV-3)

4.4. AATCS in the Enterprise Architecture Framework

Although engineering perspectives and views are important in the definition and development of a system, engineers do not represent the entire domain of project stakeholders. A business perspective is oftentimes overlooked when designing a system – an attitude which at best will lead to spirited discussions between the program office and engineers, and at worst will lead to undesirable and unrecoverable cost overruns. Thus, enterprise and related business views are necessary in order to successfully architect a system. The concept of the enterprise architecture (EA) attempts to capture the business perspective in the development of the system.

As with the Department of Defense Architectural Framework, architectural frameworks also exist for the enterprise. One definition of an Enterprise Architecture Framework is "a collection of tools, process models, and guidance used by architects to assist in the production of organization-specific architectural descriptions."[27] This paper presents analyses of two different EA frameworks: the Zachman Framework and the Federal Enterprise Architecture (FEA) framework.

4.4.1. Zachman Framework

"The Zachman Framework is a classification structure often used…for developing and documenting an Enterprise Architecture….It uses a two dimensional classification model based on the six basic interrogatives (What, How, Where, Who, When, and Why) intersecting six distinct perspectives, which relate to stakeholder groups."[28] The use of the Zachman Framework is beneficial, particularly at the beginning of the project, because it integrates the necessary stakeholder views and objectives – including business views and objectives – into a single diagram for the project. The drawback, as can be seen by the example in Figure 13, is that the "spreadsheet is either unwieldy and huge if any significant detail in spreadsheet cells…or else too high-level to be meaningful if spreadsheet cells are of a reasonable size."[29] Still, the Zachman Framework is perhaps most valuable in assembling and organizing the numerous stakeholders' views and objectives at the '40,000 foot' level; the Zachman Framework diagram is generated as part of step 3 in the spiral development model.

The six interrogatives mentioned above are shown in the columns in Figure 13 and the six stakeholder perspectives (Planner, Business Owner, Architect, Designer, Builder, Functioning System) are represented in the rows. In this example, the planner's focus is to keep a social and economic perspective at the forefront; correspondingly, the entries in the table are all concerned with the successful operation of the system as seen by the public. The business owner's view naturally focuses on the concerns of the business developing the system and, as such, the entries in the spreadsheet for that row are all related to what's best for the business. The architect has to keep the customers in mind as well as the system framework which will define the solution – this is perhaps most evidenced by the many entries in the Who column which represent the various external stakeholders. There is some overlap with the planner's view, noted in the Why column and the architect's interest in social and business benefits of the applied technologies.

AATCS Zachman Framework	Structure (What)	Activities (How)	Location (Where)	People (Who)	Time (When)	Motivation (Why)
Objectives/Scope (Planner's View)	National Air Traffic Control System	Flight operations at airports and surrounding airspace	Local airports	Arriving and departing airplanes	In conjunction with NextGen upgrades	Better utilization of existing airspace
Enterprise Model (Business Owner's View)	Business development practices for enterprise applications	Program management organization	Global development team	Project team, supply chain	Project schedule	More revenue, business case for AATCS
Model of Fundamental Concepts (Architect's View)	AATCS system architecture	System architecture models, regulations, processes and standards	System deployment locations	Regulatory agencies, airlines, pilots, air traffic controllers, NextGen developers, customers	Life of the project, including operation, maintenance and disposal	Social and business benefits of applied technologies, customer satisfaction
Technology Model (Designer's View)	Available technologies	Requirements, development processes and standards	Suppliers' and developers' companies, engineers' desks	Design team, engineers at suppliers' companies, tool developers	Technology maturity, obsolescence planning	Technology permits safe management of complexity of increased air traffic density
Detailed Representation (Builder's View)	Hardware and software components in AATCS, simulations	Implementation of design, integration and V&V of HW/SW components	Integration and development labs, engineers' desks	HW, SW and systems build teams, integrators and testers	Quick response turnarounds, timeliness	Putting everything together from the ground up and ensuring the finished product meets the requirements
Functioning System	Deployed AATCS	Operation of installed system (could be incremental)	Airports, airplanes, ground stations	Pilots, ground station operators, passengers	24/7/365	Error-free, safe operation which meets mission requirements

Figure 13 – AATCS Zachman Framework Diagram

The designer and builder views in the spreadsheet show the typical concerns and emphases that engineers have during product development. The last perspective, that of the functioning system itself, shows what a deployed, working system would 'want'. Although the idea of anthropomorphizing a system might seem to be a silly or trivial way to approach system design, it is not necessarily pointless to think about such things, particularly with the observation that improvements in artificial intelligence and biotechnology will eventually warrant such approaches as a matter of course.

4.4.2. Federal Enterprise Architecture System Component Reference Model

The Federal Enterprise Architecture (FEA) is U.S. government-defined and "collectively, the reference models comprise a framework for describing important elements of the FEA in a common and consistent way….The SRM is a business-driven, functional framework classifying Service Components according to how they support business and performance objectives."[30] The Service Components and SRM define a service-oriented architecture of sorts within the FEA and describe typical business services. Although such services are not explicitly used by the AATCS, it is still worth noting that modeling systems from a business perspective can be useful in uncovering behavior or functionality that might be missed by modeling the system from a purely technical perspective.

Within the SRM are seven service domains – this paper uses the Process Automation Services Domain to exemplify some of the interactions in the AATCS. "The Process Automation Services domain represents those services and capabilities that serve to automate and facilitate the processes associated with tracking, monitoring, and maintaining liaison throughout the business cycle of an organization."[31] The service type that will be used is "Routing and Scheduling" – from an AATCS perspective of routing and scheduling airplane traffic in the AATCS.

Figure 14 shows the routing and scheduling service type example for the AATCS. "Capabilities within this Service Type provide automatic directing, assignment, or allocation of time for a particular action or event."[32] The two service components within this service type include inbound and outbound correspondence management, with inbound and outbound being relative to the ground station. These service components describe at a high level the methods by which the ground station collects and organizes data from various sources needed to create a method by which airport spatial and temporal slots are automatically allocated into a 'queuing schedule' and used as a primary factor for how the flight commands are calculated, such that the slots are maintained by the entire air traffic system. As with the other architectural models and frameworks mentioned in this paper, the up-front benefit of this technique is utilized in step 3 of the spiral development model.

Finally, it should be noted that the previous assertion of benefiting by modeling a non-business system using business service models was exemplified by the author in discovering the need for and creating the "queuing schedule" concept for the AATCS.

Process Automation Services Domain Routing and Scheduling of Airplane Traffic in the AATCS		
Service Component	Defines the set of capabilities that…	Services provided by Ground Station
Inbound Correspondence Management (Data From Airplanes)	Manage externally initiated communication between an organization and its stakeholders	The ground station receives the inbound message packets from all of the airplanes in the local traffic pattern and uses the flight vector and status data to automatically allocate airport spatial and temporal slots by creating arrival and departing queuing schedules as well as any alerts for ground station personnel.
Outbound Correspondence Management (Messages to Airplanes)	Manage internally initiated communication between an organization and its stakeholders	The ground station uses the queuing schedules and last received flight vector data as the basis for calculating flight commands for each airplane and organizes the data, along with status, for each airplane and transmits the outbound message packets to the airplanes.

Figure 14 – AATCS FEA SRM Example

4.5. Ontologies and Semantic Models

"A semantic data model in software engineering is a data modeling technique to define the meaning of data within the context of its interrelationships with other data."[33] Semantic models can be similarly used to define a common data-centric 'language' for the interactions between elements in a system. Ontologies are related to semantic models in that they define "what is meant by a particular terminology: semantics and meaning."[34] This section describes the development of ontologies (semantic models) for three elements or objects in the AATCS.

4.5.1. AATCS Ontology Examples

The three objects for which ontologies will be developed are a flight control computer, a transceiver and a hydraulic actuator. A brief description of the object will be presented, along with the contextual domains and ranges for the object. Finally, a description of the behavior of the object rendered as 'if…then' statements is used to establish the rules by which the object can be used. Although the domains, ranges, and behavior characterizations are not exhaustive, some of the key values of each are included in the models.

Key object 1: Flight Control Computer (FCC)

Description: Flight control computers are line-replaceable units (LRUs) that assist with the flight and operation of the airplane and can provide work alleviation to reduce fatigue and allow the flight crew to concentrate on other tasks while still maintaining situational awareness.

Domain: Electronic line replaceable units (LRUs) on AATCS-compatible airplanes

Range: Parts are: chassis, connectors, processor, databus interface, power input, external cables, control laws, software

Behavior: FCC semantic model behavior characterization is shown in Table 5

Table 5 – AATCS Semantic Model Behavior Characterization, Flight Control Computer

If…	Then…
An FCC and a new valid data packet is received from the air-to-ground network transceiver	Process packet data and extract constituent flight commands (heading and altitude)
An FCC and constituent flight commands have been extracted	Perform control law processing on constituent flight commands and generate actuation commands
An FCC and control law processing has been performed	Transmit actuation commands to surface and backdrive actuators on actuation data bus network

Table 5 – AATCS Semantic Model Behavior Characterization, Flight Control Computer	
If…	Then…
An FCC and one or more other FCCs are inoperative	Annunciate loss of redundancy to flight displays
An FCC and AATCS disengagement signal received	Set AATCS disengagement bit in status word and accept inputs to control laws only from inertial sensors and pilot inceptors (column, wheel, pedals)
An FCC and power is applied	Perform start-up built-in test (SBIT) prior to execution of normal processing sequence
An FCC and fault is detected in SBIT	Disable all FCC outputs and force processor into idle state

Key object 2: Transceiver

Description: The transceiver is a line replaceable unit (LRU) on an airplane which sends and receives data from a system which is external to the airplane.

Domain: Electronic line replaceable units (LRUs) on AATCS-compatible airplanes

Range 1: Types are: Air-to-ground network transceiver, Air-to-air network transceiver

Range 2: Parts are: transmitter, receiver, power input, connectors, antenna, AATCS network databus interface, chassis, external cables

Range 3: Transmitting characteristics are: power, frequency, bandwidth, modulation, direction

Behavior: Air-to-Ground network transceiver semantic model behavior characterization is shown in Table 6

Table 6 – AATCS Semantic Model Behavior Characterization, Air-to-Ground Network Transceiver	
If…	Then…
An Air-to-Ground network transceiver receives input packet from Ground station	Transduce incoming packet and store data in available input queue
An Air-to-Ground network transceiver has data in incoming input queue	Calculate CRC on data and compare to CRC received with packet

Table 6 – AATCS Semantic Model Behavior Characterization, Air-to-Ground Network Transceiver	
If…	Then…
An Air-to-Ground network transceiver detects difference between received CRC and calculated CRC	Set CRC error flag in status word, store error status to flight data recorder for non-repudiation
An Air-to-Ground network transceiver detects no difference between received CRC and calculated CRC	Clear CRC error flag in status word, store error status to flight data recorder for non-repudiation
An Air-to-Ground network transceiver receives data from FCC	Store incoming data in available output queue and generate CRC for outgoing packet
An Air-to-Ground network transceiver available and data is in input queue	Transmit packet data to FCCs,
An Air-to-Ground network transceiver available and data is in output queue	Transduce outgoing packet and transmit to ground station

Key object 3: Actuators

Description: Actuators are force-multiplying mechanical devices which move objects attached to the movable piston or ram using pneumatic, hydraulic or electrical power.

Domain: Airplane hydraulic actuators

Range 1: Types are: wing surface actuators, tail surface actuators, speedbrake actuators, backdrive actuators, landing gear actuators

Range 2: Movement types are: linear, rotary

Range 3: Control types are: rate, position

Range 4: Sensor types are: position, pressure

Range 5: Physical characteristics are: rate, position, force, stiffness, pressure

Behavior: Hydraulic actuator semantic model behavior characterization is shown in Table 7

Table 7 – AATCS Semantic Model Behavior Characterization, Hydraulic Actuator	
If…	Then…
Rate-commanded hydraulic actuator receives non-zero rate command from FCC	Open fluid port to move actuator at commanded rate
Rate-commanded hydraulic actuator receives zero rate command from FCC	Close fluid port to stop actuator motion
Position-commanded hydraulic actuator receives different position command from FCC	Open fluid port to move actuator to commanded position
Position-commanded hydraulic actuator receives position command from FCC which matches current position	Close fluid port to stop actuator motion
Hydraulic actuator receives power to shut-off solenoid	Move shut-off solenoid to position to allow fluid flow (note: the shut-off solenoid stops fluid flow when de-powered)
Hydraulic actuator receives VDT excitation to position sensor	Transmit position
Hydraulic actuator receives VDT excitation to pressure sensor	Transmit fluid pressure

5. Summary/Conclusions

This paper has presented an overview of the automated air traffic control system (AATCS) and has performed analyses using different systems architectural modeling methods. A description of the system was provided along with the many architectural views which are indicative of the complex nature of the system. Although the entire system architecture can never be modeled exactly, it can certainly be approximated to be 'good enough' to design to. Several of the analyses presented in this paper are exemplary of the ability of models to reduce complexity and increase comprehension of the system architecture in an unambiguous manner. Models also serve to reduce miscommunication and can potentially reduce overruns in development cost, especially if the modeling activity is done sufficiently early in the program, as demonstrated by the spiral development model.

The architectural analyses highlighted primary areas of interest in defining the AATCS. The UML analyses presented examples of the system architecture from an object-oriented perspective: the use case, the class hierarchy diagram, and the sequence diagram. DoDAF diagrams similarly depicted the operation and composition of the system by using operational view (OV) models and system view (SV) models. Enterprise architecture models which incorporated the enterprise and business perspectives provided new insight into the operation of the system. Finally, semantic modeling provided a means of unambiguously describing the very nature of objects (the characteristics of their 'being') within the system.

The system architecture models presented in this paper represent only a small number of the modeling diagram and language types that are available. The spiral development model shows that incremental prototyping, evaluation and review of functionality during a program can be a viable method of program development if proper planning and modeling are done as early as possible instead of later, as an afterthought. New 'breakthrough' systems will by definition be even more complex than the systems which preceded them. However, systems architects who are armed with tools to model the many views of the system's architecture can successfully develop and deploy complex, net-centric systems like the AATCS.

6. Acknowledgements

The concept that this paper is based on originated from a discussion with Brad Betters of Tandel Systems, Bruce Vacey of Honeywell International and the author. The participants of the discussion have given the author verbal permission to further expand upon the original idea of an automated air traffic control system. The analyses and all unattributed material in this paper are solely the creation of the author.

7. References

[1] *"NASA & The Next Generation Air Transportation System (NextGen)"* (n.d.), retrieved October 5, 2008, from www.aeronautics.nasa.gov/docs/nextgen_whitepaper_06_26_07.pdf.

[2] "Free flight (air traffic control)" (July 18, 2008), retrieved October 21, 2008 from Wikipedia (Free Flight): http://en.wikipedia.org/wiki/Free_flight_(air_traffic_control)

[3] Cureton, K. (Aug. 26, 2008). *SAE 574 Lecture #1: Net-Centric Systems Architecting and Engineering*. University of Southern California.

[4] Cureton, K. (Aug. 26, 2008). *SAE 574 Lecture #1: Net-Centric Systems Architecting and Engineering*. University of Southern California.

[5] "Fact Sheet – System-Wide Information Management (SWIM)", (May 2, 2006), retrieved October 23, 2008 from Google (System-Wide Information Management): http://www.faa.gov/news/fact_sheets/news_story.cfm?newsId=7129

[6] "OEP Plan Reference Sheet NNEW" (June 19, 2007), retrieved October 23, 2008 from Google (NNEW): http://www.faa.gov/about/office_org/headquarters_offices/ato/publications/oep/version1/reference/nnew/

[7] Spitzer, C., (Ed.). (2001). *The Avionics Handbook*. CRC Press LLC.

[8] Wasson, C. (2006). *System Analysis, Design, and Development : Concepts, Principles and Practices*. New Jersey : John Wiley & Sons, Inc

[9] Rechtin, E. (1991). *System Architecting: Creating and Building Complex Systems*. New Jersey : Prentice-Hall, Inc.

[10] Hines, J. (June 2, 2008). *Systems Engineering Theory and Practice, SAE 541, Summer 2008, Session 1*. University of Southern California.

[11] "Spiral model" (Dec. 2, 2008), retrieved Dec. 6, 2008 from Google (spiral development model): http://en.wikipedia.org/wiki/Spiral_model

[12] "Spiral model" (Dec. 2, 2008), retrieved Dec. 6, 2008 from Google (spiral development model): http://en.wikipedia.org/wiki/Spiral_model

[13] Cureton, K. (Oct. 14, 2008). *SAE 574 Lecture #6: Architecture Modeling Concepts*. University of Southern California.

[14] Cureton, K. (Oct. 14, 2008). *SAE 574 Lecture #6: Architecture Modeling Concepts*. University of Southern California.

[15] Cureton, K. (Oct. 14, 2008). *SAE 574 Lecture #6: Architecture Modeling Concepts*. University of Southern California.

[16] Cureton, K. (Oct. 28, 2008). *SAE 574 Lecture #7: Architecture Modeling Concepts*. University of Southern California.

[17] Schmuller, J. (2004). *Sams Teach Yourself UML in 24 Hours, Third Edition*, Sams Publishing.

[18] "DoD Architecture Framework Version 1.5" (April 23, 2007), retrieved Dec. 10, 2008 from U.S. Department of Defense (DoDAF): http://www.defenselink.mil/dbt/products/2008_BEA_ETP/bea/products/DoDAF_Volume_II.pdf

[19] Cureton, K. (Oct. 28, 2008). *SAE 574 Lecture #7: Architecture Modeling Concepts*. University of Southern California.

[20] "DoD Architecture Framework Version 1.5" (April 23, 2007), retrieved Dec. 10, 2008 from U.S. Department of Defense (DoDAF): http://www.defenselink.mil/dbt/products/2008_BEA_ETP/bea/products/DoDAF_Volume_II.pdf

[21] "DoD Architecture Framework Version 1.5" (April 23, 2007), retrieved Dec. 10, 2008 from U.S. Department of Defense (DoDAF): http://www.defenselink.mil/dbt/products/2008_BEA_ETP/bea/products/DoDAF_Volume_II.pdf

[22] "DoD Architecture Framework Version 1.5" (April 23, 2007), retrieved Dec. 10, 2008 from U.S. Department of Defense (DoDAF): http://www.defenselink.mil/dbt/products/2008_BEA_ETP/bea/products/DoDAF_Volume_II.pdf

[23] "DoD Architecture Framework Version 1.5" (April 23, 2007), retrieved Dec. 10, 2008 from U.S. Department of Defense (DoDAF): http://www.defenselink.mil/dbt/products/2008_BEA_ETP/bea/products/DoDAF_Volume_II.pdf

[24] "DoD Architecture Framework Version 1.5" (April 23, 2007), retrieved Dec. 10, 2008 from U.S. Department of Defense (DoDAF): http://www.defenselink.mil/dbt/products/2008_BEA_ETP/bea/products/DoDAF_Volume_II.pdf

[25] "DoD Architecture Framework Version 1.5" (April 23, 2007), retrieved Dec. 10, 2008 from U.S. Department of Defense (DoDAF): http://www.defenselink.mil/dbt/products/2008_BEA_ETP/bea/products/DoDAF_Volume_II.pdf

[26] "DoD Architecture Framework Version 1.5" (April 23, 2007), retrieved Dec. 10, 2008 from U.S. Department of Defense (DoDAF): http://www.defenselink.mil/dbt/products/2008_BEA_ETP/bea/products/DoDAF_Volume_II.pdf

[27] "Enterprise architecture" (Dec. 11, 2008), retrieved Dec. 15, 2008 from Google (enterprise architecture): http://en.wikipedia.org/wiki/Enterprise_architecture

[28] "Zachman framework" (Nov. 7, 2008), retrieved Nov. 27, 2008 from Google (Zachman framework): http://en.wikipedia.org/wiki/Zachman_framework

[29] Cureton, K. (Nov. 4, 2008). *SAE 574 Lecture #8: Government Enterprise Architecture Concepts*. University of Southern California.

[30] "FY07 Budget Formulation: FEA Consolidated Reference Model Document" (May 2005), retrieved August 24, 2008 from USC Blackboard Learning System, SAE 574 class website: http://www.uscden.net/courses/1/20083_SAE574/content/_88117_1/CRM.pdf

[31] "FY07 Budget Formulation: FEA Consolidated Reference Model Document" (May 2005), retrieved August 24, 2008 from USC Blackboard Learning System, SAE 574 class website: http://www.uscden.net/courses/1/20083_SAE574/content/_88117_1/CRM.pdf

[32] "FY07 Budget Formulation: FEA Consolidated Reference Model Document" (May 2005), retrieved August 24, 2008 from USC Blackboard Learning System, SAE 574 class website: http://www.uscden.net/courses/1/20083_SAE574/content/_88117_1/CRM.pdf

[33] "Semantic data model" (Nov. 27, 2008), retrieved Dec. 15, 2008 from Wikipedia(semantic model): http://en.wikipedia.org/wiki/Semantic_data_model

[34] Cureton, K. (Nov. 11, 2008). *SAE 574 Lecture #9: Net-Centric Systems Engineering and Processes*. University of Southern California.