Developing DoD Architecture Framework Integrated Architecture Using an Object-Oriented Method

Headquarters Air Force Space Command
Directorate of Requirements

19 August 2005

Point of Contact:

Joanne Schissel, GS-14
Chief, C3 & Navigation Integration and Architecture Branch
CCIC2S Chief Architect
AFSPC/DRN
Phone: 719-554-5057

Table of Contents

List of Figures

List of Tables

Executive Summary

This paper is intended for DoD professionals interested in exploiting new systems engineering capabilities, enabled by the DoD Architecture Framework (DoDAF), to better manage the modernization of information technology-intensive enterprises. Both senior acquisition managers and practicing DoDAF architects will find value in reading some or all of this paper. These new engineering capabilities involve the use of enterprise architectures and promise to improve systems acquisition practices by providing a better means to: 1) define the enterprise vision, objectives, and boundaries, 2) identify the enterprise stakeholders, 3) reach community consensus on required end-to-end enterprise capabilities (operational concepts and requirements) needed to meet the needs of the stakeholders, 4) identify capability gaps, 5) define material and non-material solutions needed to close the gaps (DOTMLPF), 6) map required operational capabilities directly to specific enterprise components (organizations, systems, etc) needed to deliver the end-to-end capabilities demanded of the enterprise, 7) identify interdependencies and interoperability needs among enterprises to ensure mission success, and 8) develop a defensible baseline to successfully advocate for the resources needed to achieve the vision.

The complexity of today's information systems, and the division of system development responsibilities among agencies, drives the need to incorporate architecture into systems engineering practice. Dependencies among disparate military systems operating in a net-centric environment bring the art and science of system design and integration to a new level of complexity. Employing enterprise architecture, especially early in the life cycle of a system, or system of systems, offers promise as a means of managing this complexity and fielding effective and affordable war fighting IT systems. This paper describes an approach for developing enterprise architectures using an object-oriented (OO) method and the Unified Modeling Language (UML) – the software industry's standard language for visualizing and documenting system software. The primary objective of this approach is to ensure warfighter-required capabilities and priorities drive system development and that it is easy to track and assess the contribution of each critical enterprise component to mission success.

Just a few of the key features of this approach include the ability to: 1) design systems that conform with operator-defined outcomes; 2) use a common language to express operational concepts, requirements, and system design; 3) demonstrate that warfighter capabilities directly link to modular system components; 4) identify reusable operations and capabilities leading to more efficient systems reducing ownership cost and development risk; 5) scale the level of detail to the size and complexity of different programs; and 6) respond quickly and effectively to fact-of-life changes (e.g., threats; missions; tactics, techniques and procedures; organizational realignments; emerging technology; budget constraints; etc). One of the remarkable aspects of this architecture approach is that it introduces a common, standard information technology vocabulary, which warfighters, architects, engineers, trainers, testers, and acquirers can use to communicate better with one another.

Development of this method was led by HQ AFSPC/DRN with the support of *SI International, Inc.* and The MITRE Corporation. Furthermore, the AFSPC team consulted with the IBM Rational Company, whose founding members led the formulation of the UML, to exploit the knowledge and experience of the world's leading provider of UML-compliant software development tools.

After providing some background on the development of this method, and a short description of DoDAF concepts, the paper details the process for building an enterprise architecture containing two of the three DoDAF prescribed views for complex systems: Operational and Systems. The third principal view, the Technical Standards View, is not addressed in the paper at this time. The architecture development process starts with some prerequisite steps needed only at the start of the process. It then describes a set of steps that repeat as many times as is necessary to derive system elements and their structure suitable for the purpose of the architecture. The data generated by this process is suitable for generating Systems View products. To illustrate how this method works, the paper uses an example drawn from a portion of HQ AFSPC's largest and most mature enterprise architecture.

The approach requires the collaboration of the operational and systems architects to be effective. With this collaboration, the approach balances both operational and system concerns in a manner that leads to effective system implementation. Without this collaboration, the warfighter's perspective could well impose requirements on the system that are unrealistic, or not understood, or both.

Readers interested in implementing this methodology or contributing to its further development should contact the CCIC2S Chief Architect, Ms. Joanne Schissel, GS-14, HQ AFSPC/DRN at 719-554-5057 for further information.

1.0 Introduction

The purpose of this document is to:

- Demonstrate the value in using architecture to manage IT acquisitions
- Describe to DoD architects how this OO approach and other methods produce integrated architecture and explain the unique advantages this approach offers in acquiring IT systems
- Illustrate how the OO process works illustrated using a real world architecture
- Recommended action to incorporate this method into management processes

This document is written for senior acquisition decision makers (sections 0 and 1.0), DoD architects who practice structured analysis (same as senior execs plus sections 2.0 and 4.0) to understand the approach, and those architects who wish to implement an OO-based architecture method (all sections, assumes a basic understanding of UML). The focus of this paper is on the integration of the Operational and Systems Views. The Technical Standards View is part of the integrated architecture but is not addressed in the paper at this time.

The genesis of the paper traces to a report submitted by IBM Rational Software consultants to AFSPC/DRN documenting the results of several workshops they facilitated in late 2004. At the workshops, IBM, SI International, Inc., and The MITRE Corporation collaborated on OO/UML techniques for bridging together DoDAF Operational and Systems Views. The IBM Rational consultants who authored the report were James Densmore and David Brown. Building on the concepts captured in this report, Rob Byrd, John Manzi and Ed Seward of SI International, Inc., and Tom Folk and Eric Todd of The MITRE Corporation, expanded the discussion to provide a more comprehensive explanation of the AFSPC architecture development approach.

1.1 Architecture Development Evolution

As the DoD moves toward Net-Centric Operations and Warfare (NCOW), a new approach to system engineering centered on the use of enterprise architecture is emerging to: 1) define new operational concepts and capabilities enabled by information technology, 2) better understand and communicate the complexities and interdependencies within and among enterprises to achieve truly effective operations in a Net-Centric environment, and 3) effectively manage the definition, development, fielding, sustainment, and evolution of materiel and non-materiel solutions needed to achieve Net-Centric capabilities.

A critical step in the development of this architecture-centric approach to systems engineering was taken by the DoD Architecture Framework Working Group (AFWG) when it defined standardized architecture representation across the DoD. In doing so, the AFWG succeeded in providing stakeholders a better means of achieving interoperability and integration by identifying and addressing inter- and intra-enterprise dependencies. Today, the Department of Defense (DoD) Architecture Framework (DoDAF), Version 1.0, mandates the use of a common approach for representing DoD architectures. The DoDAF approach enables architecture descriptions to be shared across Joint and Combined organizations. Recognizing the potential of taking an architecture-centric approach to capabilities acquisition, the Joint Chiefs of Staff mandated the use of DoDAF "views" to justify the need for and guide the acquisition of all major military systems per CJCSI 3170 (Joint Capabilities Integration and Development System) and CJCSI 6212 (Interoperability and Supportability of National Security Systems, and Information Technology Systems).

An architecture-centric approach to systems engineering can improve acquisitions by enhancing discipline, improving communications/collaboration among stakeholders, clarifying definition of the enterprise, documenting the solution, and synchronizing concepts, requirements, products, and systems. This approach fosters greater understanding of organizational, system, and enterprise relationships, identifies reusable components, aids integration of acquisition and warfighter Decision Support System (DSS) processes, and provides flexibility needed to assess impacts and manage change. In short, architecture provides a framework for effectively defining and managing the evolution of an information technology intensive enterprise.

Some significant challenges in pursuing architecture-centric acquisition include:

- Effectively defining, modeling, and communicating enterprise complexity
- Ensuring required operational capabilities drive system development – linking capabilities directly to enterprise components
- Identifying and managing enterprise and program boundaries
- Recognizing reusable operations and components to reduced ownership cost
- Coping with change (threats, missions, operations, organizations responsibilities, technology, etc)

This paper presents a promising solution to meet these challenges. Drawing upon principles of modern software engineering, we extended industry best practices in object-oriented analysis and design to address enterprise architecture development. Key features of the object-oriented approach presented in this paper include:

- Scalable to enterprise size and complexity and broadly applicable to many different programs
- Common modeling language understood by both warfighters/operators and system developers to facilitate clear and unambiguous collaboration/communication and "stakeholder buy-in"
- Effects-based approach ensures fielded systems deliver stakeholder required outcomes (services and products)
- Synchronized evolution of operational capabilities and systems solutions
- Adaptation mechanisms readily identify reusable operations and supporting capabilities
- Flexible structure rapidly accommodates large and/or frequent change

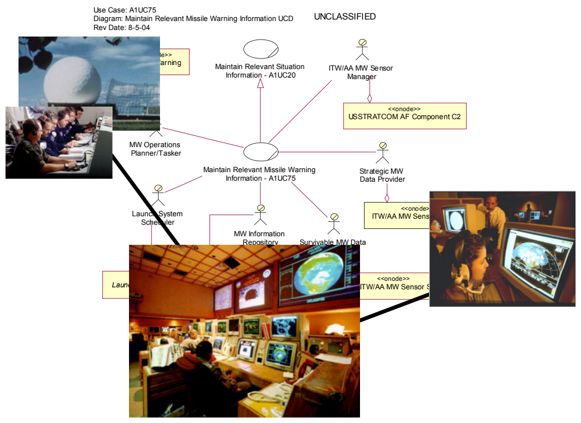

Use Case: A1UC75
Diagram: Maintain Relevant Missile Warning Information UCD
Rev Date: 8-5-04
UNCLASSIFIED
Maintain Relevant Situation Information - A1UC20
ITW/AA MW Sensor Manager
<<onode>>
USSTRATCOM AF Component C2
MW Operations Planner/Tasker
Maintain Relevant Missile Warning Information - A1UC75
Strategic MW Data Provider
<<onode>>
ITW/AA MW Sensor
MW Information Repository
Survivable MW Data

Figure 1 - *Use Cases* describe participants, roles and responsibilities, and desired outcomes (i.e., results of value) for a given process (e.g., timely missile warning to a forward user)

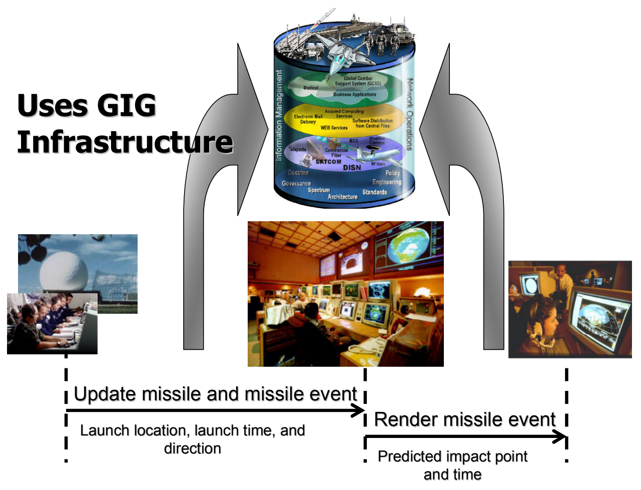

Uses GIG Infrastructure
Update missile and missile event
Launch location, launch time, and direction
Render missile event
Predicted impact point and time

Figure 2 - *Operational Sequence Diagrams* model operational activities and supporting information exchanges

This object-oriented approach delivers all DoDAF-required views. For the purpose of this paper, key architecture products are *use cases*, *operational sequence diagrams*, and *system sequence diagrams*, which used in combination, achieve an integrated architecture as described in the DoDAF. This paper focuses on using these products to provide clear traceability between required operational capabilities and supporting systems.

Use Cases (Figure 1) capture and describe operational capabilities by defining key processes (e.g., Maintain Relevant Missile Warning Information), a desired outcome (e.g., timely missile warning to a forward user), the participants (organizations and systems), and the participants individual roles and responsibilities to produce the desired outcome.

Operational Sequence Diagrams (Figure 2) model the interactions and information flows between warfighters/operators and supporting systems to capture the process detail of the desired capability and desired outcomes.

Once *Use Cases* and *Operational Sequence Diagrams* clearly capture stakeholder needs, *System Sequence Diagrams* (Figure 3) allocate the responsibilities (required functionality) to system elements to achieve the end-to-end capability to produce the desired outcomes.

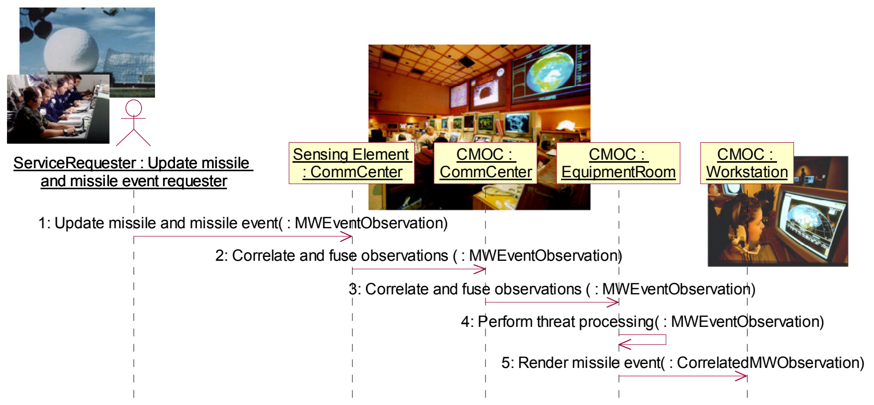

ServiceRequester : Update missile and missile event requester
Sensing Element : CommCenter
CMOC : CommCenter
CMOC : EquipmentRoom
CMOC : Workstation
1: Update missile and missile event(: MWEventObservation)
2: Correlate and fuse observations (: MWEventObservation)
3: Correlate and fuse observations (: MWEventObservation)
4: Perform threat processing(: MWEventObservation)
5: Render missile event(: CorrelatedMWObservation)

Figure 3 - *System Sequence Diagrams* model the interactions among system elements needed to achieve the required operational capability

These integrated views provide the developer a clearly defined design framework to ensure full traceability between the desired operational capability, the desired outcomes (results of value) and the organizations/systems/elements required to achieve that operational capability.

As the software development industry is acutely aware, a significant power of an object-oriented approach is the ability to more easily identify reusable elements within and among enterprises to include: 1) Processes such as "maintaining relevant situational awareness" (Figure 4), 2) System Components such as enterprise workstations, 3) Interfaces such as those between sensors and C2 systems, 4) Nodes such as Cheyenne Mountain Operations Center and its alternate center, and 5) Architectural Components such as information exchanges. The power of reuse can, if properly exploited, reduce development costs, schedule, and risks, while facilitating integration and interoperability.

This object-oriented approach to architecture-centric acquisition has the potential to produce more highly-integrated, agile, and affordable systems, fielded in a shorter amount of time compared to traditional approaches.

1.2 Background

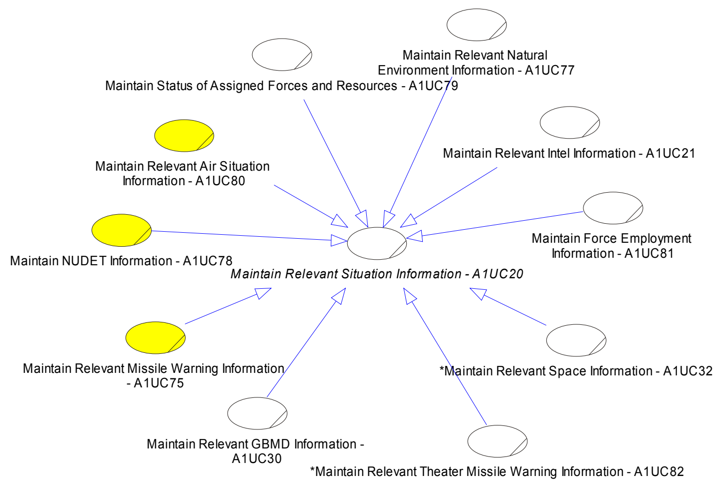

Maintain Relevant Natural Environment Information - A1UC77
Maintain Status of Assigned Forces and Resources - A1UC79
Maintain Relevant Air Situation Information - A1UC80
Maintain Relevant Intel Information - A1UC21
Maintain Force Employment Information - A1UC81
Maintain NUDET Information - A1UC78
Maintain Relevant Situation Information - A1UC20
Maintain Relevant Missile Warning Information - A1UC75
*Maintain Relevant Space Information - A1UC32
Maintain Relevant GBMD Information - A1UC30
*Maintain Relevant Theater Missile Warning Information - A1UC82

Figure 4 - An object-oriented approach enables the architect to easily identify common elements that can be developed once and reused many times across the enterprise

Launched in the year 1996 the CCIC2S program sought to field flexible, affordable and interoperable command and control (C2) and war fighting support systems. Breaking the decades-long trend of developing stove-piped C2 solutions called for innovation in managing the development of this integrated capability. One such innovation was the use of a DoDAF-compliant enterprise architecture to bound the enterprise, identify key stakeholders and communicate the warfighter's requirements for new C2 capabilities and to drive the design of the system solution. After carefully considering the challenge of accurately describing such a large and complex C2 enterprise, the CCIC2S Architecture team developed an architecture approach based on the same analysis and design methods, and expressed using the same modeling language, as the vast majority of today's software developers. CCIC2S capabilities are modeled within an object-oriented (OO)/Unified Modeling Language (UML) enterprise architecture.

Encouraged by the robustness of the UML to express operational concepts, and the ease with which the warfighters could participate in the modeling sessions, the CCIC2S Enterprise Architecture team set out to extend this technology into the realm of system architecture. Defining the mechanism for producing a fully integrated architecture seemed to offer hope that complex requirements could be traced into the system's design, thereby reducing the risk that the delivered system would fail to satisfy the warfighter's C2 and mission support needs. To that end, HQ AFSPC/DRN funded a collaborative study with members of the IBM Rational Corporation (originators of much of the OO/UML theory in practice today) aimed at defining an end-to-end UML method for translating operational capabilities into a system design. Based on IBM Rational's "Rational Unified Process for Systems Engineering (RUP-SE)[see Appendix A: RUP SE Process Outline], the study group successfully framed an approach having the potential to take a specification of needed operational services and derive detail sufficient to begin creating a design model.

Within the DoD, program responsibilities are split between the Major Command (e.g. AFSPC) and the System Program Office (e.g. ESC) for development of the operational and system requirements, respectively. This separation of concerns often leads to the "throw the requirements over the fence" syndrome. The approach requires the collaboration of the operational and systems architects to be effective. With this collaboration, the approach balances both operational and system concerns in a manner that leads to effective system implementation. Without this collaboration, the warfighter's perspective could well impose requirements on the system that are unrealistic, or not understood, or both.

Implementation of this collaborative approach by the operational and systems architects should not have a negative impact on combined budget and schedule. While the activities of the approach are different from what the teams are currently performing, these activities would replace some of the current activities rather than add an additional burden. Moreover, the approach provides a higher focus on architecture interfaces making it easier to identify modular test points and operator responsibilities in the context of the system usage. This can be useful for earlier training development. In the long term, the collaborative nature of this approach results in less rework and higher quality software due to a clearer understanding of the specified requirements and desired capabilities.

The remainder of the paper delves into the more technical aspects of this integrated architecture approach.

2.0 Approach to Integrated Architecture

2.1 Enterprise Architecture

The OMB memo for the Management of Federal Information Resources, also known as Circular No. A-130, defines *enterprise architecture* as the explicit description and documentation of the current and desired relationships among business and management processes and information technology. The Enterprise Architecture includes principles, an Enterprise Architecture framework, a standards profile, current and target architectures, and a transition strategy to move from the current to target architecture.

2.2 DoD Architecture Framework[1]

In the mid 1990s with increasing focus on joint and multinational operations, the U.S. Department of Defense (DoD) realized the need for a common approach for describing architectures. Until that time, the individual Commands, Services, and Agencies in DoD traditionally described their architectures using unique techniques, vocabularies, and presentation schemes. The Assistant Secretary of Defense for Command, Control, Communications, and Intelligence (ASD[C3I]) established a C4ISR Integration Task Force. The C4ISR Architecture Framework, Version 1.0, was developed in 1996 as a product of the Integrated Architectures Panel, one of several panels established by the ITF. Subsequent to this, the IAP published Version 2.0 of the C4ISR Architecture Framework in December 1997.

The utility of the C4ISR Architecture Framework, combined with Federal and DoD policy encouraging the use of architectures, led the DoD to evolve the document into the DoD Architecture Framework (DoDAF) in 2003. Version 1.0 of the DoDAF consists of two volumes (I & II) and a Deskbook companion. Volume I provides definitions, guidelines, and related background material.

Volume II contains a detailed discussion of each of the Framework products and provides examples of each product using Structured Analysis and Unified Modeling Language representations. The Deskbook provides example approaches for developing architectures and using architecture products and data elements.

In the DoD Architecture Framework, there are three perspectives (i.e., views) that logically combine to describe an architecture description. These are the Operational View (OV), Systems View (SV), and Technical Standards View (TV). Per the DoDAF, an *integrated architecture* is comprised of the three DoDAF views of a single architecture such that references to a specific element in different views correlate to that same element. Individual architecture products are not stand-alone entities but represent depictions of subsets of architecture data describing various aspects of an architecture. As such, relationships exist among the architecture data elements that compose the various products, creating relationships among the products. Each of the three views depicts certain architecture attributes. Some attributes bridge two views and provide integrity, coherence, and consistency to architecture descriptions.

[1] DoD Architecture Framework Working Group, *DoD Architecture Framework, Volume I: Definitions and Guidelines, Version 1.0.*

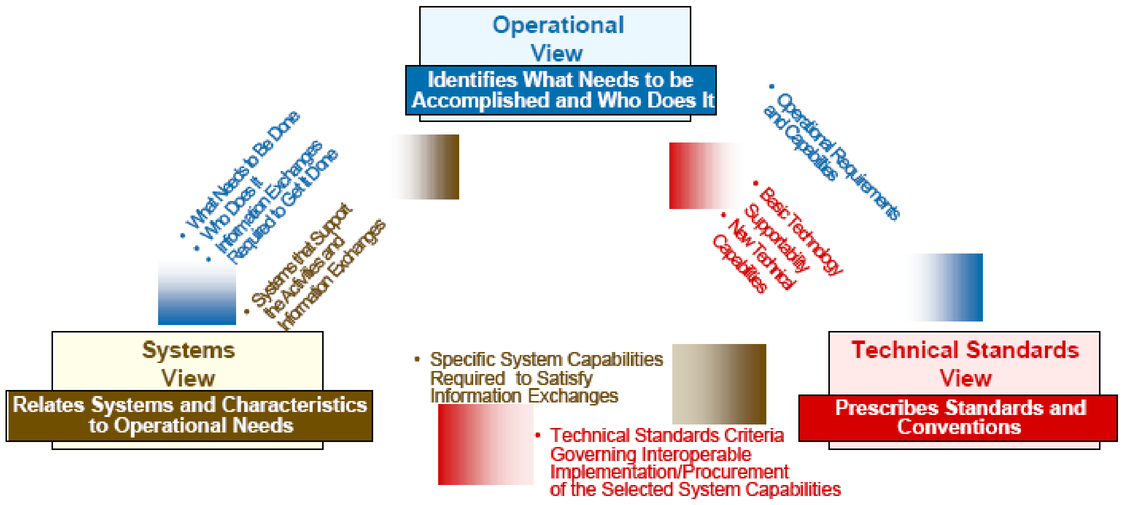

Operational View
Identifies What Needs to be Accomplished and Who Does It
• What Needs to Be Done
• Who Does It
• Information Exchanges Required to Get It Done
• Systems that Support the Activities and Information Exchanges
Systems View
Relates Systems and Characteristics to Operational Needs
• Operational Requirements and Capabilities
• Basic Technology Supportability
• New Technical Capabilities
Technical Standards View
Prescribes Standards and Conventions
• Specific System Capabilities Required to Satisfy Information Exchanges
• Technical Standards Criteria Governing Interoperable Implementation/Procurement of the Selected System Capabilities

Figure 5 - Linkages Among DoDAF Views

2.2.1 Operational View

The Operational View (OV) is a description of the tasks and activities, operational elements, and information exchanges required to accomplish DoD missions.

The OV contains graphical and textual products that comprise an identification of the operational nodes and elements, assigned tasks and activities, and information flows required between nodes. (A *node* by definition is a representation of an element of architecture that produces, consumes, or processes data. An *operational node* may represent an operational/human role (e.g., Air Operations Commander), an organization (e.g., AFSPC) or organization type, that is, a logical or functional grouping (e.g., Logistics Node, Intelligence Node).) The OV defines the types of information exchanged, the frequency of exchange, which tasks and activities are supported by the information exchanges, and the nature of information exchanges. It provides a basis for assessing operational concepts[2] and procedures for effectiveness and DOTMLPF impacts. The OV lends itself particularly to assessing impacts of doctrine, training, personnel, and materiel.

2.2.2 Systems View

The Systems View (SV) describes the *how* of the architecture and depicts systems elements, software, data and connectivity required to support the enterprise business process. Developed after the OV, the SV is a set of graphical and textual products that associates systems resources to the OV and describes systems and interconnections. These systems resources support the operational activities and facilitate the exchange of information among operational nodes.

2.2.3 Technical Standards View

The Technical Standards View (TV) is the minimal set of rules governing the arrangement, interaction, and interdependence of system parts or elements. Its purpose is to ensure that a system satisfies a specified set of operational requirements. The TV provides the technical systems implementation guidelines upon which engineering specifications are based, common building blocks are established, and product lines are developed. The TV includes a collection of the technical standards, implementation conventions, standards options, rules, and criteria organized into profile(s) that govern systems and system elements for a given architecture.

[2] A verbal or graphic statement, in broad outline, of a commander's assumptions or intent in regard to an operation or series of operations. (Joint Pub 1-02)

2.3 Structured Analysis Method for OV-SV Integration

Structured Analysis methods relate operational activities to system functions through abstract matrix mapping.

The DoDAF Deskbook in the chapter entitled, "Techniques for Developing Architectures," prescribes an approach for constructing the system views. The following steps are generally applicable in constructing the systems views for As Is Architectures:

- Identify physical node locations to determine communications asset availability.
- Identify and characterize available systems that support the business processes needed at the operational nodes.
- Identify the system functions in the current systems and create the SV-4 System Function Model.
- The Systems Functionality Description may be hierarchical in nature and may have both a hierarchy or decomposition model and a system data flow model. The hierarchy model documents a functional decomposition. The functions decomposed are system functions.
- Associate existing system functions with the operational activities they support
- Using the As Is system functions and the operational activity model (OV-5), map the existing system functions to the activities they support to create the As-Is Operational Activity to Systems Function Traceability Matrix (SV-5). The system functions documented in the SV-4 may be identified using a system function taxonomy. An example of a taxonomy is the Service Component Reference Model created by the federal government to identify and classify software service components that support federal agencies and their IT investments and assets.[3]

For **To-Be** Architectures the order of developing SV-4 and SV-5 is reversed:

- Based on the operational activities (OV-5), determine the required system functions (SV-4).
- Build the To-Be Operational Activity to Systems Function Traceability Matrix (SV-5), which provides the primary bridge between the Operational View and the Systems View.
- Define the relationship among system functions
- Given the system functions identified in the To-Be SV-5, develop a decomposition of those functions by identifying and organizing associated sub-functions. This provides the functional decomposition version of the To-Be Systems Functionality Description (SV-4).
- Determine systems' behavior
- Once the system functions (SV-4) and operational activity to system function matrix (SV-5) are developed, the systems functionality sequence and timing description (SV-10) and the systems interface description (SV-1) can be developed. This step will typically require several cycles of iteration. Work on the SV-10 product set will likely identify system functions that must be folded into SV-4 and SV-5. Similarly, as the SV-4/SV-5 is matured, any new system functions must be incorporated into the SV-10.
- Assign systems and their interfaces to the system nodes to develop the systems interface description (SV-1).
- Map information exchange requirements to candidate systems
- Referring to the **OV-5** for information exchanges, **SV-5** for the relation between operational activities and system functions, and **SV-11** for the physical data model, develop the systems-related data exchange requirements that match the IEs presented in **OV-3**. Such an analysis supports a determination of how well information would flow during the operation. Produce the **Systems Data Exchange Matrix (SV-6)**.

Ring, et al, in their Activity Based Methodology for Development and Analysis of Integrated DoD Architectures, which uses Structured Analysis as its foundational method, prescribes the following steps for creating the System views for a To Be model:

- Create the SV-4 System Function Model
- Create the SV-1 System Nodes

[3] U.S. Office of Management and Budget, "Federal Enterprise Architecture."

- Create the SV-1 Systems
- Manually associate the System Functions with System Nodes with Systems
- Using automation, form the associations between the system functions, system nodes and systems
- Using automation, render the system data exchanges, complete the SV-1 rendering, and generate the SV-6 System Data Exchange Matrix
- Map Operational Activities to System Functions to build the SV-5

2.4 Object-Oriented Method for OV-SV Integration

OO methods focus on concrete interaction between operational and system elements to assure operational activities and supporting system functions co-evolve. Allocation of operational behavior to system elements assures effective alignment of system to operations.

In contrast to the Structured Analysis method of abstract mapping for relating operational activities to system functions, the Object Oriented approach focuses on capturing the interactions between the subject system and its operators or other external systems. The OO method begins by first defining the system in the context of its environment showing who (person or external system) is using the system to achieve a desired result. Next, it allocates the overall desired behavior in discrete chunks called use cases, which have distinct results of value for users of the use case. The use cases describe actors having a dialog with the system to achieve a measurable result of value. This dialog implicitly allocates responsibilities between the system and the actors within the context of a process that produces something of value. The dialog assigned to the system is then collected into a set of services dialog, which are assigned to an interface class that is realized by the system under construction. In following stages, the services are allocated to system elements, both logical and physical, using a process called service realization. Realizing services is the process of allocating responsibility for the service across system elements. It is a key feature of object-oriented methods and is responsible for a strong relationship between design abstraction layers in the developing solution.

In summary, while structured analysis methods assert a support relationship exists between separately developed system functions and operational activities, object-oriented methods establish the relationship by modeling the interactions between the system and its environment, thus co-evolving the process and its system support.

3.0 Architecture Development Process

The following sections describe a process for building an integrated architecture containing both an operational view and system view for complex systems. The Technical Standards View is part of the integrated architecture but not included in this paper. The process starts with some prerequisite steps needed only at the start of the process. It then describes a set of steps that repeat as many times as is necessary to derive system elements and their structure suitable for the purpose of the architecture. The data generated by this process is suitable for generating system views. An example drawn from a small portion of the CCIC2S architecture illustrates the process.

3.1 Prerequisites

A properly formulated vision, mission, and scope, contained in the Overview and Summary (AV-1), ensure the architecture targets the right operations and supporting system solutions. AV-1 ensures useful results from the architecture with minimal effort. Since the DoDAF does not define an OV product to capture non-functional operational requirements, the method includes a Supplementary Specification that details the operational usability, reliability, performance, and supportability requirements of the system to ensure the system is effective over its lifetime. The Integrated Dictionary (AV-2) should also be developed but is not addressed in this paper. Prerequisite products are initially developed prior to starting the architecture project and should be refined iteratively throughout the project.

3.1.1 Develop Overview and Summary (AV-1)

The Overview and Summary is a document that describes the architecture effort, its purpose, scope, and use, the resources and tools to build it, and lessons learned throughout the architecture's development. It provides guidance to the development effort focused on a useful and appropriate outcome. Prepare a Summary and Overview (AV-1) that defines the purpose of the architecture, the data the architecture must provide to satisfy its purpose, the way the data will be used and the granularity needed. If the reader is interested, refer to the DoDAF for further guidance.

3.1.2 Create Supplementary Specification

Not captured in the DoDAF architecture products, the Supplementary Specification[4] captures the non-functional operational requirements. Supplementary specifications address non-functional requirements such as legal and regulatory requirements, usability, reliability, performance, and supportability that make the system effective over its lifecycle. Appropriate subsets of these are allocated to system elements as the system is architected and designed. These constraints ensure the system effectively supports its users in the conduct of operations. While the DoDAF does not currently define a product in the operational view for these types of architectural data, they are important to the successful implementation of the system.

3.2 Create Context Diagram

From an architecture development perspective, the system boundary is of critical importance. The context diagram identifies the key entities that must be provided to or produced by the system within the enterprise context. A context diagram also shows the actors external to the system and their exchanges with the system. UML uses actors to represent the first order system environment at the system boundary. Actors may be anything that lies outside the system with which it interacts including both people and other systems. The context diagram, represented by a UML class diagram, depicts the subject system (i.e., the architected system), associations with actors outside the system, and the classes of information those actors send to or receive from the system. The context diagram helps define the system's boundary and identify valuable results the system produces on behalf of the enterprise. The information aids in identifying use cases (uses of the system) that specify the key processes of the enterprise supported by the system.

Figure 6 depicts a small portion of the context diagram for the **ISC2 System**. The subject system is represented by the class "ISC2 System" with the <<system>> stereotype marking. Four actors—Strategic MW Data Provider, MW Operations Planner/Tasker, Launch System Scheduler, and Missile Analyst—are associated with the system indicated by the connecting lines. The actors send or receive four input/output entities to or from the system. Those received from the system are its valued results; those sent to the system represent important contributions from its environment needed to produce those results.

The actors represent a role or set of responsibilities that an organization or system must take on in order for the subject system to accomplish its purpose. By capturing the responsibilities to provide or receive information relative to the system, the actors/roles start to define the external interfaces of the system without binding them to explicit organizations or systems so that the subject system might be reusable to any organization that requires its services.

[4] Rational Unified Process® Version 2003.06.13

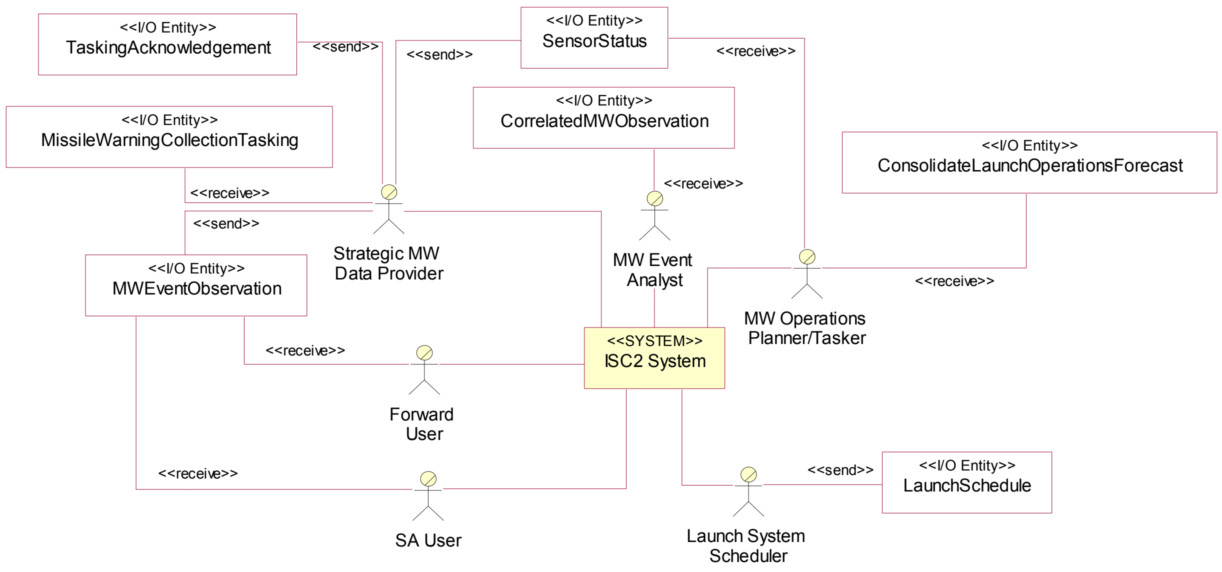

<<I/O Entity>> TaskingAcknowledgement
<<send>>
<<send>>
<<I/O Entity>> SensorStatus
<<receive>>
<<I/O Entity>> MissileWarningCollectionTasking
<<I/O Entity>> CorrelatedMWObservation
<<I/O Entity>> ConsolidateLaunchOperationsForecast
<<receive>>
<<receive>>
<<send>>
Strategic MW Data Provider
MW Event Analyst
<<I/O Entity>> MWEventObservation
MW Operations Planner/Tasker
<<receive>>
<<receive>>
<<SYSTEM>> ISC2 System
Forward User
<<receive>>
<<send>>
<<I/O Entity>> LaunchSchedule
SA User
Launch System Scheduler

Figure 6 - Context Diagram

3.3 Develop Use Cases

Use cases represent system-enabled processes by describing how the system collaborates with its environment to produce a result of value (ROV) for one of its actors. In the use case, actors enter into a "dialog" with the system to produce this result of value. Thus, the system behavior that is required to produce a specific effect is a key part of the use case. Use cases are an important mechanism for capturing system functional requirements in terms of system behavior.

Similar to effects based planning, use case development starts with identifying the results of value the system produces. The prior work on the mission, vision, scope, and context diagram provides a rich source of ideas for this activity. Create use cases to account for all of the results of value from the system and all the system environment contributions discovered in preparing the context diagram. Each use case has a bounded set of responsibilities in producing the result; these form the use case scope. Next, create a use case diagram for each use case to identify all the potential roles various organizations and related systems might play in producing the outcome. Start to account for the contributions of each role to attaining the use case outcome. Represent these as exchanges between roles using a collaboration diagram. This set of exchanges forms an outline for the use case transactions section. Write the transactions section from the point of view of the key roles. The transactions form the "service dialog" between the roles and the system. As this portion stabilizes, model the transactions in the operational sequence diagram. Derive the process steps within the use case from related transaction sets that accomplish a part of the use case goal. From these, build the activity diagram. The following sections illustrate and elaborate on this process in detail.

3.3.1 Identify Results of Value

In preparing the vision, mission, scope, and context diagram, we already have some rich sources of system ROVs. The **<<receive>>** classes on the context diagram are a good starting point for discovering ROVs. Those classes that align with the mission and vision for the system are good candidates. There may be additional ROVs within the vision and mission.

3.3.2 Identify Use Cases for each ROV

Name use cases that account for all ROVs identified. Use cases are named using verbs that are then related to the ROV they produce, such as "Plan Operations" for example. In addition, they must account for all <<sent>> data within the context diagram as well. The syntax of the UML affords numerous opportunities to factor out common elements and show variations of processes through use case relationships. (A discussion of extension, inclusion, and generalization is beyond the scope of this paper; consult the UML 2.0 specification.)

3.3.3 Determine Use Case Scopes

Use case scopes identify the responsibilities of the use cases to create the ROV. Write each use case scope such that it reflects what the use case must do to produce the ROV. For instance, if the ROV is current weather data, does the use case go to repositories of weather data and gather it up, or must it actually collect the data with sensors?

3.3.4 Create Use Case Diagrams

For each use case, determine all the roles based on the ROV, the potential information needed to produce it, and organizational elements that provide such information. Add these to the use case diagram. Identify potential triggering roles (roles that can cause the use case to start) by connecting them to the use case with an in-bound arrow. Aggregate roles to operational nodes (<<onodes>>) as operational responsibilities are worked out. The operational node to role mapping is needed in order to produce the OV-2 Node Connectivity Description and OV-3 Information Exchange Matrix, which depict the connectivity of operational nodes via information exchange needlines among them and the attributes of information exchanges, respectively.

Figure 7 depicts an example use case diagram. In addition to the use case, its associated roles, and their operational nodes; related use cases appear on the diagram.

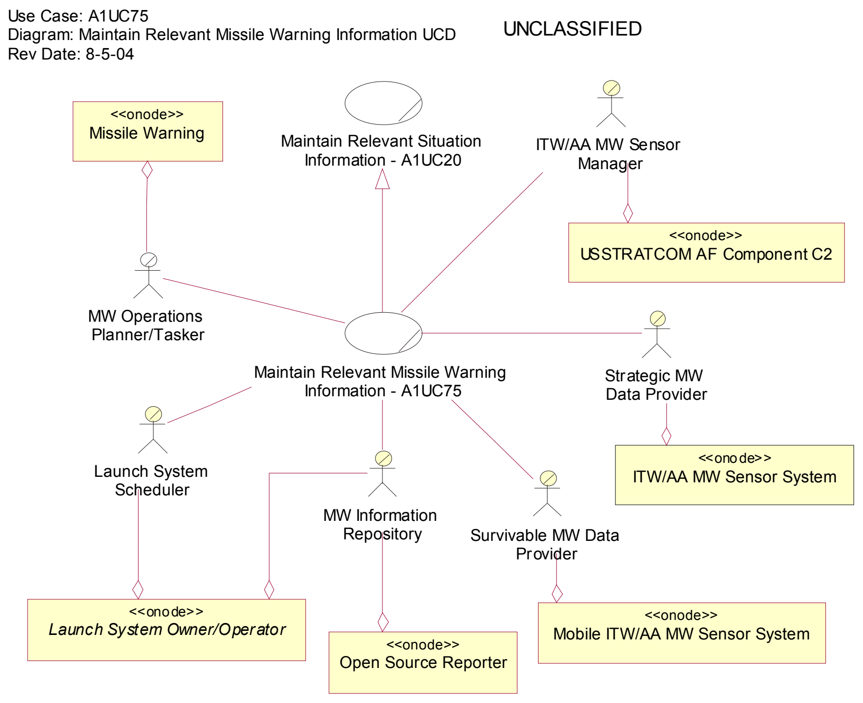

Use Case: A1UC75
Diagram: Maintain Relevant Missile Warning Information UCD
Rev Date: 8-5-04
UNCLASSIFIED
<<onode>> Missile Warning
Maintain Relevant Situation Information - A1UC20
ITW/AA MW Sensor Manager
<<onode>> USSTRATCOM AF Component C2
MW Operations Planner/Tasker
Maintain Relevant Missile Warning Information - A1UC75
Strategic MW Data Provider
<<onode>> ITW/AA MW Sensor System
Launch System Scheduler
MW Information Repository
Survivable MW Data Provider
<<onode>> Launch System Owner/Operator
<<onode>> Open Source Reporter
<<onode>> Mobile ITW/AA MW Sensor System

Figure 7 - Use Case Diagram

3.3.5 Develop the Service Dialog between Roles and the System

Start by building a collaboration diagram using roles as the objects on the diagram. Think about the contributions and needs for information among the roles. Show these as links between roles and associated Information Exchange messages that must flow to create the ROV of the use case. The inclusion of Information Exchanges (IEs) is a modification of RUP SE approach needed for consistency with the DoDAF. It provides valuable context in developing a use case.

When the IEs are reasonably complete, produce the equivalent sequence diagram and nominally order the information exchanges in time (in Rational Rose, use the F5 key to produce this diagram). This "macro" view of information needs expressed in the IEs becomes a rough outline for detailing the role-to-system service dialog. As you consider how the system supports the roles in creating the ROV, describe the interactions among the roles and the system from the primary role's viewpoint in the use case specification (textual description of the use case). This concrete perspective exposes additional information and flows as the dialog develops. It may take several iterations before the service interactions and information exchanges become stable.

Now expand the sequence diagram by including messages that represent service transactions between the roles and the system based on the description in the specification. These service transactions, or service fragments, look like UML operation calls on the system and roles. Create an interface class to hold the service transaction and build an interface realization diagram that allows the realizing role or system to take on the operation so that the messages can be properly bound (see Figure 8). The head of the service transaction arrow points toward the entity providing the service while the tail comes from the entity requesting the service. A list of parameters follows the operation name indicating classes that flow across the transaction; these classes appear in a logical data model package. A description of the service fragment is included in the documentation field of the operation; it also includes any non-functional requirements that apply. Service transactions are analyzed later to identify services (see Section 4.4)

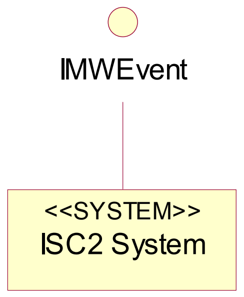


Figure 8 – Interface Realization

Figure 9 represents an early operational service interaction oriented sequence diagram that we have called a System Operational Sequence (SOS). Many of the operations are verbose English phrases and few class parameters are in place.

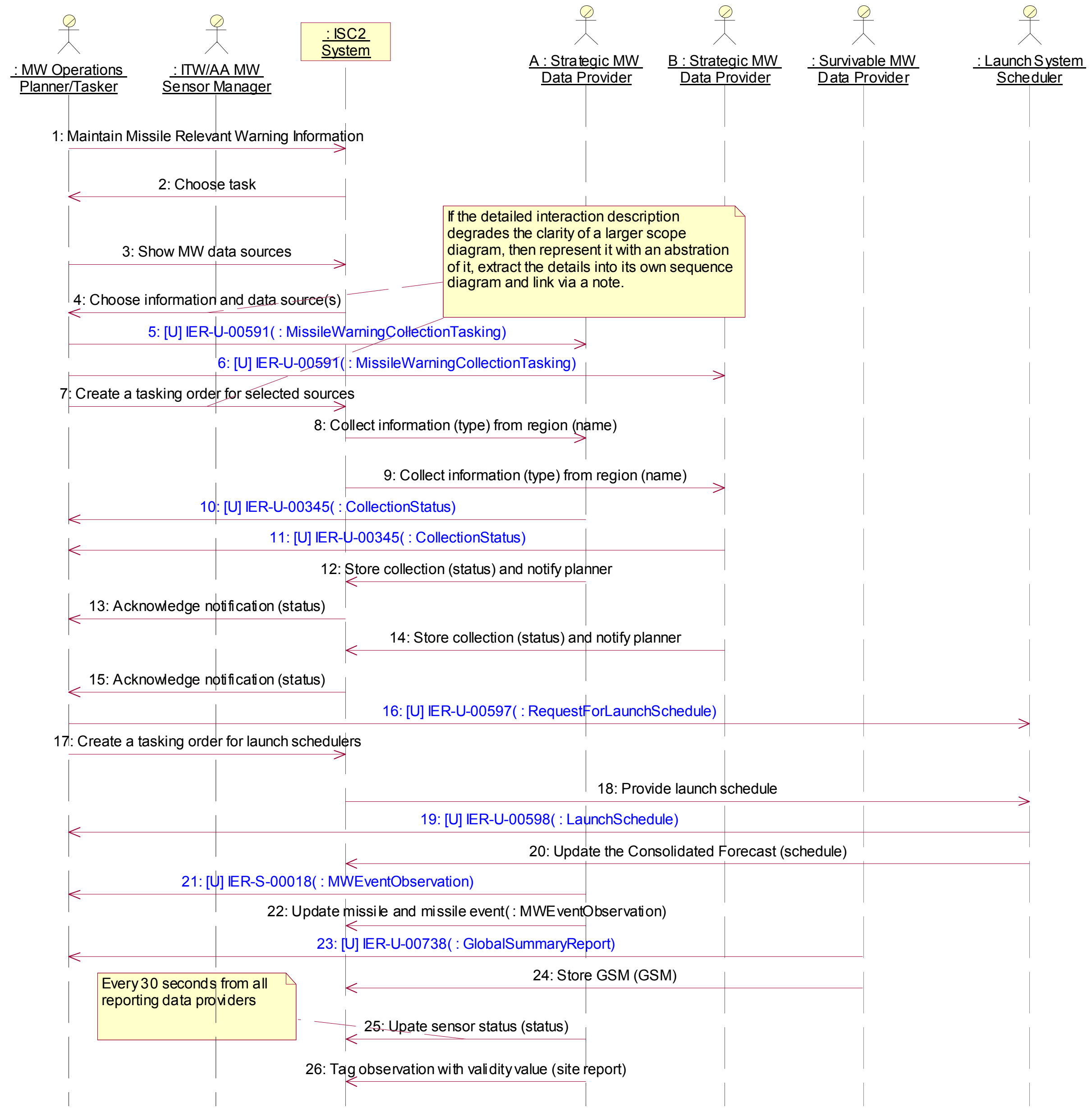

: MW Operations Planner/Tasker
: ITW/AA MW Sensor Manager
: ISC2 System
A : Strategic MW Data Provider
B : Strategic MW Data Provider
: Survivable MW Data Provider
: Launch System Scheduler
1: Maintain Missile Relevant Warning Information
2: Choose task
If the detailed interaction description degrades the clarity of a larger scope diagram, then represent it with an abstration of it, extract the details into its own sequence diagram and link via a note.
3: Show MW data sources
4: Choose information and data source(s)
5: [U] IER-U-00591(: MissileWarningCollectionTasking)
6: [U] IER-U-00591(: MissileWarningCollectionTasking)
7: Create a tasking order for selected sources
8: Collect information (type) from region (name)
9: Collect information (type) from region (name)
10: [U] IER-U-00345(: CollectionStatus)
11: [U] IER-U-00345(: CollectionStatus)
12: Store collection (status) and notify planner
13: Acknowledge notification (status)
14: Store collection (status) and notify planner
15: Acknowledge notification (status)
16: [U] IER-U-00597(: RequestForLaunchSchedule)
17: Create a tasking order for launch schedulers
18: Provide launch schedule
19: [U] IER-U-00598(: LaunchSchedule)
20: Update the Consolidated Forecast (schedule)
21: [U] IER-S-00018(: MWEventObservation)
22: Update missile and missile event(: MWEventObservation)
23: [U] IER-U-00738(: GlobalSummaryReport)
24: Store GSM (GSM)
Every 30 seconds from all reporting data providers
25: Upate sensor status (status)
26: Tag observation with validity value (site report)

Figure 9 - SOS Diagram for the Maintain Relevant MW Information Use Case

There is a tendency to develop service interactions between the system and its actors in complete detail, especially if the actor is a person or operator. For large, complex, interactive systems, this tendency leads to an overwhelming number of service transactions. While it would be accurate to reflect every step of a detailed interaction that crosses the system boundary at any model scale, this can make the diagrams difficult to read, degrade their communication value, and create an additional burden since detailed service interactions must be realized at the next level of the model. To address this, the concept of an abstract interaction is applied. At an abstract level, a sequence diagram describes the interaction sequence using less granular message parameters. An example appears in Figure 10 and Figure 11. In Figure 10, the interaction represents a single pair of messages where the system requires the operator to choose among various information elements and related sources. The operator, by so choosing, requires the system to create tasking orders for the required information sources. However, the actual intended interaction between the system and actor, illustrated in Figure 11, may contain far more detail.

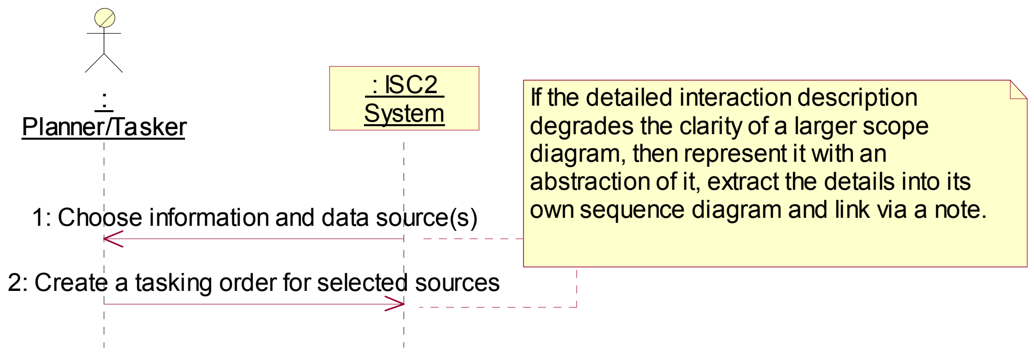

: Planner/Tasker
: ISC2 System
If the detailed interaction description degrades the clarity of a larger scope diagram, then represent it with an abstraction of it, extract the details into its own sequence diagram and link via a note.
1: Choose information and data source(s)
2: Create a tasking order for selected sources

Figure 10 - Defining Aggregate Enterprise Services

We recommend the less detailed representation of the interaction until the interaction realization must be broken up to span multiple realizing system elements or until detailed interactions are appropriate for the level of design. Adding detailed transactions too early in the design process obscures the intent of the interaction. If there are important transactions required at the more detailed level, create both representations in the model. When describing the interaction context, remain at the abstract level. Create a separate diagram to describe the detailed interaction. Bind the elements by creating a link using a diagram link on a note from the abstract representation to every related detailed interaction. Refer to these diagrams at an appropriate point in the service realization process when the transaction begins to span more than one system element

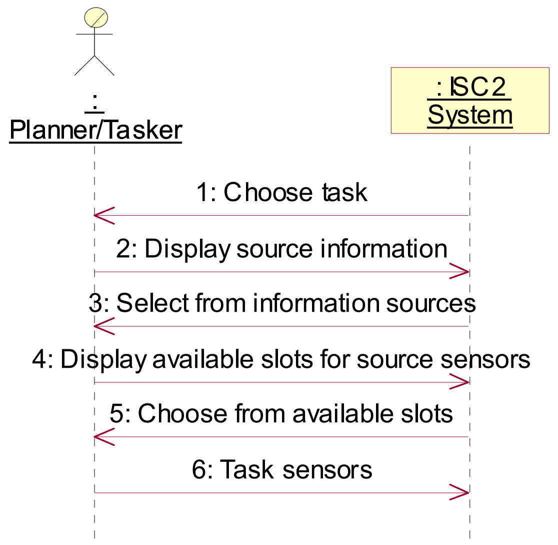
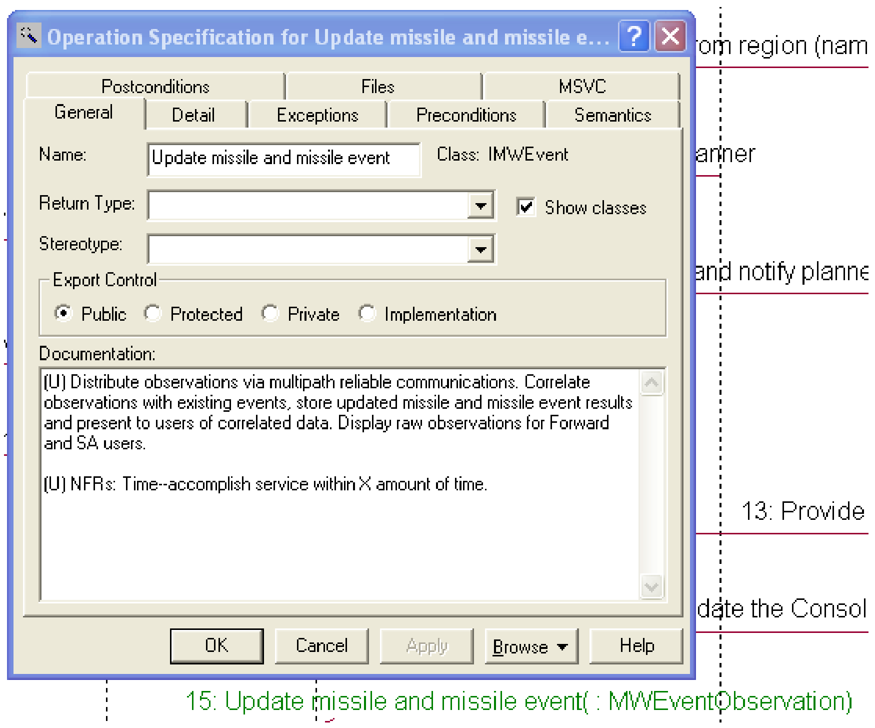

: Planner/Tasker
: ISC2 System
1: Choose task
2: Display source information
3: Select from information sources
4: Display available slots for source sensors
5: Choose from available slots
6: Task sensors

Figure 11 - Defining Detailed Enterprise Services

A sample service transaction specification is included in Figure 12. Notice that the documentation captures the scope of the service and what it should do in broad terms. Include any non-functional requirements that apply. Avoid entering too much detail until you ensure that the service transaction is not duplicated elsewhere, is named correctly, that all its actors have been identified, and any other necessary re-factoring has occurred.



Name: Update missile and missile event

Class: IMWEvent

Return Type:

Show classes [✓]

Stereotype:

Export Control: [✓ Public] Protected Private Implementation

Documentation:

(U) Distribute observations via multipath reliable communications. Correlate observations with existing events, store updated missile and missile event results and present to users of correlated data. Display raw observations for Forward and SA users.

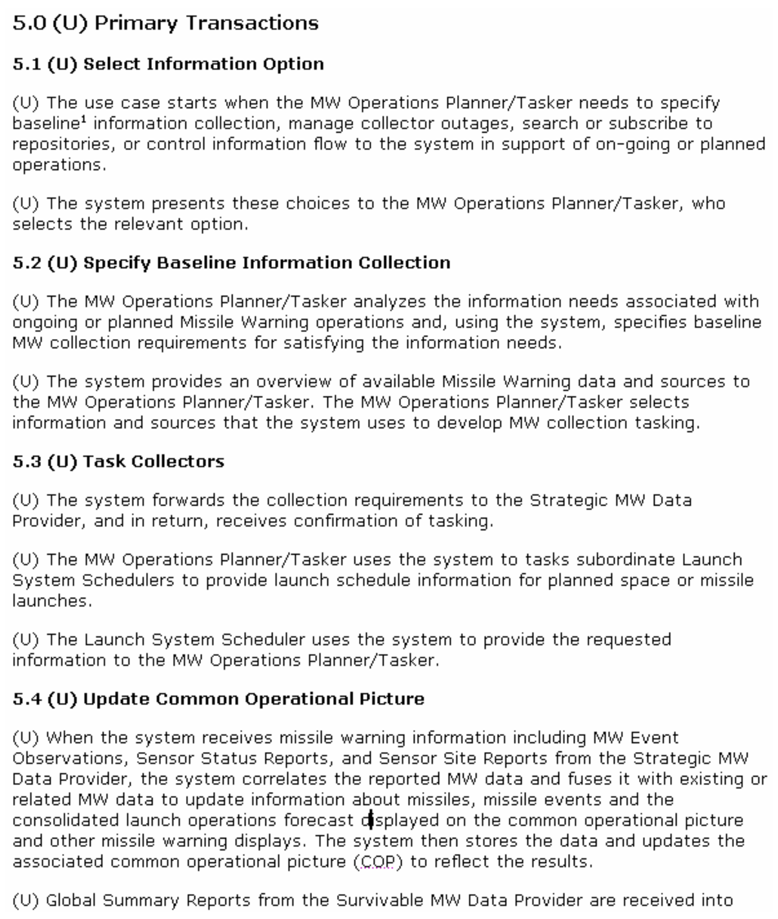
(U) NFRs: Time--accomplish service within X amount of time.

15: Update missile and missile event(: MWEventObservation)

Figure 12 - Service Specification for Update Missile and Missile Event

3.3.6 Identify Use Case Activities from Dialog Segments

Considering the service transaction dialog of the SOS (Figure 9), organize the dialog into sets of transactions that have a purpose as a group and identify a process step name based on that purpose. Add headings with the step name to the Use Case Specification to break the English description into process steps as well. Figure 13 illustrates a use case specification showing the service transaction dialog parsed into activity steps.



5.0 (U) Primary Transactions

5.1 (U) Select Information Option

(U) The use case starts when the MW Operations Planner/Tasker needs to specify baseline[1] information collection, manage collector outages, search or subscribe to repositories, or control information flow to the system in support of on-going or planned operations.

(U) The system presents these choices to the MW Operations Planner/Tasker, who selects the relevant option.

5.2 (U) Specify Baseline Information Collection

(U) The MW Operations Planner/Tasker analyzes the information needs associated with ongoing or planned Missile Warning operations and, using the system, specifies baseline MW collection requirements for satisfying the information needs.

(U) The system provides an overview of available Missile Warning data and sources to the MW Operations Planner/Tasker. The MW Operations Planner/Tasker selects information and sources that the system uses to develop MW collection tasking.

5.3 (U) Task Collectors

(U) The system forwards the collection requirements to the Strategic MW Data Provider, and in return, receives confirmation of tasking.

(U) The MW Operations Planner/Tasker uses the system to tasks subordinate Launch System Schedulers to provide launch schedule information for planned space or missile launches.

(U) The Launch System Scheduler uses the system to provide the requested information to the MW Operations Planner/Tasker.

5.4 (U) Update Common Operational Picture

(U) When the system receives missile warning information including MW Event Observations, Sensor Status Reports, and Sensor Site Reports from the Strategic MW Data Provider, the system correlates the reported MW data and fuses it with existing or related MW data to update information about missiles, missile events and the consolidated launch operations forecast displayed on the common operational picture and other missile warning displays. The system then stores the data and updates the associated common operational picture (COP) to reflect the results.

(U) Global Summary Reports from the Survivable MW Data Provider are received into

Figure 13 – Portion of Use Case Specification

3.3.7 Develop the Activity Diagram

Once the process steps stabilize, create an activity diagram highlighting control flow, based on Use Case Specification steps and conditional considerations in the use case specification. Activities represent the process steps. Decision diamonds represent conditional logic, where alternative flows leave the decision diamonds and can recombine downstream at merge points. Parallel activities, those that must start and finish in parallel, are enclosed by fork and join bars. The process has a single start point and one or more end points depending on its structure.

Figure 14 and Figure 15 illustrate an activity diagram for this use case.

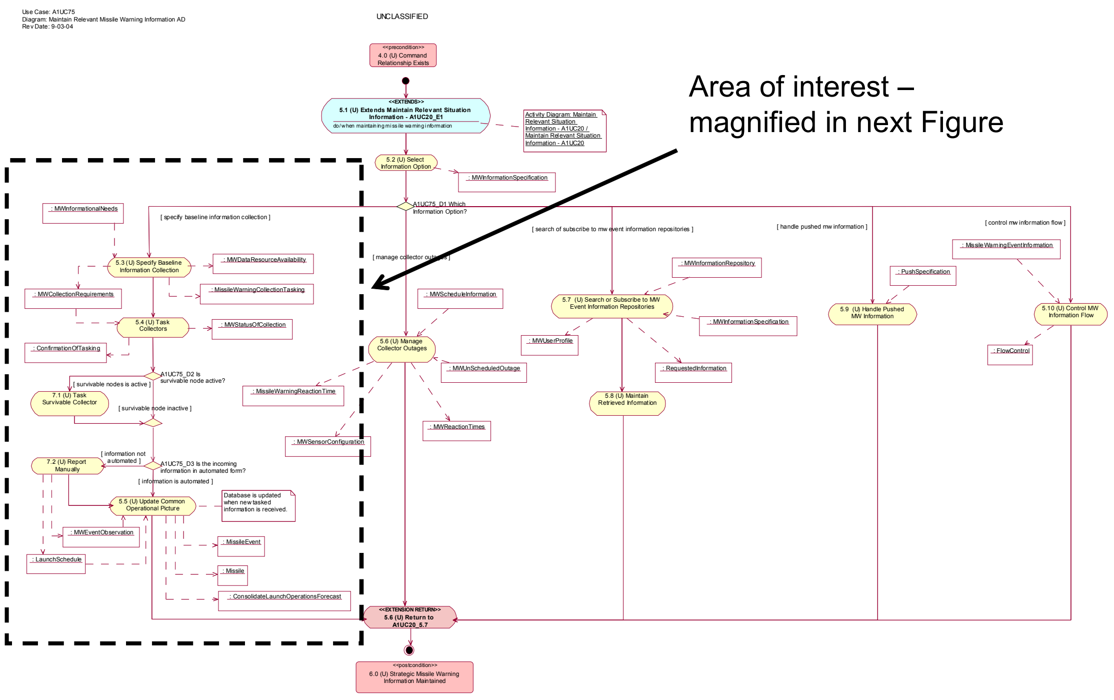

Use Case: A1UC75
Diagram: Maintain Relevant Missile Warning Information AD
Rev Date: 9-03-04
UNCLASSIFIED
Area of interest – magnified in next Figure
<<precondition>> 4.0 (U) Command Relationship Exists
<<EXTENDS>> 5.1 (U) Extends Maintain Relevant Situation Information - A1UC20_E1
do/when maintaining missile warning information
Activity Diagram: Maintain Relevant Situation Information - A1UC20 / Maintain Relevant Situation Information - A1UC20
5.2 (U) Select Information Option
: MWInformationSpecification
A1UC75_D1 Which Information Option?
[specify baseline information collection]
[manage collector outages]
[search of subscribe to mw event information repositories]
[handle pushed mw information]
[control mw information flow]
: MWInformationalNeeds
5.3 (U) Specify Baseline Information Collection
: MWDataResourceAvailability
: MissileWarningCollectionTasking
: MWCollectionRequirements
5.4 (U) Task Collectors
: MWStatusOfCollection
: ConfirmationOfTasking
A1UC75_D2 Is survivable node active?
[survivable nodes is active]
7.1 (U) Task Survivable Collector
[survivable node inactive]
[information not automated]
7.2 (U) Report Manually
A1UC75_D3 Is the incoming information in automated form?
[information is automated]
5.5 (U) Update Common Operational Picture
Database is updated when new tasked information is received.
: MWEventObservation
: LaunchSchedule
: MissileEvent
: Missile
: ConsolidateLaunchOperationsForecast
: MWScheduleInformation
5.6 (U) Manage Collector Outages
: MWUnScheduledOutage
: MissileWarningReactionTime
: MWSensorConfiguration
: MWReactionTimes
: MWInformationRepository
5.7 (U) Search or Subscribe to MW Event Information Repositories
: MWUserProfile
: MWInformationSpecification
: RequestedInformation
5.8 (U) Maintain Retrieved Information
: PushSpecification
5.9 (U) Handle Pushed MW Information
: MissileWarningEventInformation
5.10 (U) Control MW Information Flow
: FlowControl
<<EXTENSION RETURN>> 5.6 (U) Return to A1UC20_5.7
<<postcondition>> 6.0 (U) Strategic Missile Warning Information Maintained

Figure 14 - Use Case Activity Diagram

For illustrative purposes, we focused exclusively on the flow of activity on the leftmost side of the Activity Diagram, enclosed by the dashed line. Figure 15 expands this segment of the activity diagram to support the next topic of discussion. Step **5.5 Update Common Operating Picture** contains the service transaction "Update Missile and Missile Event," which provides the operational context for describing the rest of the approach.

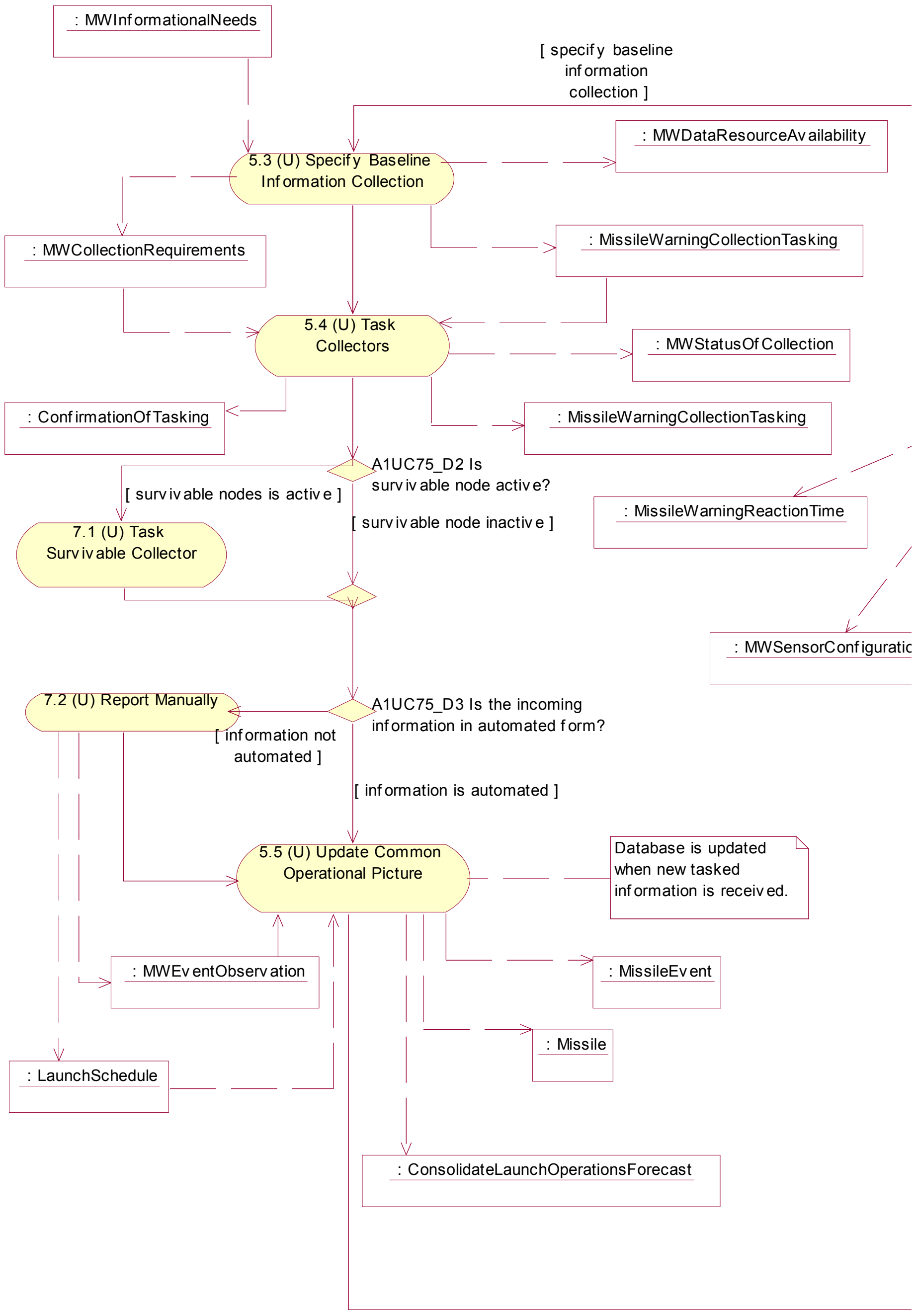

: MWInformationalNeeds
[specify baseline information collection]
: MWDataResourceAvailability
5.3 (U) Specify Baseline Information Collection
: MWCollectionRequirements
: MissileWarningCollectionTasking
5.4 (U) Task Collectors
: MWStatusOfCollection
: ConfirmationOfTasking
: MissileWarningCollectionTasking
A1UC75_D2 Is survivable node active?
[survivable nodes is active]
[survivable node inactive]
: MissileWarningReactionTime
7.1 (U) Task Survivable Collector
: MWSensorConfiguratic
7.2 (U) Report Manually
A1UC75_D3 Is the incoming information in automated form?
[information not automated]
[information is automated]
5.5 (U) Update Common Operational Picture
Database is updated when new tasked information is received.
: MWEventObservation
: MissileEvent
: Missile
: LaunchSchedule
: ConsolidateLaunchOperationsForecast

Figure 15 - Use Case Activity Diagram, Specifying Baseline Collection Fragment

3.4 Identify Services

A service is a transaction, or a collection of transactions, that provides a well-defined interface and procedure to deliver information or other outcomes to a set of service consumers. Services are based on agreements among operators and developers on useful units of system functionality that are then implemented using a service pattern. The service pattern is a design pattern that combines a programming language independent way of discovering, binding to, and invoking a defined set of functions located within a network. By implementing services in this way, service delivery is decoupled from service implementation, fostering parallel development and system evolution. Services tend to align with business processes, providing access to functionality that is meaningful in that context.

By analyzing service transactions on the sequence diagram, the architect can discover groupings of system responsibilities. When considered across the complete set of use cases, these groupings identify candidate services. Often these groupings, or access patterns, occur multiple times within and across use cases. In some cases, a single service may accommodate several variations in the access pattern. The following sections discuss service identification in more detail.

3.4.1 Analyze Use Case Event Flow to Identify Services

To identify services, look across use case sequence diagrams for common access patterns as prospective services. Such patterns may involve service transactions with common class parameters or operation names clustered by use case. Once one or more related transactions are located, look at pattern variations. For instance, there are many instances where the system issues **<<: EventAlerts>>**, but sometimes they require approval before release. There may be a service that handles event alerts and accommodates alerts that require approval. There are several event alerts based transactions in the model including:

- **alert(: EventAlert)**
- **approveAlert(inout : EventAlert)**
- **monitor(: EventIndications, out : EventAlert)**

The first issues an alert. The second approves an alert for release. The third creates an alert when event indications are present. Define a consistent set of operations across the use cases and then develop them into a service. This may require changes in service transactions within sequence diagrams, both in form and in order, to ensure that service invocations are properly represented across the model.

3.4.2 Organize Services into Interfaces

Once services are identified and normalized, collect groups of service transactions for the services together by assigning the operations to an interface[5].

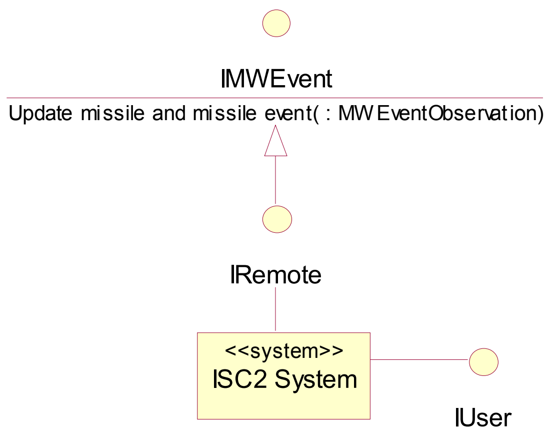


Figure 16 - Enterprise/External Interfaces

Figure 16 shows a simple interface relationship using a class diagram that represents a small subset of the ISC2 interfaces. When more than one service transaction is part of the service, a description of the service is located in the documentation field associated with the interface class; each of the service transactions is documented in the operation documentation field. In the diagram, IMWEvent is a service interface with a single operation—Update missile and missile event. In addition, the example interfaces are collected into remote access services for external systems (**IRemote)** and user services intended to support human interaction (**IUser)**. Since the Missile Warning Event service is invoked by an external system, it belongs to the IRemote collection of interfaces indicated by the generalization relationship between the two. By organizing services in this way, the architect can collect

[5] From the UML Specification: An interface declares a set of public features and obligations that constitute a coherent service offered by a classifier. They describe the services that the instances of that classifier offer to their clients. Note that a given classifier may implement more than one interface and that an interface may be implemented by a number of different classifiers. Operations and/or attributes owned by an interface are abstract and imply that the conforming instance should maintain information corresponding to the type and multiplicity of the property and facilitate retrieval and modification of that information. The set of interfaces realized by a classifier are its provided interfaces, which represent the obligations that instances of that classifier have to their clients.

and separate all the remote and user services required by the **ISC2 System**.

A reminder on UML notation – the two diagrams depicted in Figure 17 are equivalent. They both say that the implementation class realizes the service interface. The two forms are the elided (on the left) and canonical (right) forms.

3.5 Realize Services

Realizing services is the process of allocating responsibility for the service across system elements. It is a key feature of object-oriented methods and responsible for a strong relationship between design abstraction layers in the developing solution. It is used at this point in the architecture development process to transition from the operational view to the system view. Later it will be used to transition from one level of design abstraction to a more detailed level. The following subsections describe the process in more detail.

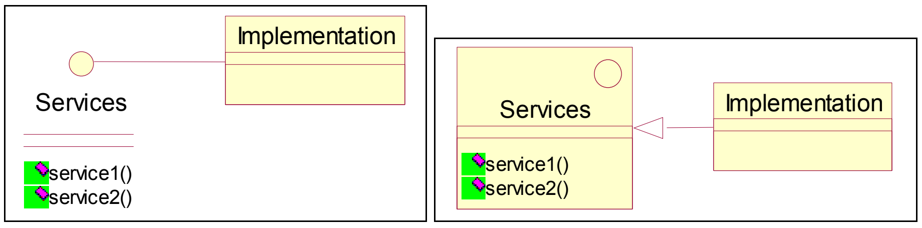


Figure 17 – Elided and Canonical Representations for Realization in UML

3.5.1 Postulate the Logical and Physical Organization of the Architecture

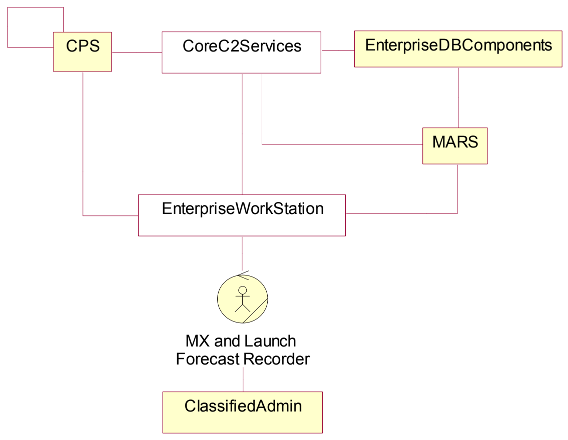
Based on analysis of operational processes and classes discovered during use case development, domain knowledge and architectural patterns, postulate the logical system structure. Many sources on architecture analysis and synthesis, including the Rational Unified Process, describe how architectures are formulated. As logical realization takes place and operational-physical node relationships develop, postulate physical equipment suites as system nodes. These system nodes implement the functionality of the logical architecture elements. Realization relationships from the physical system element to the logical system element it hosts assures that the physical system element takes on the logical interface operations. Further, physical nodes realize both operational nodes (bound to roles on the use case, Figure 7) and the system nodes that support them (at a minimum, the interfaces to the system nodes that support roles of the operational node). Physical nodes provide the facilities, people, systems, power, etc. to bear to make the operation a reality.

Figure 18 shows a set of logical system elements in the CCIC2S enterprise. In a green-field situation, where there is no existing architecture, system architects would review the documentation developed so far to identify this set of candidate system elements. With an existing architecture, one might identify the elements that already are present. The diagram evolves over time with the discovery of new elements and new relationships.

Following good systems engineering principles and practices that foster a decomposition of systems, as opposed to functions, choose names for these system elements that reflect a noun-based encapsulation of related functionality. Avoid names representing functions.

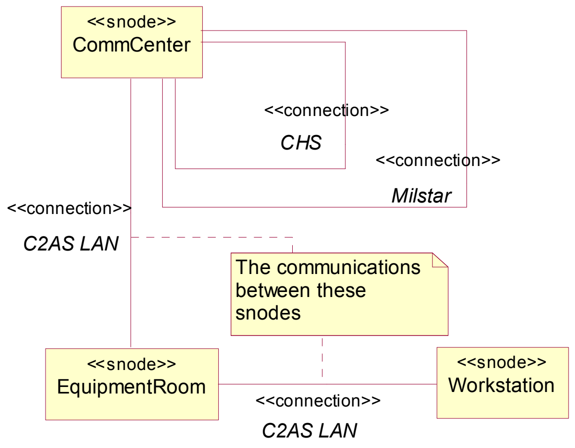
Capture the logical architecture on a series of class diagrams. Classes represent logical system elements. A line associates these classes when the two have transactions in common. Allocation of a service transaction's responsibilities to logical system element interactions in the realization process creates these associations. So, initially, add classes to the diagram for the system elements. Then, as service transactions are realized, add the interconnecting association lines.

In a similar way, classes represent physical system elements in the physical architecture. Association lines, stereotyped **<<connection>>**, interconnect them when they share transactions from the realization process. These connections represent physical links such as telephone lines, networks, or wireless links. Add names to the lines to indicate connection types (e.g., WAN--Wide Area Network) or specific connection names (e.g., SIPRNET).



CPS—Communications Processing System
MARS—Missile Analysis and Reporting System

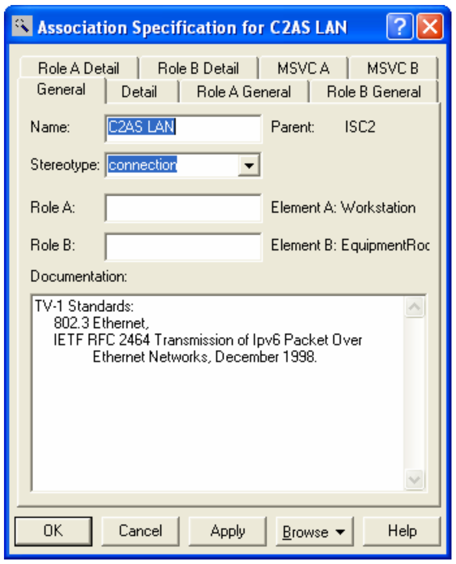
Figure 18 - Logical Architecture



CHS—Commercial High Speed
C2AS—Command and Control Automated System

Figure 19 - Physical Architecture

Figure 19 shows a set of system elements stereotyped as system nodes [**<<snode>>**] in the CCIC2S enterprise. Its connection content reflects discovery in a manner similar to the logical architecture. The system nodes are configuration groupings of physical system elements. As seen on the figure, **<<connection>>** identifies the connection types. The standards associated with the connections are in the documentation field of the connection association (see Figure 20). The diagram shows, for example, that the **CommCenter** provides **C2AS LAN**, **CHS**, and **Milstar** network connections.



Name: C2AS LAN
Parent: ISC2
Stereotype: connection
Role A:
Element A: Workstation
Role B:
Element B: EquipmentRoc
Documentation:
TV-1 Standards:
802.3 Ethernet,
IETF RFC 2464 Transmission of Ipv6 Packet Over Ethernet Networks, December 1998.

Figure 20 – Connection Standards

3.5.2 Create Logical Service Realization Diagram and Service Specification

For each service, allocate responsibility to collaborating logical system elements to realize the operations of the service. Realization must account for abstract-to-concrete transaction decomposition when data elements of the service responsibility split in order to assign them to realizing transactions assigned to different system elements. Allocate any non-functional requirements associated with the service across the realizing transactions as well.

Create the service specification as a table that captures the logical service realization information. Later, once the physical realization information is added, the tabular format facilitates a critical feature of the approach, simultaneous reasoning about the tradeoffs in the logical architecture and physical architecture. Next to the essential idea of decomposing systems instead of functions, there is perhaps no other more powerful concept than that of being able to quantitatively, precisely assess physical architecture performance in the context of logical behavior.

RequisitePro can be used to build the table, and in doing so enables the use of requirement types and traceability between various elements in the table to facilitate ad hoc queries of the information representing service realization. On the other hand, a spreadsheet application, such as Microsoft Excel used here, also facilitates ad hoc queries. In this case, elements of the service realization document other than the table are not detailed, but they are similar to a use case narrative.

Table 1 - Service Realization Table

Enterprise Service: Update missile and missile event (MWObservation)							
Flow: Determine Baseline Collection Requirements							
#	System Element Service Name	Service Description	Logical Identifier	Logical System Element	NFR	Physical Identifier	Physical System Element (snode)
1	Update missile and missile event (MWObservation)	Parse and validate raw data and create duplicate messages for each destination/comm path and transmit them via reliable protocol. Request Correlation Processor CPS to correlate and fuse observations.	Sensor	CPS	Time—complete with X amount of time	Sensor	CommCenter
2	Correlate and fuse observations (MWObservation)	Parse and validate messages and eliminate duplicate messages.	Correlation Processor	CPS	Time—A% of X	Correlation Processor	CommCenter
2	Display missile event (MWObservation)	Validate and eliminate duplicate messages.	Forward User	CPS	Time—B% of X	Forward User	CommCenter
2	Display missile event (MWObservation)	Validate and eliminate duplicate messages.	Correlation Processor	CPS	Time—B% of X	Correlation Processor	CommCenter
3	Correlate and fuse observations (MWObservation)	Store the mw data, synchronize the appropriate databases, log the data, and perform event handling. Request Correlation Processor MARS to perform threat processing.	Correlation Processor	Core C2 Services	Time--C% of X	Correlation Processor	EquipmentRoom
4	Perform threat processing (MWObservation)	Determine areas at risk, calculate d/r sensor intercept time, characterize intended targets. Request Correlation Processor MARS to render missile event.	Correlation Processor	MARS	Time—D% of X	Correlation Processor	EquipmentRoom
5	Render Missile Event (CorrelatedMWObservation)	Present information in tabular form	Correlation Processor	EWS	Time—E% of X	Correlation Processor	Workstation
6 Occurs in parallel with Step 3	Render Missile Event (MWObservation)	Present information in tabular form	Forward User	EWS	Time F% of X	Forward User	Workstation
6 Occurs in parallel with Step 3	Render Missile Event (MWObservation)	Present information in tabular form	SA User	EWS	Time--E% of X	SA User	Workstation

The table columns include: 1) #--the order in which actions occur, 2) System Element Service Name—name of the realizing service, 3) Service Description—expected service to be provided, 4) Logical Identifier—designates the specific logical element, 5) Logical Element—logical system element providing the service, 6) NFR—non-functional requirements that apply to the service, 7) Physical Identifier—designates the specific physical element, and 8) Physical System Element—system configuration that provides the service. These columns reflect the associated information about the collection of services captured by the rows. There is one row for each service transaction in the realization. The first row reflects the service being realized. Note that the description of the service in the table differs from that in the documentation field because it includes knowledge of the system structure that was intentionally unspecified when the service was originally defined. The row also corresponds to the initial transaction in the realizing sequence diagrams that follow. The remaining rows reflect the allocation of responsibility across the system elements needed to complete the service.

It is the analyst's choice whether to build the logical service realization diagram (Figure 21 - Logical Service Realization Diagram and Figure 22 –Logical Service Realization Diagram (magnified for readability) from Table 1 - Service Realization Table above or vice-versa. In fact, in practice, the team recommends building them together – it is important that the physical realization diagram, logical realization diagram, and service realization table remain synchronized. We find that each view compliments the other; from each, one discovers and can analyze different aspects of the anticipated system's behavior and architecture.

Figure 21 and Figure 22 show the logical realization diagram. It starts with a system actor representing any source of invocation of the service. Many business actors may correspond to this system actor. Transactions 3, 7, and 9 on the diagram occur in parallel as indicated by the rows numbered 2 in the table. Notice there are also linked notes that allow one to navigate to the logical or physical architecture diagrams as well as to more detailed service realization diagrams for two of the services shown here.

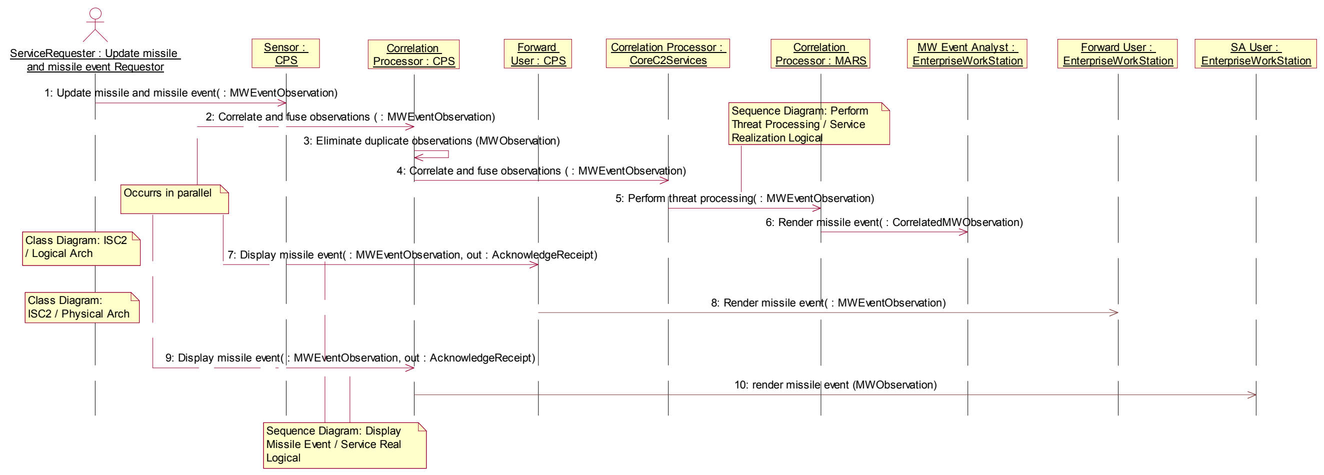

ServiceRequester : Update missile and missile event Requestor
Sensor : CPS
Correlation Processor : CPS
Forward User : CPS
Correlation Processor : CoreC2Services
Correlation Processor : MARS
MW Event Analyst : EnterpriseWorkStation
Forward User : EnterpriseWorkStation
SA User : EnterpriseWorkStation
1: Update missile and missile event(: MWEventObservation)
2: Correlate and fuse observations (: MWEventObservation)
3: Eliminate duplicate observations (MWObservation)
4: Correlate and fuse observations (: MWEventObservation)
5: Perform threat processing(: MWEventObservation)
6: Render missile event(: CorrelatedMWObservation)
7: Display missile event(: MWEventObservation, out : AcknowledgeReceipt)
8: Render missile event(: MWEventObservation)
9: Display missile event(: MWEventObservation, out : AcknowledgeReceipt)
10: render missile event (MWObservation)
Sequence Diagram: Perform Threat Processing / Service Realization Logical
Occurs in parallel
Class Diagram: ISC2 / Logical Arch
Class Diagram: ISC2 / Physical Arch
Sequence Diagram: Display Missile Event / Service Real Logical

Figure 21 - Logical Service Realization Diagram

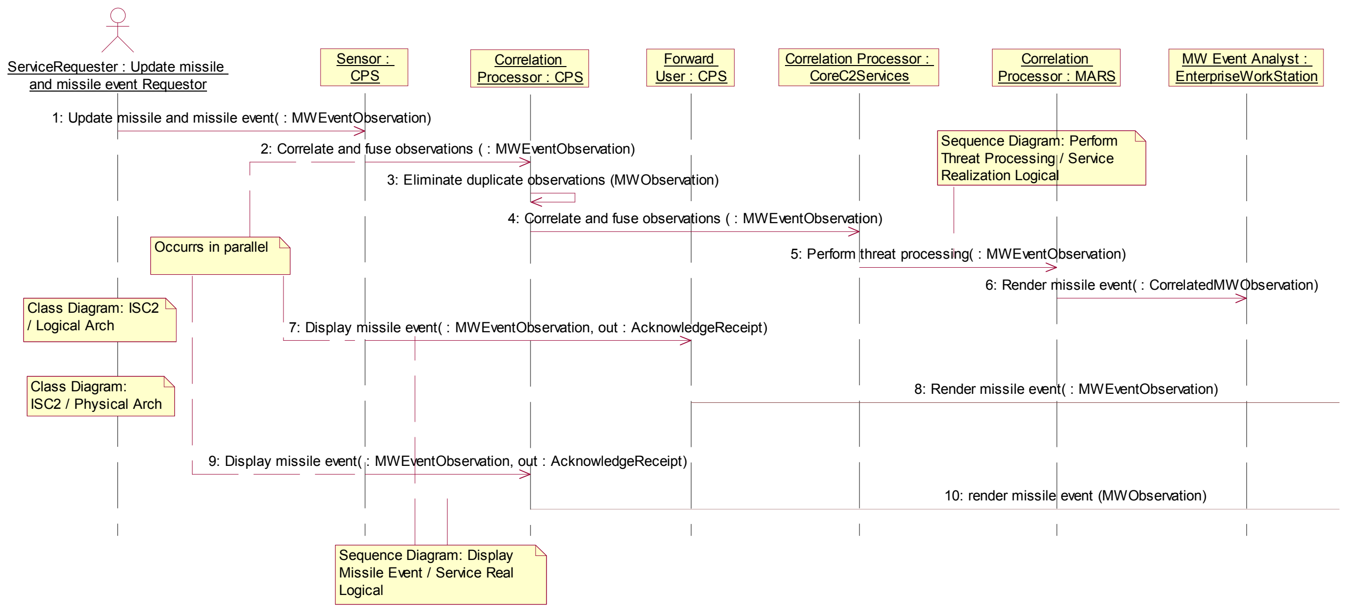

ServiceRequester : Update missile and missile event Requestor
Sensor : CPS
Correlation Processor : CPS
Forward User : CPS
Correlation Processor : CoreC2Services
Correlation Processor : MARS
MW Event Analyst : EnterpriseWorkStation
1: Update missile and missile event(: MWEventObservation)
2: Correlate and fuse observations (: MWEventObservation)
3: Eliminate duplicate observations (MWObservation)
4: Correlate and fuse observations (: MWEventObservation)
5: Perform threat processing(: MWEventObservation)
6: Render missile event(: CorrelatedMWObservation)
7: Display missile event(: MWEventObservation, out : AcknowledgeReceipt)
8: Render missile event(: MWEventObservation)
9: Display missile event(: MWEventObservation, out : AcknowledgeReceipt)
10: render missile event (MWObservation)
Sequence Diagram: Perform Threat Processing / Service Realization Logical
Occurrs in parallel
Class Diagram: ISC2 / Logical Arch
Class Diagram: ISC2 / Physical Arch
Sequence Diagram: Display Missile Event / Service Real Logical

Figure 22 –Logical Service Realization Diagram (magnified for readability)

While the messages exhibit operations bound to the system elements, these operations are stored in interface classes that are realized by the system elements. This practice allows for easy reallocation should the need arise as the architecture matures. It also fosters concordance between the logical and physical realizations of the same service (see Figure 23 below).

3.5.3 Create Physical Realization Diagram

Determine correspondence between logical and physical elements. Create realization relationships from the physical system elements to the logical ones they host. This produces the diagram shown in Figure 23.

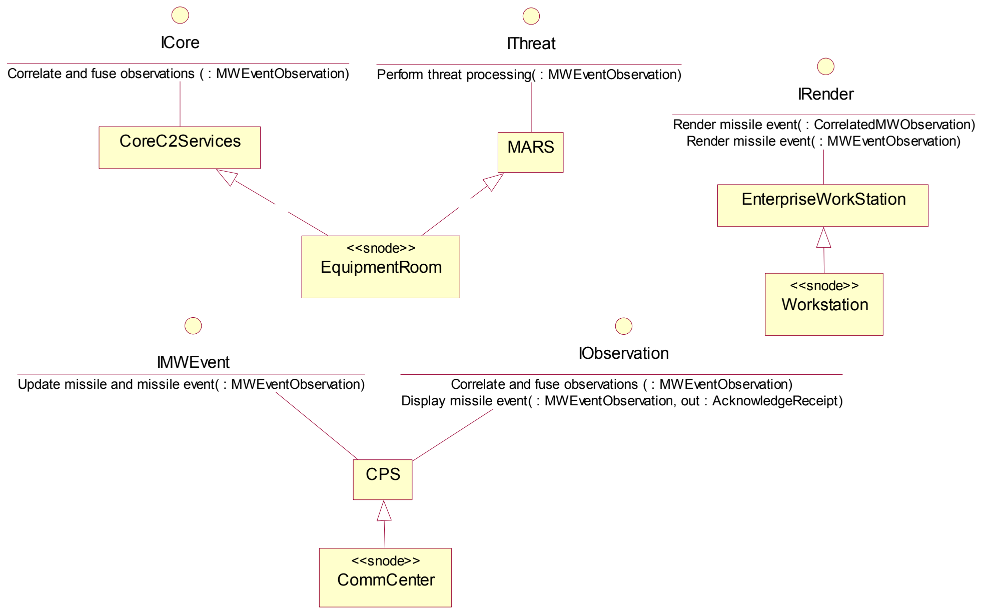

ICore
Correlate and fuse observations (: MWEventObservation)
CoreC2Services
IThreat
Perform threat processing(: MWEventObservation)
MARS
IRender
Render missile event(: CorrelatedMWObservation)
Render missile event(: MWEventObservation)
EnterpriseWorkStation
<<snode>> EquipmentRoom
<<snode>> Workstation
IMWEvent
Update missile and missile event(: MWEventObservation)
IObservation
Correlate and fuse observations (: MWEventObservation)
Display missile event(: MWEventObservation, out : AcknowledgeReceipt)
CPS
<<snode>> CommCenter

Figure 23 – Interface Realizations

Assign realization responsibilities identified during the logical realization to corresponding physical elements using a sequence diagram as shown in Figure 24 and Figure 25. Add the physical element data to the realization specification as well. On these diagrams, both a primary set of physical system elements located at the CMOC (Cheyenne Mountain Operations Center) and an alternate set at Offutt AFB correlate and fuse the missile warning event observation. Using the physical realization diagram and Table 1, one can readily analyze performance attributes and budgets as well as other nonfunctional requirements. Store the non-functional requirement allocations in the documentation fields of the applicable operations.

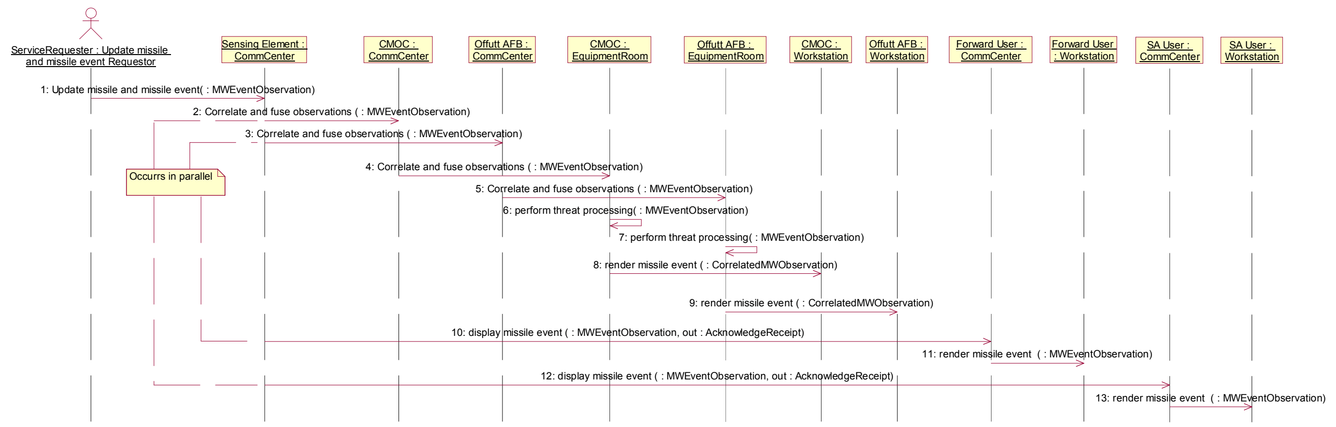

ServiceRequester : Update missile and missile event Requestor
Sensing Element : CommCenter
CMOC : CommCenter
Offutt AFB : CommCenter
CMOC : EquipmentRoom
Offutt AFB : EquipmentRoom
CMOC : Workstation
Offutt AFB : Workstation
Forward User : CommCenter
Forward User : Workstation
SA User : CommCenter
SA User : Workstation
1: Update missile and missile event(: MWEventObservation)
2: Correlate and fuse observations (: MWEventObservation)
3: Correlate and fuse observations (: MWEventObservation)
Occurs in parallel
4: Correlate and fuse observations (: MWEventObservation)
5: Correlate and fuse observations (: MWEventObservation)
6: perform threat processing(: MWEventObservation)
7: perform threat processing(: MWEventObservation)
8: render missile event (: CorrelatedMWObservation)
9: render missile event (: CorrelatedMWObservation)
10: display missile event (: MWEventObservation, out : AcknowledgeReceipt)
11: render missile event (: MWEventObservation)
12: display missile event (: MWEventObservation, out : AcknowledgeReceipt)
13: render missile event (: MWEventObservation)

Figure 24 - Physical Service Realization

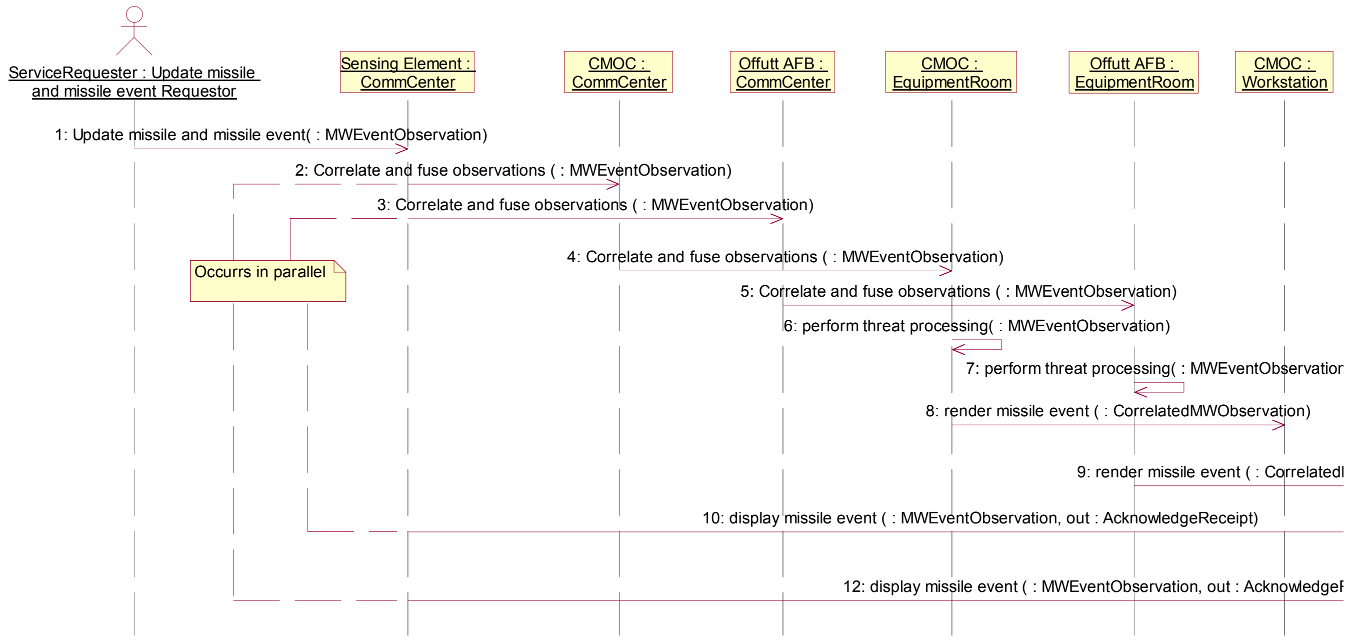

ServiceRequester : Update missile and missile event Requestor
Sensing Element : CommCenter
CMOC : CommCenter
Offutt AFB : CommCenter
CMOC : EquipmentRoom
Offutt AFB : EquipmentRoom
CMOC : Workstation
1: Update missile and missile event(: MWEventObservation)
2: Correlate and fuse observations (: MWEventObservation)
3: Correlate and fuse observations (: MWEventObservation)
Occurs in parallel
4: Correlate and fuse observations (: MWEventObservation)
5: Correlate and fuse observations (: MWEventObservation)
6: perform threat processing(: MWEventObservation)
7: perform threat processing(: MWEventObservatior
8: render missile event (: CorrelatedMWObservation)
9: render missile event (: Correlatedl
10: display missile event (: MWEventObservation, out : AcknowledgeReceipt)
12: display missile event (: MWEventObservation, out : AcknowledgeF

Figure 25 - Physical Service Realization (magnified for readability)

3.6 Reapply Architecture Development Process

Once realization at one level is complete, reapply the architecture development process to all sub-systems to develop the next level. Continue applying the process until design level elements emerge. Often, dedicated teams develop the system elements. When this is the case, this execution of the process can proceed in parallel, with each element team working on its element to accomplish the process. Should the teams encounter problems with the service responsibilities allocated to their element during the last iteration, they must renegotiate the responsibilities and reflect the new allocation in the higher-level model.

3.6.1 CPS as System

The Communications Processing System (CPS) is the system element used to illustrate the next level application of the process by considering it as though it were a subject system in its own right.

The prerequisite products do not have to be prepared as the information is already available at the start of this iteration. The AV-1 summary and overview mission, vision, and scope apply to the overall system and to this subsystem as well. The preceding logical and physical realization processes allocated non-functional requirements from the supplementary specification to this subsystem. So start the process with the context diagram.

3.6.1.1 Create CPS Context Diagram

From the logical architecture diagram in Figure 18 and the logical realization diagram in Figure 21, CPS connects to Core C2 Services, the enterprise workstation, and the Strategic MW Data Provider. CPS receives a MW Event Observation from the Strategic MW Data Provider and sends it to both the Core C2 Services and Enterprise Work Station now represented as system actors relative to CPS. Note that, while CPS also connects to itself (see Figure 18), this feature is an internal design concern and does not appear on the context diagram.

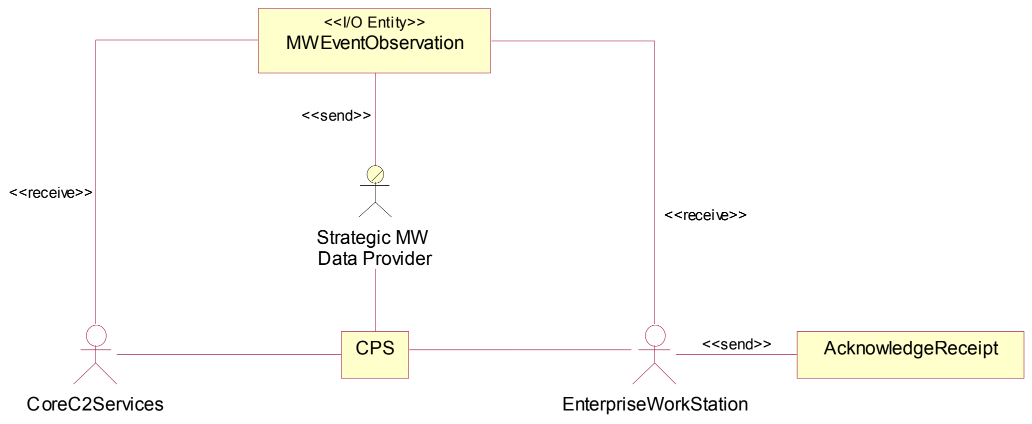


Figure 26 - CPS System Context Diagram

3.6.1.2 Develop CPS Use Cases

3.6.1.2.1 Identify CPS Results of Value

From the context diagram, CPS basically receives MW Event Observations from outside sources and invokes the appropriate services from Core C2 Services and the enterprise work station. The result of value is that the system appropriately handles these observations.

3.6.1.2.2 Identify CPS Use Cases

CPS has three service transactions—Update Missile and Missile Event (:MWEventObservation), Correlate and Fuse Observations (:MWEvent Observation), and display missile event (:MWEventObservation, out:AcknowledgeReceipt). These three occur several times throughout the CCIC2S model. In the instance explored so far, the data comes from the Strategic MW Data Provider. However, at another place in the model, the Theater MW Data Provider triggers the same set of responses. Based on this analysis, identify the use case Distribute Missile Warning Data..

3.6.1.2.3 Determine CPS Use Case Scopes

This use case is responsible for distributing the raw and correlated MW Event Observation data to various processing and presentation sites.

3.6.1.2.4 Create CPS Use Case Diagrams

Figure 27 shows the system use case diagram.

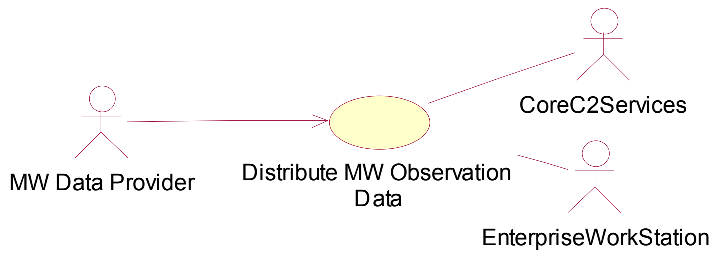


Figure 27 - CPS System Use Cases

3.6.1.2.5 Develop the CPS Service Dialog between Roles and the System

Based on the way CPS participates in the broader system logical realization diagrams, build the following sequence diagram (Figure 28).

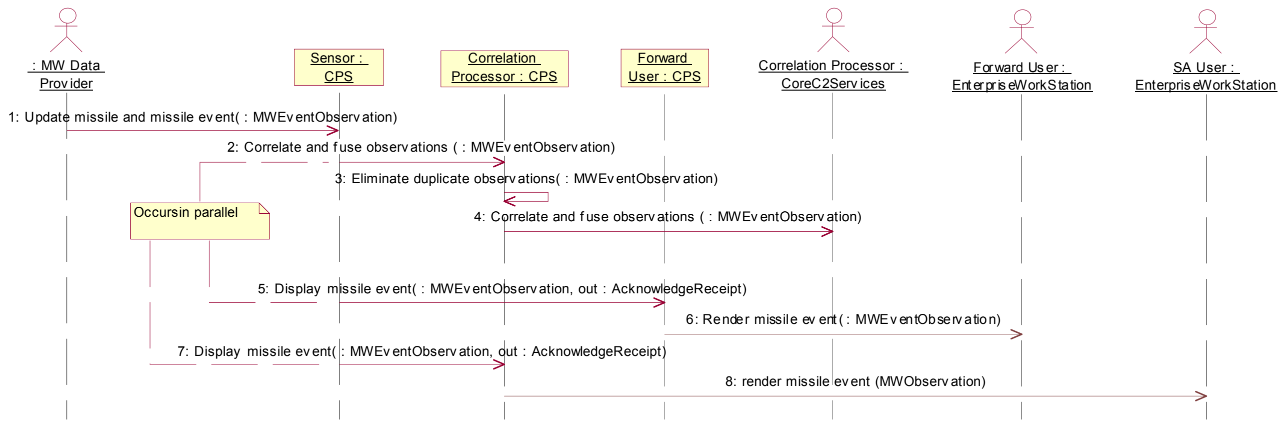
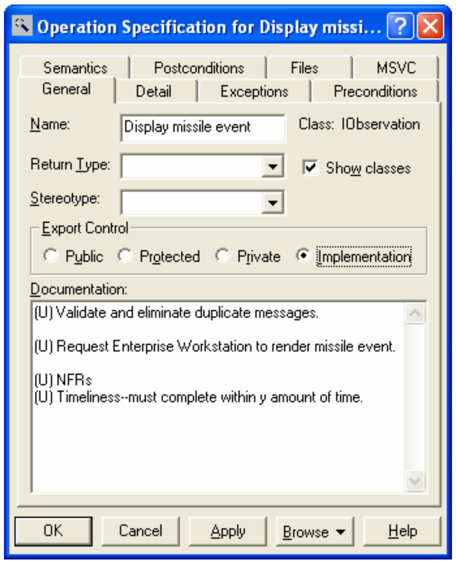

: MW Data Provider
Sensor : CPS
Correlation Processor : CPS
Forward User : CPS
Correlation Processor : CoreC2Services
Forward User : EnterpriseWorkStation
SA User : EnterpriseWorkStation
1: Update missile and missile event(: MWEventObservation)
2: Correlate and fuse observations (: MWEventObservation)
3: Eliminate duplicate observations(: MWEventObservation)
Occursin parallel
4: Correlate and fuse observations (: MWEventObservation)
5: Display missile event(: MWEventObservation, out : AcknowledgeReceipt)
6: Render missile event(: MWEventObservation)
7: Display missile event(: MWEventObservation, out : AcknowledgeReceipt)
8: render missile event (MWObservation)

Figure 28 – Distribute MW Data

The specification for "Display missile event" appears in Figure 29. Note it includes the elimination of duplicate events so there are no reflexive operation calls preceding it (as seen in transaction 3 earlier in the sequence) to eliminate the multiple requests that might arise from multi-channel messages.



Name: Display missile event
Class: IObservation
Return Type:
[✓ Show classes]
Stereotype:
Export Control: Public Protected Private [✓ Implementation]
Documentation:
(U) Validate and eliminate duplicate messages.

(U) Request Enterprise Workstation to render missile event.

(U) NFRs
(U) Timeliness--must complete within y amount of time.

Figure 29 – Display Missile Event Specification

Construct the dialog that describes this set of service transactions for the use case specification (Figure 30).

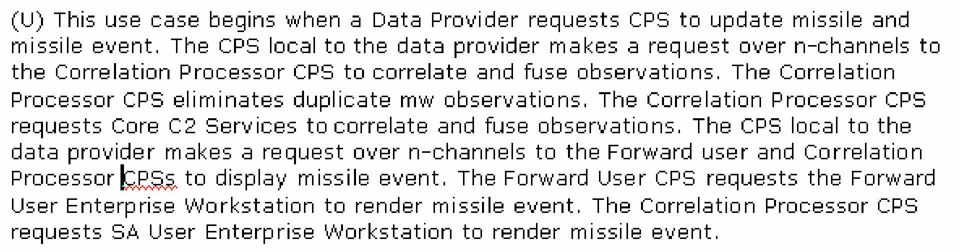
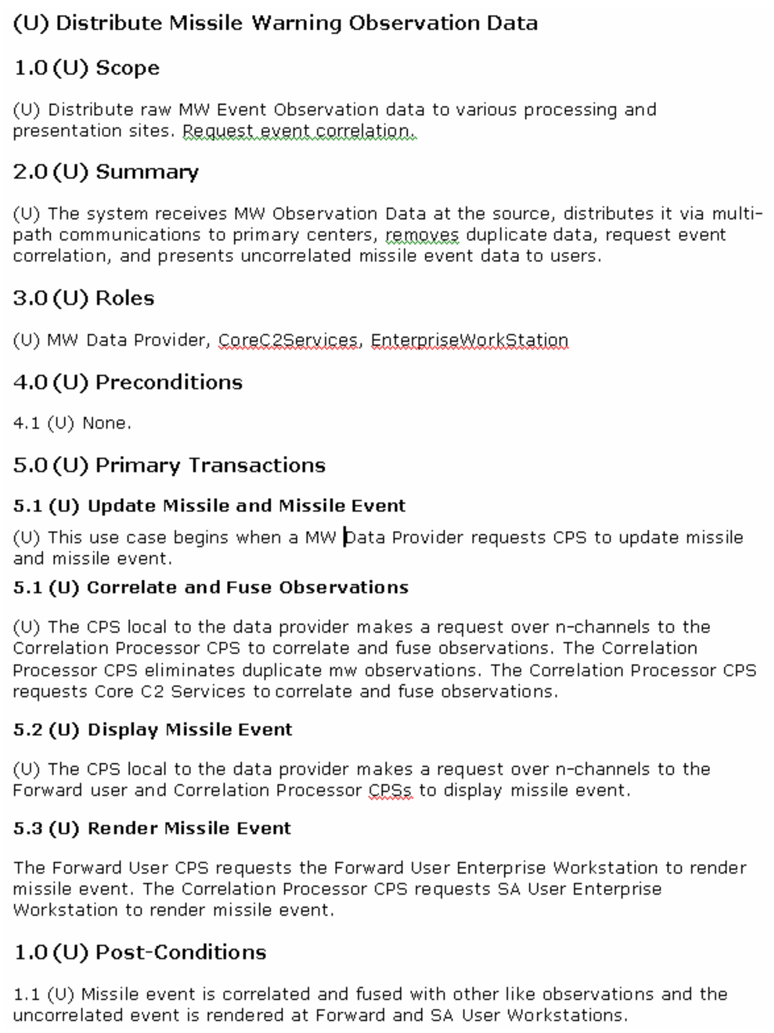

(U) This use case begins when a Data Provider requests CPS to update missile and missile event. The CPS local to the data provider makes a request over n-channels to the Correlation Processor CPS to correlate and fuse observations. The Correlation Processor CPS eliminates duplicate mw observations. The Correlation Processor CPS requests Core C2 Services to correlate and fuse observations. The CPS local to the data provider makes a request over n-channels to the Forward user and Correlation Processor CPSs to display missile event. The Forward User CPS requests the Forward User Enterprise Workstation to render missile event. The Correlation Processor CPS requests SA User Enterprise Workstation to render missile event.

Figure 30 – CPS Service Dialog Narative

3.6.1.2.6 Identify Use Case Activities from Dialog Segments

Looking across the service dialog, break it into activities based on clustering related transactions. Also, reflect these activities in the use case specification (Figure 31).



(U) Distribute Missile Warning Observation Data

1.0 (U) Scope

(U) Distribute raw MW Event Observation data to various processing and presentation sites. Request event correlation.

2.0 (U) Summary

(U) The system receives MW Observation Data at the source, distributes it via multi-path communications to primary centers, removes duplicate data, request event correlation, and presents uncorrelated missile event data to users.

3.0 (U) Roles

(U) MW Data Provider, CoreC2Services, EnterpriseWorkStation

4.0 (U) Preconditions

4.1 (U) None.

5.0 (U) Primary Transactions

5.1 (U) Update Missile and Missile Event

(U) This use case begins when a MW Data Provider requests CPS to update missile and missile event.

5.1 (U) Correlate and Fuse Observations

(U) The CPS local to the data provider makes a request over n-channels to the Correlation Processor CPS to correlate and fuse observations. The Correlation Processor CPS eliminates duplicate mw observations. The Correlation Processor CPS requests Core C2 Services to correlate and fuse observations.

5.2 (U) Display Missile Event

(U) The CPS local to the data provider makes a request over n-channels to the Forward user and Correlation Processor CPSs to display missile event.

5.3 (U) Render Missile Event

The Forward User CPS requests the Forward User Enterprise Workstation to render missile event. The Correlation Processor CPS requests SA User Enterprise Workstation to render missile event.

1.0 (U) Post-Conditions

1.1 (U) Missile event is correlated and fused with other like observations and the uncorrelated event is rendered at Forward and SA User Workstations.

Figure 31 – Distribute Missile Warning Data Use Case Specification

3.6.1.2.7 Develop the Activity Diagram

From the activities and logic described in the use case specification, develop the activity diagram. This one is so simple, it is optional (Figure 32).

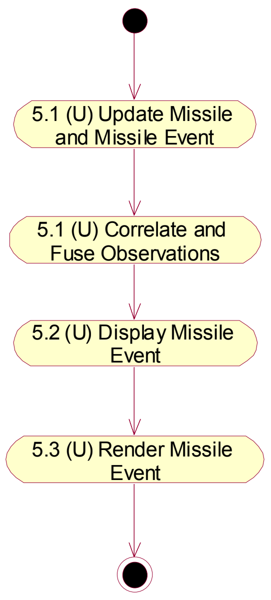

5.1 (U) Update Missile and Missile Event
5.1 (U) Correlate and Fuse Observations
5.2 (U) Display Missile Event
5.3 (U) Render Missile Event

Figure 32 – Distribute MW Data Activity Diagram

3.6.1.3 Identify CPS Services

3.6.1.3.1 Analyze CPS Use Case Event Flow to Identify Services

The last iteration of the process already identified the service transactions. All that remains is to see if there are groups of transactions that occur together as a service. In this case, it may be appropriate to group the three service transactions—Update Missile and Missile Event (:MWEventObservation), Correlate and Fuse Observations (:MWEvent Observation), and Display missile event (:MWEventObservation, out:AcknowledgeReceipt)— as well as Eliminate duplicate observations(:MWEventObservation) together into a MW Data Distribution service since they arise together at various points in the broader architecture.

3.6.1.3.2 Organize CPS Services into Interfaces

Create an interface class for the service that holds the four service transactions together (Figure 33). The realization relationships between the IDistMWData interface and the others on the figure assign the service transaction operations to the new interface.

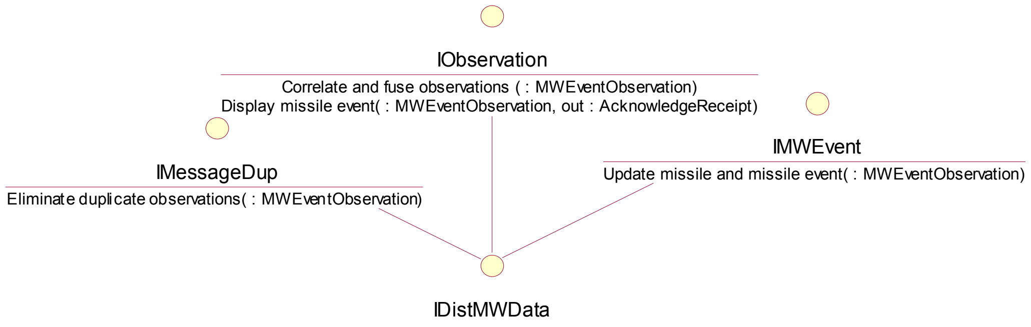

IObservation
Correlate and fuse observations (: MWEventObservation)
Display missile event(: MWEventObservation, out : AcknowledgeReceipt)
IMessageDup
Eliminate duplicate observations(: MWEventObservation)
IMWEvent
Update missile and missile event(: MWEventObservation)
IDistMWData

Figure 33 – Organizing Service Transaction into a Service—Distribute Missile Warning Data

3.6.1.4 Realize CPS Services

3.6.1.4.1 Postulate the Logical and Physical Organization of the Architecture

The Communications Processing System has four logical elements based on an analysis of its role in the larger architecture:

- CPS Protocol Converter to transcribe messages from serial communications links to network links
- CPS Client to transport and route messages
- CPS Server to analyze messages and invoke the appropriate mission processing, and
- CPS User Interface to allow for configuring the message and protocol conversion as well as checking CPS status

Figure 34 shows the postulated logical CPS architecture.

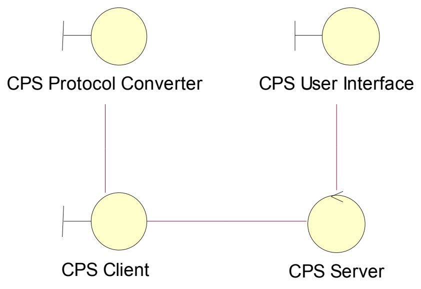


Figure 34 - CPS System Logical Architecture

The **CommCenter** that realizes CPS at the previous level, is decomposed into four physical elements: a Front End that hosts the protocol converter, a client that handles message routing and transport, a server that hosts message processing, and a workstation for the user interface (Figure 35).

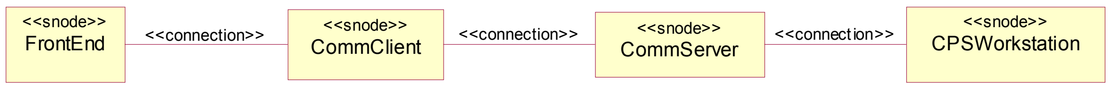


Figure 35 – Communication Center Physical Architecture Refined due to CPS

Build the diagram showing which logical elements the physical elements host (Figure 36). Note this is a partial diagram because only those element involved in the "Display Missile Event" service are in this limited realization.

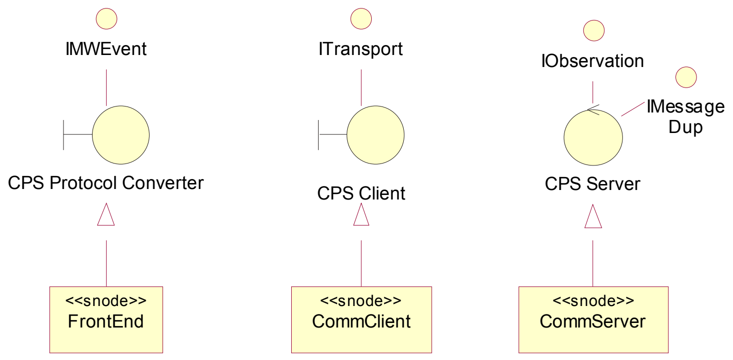


Figure 36 – Physical Elements Host Logical Elements (Partial)

3.6.1.4.2 Create CPS Logical Service Realization Diagram and Service Specification

For the Display Missile Event service transaction, divide the responsibilities among the CPS logical elements to create the tabular service specification and equivalent realization sequence diagram (Table 2 and Figure 37).

Table 2 - CPS System Service Specification for the Service Display Missile Event

	CPS Service: Display Missile Event						
Flow: Determine Baseline Collection Requirements							
#	System Element Service Name	Service Description	Logical Identifier	Logical Class	NFR	Physical Identifier	Physical Class (snode)
1	Display missile event	Ack message receipt. Convert protocol to TCP/IP and request message transport routing to process and render missile event		CPS Protocol Converter			FrontEnd
2	Process and render missile event	Validate message. Request duplicate elimination, write message to log. If not duplicate, request enterprise workstation to render missile event.		CPS Client			CommClient
3	Eliminate duplicate observations	Determine if the observation arrived earlier; respond with duplicate indication		CPS Server			CommServer

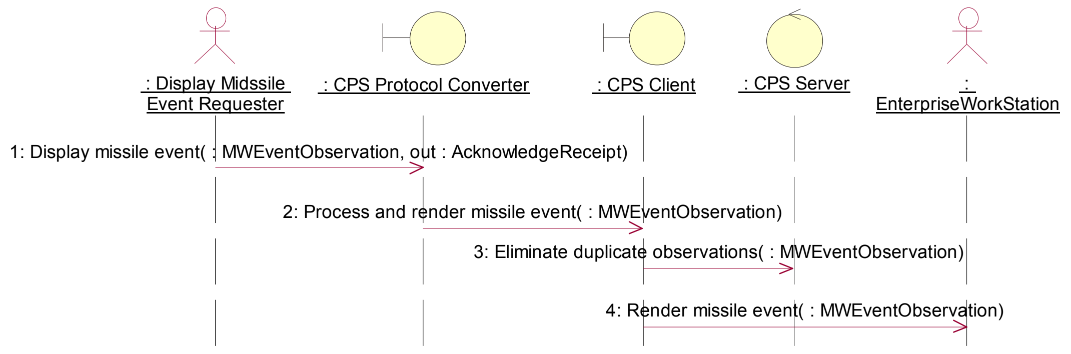

: Display Midssile Event Requester
: CPS Protocol Converter
: CPS Client
: CPS Server
: EnterpriseWorkStation
1: Display missile event(: MWEventObservation, out : AcknowledgeReceipt)
2: Process and render missile event(: MWEventObservation)
3: Eliminate duplicate observations(: MWEventObservation)
4: Render missile event(: MWEventObservation)

Figure 37 - CPS System Logical Realization Diagram for the Service Display Missile Event

Build the equivalent physical realization sequence diagram using the hosting physical elements (Figure 38).

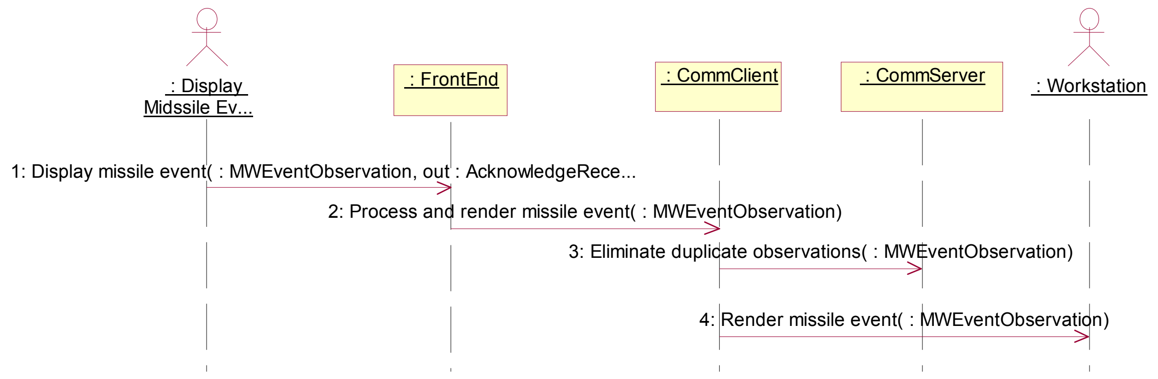

: Display Midssile Ev...
: FrontEnd
: CommClient
: CommServer
: Workstation
1: Display missile event(: MWEventObservation, out : AcknowledgeRece...
2: Process and render missile event(: MWEventObservation)
3: Eliminate duplicate observations(: MWEventObservation)
4: Render missile event(: MWEventObservation)

Figure 38 - CPS System Physical Realization Diagram for the Service Display Missile Event

This completes the example iteration of the process. To finish the process, construct the realization for the remaining service transactions assigned to CPS during the last iteration.

4.0 Develop Systems View Products

The architecture development process creates system view architectural data needed to build the SV-1 through 7, 10c and 11. The following sections contain tables with three columns—DoDAF Data Element, Method Element, and explanation. The DoDAF Data elements are those items from DoDAF Volume 2, appearing at the end of each product section, that support integrated architecture (these items bear an asterisk in the CADM tables in volume 2). The Method Element references diagrams or subsections from section four above that render the architecture data identified in the first column. The explanation describes how the data element is captured by the method element. Thus, the tables illustrate how the architectural data needed to build the system view products is captured by the method. All of the data cited can be extracted from the Rational Rose tool and used to build the tables or diagrams that comprise the system view products.

The following subsections present excerpts from volume 2 describing the relevant system view products and present the tables showing how the supporting architectural data is captured.

4.1 SV-1 Systems Interface Description

Depicts systems nodes and the systems resident at these nodes to support organizations/human roles represented by operational nodes of the Operational Node Connectivity Description (OV-2).

Identifies the interfaces between systems and systems nodes. Can also identify interfaces that cross-organizational boundaries (key interfaces). Some systems can have numerous interfaces.

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.

Table 3 – SV-1 to Method Elements Trace

DODAF Data Element	**Method Element**	**Explanation**
Systems Node Name Description	Figure 19 - Physical Architecture Figure 35 – Communication Center Physical Architecture Refined due to CPS	The classes on the physical architecture diagrams represent system nodes. To build the SV-1, choose the level of system refinement suitable to the purpose for the diagram and use the system nodes at that level as the basis.
System Name Description	Figure 18 - Logical Architecture	The classes on the logical diagram represent systems. Choose the level of granularity appropriate for the diagram and use the logical elements as the systems. The description is in the documentation field.
Interface Name Description Endpoints	Figure 19 - Physical Architecture Figure 35 – Communication Center Physical Architecture Refined due to CPS	Physical interfaces are represented by connections on the physical architecture diagram. The name is the name of the line and end points are depicted on the diagram for connections.
Operational Node	Figure 7 - Use Case Diagram	Operational nodes are associated with roles on the use case diagram via aggregation associations.
Needline	Reference will be a Rose Communications Diagram	Needlines connect operational nodes that exchange information. These are represented on communication diagrams in UML. The current architecture uses Popkin System Architect to render these diagrams on a mission basis.
Standard	Figure 20 – Connection Standards	Standards that apply to physical interfaces are held in the documentation field of the connection associations.
Systems Node Contains System Node System names	Figure 23 – Interface Realizations	The realization relationships on these diagrams show what systems are hosted by a system node.

DODAF Data Element	Method Element	Explanation
System Contains (Sub)-System(s)* System name (Sub)-System(s) Name	Figure 18 - Logical Architecture Figure 34 - CPS System Logical Architecture	As the architecture development process is recursively applied, the logical system elements are further divided into sub-systems. The figures illustrate the subsystems of the Communication Processing System.
Systems Node Supports Operational Node Systems Node Name Operational Node Name	Figure 7 - Use Case Diagram Figure 25 - Physical Service Realization (magnified for readability)	The use case diagram shows the association of operational nodes to actors who invoke services from system nodes as shown on the physical realization diagram. One or more actors on the use case diagram may play an actor on the realization diagram.
Interface Depicts Automated Portion of Needline Interface Name Needline Name	Figure 19 - Physical Architecture Figure 18 - Logical Architecture Figure 9 - SOS Diagram for the Maintain Relevant MW Information Use Case Figure 7 - Use Case Diagram	Interfaces are represented by connecting lines on the physical/logical diagrams. These lines are generated by service transactions in the original iteration of the process. The service transactions, in some cases, implement Information Exchanges on the System Operational Sequence. When they do (currently indicated by attached notes on the messages), the needline(s) between the operational nodes associated with the roles (shown on the use case diagram) are automated.
System Uses Standard System Name Standard Name	Figure 18 - Logical Architecture Figure 19 - Physical Architecture Figure 23 – Interface Realizations Figure 20 – Connection Standards	The documentation field of the system element may reference applicable standards. Also, interfaces are used by system nodes in the physical architecture. These are related to systems in in the logical architecture through the interface realization diagrams. Standards for interfaces are held in the associated documentation field for the connection association.

4.2 SV-2 Systems Communications Description

Depicts pertinent information about communications systems, communications links, and communications networks. SV-2 documents the kinds of communications media that support the systems and implement their interfaces as described in SV-1. Thus, SV-2 shows the communications details of SV-1 interfaces that automate aspects of the needlines represented in OV-2. Can be used to document how interfaces (described in SV-1) are supported by physical media. Documents the specific communications links or communications networks (e.g., Intelink or Joint Worldwide Intelligence Communications System [JWICS]) and the details of their configurations through which systems interface. Contains a detailed description of how each SV-1 interface is implemented (e.g., composing parts of the implemented interface including communications systems, multiple communications links, communications networks, routers, and gateways).

Table 4 – SV-2 to Method Element Trace

DODAF Data Element	**Method Element**	**Explanation**
Systems Node	Figure 19 - Physical Architecture	System nodes are identified as classes on the physical architecture.
System	Figure 18 - Logical Architecture	Systems are identified as classes on the logical architecture diagram.
Communications System Name Description Communications System Standard, Protocols Supported	Figure 19 - Physical Architecture Figure 35 – Communication Center Physical Architecture Refined due to CPS	The linked documentation of the association between the system nodes contains the relevant standards.
Communications Link Name Communication Standard, Protocols Supported Capacity Infrastructure Technology Endpoint: Systems Node/System Name	Figure 19 - Physical Architecture Figure 35 – Communication Center Physical Architecture Refined due to CPS	This shows inter and intra-nodal connections. Reference documentation would contain the relevant standards.
Communications Path Name Description Endpoint Systems Node/System Name Number of Communications Links	Figure 19 - Physical Architecture Figure 35 – Communication Center Physical Architecture Refined due to CPS	This shows inter and intra-nodal connections. Reference documentation would contain the relevant standards.
Standard	Not addressed in this paper.	TV-1 standards are associated with communications systems via hyperlinked table.
Communications Path Contain Communications Link(s) Communications Path Name Communications Link Communications Link Position in	Figure 19 - Physical Architecture Figure 25 - Physical Service Realization (magnified for readability)	A communications path may be associated with a service realization. The physical architecture identifies the communication links of the system as connections. These are traversed among physical

DODAF Data Element	Method Element	Explanation
Communications Path		nodes by the realizing service transactions on the physical realization diagram. The position of the links on the path is driven by the order of link transits on that diagram as well.
Communications Path Implements Interface Communications Path Name Interface Name	Figure 23 – Interface Realizations Figure 25 - Physical Service Realization (magnified for readability)	A communications path may be associated with a service realization. The physical architecture identifies the communication links of the system as connections. These are traversed among physical nodes by the realizing service transactions on the physical realization diagram. Link interfaces are described in the documentation fields for the link.
Communications System Uses Standard Communications System Name Standard Name	Not addressed in this paper.	TV-1 standards are associated with communications systems via hyperlinked table.
System Hardware/Software Item Uses Standard System Hardware/Software Item Name Standard Name	Not addressed in this paper.	TV-1 standards are associated with communications systems via hyperlinked table.
Communications Link Uses Standard Communications Link Name Standard Name	Not addressed in this paper.	TV-1 standards are associated with communications systems via hyperlinked table.

4.3 SV-3 Systems-Systems Matrix

Provides detail on the interface characteristics described in SV-1 for the architecture. Supports a rapid assessment of potential commonalities and redundancies (or, if fault-tolerance is desired, the lack of redundancies). Can be organized in a number of ways (e.g., by domain, by operational mission phase). Can be a useful tool for managing the evolution of systems and system infrastructures, the insertion of new technologies/functionality, and the redistribution of systems and processes in context with evolving operational requirements.

Table 5 – SV-3 to Method Element Trace

DODAF Data Element	Method Element	Explanation
Code Legend Type Name Description		Not modeled

DODAF Data Element	Method Element	Explanation
Code Legend Code Type Name Description		Not modeled
System	Figure 18 - Logical Architecture	Systems are defined on the logical architecture diagram.
Interface Entry Name Description	Figure 19 - Physical Architecture	Interfaces appear as connections on the physical architecture diagram.
Interface	Figure 19 - Physical Architecture	Interfaces appear as connections on the physical architecture diagram.
Code Legend Represents Characteristics of Interface Entry Code Legend Code Interface Entry Name		Not modeled
Code Legend is Categorized by Code Legend Type Code Legend Name Code Legend Type Name		Not modeled
Interface Entry Details Interface Characteristics Interface Entry Name Interface Name	Figure 19 - Physical Architecture	Details interfaces defined and documented as connections.

4.4 SV-4 Systems Functionality Description

Documents system functional hierarchies and system functions, and the system data flows between them. Develops a clear description of the necessary system data flows that are input (consumed) by and output (produced) by each system. Ensures that the functional connectivity is complete (i.e., that a system's required inputs are all satisfied). Ensures that the functional decomposition reaches an appropriate level of detail. Scope of this product may be enterprise wide, without regard to which systems perform which functions, or it may be system specific. While the Operational Activity Model (OV-5) or business-process hierarchies correlate with the system functional hierarchy of SV-4, it need not be a one-to-one mapping, hence, the need for the Operational Activity to Systems Function Traceability Matrix (SV-5), which provides that mapping.

Table 6 – SV-4 to Method Element Trace

DODAF Data Element	Method Element	Explanation
System Function	Figure 23 – Interface Realizations	System functions are represented as operations of the system, defined on realization diagrams.
External System Data Source/Sink Name Description	Figure 9 - SOS Diagram for the Maintain Relevant MW Information Use Case	Name and description is contained in the documentation field of the external actor.

DODAF Data Element	Method Element	Explanation
System Data Flow Name Description From System: Function/External System Data Source/System Data Repository To System: Function/External System Data Sink/System Data Repository	Figure 22 –Logical Service Realization Diagram (magnified for readability)	Persistence mechanisms can be identified using the Logical Service Realization Diagram. Dataflow among the logical system elements is via service dialog. The system functions are represented by the operations on the system elements. The dataflow is represented by the parameters in the operation invocation. External datasinks and source are represented by actors.
System (or Subsystem)	Figure 18 - Logical Architecture	Systems/subsystems are defined on the logical architecture diagram.
External Sink is Sink for System Data Flow External Sink Name System Data Flow Name	Figure 6 - Context Diagram Figure 22 –Logical Service Realization Diagram (magnified for readability)	Actors on the sequence and class diagrams represent the external sink. Dataflow on the context is addressed by the <<send>> and <<receive>> stereotype. Dataflow is represented by operation invocation on the sequence diagram.
External Source is Source for System Data Flow External Source Name System Data Flow Name	Figure 6 - Context Diagram Figure 22 –Logical Service Realization Diagram (magnified for readability)	Actors on the sequence and class diagrams represent the external source. Dataflow on the context is addressed by the <<send>> and <<receive>> stereotype. Dataflow is represented by operation invocation on the sequence diagram.
System Function Produces System Data Flow System Function Name System Data Flow Name	Figure 22 –Logical Service Realization Diagram (magnified for readability)	System functions are represented as operations of the system. Their parameters represent information flow.
System Function Processes (Consumes) System Data Flow System Function Name System Data Flow Name	Figure 22 –Logical Service Realization Diagram (magnified for readability)	System functions are represented as operations of the system. Their parameters represent information flow.
System Function is Allocated to System System Function Name System Name	Figure 22 –Logical Service Realization Diagram (magnified for readability)	System functions are allocated to systems via the arrowhead on the message line.

DODAF Data Element	Method Element	Explanation
System Function Uses Standard System Function Name Standard		This method conforms to the object-oriented coding conventions in determining system function names.

4.5 SV-5 Operational Activity to Systems Function Traceability Matrix

An explicit link between the OV and SV. A specification of the relationships between the set of operational activities applicable to an architecture and the set of system functions applicable to that architecture. Identifies the transformation of an operational need into a purposeful action performed by a system. Can be extended to depict the mapping of capabilities to operational activities, operational activities to system functions, system functions to systems, and thus relates the capabilities to the systems that support them. Correlates capability requirements that would not be satisfied if a specific system is not fielded to a specific DoD unit. Given that operational nodes do not map one-to-one to systems nodes, it is natural that operational activities do not map one-to-one to system functions.

Table 7 – SV-5 to Method Element Trace

DODAF Data Element	Method Element	Explanation
Operational Activity	Figure 15 - Use Case Activity Diagram, Specifying Baseline Collection Fragment	Operational activities appear as activities on activity diagrams.
System Function	Figure 9 - SOS Diagram for the Maintain Relevant MW Information Use Case	System functions appear as messages on sequence diagrams as well as operations bound to interfaces and realized by systems.
System Function Implements Operational Activity System Function Name Operational Activity Name	Figure 9 - SOS Diagram for the Maintain Relevant MW Information Use Case Figure 15 - Use Case Activity Diagram, Specifying Baseline Collection Fragment	One or more system functions support an activity as indicated by the notes attached to messages on the sequence diagram that reference activities on the activity diagram.

4.6 SV-6 Systems Data Exchange Matrix

Specifies the characteristics of the system data exchanged between systems. Focuses on *automated* information exchanges (from OV-3) that are implemented in systems. System data exchanges express the relationship across the three basic architecture data elements of an SV (systems, system functions, and system data flows) and focus on the specific aspects of the system data flow and the system data content. Relates to, and grows out of, OV-3. The operational characteristics for the OV-3 information exchange are replaced with the corresponding system data characteristics. SV-1 graphically depicts system data exchanges as interfaces that represent the automated portions of the needlines. The implementation of SV-1 interfaces is described in SV-2 (if applicable). The system data exchanges documented in SV-6 trace to the information exchanges detailed in OV-3 and constitute the automated portion(s) of the OV-3 information elements

Table 8 – SV-6 to Method Element Trace

DODAF Data Element	Method Element	Explanation
System Data Exchange Identifier Name Sending System Name Sending System Function Name Receiving System Name Receiving System Function Name Transaction Type Triggering Event Periodicity Timeliness Throughput Size Classification Classification Caveat	Figure 22 –Logical Service Realization Diagram (magnified for readability) Figure 37 - CPS System Logical Realization Diagram for the Service Display Missile Event	An identifier and name for the system data exchange have not been created at this point, but would be created at the time of SV-6 development. The sending and receiving systems and the receiving system function names are easily identifiable from these diagrams. A sending system function name is identified by looking at the closest previous message coming into to the sending system lifeline for the exchange in question. (“Out” or “inout” designations on system data change the sense of direction of the data flow.) The triggering event is the sending function. Timeliness and throughput could be derived from the NFRs Size may be identified through analysis of parameters. Classification will be derived from either the related IER or the appropriate classification guide at the time of SV-6 production.
System Data Element Identifier Name Content Format Type Media Type Accuracy Units of Measurement System Data Standard		Held in the systems data model, which is reflected in the parameters of the operations assigned to systems.
Information Exchange	Figure 9 - SOS Diagram for the Maintain Relevant MW Information Use Case Figure 22 –Logical Service Realization Diagram (magnified for readability)	On the Logical Service Realization Diagram, when the service being realized is tied to an IE (from the SOS diagram), all subsequent exchanges among system elements are associated with the IE.

DODAF Data Element	**Method Element**	**Explanation**
Interface	Figure 23 – Interface Realizations Figure 19 - Physical Architecture	Interfaces are associated with the connections between snodes, assuming the systems are in different snodes
System		Addressed above in System Data Exchange Element
System Function		Addressed above in System Data Exchange Element
System Data Flow		Addressed above in System Data Exchange Element
Triggering Event		Addressed above in System Data Exchange Element
Standard		Addressed in SV-1
System Data Exchange is Carried by Interface System Data Exchange Identifier Interface Name	Figure 22 –Logical Service Realization Diagram (magnified for readability) Figure 19 - Physical Architecture Figure 23 – Interface Realizations	System data exchanges are represented as parameters of a message on the realization diagram. They flow across connections on the physical architecture that represent interfaces. Logical interfaces are associated with systems and system nodes via the interface realization diagram.
System Data Exchange Automates Information Exchange System Data Exchange Identifier Information Exchange Identifier	Figure 22 –Logical Service Realization Diagram (magnified for readability) Figure 9 - SOS Diagram for the Maintain Relevant MW Information Use Case	The Logical Realization Diagram is the diagram, or traces to the diagram, that realizes an SR found on the SOS linked to an IE by a note.
System Data Exchange has Sending System System Name System Data Exchange Identifier	Figure 22 –Logical Service Realization Diagram (magnified for readability)	In general, the system data exchange tail connects to the sending system. (“Out” or “inout” designations on system data change the sense of direction of the data flow.)
System Data Exchange has Receiving System System Name System Data Exchange Identifier	Figure 22 –Logical Service Realization Diagram (magnified for readability)	In general, the system data exchange head connects to the sending system. (“Out” or “inout” designations on system data change the sense of direction of the data flow.)

DODAF Data Element	Method Element	Explanation
System Data Exchange has Sending System Function System Function Name System Data Exchange Identifier	Figure 22 –Logical Service Realization Diagram (magnified for readability)	A sending system function name is identified by looking at the closest previous message coming into to the sending system lifeline for the exchange in question. ("Out" or "inout" designations on system data change the sense of direction of the data flow.)
System Data Exchange has Receiving System Function System Function Name System Data Exchange Identifier	Figure 22 –Logical Service Realization Diagram (magnified for readability)	Generally, the enclosing operation is the receiving system function. If the system data is marked "out," the receiving function is the previous inbound arrow on the system at the tail of the message.
System Data Element is Exchanged Via System Data Exchange System Data Element Identifier System Data Exchange Identifier	Figure 22 –Logical Service Realization Diagram (magnified for readability)	The system data element is associated with the systems data exchange when the SV-6 entry is created.
System Data Exchange has Triggering Event Event Name System Data Exchange Identifier	Figure 22 –Logical Service Realization Diagram (magnified for readability)	The triggering event is the sending function.
System Data Exchange Conforms to Security Standard System Data Exchange Identifier Standard Name		When such standards exist, they can be captured in the documentation field.

4.7 SV-7 Systems Performance Parameters Matrix

Specifies the quantitative characteristics of systems and system hardware/software items, their interfaces (system data carried by the interface as well as communications link details that implement the interface), and their functions. Communicates which characteristics are considered most crucial for the successful achievement of the mission goals assigned to the system. Builds on SV-1, SV-2, SV-4, and SV-6 by specifying performance parameters for systems and system hardware/software items and their interfaces (defined in SV-1), communications details (defined in SV-2), their functions (defined in SV-4), and their system data exchanges (defined in SV-6).

Table 9 – SV-7 to Method Element Trace

DODAF Data Element	**Method Element**	**Explanation**
Performance Parameter Set Name Number of Performance Parameters in Set	3.1.2 Create Supplementary Specification	The supplementary specification holds the set of operation performance requirements from which the parameter sets are derived.
Performance Parameter Type Name Description	3.1.2 Create Supplementary Specification	The types of parameters are defined in the supplementary specification.
Time Time Identifier Timestamp	3.1.2 Create Supplementary Specification	The timelines for reaching specified performance levels are defined in the supplementary specification.
Performance Measurement Performance Parameter Type Name Time Identifier Measured Value		Not addressed in this method. Developed as part of system test conduct.
Required Performance Parameter Range Performance Parameter Name Performance Parameter Type Name Time Identifier Threshold Value	Figure 12 - Service Specification for Update Missile and Missile Event	Allocated through the Service Specification at various levels of development.
System	Figure 18 - Logical Architecture	Systems are defined on the logical architecture diagram.
System has Performance Parameter Set System Name Performance Parameter Set Name	Table 1 - Service Realization Table Figure 12 - Service Specification for Update Missile and Missile Event	Performance is allocated across service transactions in the service realization tables as non-functional requirements. These are then captured as part of the service documentation field.
Performance Parameter Set Includes Performance Measurement Performance Parameter Set Name Performance Measurement Name	Figure 12 - Service Specification for Update Missile and Missile Event	The measurement is associated with parameters in the service specification in the documentation field.

DODAF Data Element	Method Element	Explanation
Performance Parameter Set Includes Required Performance Parameter Range Performance Parameter Set Name Performance Parameter Range Identifier	Figure 12 - Service Specification for Update Missile and Missile Event	Performance allocation is resides in the service specification documentation field.
(Required Performance Range Depends on Timed Technology Forecast) Timed Technology Forecast Name Performance Parameter Range Identifier		Not addressed in this method. Developed as part of system test conduct
(Required Performance Range Supports Evolution Milestone) Performance Parameter Range Identifier Milestone Name		Not addressed in this method. Developed as part of system test conduct

4.8 SV-10c Systems Event-Trace Description

May reflect system-specific aspects or refinements of critical sequences of events described in the Operational View. Valuable for moving to the next level of detail from the initial systems design, to help define a sequence of functions and system data interfaces, and to ensure that each participating system, system function, or human role has the necessary information it needs, at the right time, in order to perform its assigned functionality. Different scenarios should be depicted by separate diagrams. SV-10c can be used by itself or in conjunction with a SV-10b to describe dynamic behavior of system processes or system function threads.

Table 10 – SV-10c to Method Element Trace

DODAF Data Element	Method Element	Explanation
Lifeline Lifeline Name Description	Figure 21 - Logical Service Realization Diagram	System and role lifelines appear on the sequence diagram.
Event	Figure 21 - Logical Service Realization Diagram	Events are represented by messages on the sequence diagram.
Event Time	Figure 21 - Logical Service Realization Diagram	Event time can be represented by its vertical position on the sequence diagram.
Human Role	Figure 21 - Logical Service Realization Diagram	Human roles are represented by role instances on the sequence diagram.
System	Figure 21 - Logical Service Realization Diagram	Systems are represented by system instances on the sequence diagram.

DODAF Data Element	Method Element	Explanation
Action	Figure 21 - Logical Service Realization Diagram	Actions are indicated by operations invoked by messages on the sequence diagram.
Lifeline Represents a Human Role, System, or System Function Lifeline Name Human Role, System, or System Function Name	Figure 21 - Logical Service Realization Diagram	Lifelines associated with roles on the diagram can represent human roles.
Event Starts at Time	Figure 21 - Logical Service Realization Diagram	Event start times correspond to message vertical position on the diagram.
Event Ends at Time	Figure 21 - Logical Service Realization Diagram	Event end times correspond to message vertical position on the diagram.
Event is Associated with Action	Figure 21 - Logical Service Realization Diagram	Evnts invoke actions via the messages on the diagam.
Event is Related to System/Function Action Name System Function Name Relationship Description	Figure 21 - Logical Service Realization Diagram	The arrow head associates an operation with a system.
Event Maps to System Data Produced or Consumed by System/Function	Figure 21 - Logical Service Realization Diagram	Parameters on the meassage line associate system data with events. The operation then associates the data with system functions.
Event is Associated with System Data Exchange Event Name System Data Exchange Identifier	Figure 21 - Logical Service Realization Diagram	Meassages represent system data exchanges throughthe associated parameters.
Event Originates from Lifeline	Figure 21 - Logical Service Realization Diagram	Message arrow tail represents the originating lifeline.
Event Terminates at Lifeline	Figure 21 - Logical Service Realization Diagram	Message arrow head represents the originating lifeline.

4.9 SV-11 Physical Schema

An implementation-oriented data model that is used in the Systems View to describe how the information requirements represented in Logical Data Model (OV-7) are actually implemented.

Defines the structure of the various kinds of system data that are utilized by the systems in the architecture. Provides as much detail as possible on the system data elements exchanged between systems, thus reducing the risk of interoperability errors. Provides system data structures for use in the system design process, if necessary.

Table 11 – SV-11 to Method Element Trace

DODAF Data Element	Method Element	Explanation
Physical Schema Model Name Description Number of Constituent Models	Not demonstrated in the paper	However, the packaging of the model divides the physical schema into constituent models
Physical Schema Model Contains System Data/Message Model, File Structured Model, ERD Model, DDL Model, or OO Class Model Physical Schema Model Name System data/Message Model, File Structured Model, ERD Model, DDL Model, or OO Class Mode Name	Figure 21 - Logical Service Realization Diagram	Will create a class model from the parameters passed among systems elements as depicted in the Logical Service Realization Diagrams
Logical Model Maps to Physical Schema Model Logical Model Name Physical Schema Model Name Reference to Mapping Document	Figure 21 - Logical Service Realization Diagram	Logical data elements are associated with system data elements via the associated realization diagrams.

5.0 Way Ahead

From the previous sections, one can see we have developed a process for creating integrated architecture at increasing level of detail. However, there are several areas of improvement we expect to address in subsequent versions of the paper:

5.1 Add detail to the description of the method

Section 4 of this first version of the paper will be expanded over time to provide more detail about the architecture development process, and to incorporate improvements as this approach is exercised. The AFSPC architecture team will test this concept further in an actual program, CCIC2S or another within in AFSPC's portfolio. Sections 4.4 and 4.5 of the paper will evolve most rapidly as the team gains experience using the Systems View products

5.2 Provide examples of DoDAF SV products

In Section 5, the paper traces the data produced by this architecture development approach to the DODAF SV data elements. In subsequent versions of the paper, the team will provide the actual SV products

5.3 Research tools needed to either move to a UML 2.0 compliant tool or create workarounds in the current application, IBM Rational Rose

Rational Rose is compliant with the UML version 1.4. IBM Rational is not performing significant development of Rose, choosing instead to invest most of its effort in its Rational Software Architect and Modeler products. With the adoption of the UML 2.0 superstructure by the OMG, migration to a UML 2.0 compliant tool will eventually be necessary to exploit new features of the language. For instance, using Interaction Overview diagrams in place of sequence and activity diagrams would create a more effective representation of desired operational and system behavior

Readers interested in implementing this methodology or contributing to its further development should contact the CCIC2S Chief Architect, Ms. Joanne Schissel, GS-14, HQ AFSPC/DRN at 719-554-5057 for further information.

Appendix A: RUP SE Process Outline

1.0 Enterprise Level (level 0)

This level captures the information relevant only to the top-most item of concern (the subject of the model). The elements that make up this subject are in the lower levels.

1.1 Black Box Views

These views represent the subject as an opaque black box. We can observe all of its external interactions, but not any of the implementation and realization details.

1.1.1 Context Diagram

The purpose of this diagram is to document the scope of the subject and anything that is external to the subject represented as Actors. Thus, an Actor must perform responsibilities that are outside the scope of the subject. Represented on a UML class diagram, information exchanges are shown between the subject and the Actors. The subject is represented as a class on the diagram with a **<<proxy>>** stereotype, indicating this is not the class that implements the subject, but a placeholder for the elements that compose it. The Actors relate directly to the subject using associations. The items exchanged are related only to the Actors with associations. These associations are stereotyped **<<send>>** or **<<receive>>** to indicate if the Actor sends or receives the item.

1.1.2 Use Case Diagram

This diagram captures the purpose for the Actor's interactions with the subject – use case diagrams represent these interactions. The sum of all the use cases represents all the functionality required of the subject with the exception of common functionality that is applicable to all use cases – the Supplementary Specification captures this functionality.

1.1.3 Supplementary Specification

This document captures all of the non-functional requirements (NFR) on the subject. It also includes any design constraints and functionality that is common across all use cases.

1.1.4 Use Case Specifications

For each of the use cases defined in the use case diagram, a use case specification is developed. The three elements of the specification follow.

1.1.4.1 Text Document

This document captures basic information about the use case such as the author, references, actors involved, preconditions, post conditions, special requirements, and flows. The special requirements are derived from requirements in the supplementary specification that can be specifically applied to this use case. The flows describe the flow of events of the interactions between the subject and the actors to accomplish the behavior. Multiple flows (if applicable) account for the different conditions under which the subject must perform, and the resulting different outcomes of the interaction. If any special requirements are applicable to a flow, then the requirement is budgeted across the steps in the flow. For example, if the flow has a ten-second requirement, then some portion of the time is budgeted to each step in ensure that the entire flow will meet the overall requirement.

1.1.4.2 Activity Diagram

Activity diagrams are an optional part of the use case specification based on RUP SE. However, CCIC2S operators find them useful to describe the conditional logic of the flows. If there are multiple conditions, then making sure all paths are covered can be non-trivial and their depiction is useful to the developer.

1.1.4.3 Sequence Diagrams

A sequence diagram represents a single use case flow. Multiple sequence diagrams for a use case represent all the significant flows of the use case. While the sequence diagram represents essentially the same information in the text document for one flow, this representation adds formality and visualization to the free flowing text of the document. The visual representation ensures that there is a continuous stimulus flowing through the interaction, and clearly illustrates what functionality is being requested of who, and by whom. UML messages represent the interactions between the actors and the subject. Discussed later, services replace the messages stimulating the subject.

1.1.5 Define Services

After creating the use case specifications, the architect identifies the services the subject requires to offer to support the interactions. By looking at all the messages requesting functionality of the subject, on all the sequence diagrams, across all the use cases, the architect collects a set of candidate services. By analyzing these candidate services, they are re-factored and refined into a coherent set of services offered by the subject. Initially, these services are operations on the subject <<proxy>> class.

1.1.5.1 Service Specification

For each of the identified services, a service specification is developed. The service specification is a text document (or documentation field in the model) that captures basic information such as preconditions, post conditions, references, budgeted special requirements, etc.

1.1.5.2 Interface Classes

With the set of services defined, it is useful to group them as operations on interface classes. This makes the number of services more manageable and enables the joint realization of interface classes done in the white box. The mechanism establishes a realization relationship from the subject proxy class to the interface class, and then provides a way to move the service operation from the subject proxy class to the interface class. The result – all services are still visible as operations on the subject proxy class.

1.1.5.3 Refine Sequence Diagrams

Revise the sequence diagrams to change operations on the subject proxy class to messages stimulating the subject collaboration changing the messages stimulating the subject into one of the service operations on the subject class.

1.2 White Box Views

These views represent how the elements that compose the subject collaborate to satisfy the required capabilities specified in the black box views.

1.2.1 Architecture Diagrams

Architecture diagrams capture the top-level architecture of the subject. Multiple architecture views represent different stakeholder viewpoints. At a minimum, the architect should create a logical and physical view. Create additional architecture diagrams for process, worker, data, etc.

1.2.1.1 Logical Architecture

The logical architecture identifies the coherent, loosely coupled, abstractions of capability that collaborate to satisfy the top-level required capabilities. Classes on class diagrams represent the elements. Depict associations between the architectural elements to show communication relationships.

1.2.1.2 Physical Architecture

The physical architecture identifies the abstract locations identifying functionality. Class diagram represent these entities as classes with a <<locality>> stereotype. Represent the physical transmission channels with associations stereotyped <<connection>>.

1.2.2 Service Realization

For each service offered by the subject, a service realization is developed. The three elements of the service realization follow.

1.2.2.1 Text Document

The best format for representing the required information is a text document table, or a spreadsheet workbook – possibly in RequisitePro. A separate table (or spreadsheet) captures the flow of the service. The table contains rows for the steps of the flow, and columns to capture information about the step. One of the columns captures the portion of the special requirements (if any) budgeted to the step. The other columns capture the functionality the step represents, which logical element performs the functionality, and at which locality the functionality is performed. If additional architectures are important, then additional columns capture the appropriate information about the step.

1.2.2.2 Activity Diagram

Activity diagrams are an optional part of the service realization, but important for enterprise understanding. They are useful to describe the conditional logic of the flows. If there are multiple conditions, then making sure all paths are covered can be non-trivial.

1.2.2.3 Sequence Diagrams

A set of sequence diagrams represent a single service flow. Use multiple sequence diagrams to represent all the significant flows of the service.

1.2.2.3.1 Logical Realization

This diagram represents the interactions between the actors and the logical architecture elements. It shows how the responsibility of the subject satisfies the collaboration of the logical elements, thus deriving requirements on these elements. At this point, messages depict the interactions between the logical elements and actors. Later, services replace the messages stimulating the elements.

1.2.2.3.2 Physical Realization

While logically, the physical realization is done immediately following the logical realization, in practice, the physical realization is delayed until after a set of logical element services have been established. The reason is that the physical realization diagram will utilize these same services. If they are created before the services exist, then duplicate entry is required. Step 2.1.5 below establishes the services offered by the logical elements.

This diagram represents the interactions between the actors and the physical architecture elements. It shows how the responsibility of the subject is distributed across the physical elements. The resulting load on the localities and the connections drives non-functional requirements on these elements.

2.0 System Level (level 1)

For each of the logical architecture elements identified above, the process is repeated with that element as the subject.

2.1 Black Box Views

The black box represents the logical architecture element.

2.1.1 Context Diagram

The context diagram shows the subject logical architecture element and all the external entities it interacts with, both enterprise actors and other logical elements.

2.1.2 Use Case Diagram

This diagram captures all the subject element’s use cases. The use cases are implemented by sequences of interactions between the subject element and its actors. Thus, by analyzing how the subject element participates in the collaborations of the service realizations at the higher level, we can identify the patterns that form its use cases. Even though the subject element’s services can be derived directly from the higher-level service realizations, it is important to capture the subject element’s use cases in order to support (at least) user documentation, test planning, and iteration planning.

2.1.3 Supplementary Specification

This document captures all the non-functional requirements derived for this logical element.

2.1.4 Use Case Specifications

For each of this logical element’s use cases, a use case specification is developed.

2.1.4.1 Text Document

This document captures basic information about the use case such as the author, references to higher-level service realizations, actors involved, preconditions, post conditions, special requirements, and flows.

2.1.4.2 Activity Diagram

The activity diagram can show the conditional logic of the use case.

2.1.4.3 Sequence Diagrams

Each sequence diagram shows a single flow of the use case. At this point, the interactions between the subject logical element and its actors are shown using messages. Later, the messages stimulating the subject logical element will be replaced by the services it offers.

2.1.5 Define Services

The subject element’s services can be identified be analyzing the higher-level service realization sequence diagrams the subject element participates in. By collecting all the messages that stimulate the subject element, the result is a set of candidate services. These are refactored and refined into a coherent set of services offered by the subject element. Initially, these services are created as operations on the subject element class.

2.1.5.1 Service Specification

For each identified service, a service specification is created to capture basic information about the service.

2.1.5.2 Interface Classes

Now it is useful to group the services into interface classes. Generally, the services are grouped by the role that would utilize the services. In addition, it is necessary to separate the services into different interface classes when the services are located on different localities. The mechanism is to first establish a realization relationship from the subject element class to the interface class, then move the service operation from the subject element class to the interface class. In addition, if the subject element realizes any of the external services offered by the higher level subject of the model, then realization relationships must be established to those interface classes. Note that if the subject element does not realize all the services in the interface class, then the services should be separated into multiple interface classes to support the realization. The result is the complete set of interface classes the subject element realizes. Next, it is necessary to establish a realization relationship from each locality where services are performed to the interface class containing those services.

2.1.5.3 Refine Sequence Diagrams

With the services available as operations on the subject element, it is now possible to revise the higher-level service Logical Realization sequence diagrams to use the services instead of just messages. In addition, the corresponding Physical Realization sequence diagrams can now be easily created using the same services.

2.2 White Box Views

Now we look at how the subject element satisfies its responsibilities.

2.2.1 Architecture Diagrams

Architecture diagrams show how the subject element is structured.

2.2.1.1 Logical Architecture Creation

Just like at the higher level, the logical architecture of the subject element captures the elements that make up the subject element.

2.2.1.2 Physical Architecture Refinement

Unlike at the higher level, the physical architecture does not belong to this logical element. However, it is still necessary to refine the physical architecture to describe the distribution of the subject element's capabilities. Therefore, working with the owners of the physical elements to be refined, the team responsible for this subject element identifies the needed distribution elements. They are captured as a decomposition of the higher-level locality. A new class diagram is created to illustrate the decomposition. The distribution elements are still localities, just at a greater level of refinement.

2.2.2 Service Realization

Just as at the higher level, we create a service realization for each offered service.

2.2.2.1 Text Document

The table format is the same as at the higher level. The logical elements used are the elements of the logical architecture of this subject element. The physical elements used are the refined localities.

2.2.2.2 Activity Diagram

Activity diagrams serve the same purpose as previously, to understand the conditional logic of the service.

2.2.2.3 Sequence Diagrams

Again, just as above, we create a set of sequence diagrams for each service flow.

2.2.2.3.1 Logical Realization

The logical realization shows how the subject element's components collaborate to satisfy its requirements.

2.2.2.3.2 Physical Realization

The physical realization shows how the same collaboration is distributed across the refined localities.

3.0 Lower Levels (level n)

The process continues, repeating all of the System Level (level 1) actives for each of the newly identified logical elements. The approach continues in this fashion until it is possible to realize the logical elements in hardware, software, or people.

Appendix B: References

Rational Unified Process for Systems Engineering, Part I: Introducing RUP SE Version 2.0, by Murray Cantor, IBM Rational, http://www.therationaledge.com/content/aug_03/f_rupse_mc.jsp

Rational Unified Process for Systems Engineering, Part II: System Architecture, Murray Cantor, IBM Rational, http://www.therationaledge.com/content/sep_03/f_m_systemarch_mc.jsp

Rational Unified Process for Systems Engineering, Part III: Requirements Analysis and Design, Murray Cantor, IBM Rational, http://www.therationaledge.com/content/oct_03/f_rupse_mc.jsp

DoD Architecture Framework Version 1.0, DoD Architecture Framework Working Group, 15 August 2003

UML 2.0 Superstructure Specification, August 29, 2004 Interim Drop, www.omg.org

U.S. Office of Management and Budget, Circular No. A-130, "Management of Federal Information Resources", (OMB, 2000). Circular No. A-130.

DoD Architecture Framework Working Group, DoD Architecture Framework, Volume I: Definitions and Guidelines, Version 1.0, http://www.defenselink.mil/nii/doc/docArchive.html#NII, created 9 February 2004, accessed 16 May 2005.

U.S. Office of Management and Budget, "Federal Enterprise Architecture," http://www.whitehouse.gov/omb/egov/a-4-srm.html, accessed 6 July 2005.

Ring, Nicholson, and Thilenius, "An Activity-Based Methodology for Development and Analysis of Integrated DoD Architectures," (The MITRE Corporation, 2003).

Appendix C: Rose Tricks

To make an operation from a message text, do not select the message line [that will create instead an operation named opname()]. Instead, select the text itself and right-click to new-operation.

An operation O that is in an implementation class/system S, but which should be in an interface I which the implementation class S realizes, can be moved there without severing the operation binding by dragging o in the browser from S to I, provided that the realization relationship between S and the interface I already exists.

Appendix D: DoDAF Products

Table 12 shows the architecture products defined in the DoD Architecture Framework. The applicable view denotes whether the product belongs to the Operational View, Systems View or Technical View. Additionally, two products the AV-1, Overview and Summary Information, and the AV-2, Integrated Dictionary, provide information relevant to the entire architecture by setting the scope and context of the architecture.

Table 12 - List of DoDAF Products

Applicable View	Framework Product	Framework Product Name	General Description
All Views	AV-1	Overview and Summary Information	Scope, purpose, intended users, environment depicted, analytical findings
All Views	AV-2	Integrated Dictionary	Architecture data repository with definitions of all terms used in all products
Operational	OV-1	High-Level Operational Concept Graphic	High-level graphical/textual description of operational concept
Operational	OV-2	Operational Node Connectivity Description	Operational nodes, connectivity, and information exchange needlines between nodes
Operational	OV-3	Operational Information Exchange Matrix	Information exchanged between nodes and the relevant attributes of that exchange
Operational	OV-4	Organizational Relationships Chart	Organizational, role, or other relationships among organizations
Operational	OV-5	Operational Activity Model	Capabilities, operational activities, relationships among activities, inputs, and outputs; overlays can show cost, performing nodes, or other pertinent information
Operational	OV-6a	Operational Rules Model	One of three products used to describe operational activity—identifies business rules that constrain operation
Operational	OV-6b	Operational State Transition Description	One of three products used to describe operational activity—identifies business process responses to events
Operational	OV-6c	Operational Event-Trace Description	One of three products used to describe operational activity—traces actions in a scenario or sequence of events
Operational	OV-7	Logical Data Model	Documentation of the system data requirements and structural business process rules of the Operational View
Systems	SV-1	Systems Interface Description	Identification of systems nodes, systems, and system items and their interconnections, within and between nodes
Systems	SV-2	Systems Communications Description	Systems nodes, systems, and system items, and their related communications lay-downs
Systems	SV-3	Systems-Systems Matrix	Relationships among systems in a given architecture; can be designed to show relationships of interest, e.g., system-type

Applicable View	Framework Product	Framework Product Name	General Description
			interfaces, planned vs. existing interfaces, etc.
Systems	SV-4	Systems Functionality Description	Functions performed by systems and the system data flows among system functions
Systems	SV-5	Operational Activity to Systems Function Traceability Matrix	Mapping of systems back to capabilities or of system functions back to operational activities
Systems	SV-6	Systems Data Exchange Matrix	Provides details of system data elements being exchanged between systems and the attributes of that exchange
Systems	SV-7	Systems Performance Parameters Matrix	Performance characteristics of Systems View elements for the appropriate time frame(s)
Systems	SV-8	Systems Evolution Description	Planned incremental steps toward migrating a suite of systems to a more efficient suite, or toward evolving a current system to a future implementation
Systems	SV-9	Systems Technology Forecast	Emerging technologies and software/hardware products that are expected to be available in a given set of time frames and that will affect future development of the architecture
Systems	SV-10a	Systems Rules Model	One of three products used to describe system functionality—identifies constraints that are imposed on systems functionality due to some aspect of systems design or implementation
Systems	SV-10b	Systems State Transition Description	One of three products used to describe system functionality—identifies responses of a system to events
Systems	SV-10c	Systems Event-Trace Description	One of three products used to describe system functionality—identifies system-specific refinements of critical sequences of events described in the Operational View
Systems	SV-11	Physical Schema	Physical implementation of the Logical Data Model entities, e.g., message formats, file structures, physical schema
Technical	TV-1	Technical Standards Profile	Listing of standards that apply to Systems View elements in a given architecture
Technical	TV-2	Technical Standards Forecast	Description of emerging standards and potential impact on current Systems View elements, within a set of time frames

Appendix E: Glossary

Abstraction Level	-	A perspective of the enterprise that governs the level of detail exposed and separates concerns. For example, at the enterprise abstraction level, the interactions of the entire enterprise are exposed, and the architecture needed by the enterprise to achieve those interactions. At the system level, the focus is a perspective that deals with a single system in the enterprise architecture at a time.
Actor	-	An entity, outside the system, that directly interacts with the system. It could represent either a human role or another system.
Context	-	The environment within which a system must operate.
Context Diagram	-	A diagram showing the context for a given system
Deployment Architecture	-	A specific instance of a configuration of hardware and software established for the purpose of installing an running the developed systems for their intended use.
Deployment Node	-	A physical location, an instance of a design node.
Design Node	-	A node realizing a locality
DoDAF	-	Department of Defense (DoD) Architecture Framework Version 1.0, 15-Aug-2003, published by the DoD Architecture Working Group
Enterprise	-	The highest abstraction level. The system at the highest abstraction level, often the entire company or mission unit.
I/O Entity	-	Some thing that is exchanged between the system and an actor.
IE/IER	-	Information Exchange Requirement. The DoDAF defines an information exchange as an act of exchanging information between two distinct O-Nodes and the characteristics of the act, including the information element (which we can refer to as the IE class) that needs to be exchanged and the attributes associated with that element, as well as the attributes associated with the exchange. For purposes of this document, the information exchange and the information exchange requirement refer to the same thing/concept.
Interface	-	A grouping of related services. A named set of operations/services that characterize the behavior of an element. An interface is eligible for realization by any logical architecture element – a class. An interface is eligible for realization at any physical architecture element – a locality.
Locality	-	An abstract grouping of physically distributed system capability.
Logical Architecture	-	(of a system) The set of logical units of functionality (subsystem classes) that collaborate to achieve the goals and interactions of the system.
Missile Event	-	A reference to the predicted missile impact and its Launch and Predicted Impact (LPI).
ONode	-	A zero-footprint grouping of operational functionality. An aggregation of roles and therefore of responsibilities. The DoDAF defines an operational node as an element of the operational architecture that produces, consumes, or processes information.
Operation	-	A service that can be requested from a system object (an instance of a system class) to effect behavior. An operation has a signature, which may restrict the actual parameters that can be supplied to influence to behavior within the operation.
Physical Architecture	-	(of a system) The set of localities that collaborate to achieve the goals and interactions of the system.
PNode	-	A place, a real place, where instances of S-Nodes and O-Nodes are realized.

Service	-	A named behavior that can be specifically requested of, and performed by, a system or subsystem. Services are modeled in UML using operations on classes. A system class describes the service but cannot actually execute it. Objects instantiated from a system class can actually execute the service, and presumably does so on request. Note: there is significant confusion regarding the word service due to a conceptual description of the structure of a system called Service Oriented Architecture (SOA). What is often called a service in SOA parlance is what RUP SE would refer to as a system object or a system class (which one depends on the context). So please be careful when saying service; you mean operation of a class, but your listener might construe the class itself.
Service Realization	-	The act of finding a suitable architecture for a system, composed of a set of discovered logical and physical subsystems, by developing a set of collaborations between those subsystems which permit that system to meet its contract to provide the set of services it is assigned.
SNode	-	Locality. See Locality.
System	-	The abstraction level just below the enterprise abstraction level. A group of interacting, interrelated, or interdependent elements forming a complex whole. A system provides a set of services that are used by an enterprise (its context) to carry out a business purpose (mission). System elements consist of hardware, software, data and workers. (Blanchard and Fabricky.)
Use Case	-	An observable result of value achievable by one of a system's actors, which that actor achieves by using the system. The collection of use cases for a system is one way to represent that system's functional behavior, or the requirements for that behavior. A use case can be considered a container for all the scenarios (formally, instances of the use case) in which the actors achieve, or attempt to achieve, the named result of value.
Use Case Flowdown	-	A technique for deriving functional (derived) requirements for the analysis (and then design) elements.