Telelogic

Modeling DoDAF Compliant Architectures

The Telelogic Approach for Complying with the DoD Architectural Framework

A Telelogic White Paper

Authors: Cris Kobryn and Chris Sibbald
Published: 25 October 2004

Abstract

No industry is more demanding than aerospace-defense when it concerns the complete integration of software and hardware architectures. Tasked with defending our nascent Information Age economy, the aerospace-defense industry must be capable of waging Information Age warfare, whose key concepts include *information superiority* and *network-centric warfare*. By most quantitative metrics, aerospace-defense systems are among the largest and most complex ever constructed. The complexity of individual aerospace-defense systems (e.g., an aircraft) is frequently compounded when we integrate many of them together to form a system-of-systems (e.g., an air traffic control system) that satisfies more global requirements.

This white paper describes a technical approach for improving how we specify system and system-of-systems architectures using frameworks in general, and the Department of Defense Architecture Framework (DoDAF) in particular. The model-driven approach to architectural frameworks explained here, which is based on UML™ 2.0, Telelogic TAU® Generation2™, and Telelogic DOORS®, can substantially improve productivity and quality.

CONTENTS

FIGURES

TABLES

INTRODUCTION

As we struggle to transform ourselves from an Industrial Age economy to an Information Age economy, our appetite for software-intensive systems has become voracious. This shouldn't be surprising, since software provides both the content and the processing instructions for the information on which our evolving economy is based. As a consequence, software-intensive systems are expanding and propagating into all facets of our lives, from the enterprise systems that run our businesses, to the embedded systems that tend our machines, and the multimedia systems that entertain us.

Of course, software-intensive systems do not exist in isolation. Software is typically deployed upon, or embedded in, hardware systems. In addition, software-intensive systems are frequently integrated with other systems to form systems-of-systems (SoS). For example, a system-of-systems for investment services might integrate various kinds of stock trading and bond trading systems. As another example, a system-of-systems for air traffic control might integrate aircraft, terminal radars, control towers, and flight service stations.

As the operational requirements of our systems and SoSs increase exponentially, so does our need to manage their complexity. Towards this end, enterprise architecture has emerged as a strategic engineering discipline, and it has become commonplace for software architects to collaborate with systems and hardware architects on large systems that integrate software and hardware components. Indeed a new role, building upon the traditional systems and software architecture experience has emerged, the role of the Enterprise Architect.

No industry is more demanding than aerospace-defense when it comes down to the complete integration of software and hardware architectures. Tasked with defending our nascent Information Age economy, the aerospace-defense industry must be capable of waging Information Age warfare, whose key concepts include *information superiority* and *network-centric warfare* [Alberts 01]. By most quantitative metrics, aerospace-defense systems are among the largest and most complex ever constructed, rivaled only by the global telecommunications network. The complexity of individual aerospace-defense systems (e.g., an aircraft) is frequently compounded when we integrate many of them together to form a SoS (e.g., air traffic control or joint forces targeting) that satisfies more global requirements and provides new or enhanced capabilities. Furthermore, in practice many of these SoSs are ephemeral and created "on the fly" to satisfy urgent needs.

This white paper describes a technical approach for improving how we specify system and SoS architectures using frameworks in general, and the Department of Defense Architectural Framework (DoDAF) in particular. The model-driven approach to architectural frameworks explained here, which is based on architectural blueprint languages such as UML 2.0 and power tools such as Telelogic TAU Generation2 and Telelogic DOORS, can substantially improve productivity and quality.

The first part of the paper introduces the concepts of architectural frameworks as they apply to systems development. It next discusses the DoDAF, which is emerging as the leading standard for frameworks in the aerospace-defense industry.

The second part of the paper shows how the powerful concepts described in the first part can be applied to solve practical problems. In particular, it shows how UML 2.0 and TAU Generation2 have been tailored to specify a DoDAF-compliant architecture and presents an example for a joint force targeting system. The paper concludes with some speculation about the future of DoDAF as it evolves from a conceptual architecture into a technical architecture supported by tools.

THE ARCHITECTURE FRAMEWORK ADVANTAGE

We can trace the concept of an architectural framework to prefabrication in the traditional building industry, which first started manufacturing standard components for homes in the first half of the twentieth century.[1] Although the first attempts to prefabricate homes produced mixed results, these early experiences had a profound influence on the building industry, and contributed to the tract-home building revolution that occurred after World War II.

The fundamental insight behind architectural frameworks and prefabrication is that efficiencies of scale can be realized if building components are efficiently fabricated in a factory and later assembled at the delivery locations. In the case of the prefabricated homes, building components (e.g., walls, ceilings, roofs) are constructed in a factory, so that they can be later assembled (e.g., nailed, screwed, welded) and customized at the building site. The fabrication-and-assembly process can also stimulate innovation by encouraging the assembly of components in novel ways to yield new or enhanced capabilities (e.g., integrating electrical and communication wiring in pre-fabricated wall units to reduce onsite installation costs).

Architectural frameworks have been effectively applied to a wide range of industries with impressive gains in productivity and quality. These industries include, but are not limited to, automotive, telecommunications, and computer hardware. Considering the current demands to increase system productivity and quality, it shouldn't be surprising that there is keen interest to apply the concept to systems and SoSs.

The advantages of an architectural framework approach are summarized in Table 1 [MDE 04]:

Table 1: Advantages of Architecture Frameworks

FRAMEWORK FEATURE	TECHNOLOGY ADVANTAGES	BUSINESS ADVANTAGES
prefabricated structural skeleton	Codifies proven architecture best practices.	Reduce costs and accelerate time to market.
multiple layers	Separates business logic from technical infrastructure	Captures and protects business intellectual property.
multiple views	Supports appropriate views various stakeholders	Ensure right business system is being built.
consistent designs, interfaces, and standards	Promotes interoperability.	Reduce costs and accelerate time to market.

[1] Sears, Roebuck and Company sold houses via mail-order catalog and through their sales offices to nearly 100,000 clients between 1908 and 1940. Source: *Journal of Design History*, Volume 14, Issue 1, 2001: pp. 53-70.

An explanation of the advantages of specific framework features follows:

- prefabricated structural skeleton: Frameworks furnish a reusable structure that includes reliable components that are constructed and integrated using best practices.
- customizability and extensibility: Engineers must be able to adapt the frameworks to satisfy the individual needs of their target system. Consequently, they must be able to change, replace, or add components in order to customize it.
- multiple layers: Frameworks usually supports multiple layers to insulate business logic from construction and platform details essentially decoupling the two, thus simplifying technology insertion to yield enhanced capabilities
- multiple views: Frameworks typically supports multiple views for various stakeholders, so that each stakeholder can evaluate the system from a perspective appropriate for her role.
- consistent designs, interfaces and standards: Frameworks promote interoperability with consistent designs, standards, and interfaces across applications. This also simplifies the analysis of interoperability and re-use.

DODAF OVERVIEW

The sub-sections below provide a brief history and technical summary of DoDAF that will serve as useful background for the example that follows.

Brief History

In the mid 1990s the DoD determined that a common approach was needed for describing its architectures, so that DoD systems could efficiently communicate and interoperate during joint and multinational operations. Consequently, the C4ISR Integration Task Force was formed and developed version 1.0 of the *C4ISR Architecture Framework* in 1996. The C4ISR Architecture Working Group completed version 2.0 in 1997.

After working with the first two versions of C4ISR framework, and recognizing the need to strengthen it prior to adoption, the DoD began work on a third version. In order to emphasize the applicability of the framework beyond the C4ISR community, the third version was renamed the DoD Architecture Framework (DoDAF) v. 1.0 and released in August 2003. The DoD approved DoDAF v. 1.0 for official use on February 9, 2004 [DoDmemo 04]. All DoD architectures developed or approved subsequent to December 1, 2003 must comply with this framework. Architectures developed prior to this date must comply with DoDAF upon their next version update.

Technical Summary

DoDAF is intended to ensure that architecture descriptions developed by the DoD commands, services, and agencies are interoperable across each organization's operational, systems, and technical architecture views, and can also interoperate across joint and combined organizational boundaries. The framework provides rules and guidance for developing and presenting architecture descriptions. It also defines work products related to architecture development, which are descriptive artifacts that communicate the architecture.

The framework provides guidance on how to describe architectures; it does not provide guidance in how to construct or implement a specific architecture or how to *develop* and *acquire* systems or systems-of-systems. DoDAF addresses the military domain and is primarily used by the DoD.

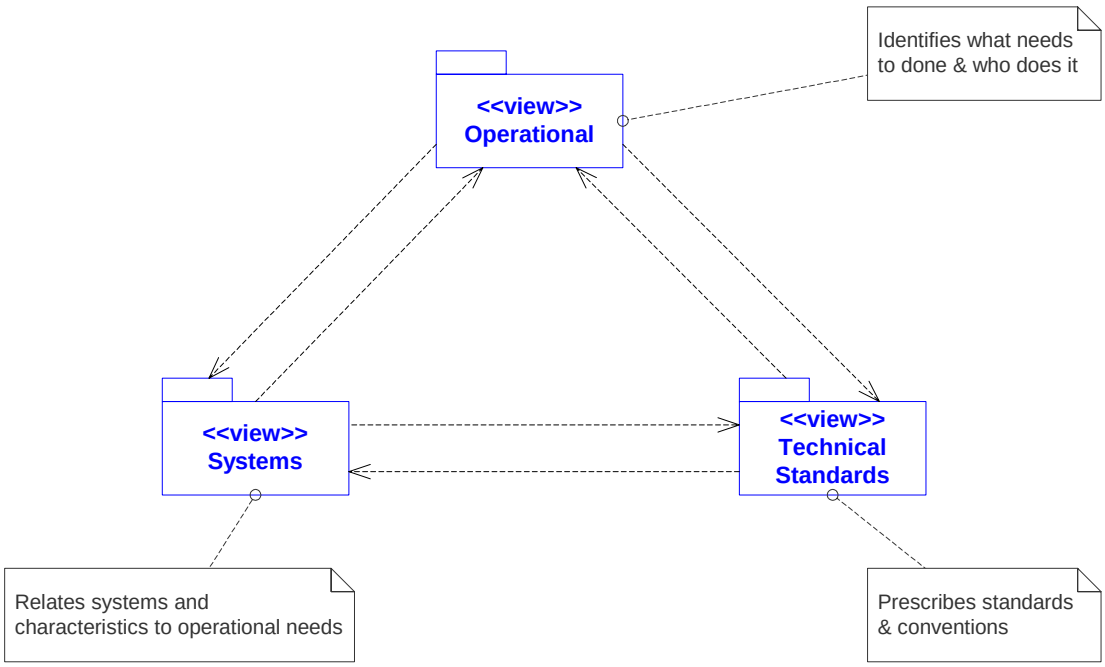
Views

DoDAF architectural descriptions are defined in terms of multiple views, each of which conveys different aspects of the architecture in several products (descriptive artifacts or models). DoDAF defines the following views:

- **Operational View (OV)**. The OV describes tasks, activities, participating nodes, and associated information exchanges required to perform a mission.

- **Systems View (SV).** The SV describes the systems of concern and their connections in the context of the OV.
- **Technical Standards View** (TV). The TV describes a profile of the minimum standards and rules that govern the implementation, arrangement, interaction and interdependence of the systems described in the SV.

In addition to the three major view types, there are also two All Views, which apply across all other views. One of these views (AV-1) describes the scope, purpose and objectives of the architecting effort and provides an executive summary including



analysis results, and the other (AV-2) furnishes an integrated dictionary.

Figure 1: Relationships Among DoDAF Views

Figure 1 shows the basic relationships among the top-level DoDAF views.. In order to maintain its architectural integrity, a DoDAF compliant architecture must maintain explicit linkages among its various views. The OV describes detailed information exchanges so that the degree of operational interoperability can be determined. The SV identifies the systems that support the operational requirements, translates the interoperability requirements into a set of system data exchanges that can be executed by system functions, and compares implementations with required operational capabilities. The TV describes the criteria for each required system that will satisfy the interoperability requirements. Collectively the three views and their interrelationships provide the basis for measuring system interoperability and performance, as well as their impact on mission and task effectiveness.

Products

The set of DoDAF products are listed in Table 2 [DoDAFv1 04]. The *UML Diagrams* column has been added to indicate the UML diagrams that are applicable for each view [MDE 04]. It must be noted that that more than one mapping from DoDAF products to UML diagrams is possible. Those listed in Table 2 correspond to those used in developing Telelogic Enterprise Architect for DoDAF (a solution consisting of the integrated tools Telelogic DOORS and Telelogic TAU Generation2, with DoDAF-specific extensions). It is also important to note that in developing the DoDAF profile for TAU Generation2 a conscious effort was made to make the tool "speak the language of the Domain" in order to minimize the need to learn UML. Hence users for Telelogic Enterprise Architect for DoDAF will create OV-2 diagrams, for example without the need to know that this is an extension of a UML 2.0 Composite Structure Diagram. Detailed descriptions of selected products accompany the examples in the following section.

Table 2: DoDAF Architecture Products

APPLICABLE VIEW	FRAME-WORK PRODUCT	FRAMEWORK PRODUCT NAME	GENERAL DESCRIPTION	UML DIAGRAMS
All Views	AV-1	Overview and Summary Information	Scope, purpose, intended users, environment depicted, analytical findings	Text (or DOORS) documents
All Views	AV-2	Integrated Dictionary	Data repository with definitions of all terms used in all products	UML model queries + report generator
Operational	OV-1	High-Level Operational Concept Graphic	High-level graphical/ textual description of operational concept	Class or Use Case diagram
Operational	OV-2	Operational Node Connectivity Description	Operational nodes, operational activities performed at each node, connectivity and information exchange needlines between nodes	Composite Structure Diagram
Operational	OV-3	Operational Information Exchange Matrix	Information exchanged between nodes and the relevant attributes of that exchange	UML model queries + report generator
Operational	OV-4	Organizational Relationships Chart	Organizational, role, or other relationships among organizations	Class diagram
Operational	OV-5	Operational Activity Model	Operational Activities, relationships among activities, inputs and outputs. Overlays can show cost, performing nodes, or other pertinent information	Activity diagram with Object Flows
Operational	OV-6a	Operational Rules Model	One of the three products used to describe operational activity sequence and timing - identifies business rules that constrain operation	Text (or DOORS) document linked to Activities

APPLICABLE VIEW	FRAME-WORK PRODUCT	FRAMEWORK PRODUCT NAME	GENERAL DESCRIPTION	UML DIAGRAMS
Operational	OV-6b	Operational State Transition Description	One of three products used to describe operational activity sequence and timing - identifies business process responses to events	State Machine Diagram
Operational	OV-6c	Operational Event-Trace Description	One of three products used to describe operational activity sequence and timing - traces actions in a scenario or sequence of events and specifies timing of events	Sequence Diagram
Operational	OV-7	Logical Data Model	Documentation of the data requirements and structural business process rules of the Operational View.	Class diagram
Systems	SV-1	Systems Interface Description	Identification of systems and system components and their interconnections, within and between nodes	Composite Structure diagram
Systems	SV-2	Systems Communications Description	Systems nodes and their related communications lay-downs	Composite Structure diagram
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 interfaces, planned vs. existing interfaces, etc.	DOORS Traceability View automatically populated as interfaces are defined in model.
Systems	SV-4	Systems Functionality Description	Functions performed by systems and the information flow among system functions	Activity diagram with Object Flows
Systems	SV-5	Operational Activity to Systems Function Traceability Matrix	Mapping of systems back to operational capabilities or of system functions back to operational activities	DOORS Traceability View automatically populated as allocation done in model.
Systems	SV-6	Systems Data Exchange Matrix	Provides details of systems data being exchanged between systems	UML model queries + report generator
Systems	SV-7	Systems Performance Parameters Matrix	Performance characteristics of each system(s) hardware and software elements, for the appropriate timeframe(s)	UML model queries + report generator Or linked DOORS document(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	Project planning doc linked to model elements
Systems	SV-9	Systems Technology Forecast	Emerging technologies and software/hardware products that are expected to be available in a given set of timeframes, and that will affect future development of the architecture	Text (or DOORS) document

APPLICABLE VIEW	FRAME-WORK PRODUCT	FRAMEWORK PRODUCT NAME	GENERAL DESCRIPTION	UML DIAGRAMS
Systems	SV-10a	Systems Rules Model	One of three products used to describe systems activity sequence and timing—Constraints that are imposed on systems functionality due to some aspect of systems design or implementation	Text (or DOORS) document linked to System Functions
Systems	SV-10b	Systems State Transition Description	One of three products used to describe systems activity sequence and timing—Responses of a system to events	State Machine diagram
Systems	SV-10c	Systems Event-Trace Description	One of three products used to describe systems activity sequence and timing -- System-specific refinements of critical sequences of events and the timing of these events	Sequence diagram
Systems	SV-11	Physical Schema	Physical implementation of the information of the Logical Data Model, e.g., message formats, file structures, physical schema	Class diagram
Technical	TV-1	Technical Standards Profile	Extraction of standards that apply to the given architecture	Text (or DOORS) document linked to Systems
Technical	TV-2	Technical Standards Forecast	Description of emerging standards that are expected to apply to the given architecture, within an appropriate set of timeframes	Text (or DOORS) document linked to Systems

LANGUAGE AND TOOLS: UML 2.0, TAU GENERATION2, DOORS

In order to successfully implement architectural frameworks, both modeling language standards and tools that implement them are required. In this section we explore the recently adopted UML 2.0 standard, and the first commercial tool that has implemented a large portion of it, TAU Generation2.

Architecture Description Language: UML 2.0

In order to successfully implement a complete, correct and robust architectural framework, we require an architectural description language that is precise and concise.

The major improvements to UML 2.0 include, but are not limited to, the following [Kobryn 04]:

- **Support for component-based development via composite structures.** Structured classifiers (both Classes and Components) can be decomposed and assembled ("wired") via Parts, Ports, and Connectors. In addition, UML 2.0 supports both black-box and white-box views of structured classifiers. For DoDAF this capability is critical for decomposing SoSs into systems, subsystems and components.
- **Hierarchical decomposition of structure and behavior.** In addition to Classes and Components, which are structural constructs, UML 2.0 supports the hierarchical decomposition of the major behavioral constructs, such as Interactions, State Machines, and Activities. For DoDAF, the ability to decompose functionality (behavior) as well as structure is essential for scalability.
- **Cross integration of structure and behavior.** The decomposed constructs described above can be flexibly integrated with each other. For example, the same Parts that are used in a Composite Structure diagram of a Class to show its internal structure can also be used in a Sequence diagram to show how the internal structures communicate with each other. For DoDAF, this capability is critical for maintaining architectural integrity across the various Operational View and System View products.
- **Integration of action semantics with behavioral constructs**. UML actions are now defined in as much detail as a programming language's actions (or statements), so that you can define executable models for simulations and code generation. For DoDAF, this feature allows architects to debug their logic as early as practical in the system lifecycle, and also supports simulation and code generation.
- **Layered architecture to facilitate incremental implementation and compliance testing.** UML 1.x was a large language, and UML 2 is larger still. Taking a lesson from other large languages (e.g., SQL), UML 2 packages are organized into three levels (Basic, Intermediate, and Complete) in order to make it easier for vendors to implement and more efficient for standards organizations to test compliance. For DoDAF, this allows vendors to incrementally implement UML 2.0 features as they are needed to support DoDAF views.

- **Powerful extension mechanism via profiles.** The ability to develop domain specific extensions via profiles and meta-models developed in UML 2.0 permits domain specific extension such as DoDAF profiles and SysML.

Cumulatively these improvements mark a significant evolution of the language, increasing its precision and expressiveness so that it can be effectively used to model large, complex architectures.

Power Tool: TAU Generation2

Although a precise and concise architectural blueprint language is required for a successful model-driven development approach, it alone is insufficient. The language must be accompanied by a power tool that faithfully and efficiently implements the language, so that it can automate the mapping transformations across the various models.

TAU Generation2 (TAU G2) is a family of model-centric and role-based tools that are among the first to implement the recently adopted UML 2.0 standard. The tool family consists of TAU/Developer™ for Software Engineers, TAU/Architect™ for Systems Engineers and Architects, and TAU/Tester™ for Test Engineers. TAU G2 builds on the model-driven compilation technology perfected in TAU SDL Suite™ (a.k.a. TAU G1). TAU G1 proved that software development, in the most complex, integrated “C4ISR” systems in the world – the telecommunications network, can be automated using mature specifications languages such as Specification and Description Language (SDL) and Message Sequence Chart (MSC). Given that many of the advanced language features offered by SDL and MSC were adapted and incorporated into UML 2.0, there were compelling technical and market reasons to combine TAU G1’s model-driven compilation technology and discrete event simulation with UML 2.0 to produce TAU G2.

TAU G2 provides the following capabilities and benefits:

- Precise and unambiguous system specification - Engineers can visually specify systems using the precise, standardized and non-proprietary language of UML 2.0. This results in easy-to-understand, clear and unambiguous specifications.
- Specification of behavior – Whereas most system modeling tools allow only the specification of the system’s architecture or structure, TAU G2 also allows engineers to visually specify the dynamic aspects of the system's behavior.
- Automatic application generation - TAU/Architect and TAU/Developer are the only tools that support executable UML 2.0 models with behavioral specifications. Developers have access to pre-defined, verifiable code patterns that ensure high quality standards. With these capabilities, developers can automatically generate complete applications.
- Dynamic model verification - With fully controllable model simulation, engineers can verify their work in the analysis, design, and implementation phases. As a result,

they can quickly locate and remove errors early when corrections are relatively easy and inexpensive.

- Scalability - Large scale systems can be specified and models can be mapped to how teams want to work, rather than having restrictions imposed by the tool. System architecture and behavior also can be modeled and viewed at the appropriate level of abstraction for the user.
- Extensibility – TAU G2 provides powerful extension, customization and automation mechanisms. The tool is meta-model driven, permitting user defined and domain specific diagram types, model elements, menus, etc. to be specified via UML profiles. Scripting capabilities are available to automate routine tasks and query & report on the model. Agents can be developed to perform automated tasks when the associated trigger event is activated.
- Open Architecture - COM and TCL APIs permit third party tool access to the underlying model. The model is stored in XML format simplifying translation and data interchange.
- Integrated requirements management via DOORS® - TAU G2 is integrated with DOORS, the market leading requirements management solution. The permits architects to view DOORS information and establish Traceability within the modeling environment (“Link as you think”) and analysts to view traceability reports and to perform impact, gap and coverage analysis across requirements and model elements within the DOORS environment.
- Automated documentation via DocExpress® - TAU G2 is integrated with DocExpress, which provides automatic extraction and formatting of system or software application documentation.
- Change and configuration management via SYNERGY™ - SYNERGY provides change and configuration management for TAU G2 and related products.

DODAF EXAMPLE: USCENTCOM ARCHITECTURE

In this section we show an example of how UML 2.0, extended with a DoDAF-specific profile and implemented in TAU G2 can be used together to specify a DoDAF model for a typical DoD application. It is important to note that this implementation delivers an environment that will be easy to adopt for the defense architect, since the terminology used is that used in DoDAF, not UML. When the 'enterprise' architecture is handed off to systems and software engineers this view can be instantly toggled to show the same architecture model but now using UML terminology.

For this illustration we have chosen to adapt the USCENTCOM (US Central Command) architecture example for *Conduct Joint Force Targeting* found in Volume 3 (Deskbook) of the DoDAF specification [DoDAFv3 04].

This architecture describes the various air tasking activities that produce a daily Air Tasking Order (ATO). An ATO directs air operations for a 24-hour period and addresses mission objectives, targets and resources. The system assesses the effectiveness of air operations against targets, and provides feedback to the ATO development process. For example, if a primary target is not destroyed during an air strike, it will be included for re-strike immediately or in a subsequent ATO.

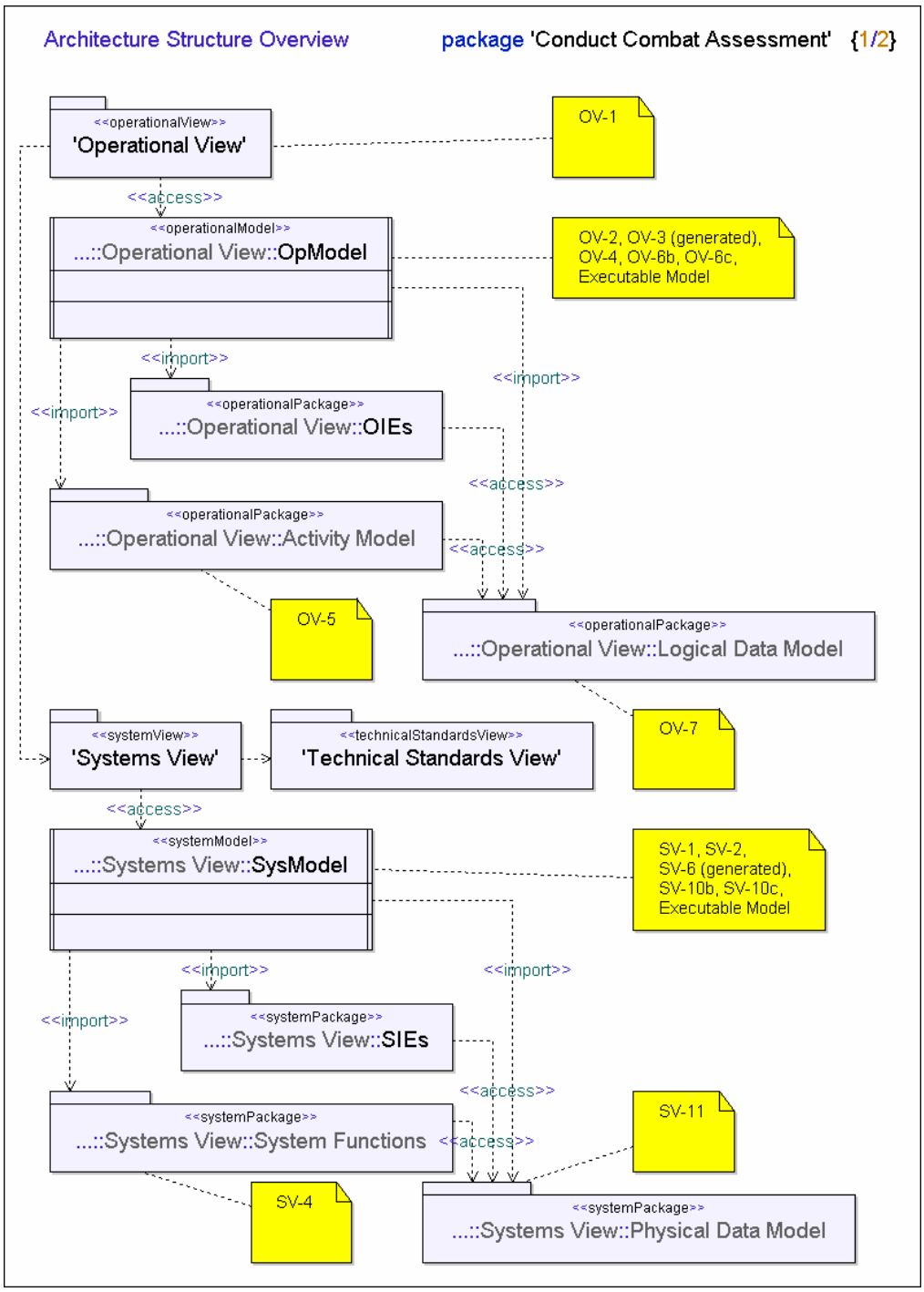


Figure 2: Architecture Framework Package Diagram

Figure 2 shows the top-level package diagram for the architecture framework, which organizes the various views and provides guidance on the structure of the framework as well as a convenient means of navigating directly to desired information.

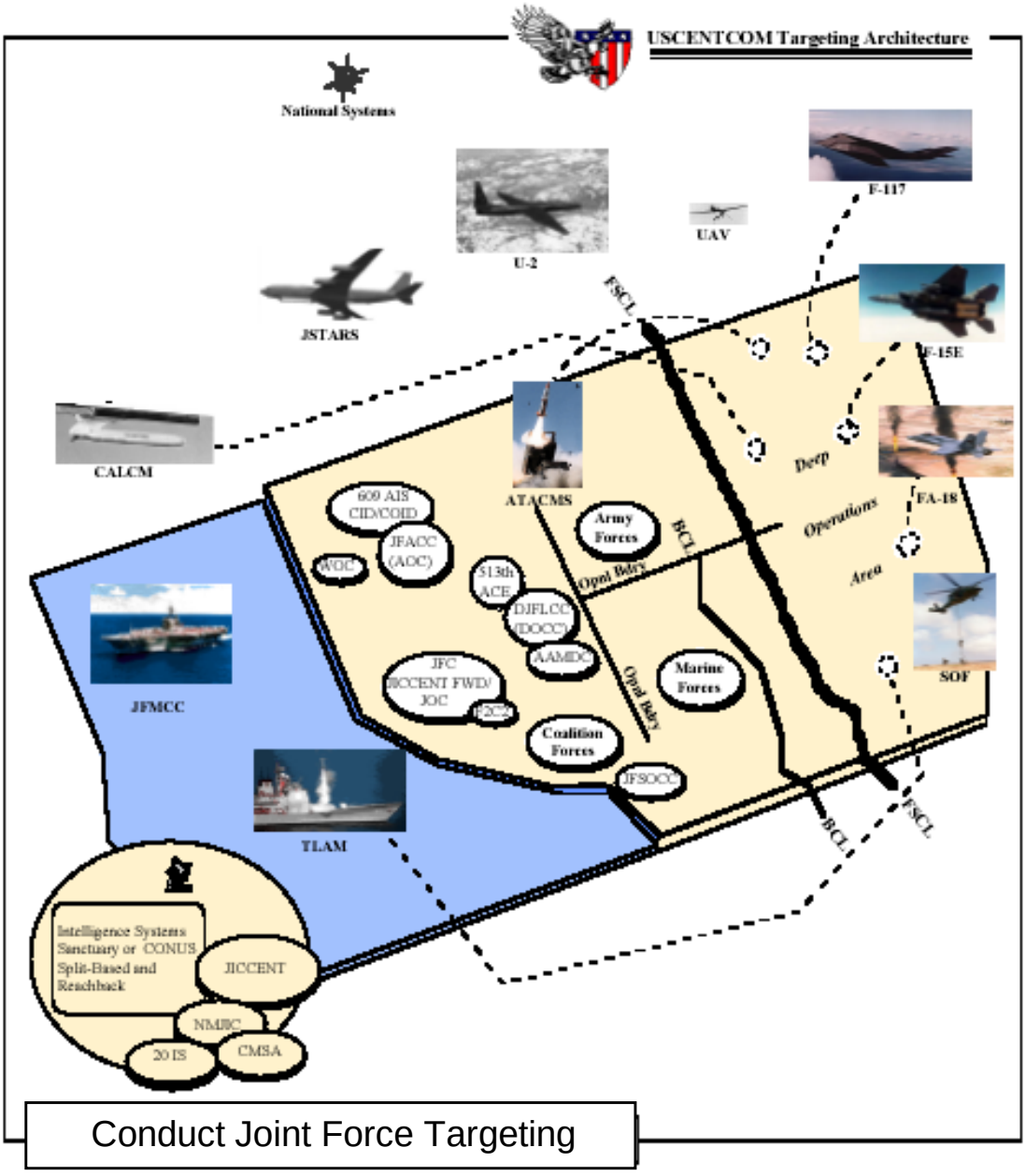


Figure 3: OV-1 High Level Operational Concept (graphic)

Figure 3 illustrates the high-level conceptual view for the operational system that conducts joint force targeting using intuitive graphics that are not based on UML. Contrast this diagram with Figure 4 and Figure 5, which illustrate how the high-level conceptual view can be shown more precisely using UML.

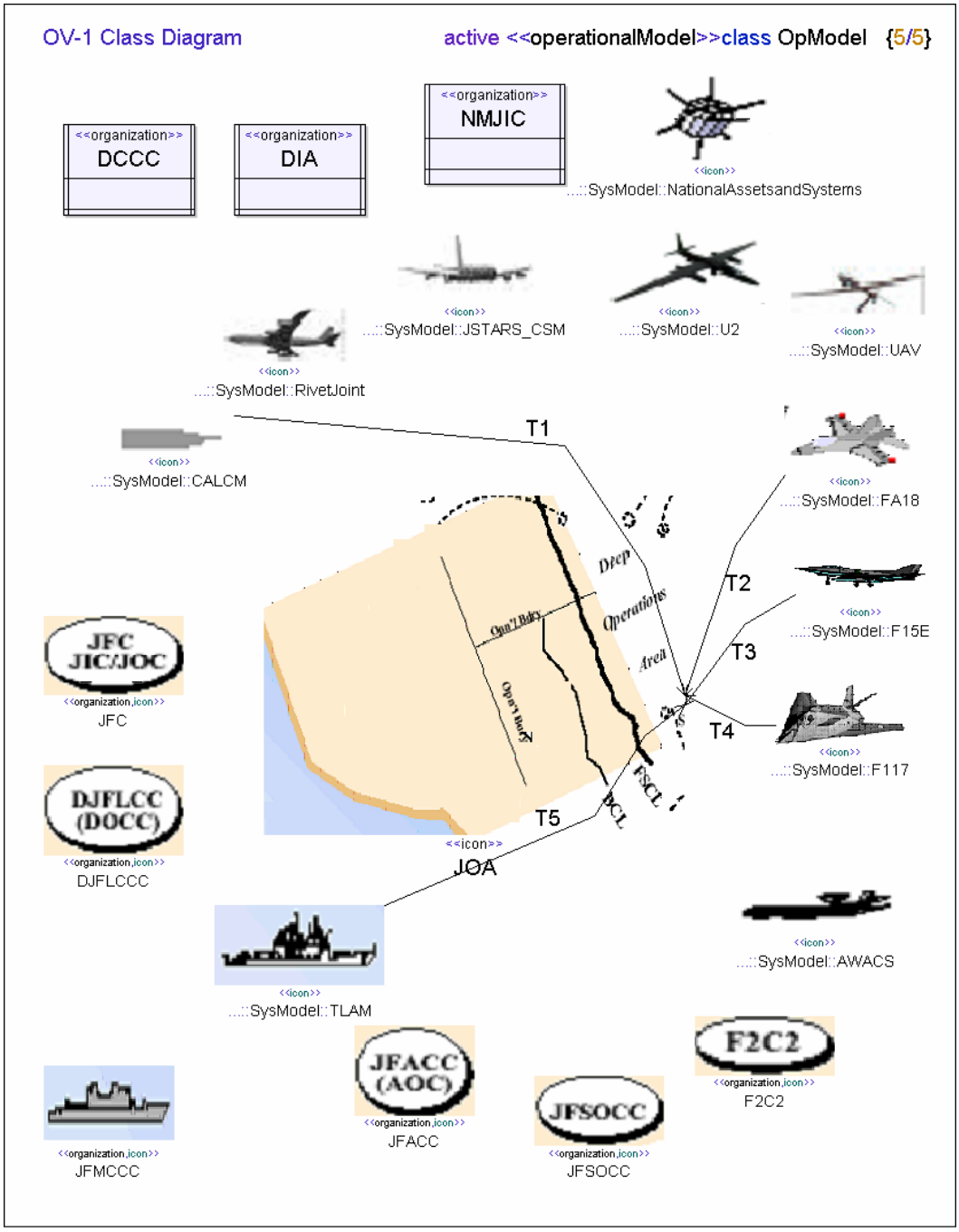


Figure 4: OV-1 High Level Operational Concept (Class diagram)

Figure 4 shows the high-level conceptual view for the architecture as a Class diagram. The example shows various organizational (e.g., *DCCC*, *DIA*, *F2C2*) and asset entities (e.g., *UAV*, F-*117*, *AWACS*). Note that the rectangles that typically represent UML Classes have been replaced with icons meaningful to domain experts.

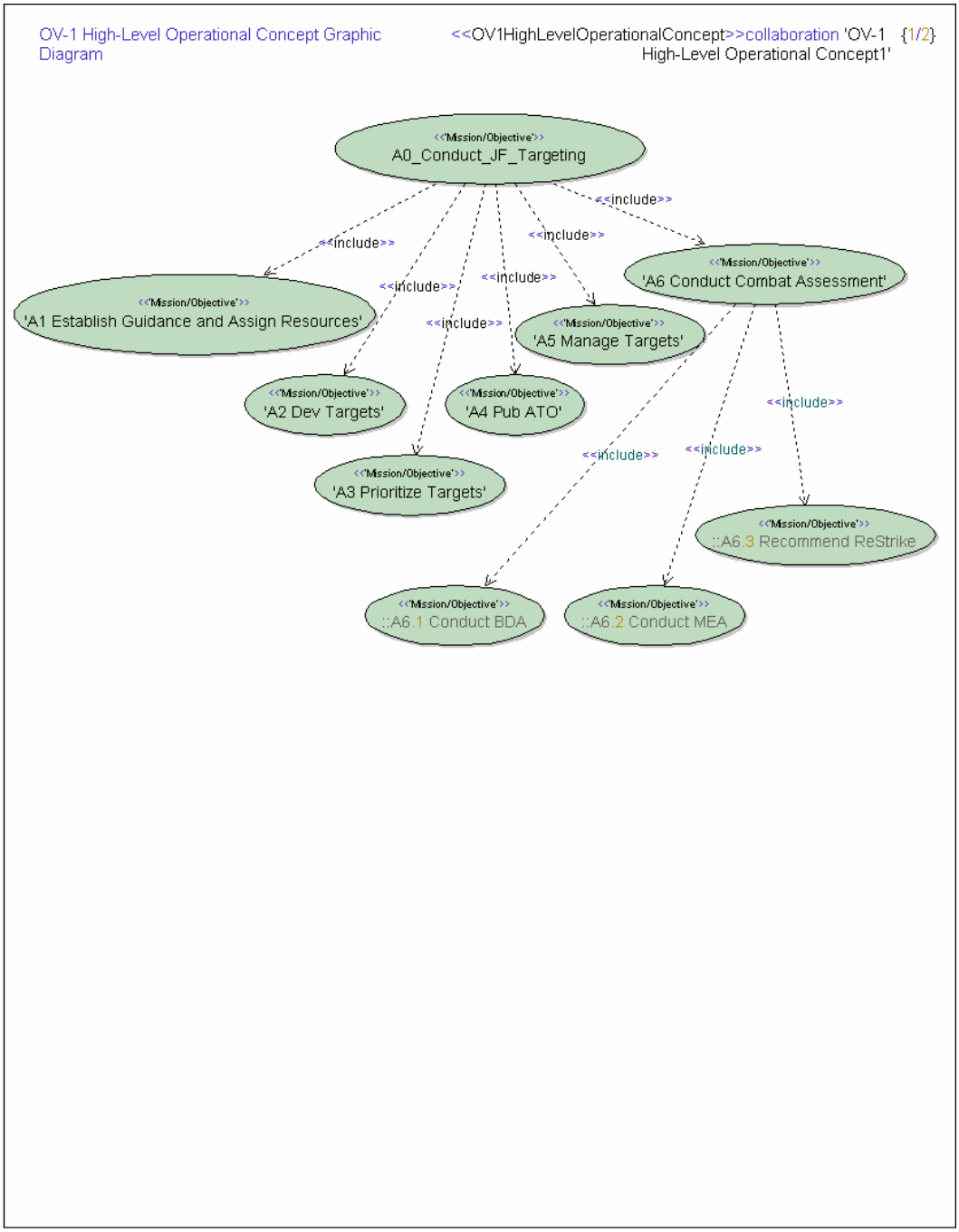


Figure 5: OV-1 High Level Operational Concept Graphic (Use Case diagram)

Figure 5 shows the high-level conceptual view for the operational system as a Use Case diagram. Whereas the Class diagram in Figure 4 emphasizes the top-level structures of the system, the Use Case diagram emphasizes the top-level functions (e.g., *Conduct Combat Assessment*, *Recommend ReStrike*).

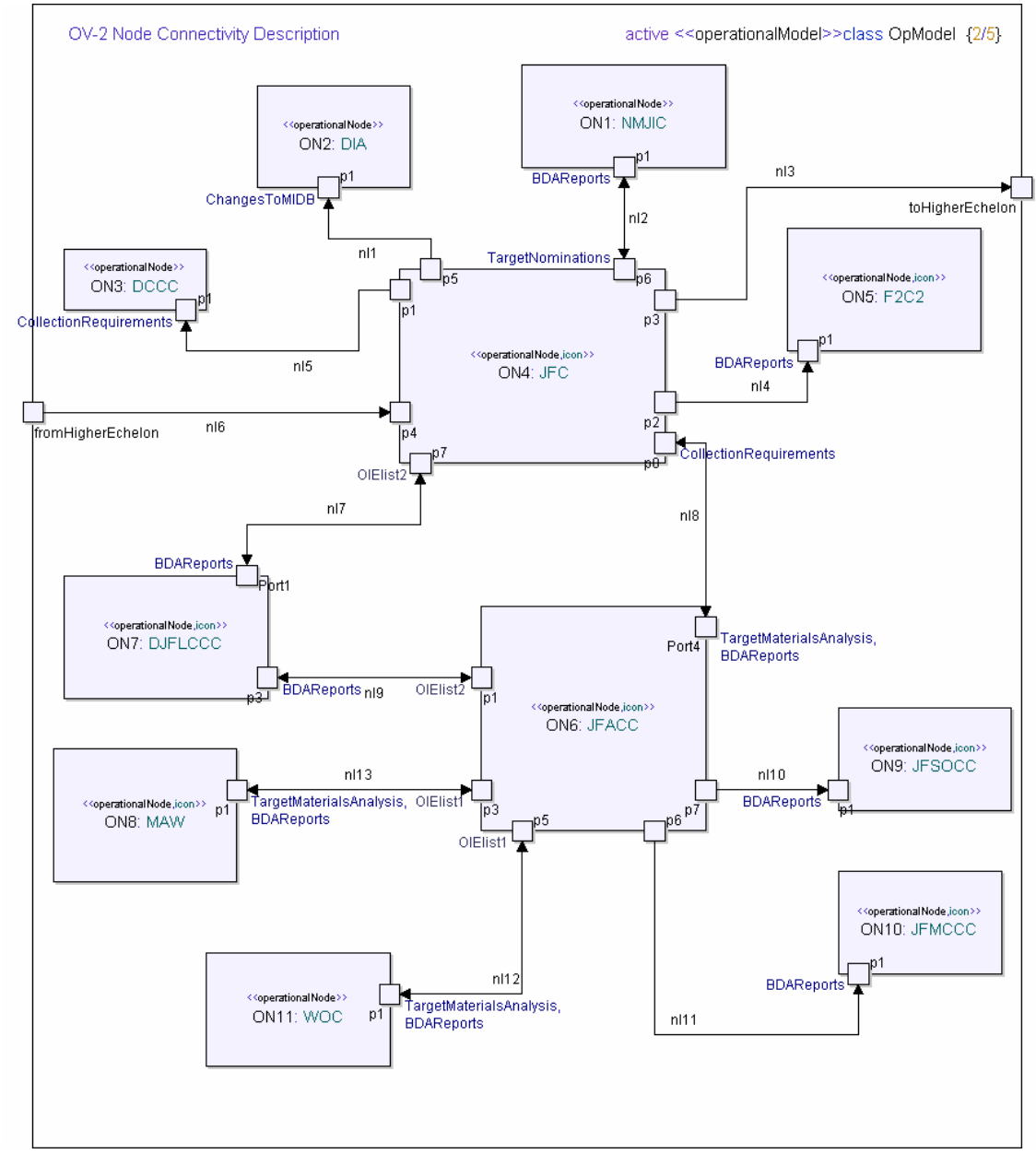


Figure 6: OV-2 Operational Node Connectivity Diagram (Composite Structure diagram)

Figure 6 shows the OV-2 Operational Node Connectivity Diagram extended from a UML Composite Structure diagram. Composite Structure diagrams allow architects to recursively decompose SoSs into systems, subsystems and components using UML 2 parts, ports and connectors.

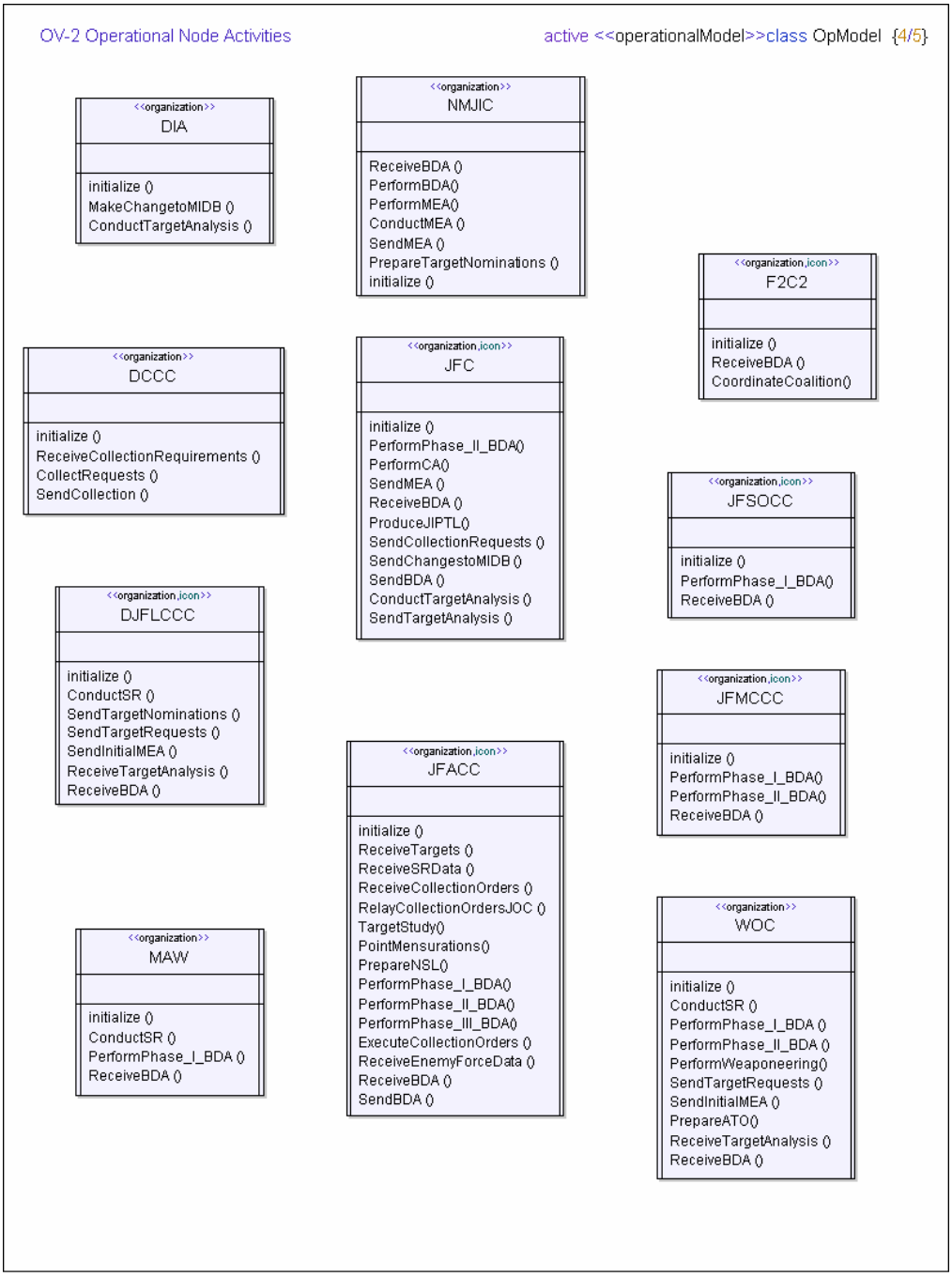


Figure 7: OV-2 Operations Node Connectivity Diagram (Class operations)

Figure 7 shows the OV-2 Operational Node Connectivity Diagram using an UML Class diagram. The Class diagram shows a collection of classes that represent operational nodes (e.g., *DIA*, *NMJIC*, *J2C2*). Each of these nodes specifies a set of operations in its bottom compartment (e.g., *F2C2* specifies the operations *initialize*, *receiveBDA*, and *CoordinateCoalition*).

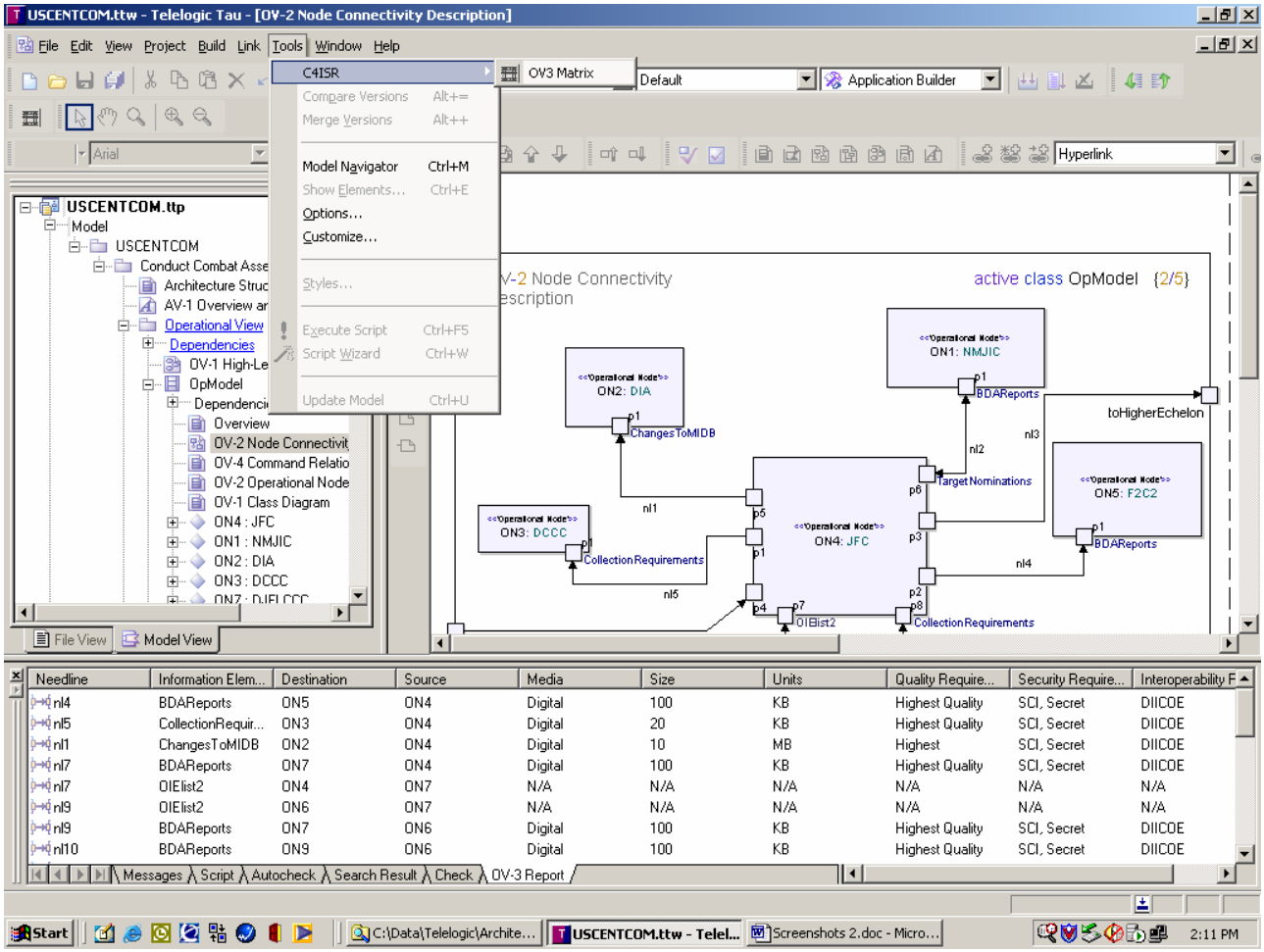


Figure 8: **OV-3 Operational Information Exchange Matrix**

Figure 8 shows the OV-3 Operational Informational Exchange Matrix report generated by TAU G2 in the bottom pane of the screen snapshot. The needlines and information elements are listed in the first two columns, and system requirements (quality, security and interoperability) are listed in the last three columns.

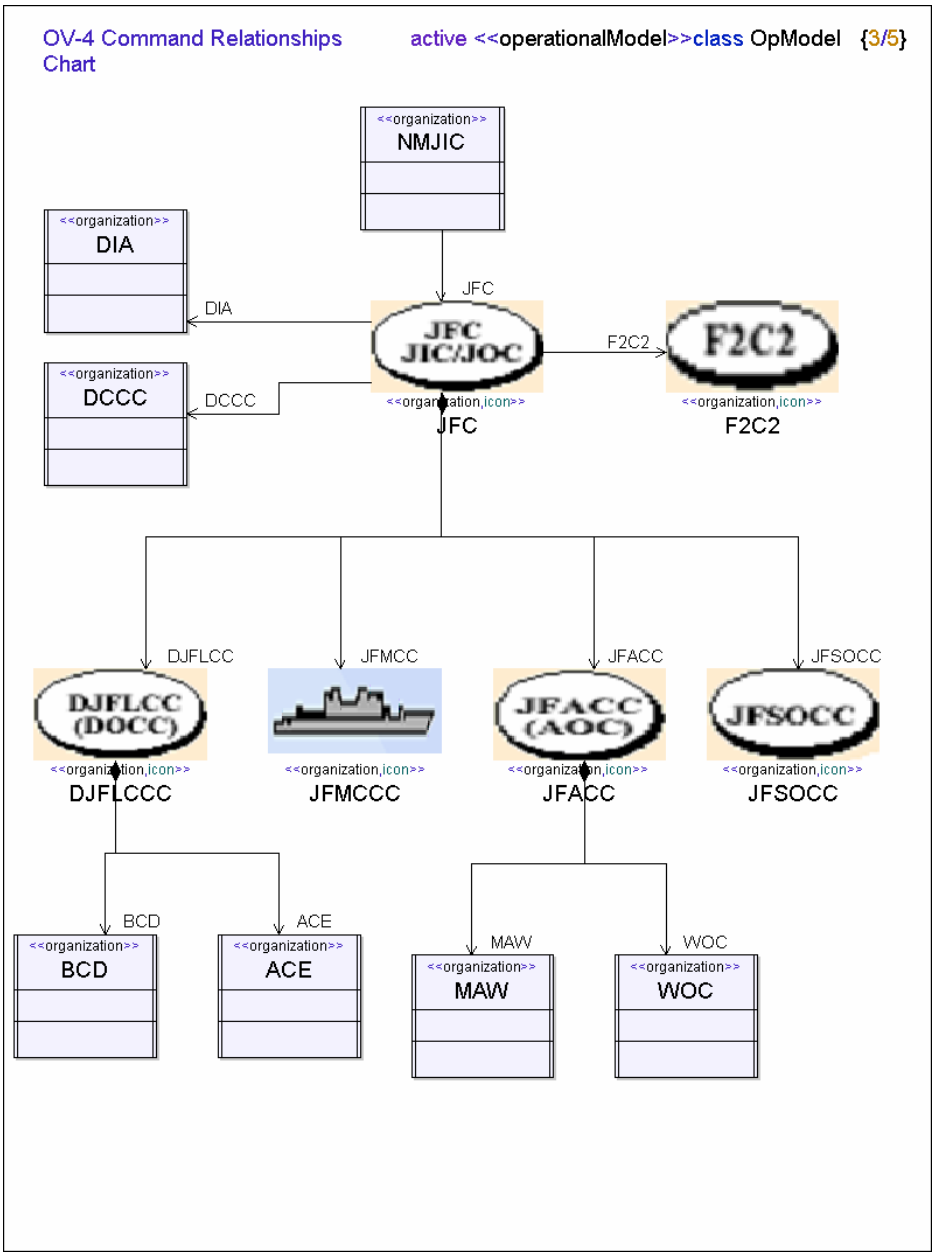


Figure 9: OV-4 Command Relationship Chart

Figure 9 shows the OV-4 Command Relationship Chart using a Class diagram. The Class diagram shows a collection of classes that represent organizational entities (e.g., *NMJIC*, *JFC*, *JFMCCC*). The hierarchical decomposition of organizations into sub-organizations is shown via composition (whole-part) relationships, where the filled diamond indicates the whole and the straight line indicates the part. For example, the *JFCC* organization is decomposed into the *DJFLCCC*, *JFMCCC*, *JFACC*, and *JFSOCC* sub-organizations.

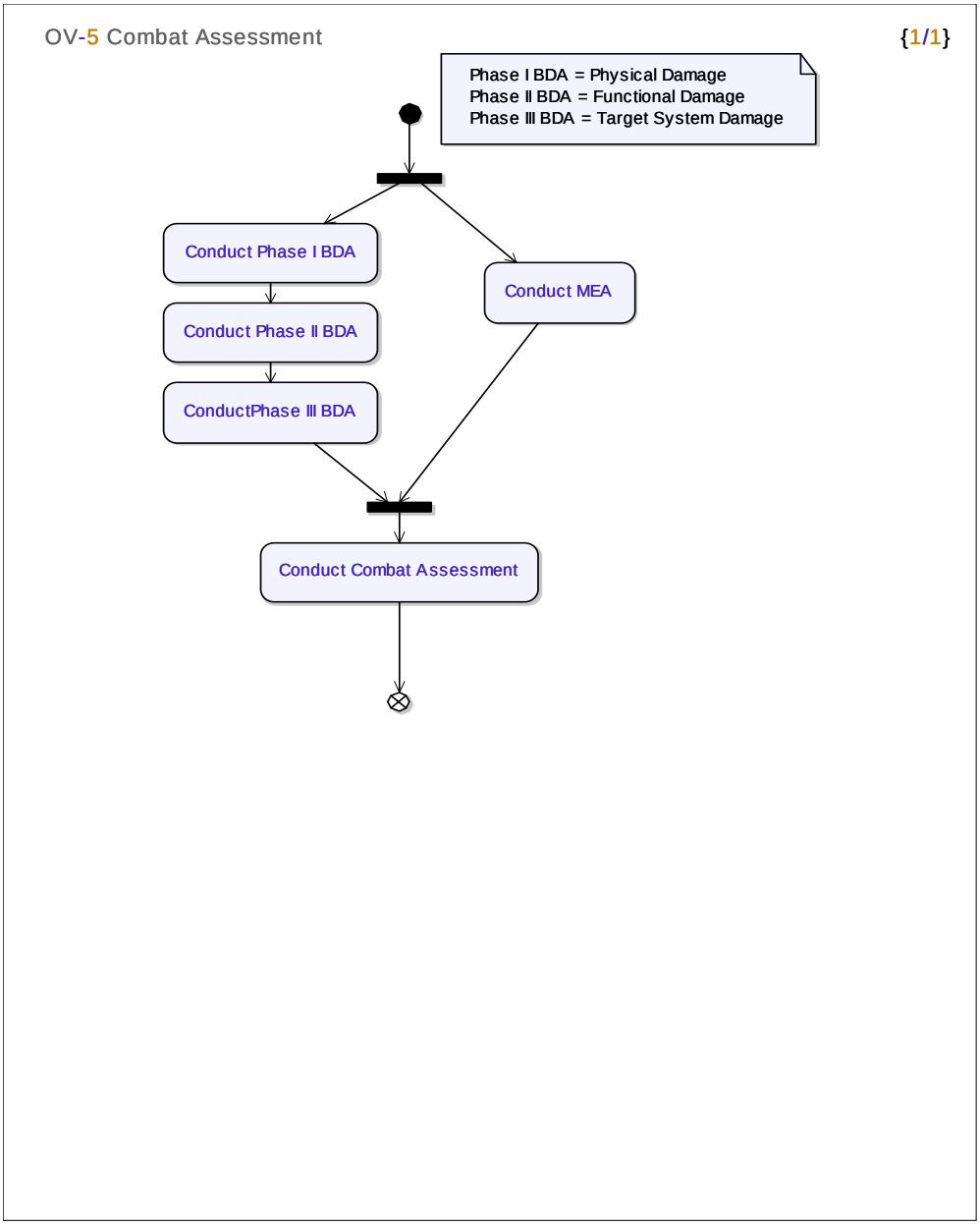


Figure 10: OV-5 Activity Model (Activity diagram)

Figure 10 shows the high-level OV-5 Activity Model using an Activity diagram without structural partitions (swimlanes). In this diagram the high-level activities are shown as action nodes (rounded rectangles), the initial node is shown as a filled circle, the flow final node is shown as a circumscribed 'X", and fork and join nodes are shown as thickened straight lines. The *Combat Assessment* activity is decomposed into the *Conduct Phase I BDA*, *Conduct Phase II BDA*, *Conduct Phase III BDA*, *Conduct MEA* and *Conduct Combat Assessment* action nodes.

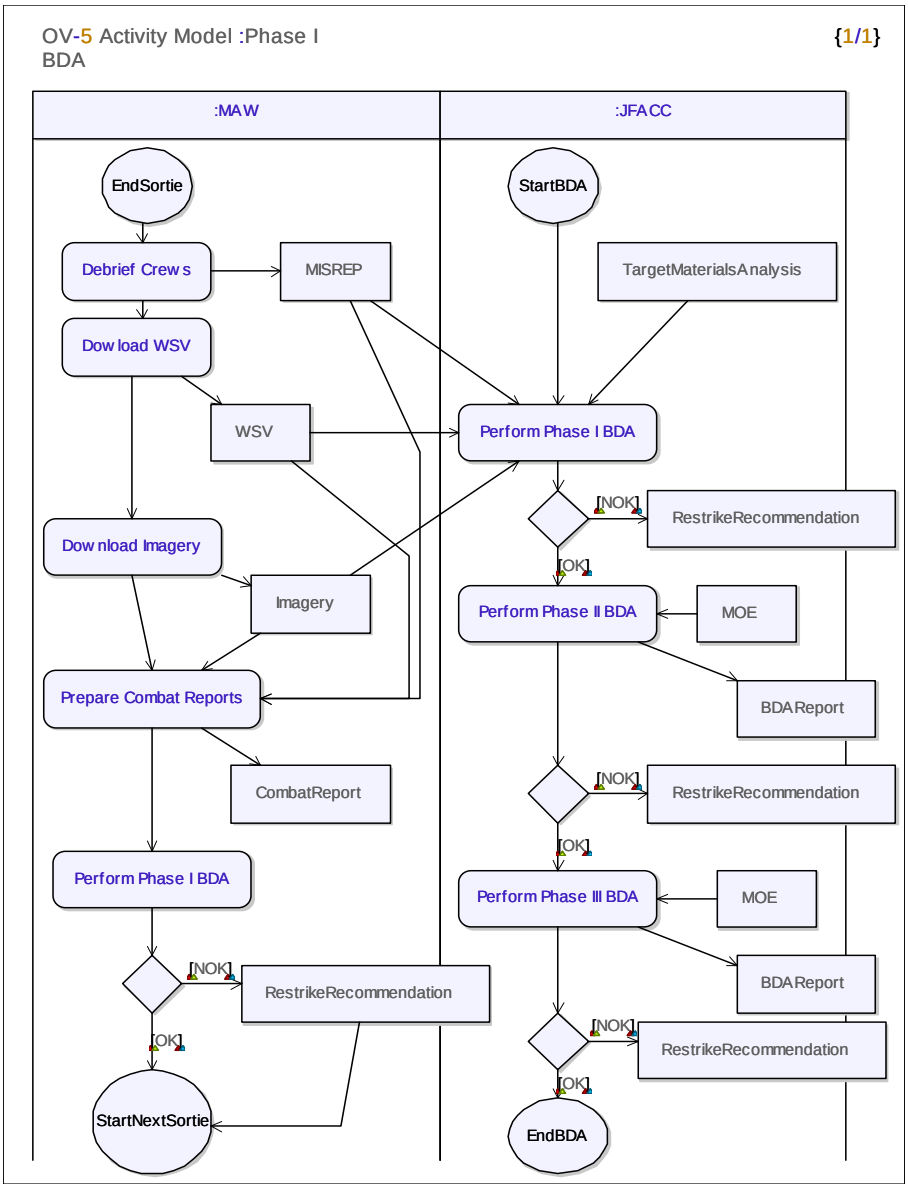


Figure 11: OV-5 Activity Model (details of Conduct Phase I BDA)

Figure 11 shows the OV-5 Activity Model for the details of the *Conduct Phase I BDA* action node previously shown in Figure 10. This Activity diagram includes two structural partitions (*:MAW* and *:JFACC*) and also shows object flows and decision nodes. For example, *Imagery* represents an object flow between the *Download Imagery* and *Prepare Combat Reports* action nodes, and the diamond-shaped node that flows from *Performance Phase I BDA* is a decision node with *[OK]* and *[NOK]* (i.e., not OK) branches.

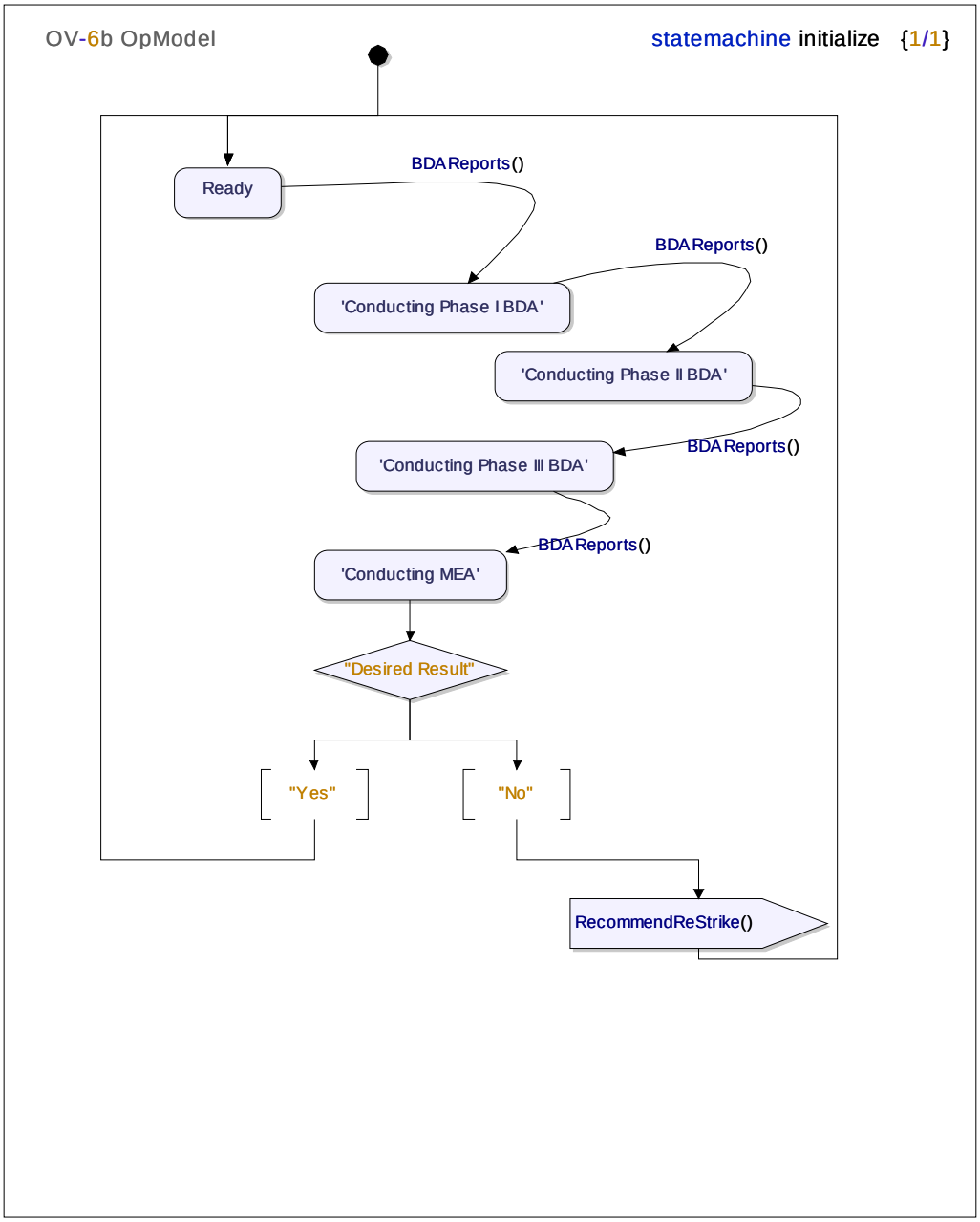


Figure 12: OV-6b Operational State Transition Diagram (state-centric)

Figure 12 shows the OV-6b Operational State Transition Diagram using a UML State Machine diagram with traditional state-centric notation, which emphasizes states more than the transitions between them. Compare with Figure 13, which shows a state machine using transition-centric notation.

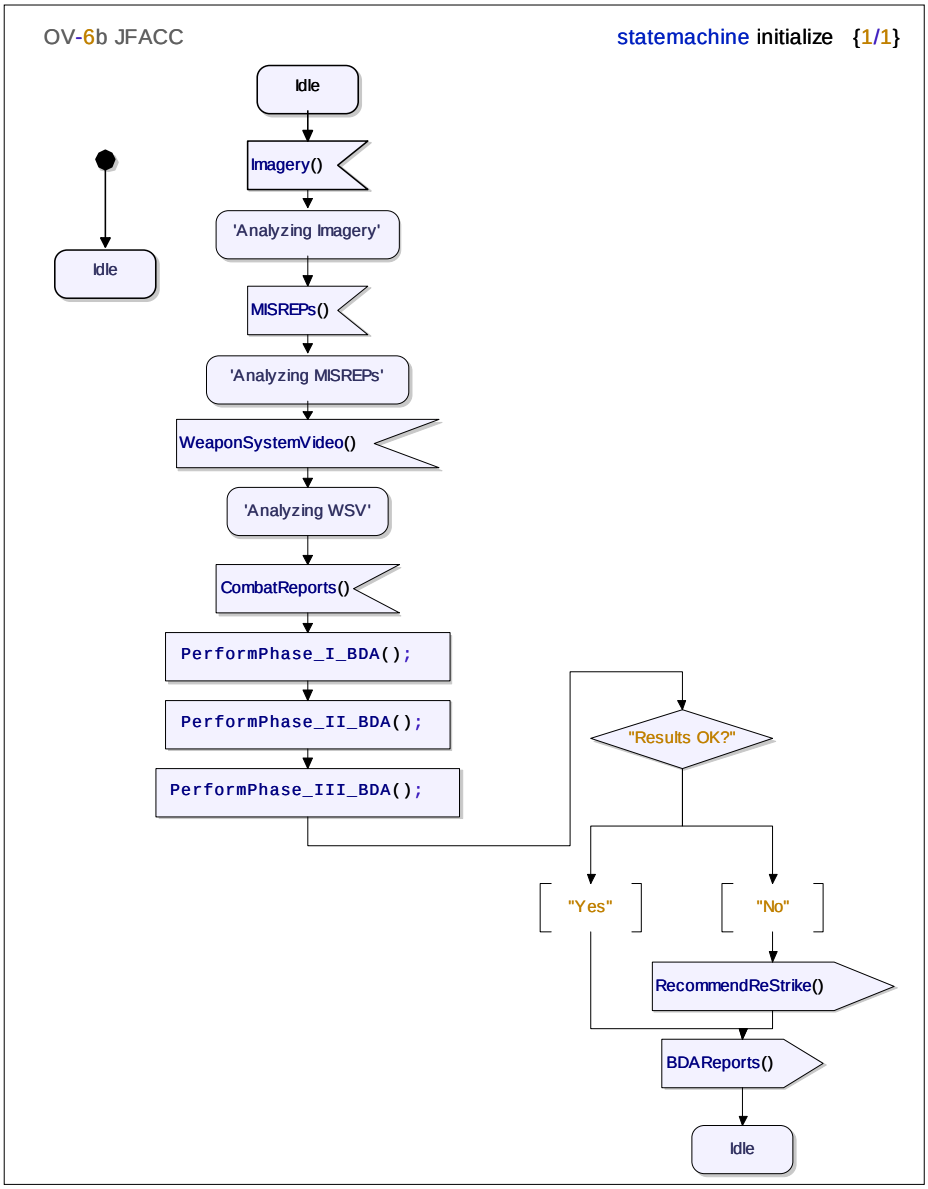


Figure 13: OV-6b Operational State Transition Diagram (transition-centric)

Figure 13 shows the OV-6b Operational State Transition Diagram using a UML State Machine diagram with transition-centric notation, which emphasizes transitions more than the states. Transition-centric state machine notation is a new UML 2.0 feature.

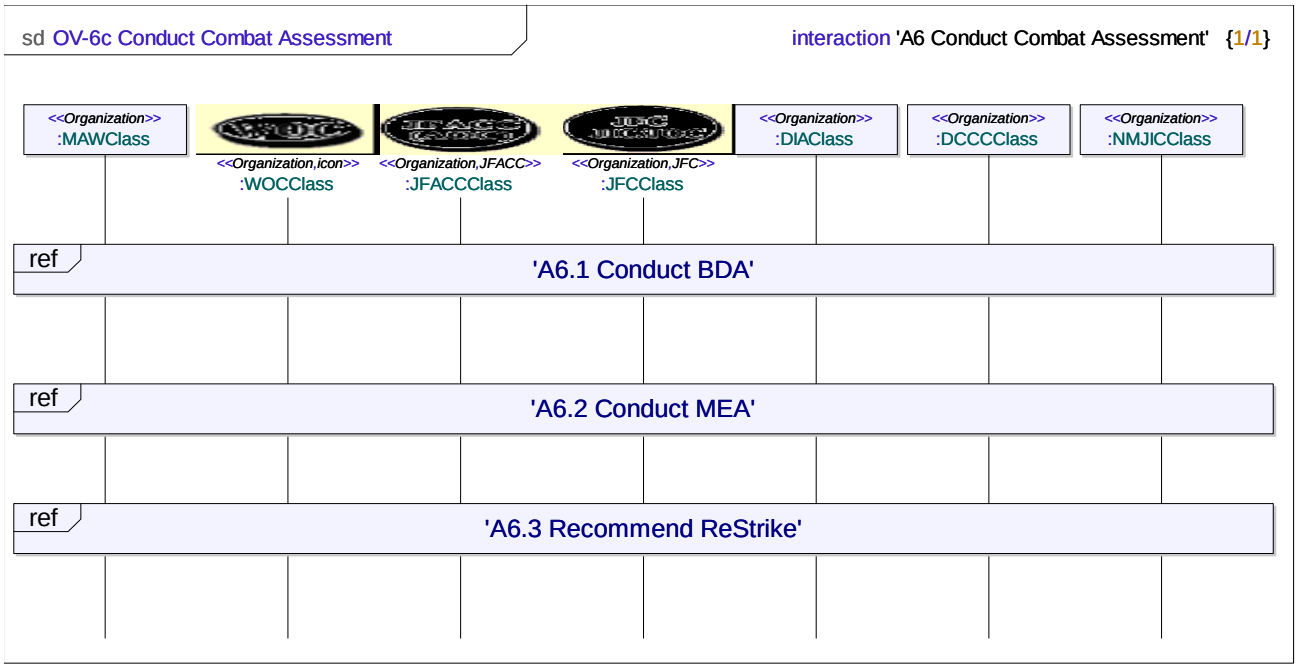


Figure 14: OV-6c Operational Event Trace Description (high-level)

Figure 14 shows the OV-6c Operational Event Trace Description using a UML Sequence diagram. The Sequence diagram decomposes the interaction Conduct Combat Assessment into three interaction occurrences (sub-sequences): *A6.1 Conduct EDA*, *A6.2 Conduct MEA*, and *A6.3 Recommend ReStrike*. Figure 15 shows how *A6.2 Conduct MEA* is further decomposed. The decomposition of interactions into interaction occurrences is a new UML 2.0 feature.

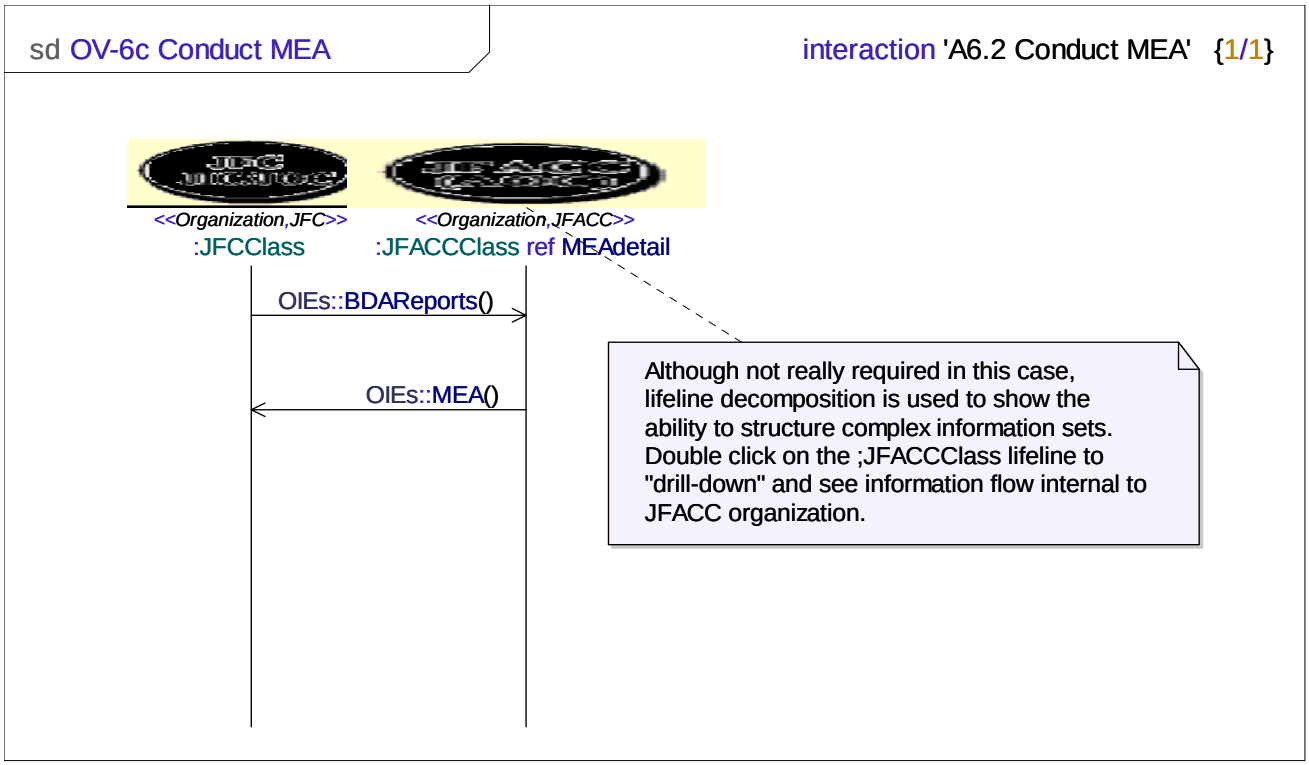
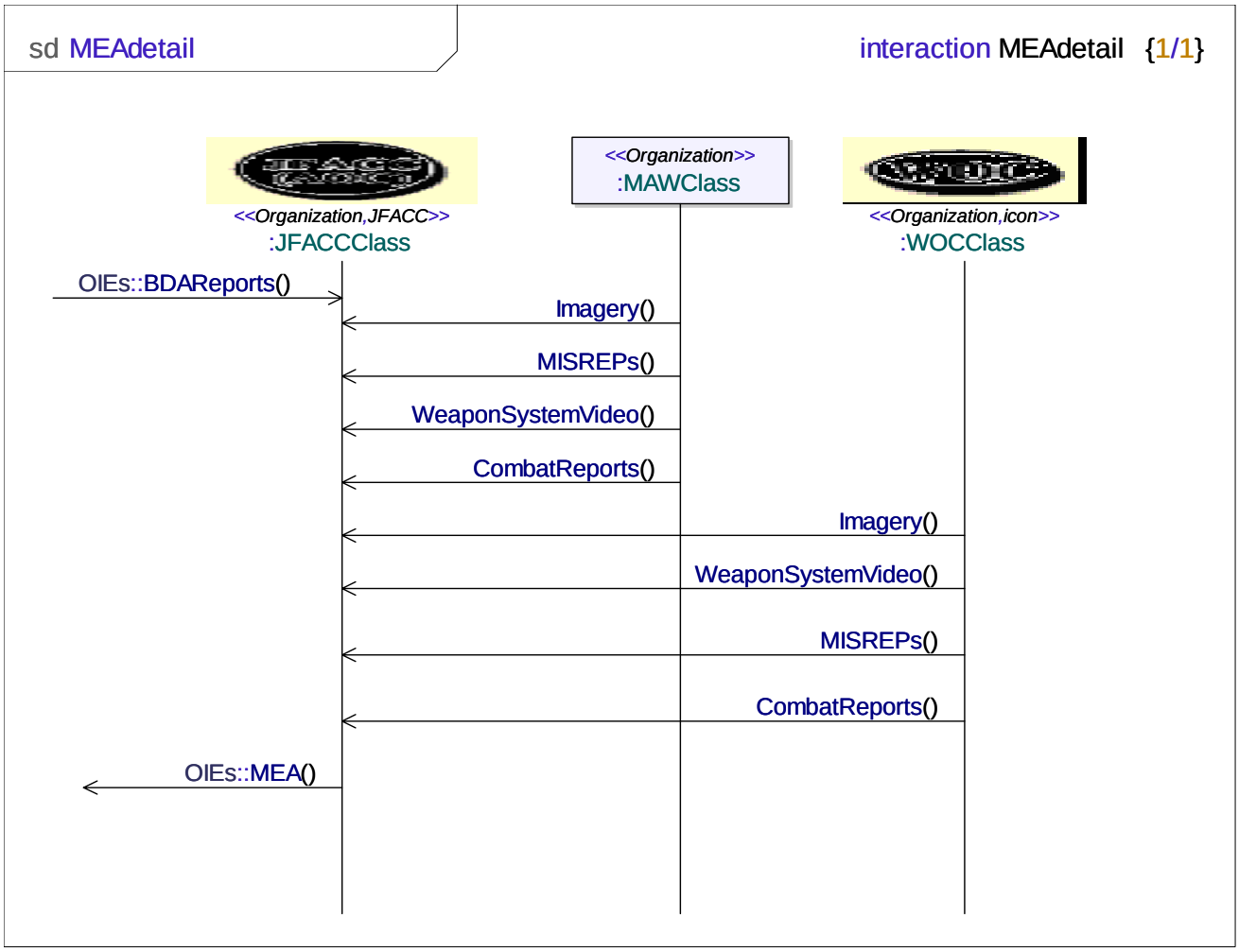


Figure 15: OV-6c Operational Event Trace Description (details of Conduct MEA interaction)

Figure 15 shows the OV-6c Operational Event Trace Description for the details of the Conduct MEA interaction occurrence previously shown in Figure 14. In this diagram there are two message sequences: the *:JFC* organization lifeline sends a *BDAreports ()* message to the *:JFACC* organization lifeline, which reciprocates with a *:MEA* message. As the comment indicates, UML 2.0 also supports the decomposition of lifelines as well as interactions. Figure 16 shows the decomposition of the *:JFACC ref MEA detail* organization lifeline.



:

Figure 16: OV-6c Operational Event Trace Description (details of :JFACC lifeline)

Figure 16 shows the OV-6c Operational Event Trace Description for the details of the *:JFACC* organization lifeline previously shown in Figure 15. The decomposition of lifelines is a new UML 2.0 feature.

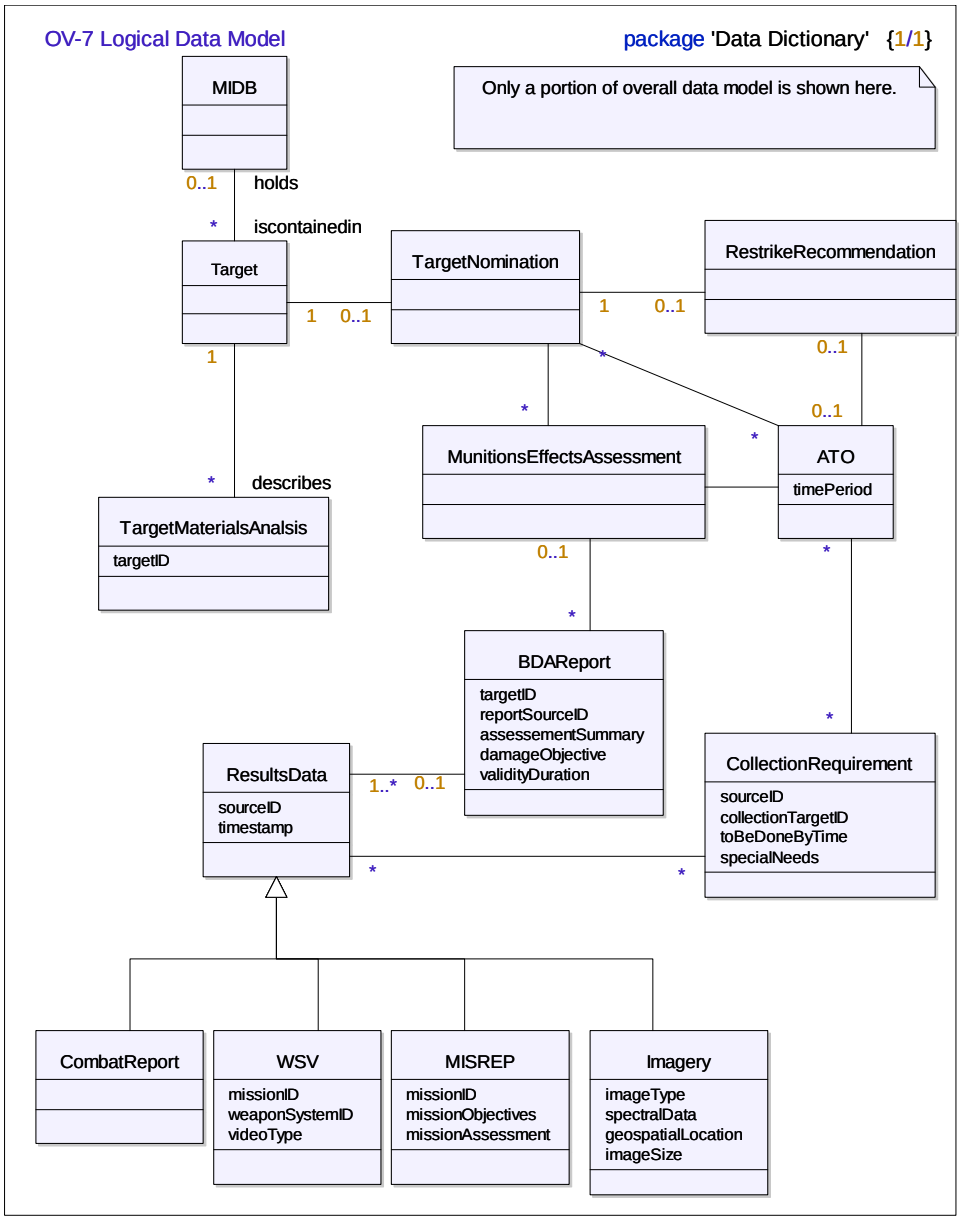


Figure 17: OV-7 Logical Data Model

Figure 17 shows the OV-7 Logical Data Model view using a Class diagram. The Logical Data Model describes the structures of the systems data types and their interrelationships. In this diagram the data types are shown as classes with attributes and without operations. The class relationships shown include associations with multiplicities (e.g., the many-to-many association between the *ResultsData* and *CollectionRequirement* classes) and generalization (e.g., *CombatReport*, *WSV*, *MISREP*, and *Imagery* classes are specializations of the *ResultsData* class).

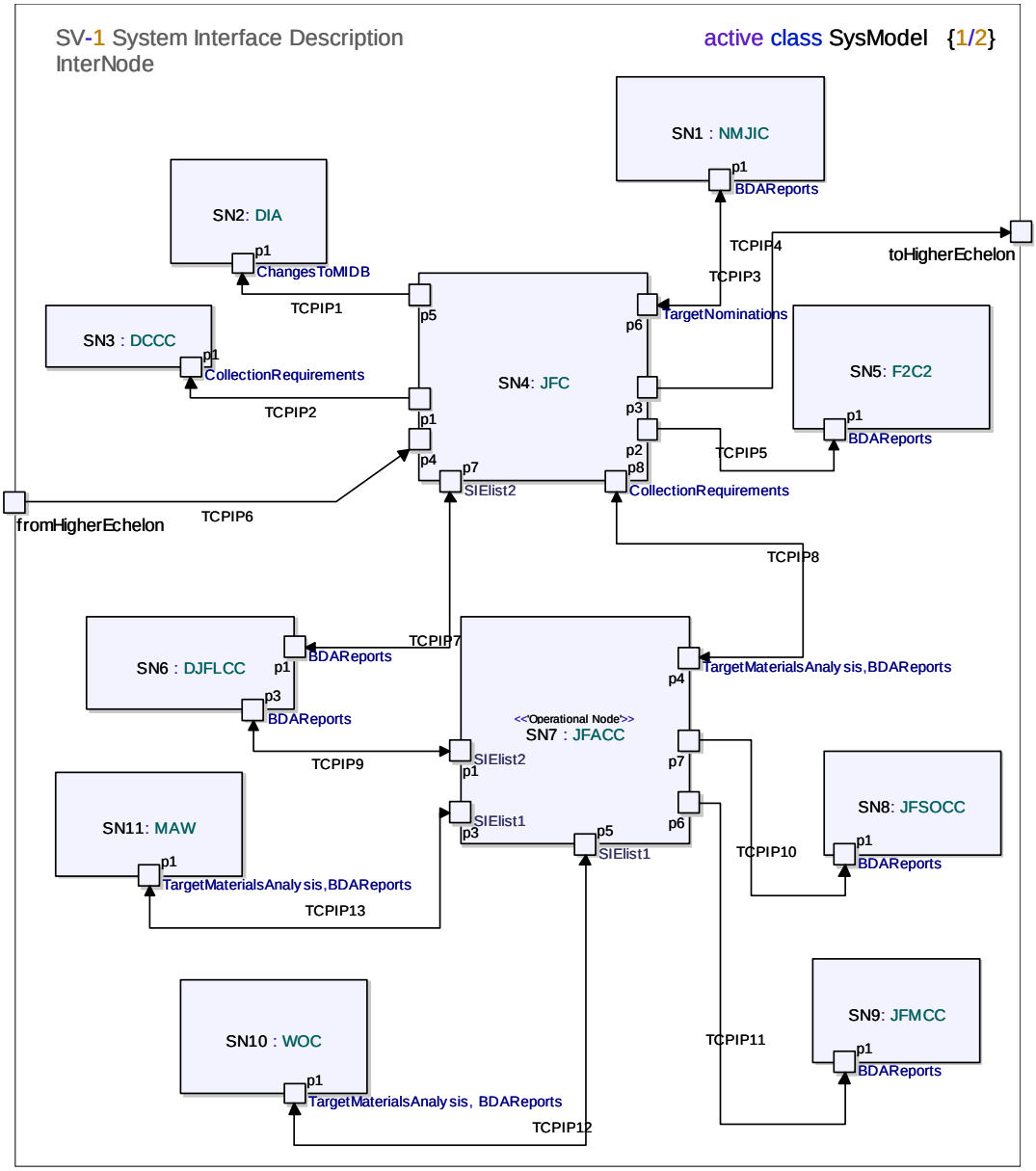


Figure 18: SV-1 System Interface Description

Figure 18 shows the SV-1 System Interface Description using a UML Composite Structure diagram. The SV-1 identifies system nodes and systems that support operational nodes, along with their interfaces. The example shows that the top-level system model (*SysModel*) can be decomposed into subsystem parts, such as *SN4:JFC*, where *SN4* is the part name and *JFC* is the part type. The parts are connected to each other via connectors, which attach to unique interaction points called ports. For example, the *SN4:JFC* part is connected to the *SN7:JFACC* part via the p8 and p4 ports, respectively. In this manner UML 2.0 parts, ports and connectors can be used to recursively decompose the system structure and precisely define the interfaces between parts at each level of decomposition.

CONCLUSIONS AND FUTURES

There is a growing realization in both commercial and defense industries that future architectures should be based on reusable architectural frameworks. In commercial industries the framework standards are frameworks and guidelines likes Zachman and FEAF; in defense industries the evolving framework standard is DoDAF, the successor to the C4ISR architectural framework specified in the late 1990s.

There are strong economic pressures to increase the quality and reduce the costs for aerospace-defense systems. Architecture frameworks, whether mandated by the government or developed internally, are a proven means for increasing quality while reducing costs.

What should we expect from DoDAF and other architectural frameworks during the next decade? We should expect them to evolve from fuzzy conceptual architectures into crisp technical architectures that solve tough engineering problems.

What should we expect from modeling tools that support DoDAF, such as TAU G2? In general, we should expect improved support for multiple views, increased automation, and better integration with other tools. The improved support for multiple views should include an integrated set of essential view templates; users should never need to develop basic views from scratch. The increased automation should include wizards and bots that will undertake the tedious bookkeeping required to maintain consistency across views. Better integration with other tools should facilitate seamless navigation with specialized tools for handing non-graphic views (e.g., matrices) and traceability to requirements, DoDAF specifications and related standards in order to demonstrate compliancy, like Telelogic DOORS.

This new generation of model-driven DoDAF solutions, such as Telelogic Enterprise Architect for DoDAF, will enable the defense community to ensure:

- **Consistency, correctness and completeness of DoDAF views and products**. Consistency (or model 'concordance') across all DoDAF views and products delivered by a powerful modeling environment and automatic propagation of changes. Correctness of DoDAF views and products is aided by the use of DoDAF-specific terminology, automatic detection of modeling errors and execution of models to verify their behavior. Completeness is enabled by support for all DoDAF products, including automatic generation of selected products from information contained in the DoDAF model.
- **Compliance of enterprise architectures with customer requirements, DoDAF specifications and standards.** Two-way integration between modeling and requirements management environments (like TAU and DOORS) eases establishment and maintenance of traceability across DoDAF products and to applicable specifications and standards. The ease of demonstrating compliance, as an enterprise architecture model evolves, leads to lower costs; and traceability enables complete assessment of the impact of changes, providing more accurate project planning.

- **Communication of enterprise architectures to customers, acquisition/procurement and systems/software engineers**. The DoDAF-specific modeling environment will enable architectures to be presented to end users in a form that is easily understood. Further, a key aim of defining enterprise architectures is to identify missing capabilities, which can then be acquired or developed. Solutions like Telelogic Enterprise Architect for DoDAF will support a smooth transition in either case: requirements for an acquisition/procurement can be further refined, RFPs (Request for Proposal) or ITTs (Invitation to Tender) produced, and responses assessed; or architectures can be reused, refined and realized by systems and software engineers, since the underlying modeling language, UML 2.0 is common throughout.

All of these key benefits add up to provide the means to achieve the aim of building defense enterprise architectures – to provide interoperable and cost-effective defense systems.

REFERENCES

Publications and Presentations

[Alberts 01] D. Alberts, et al., *Understanding Information Age Warfare*, CCRP, 2001.

[Buschmann 96] F. Buschmann, et al. *Pattern-Oriented Software Architecture: A System of Patterns.* Wiley, NY, 1996.

[Kobryn 04] C. Kobryn, “UML 3.0 and the Future of Modeling,” Journal on Software and Systems Modeling, vol. 3, no. 1, March 2004.

[Kobryn 00] C. Kobryn, "Modeling Components and Frameworks with UML," Communications of the ACM, vol. 43, no. 10, October, 2000.

[C41SR 97] *C4ISR Architectural Framework Version 2.0* December 1997 Department of Defense C4ISR Architectural Working Group.

[DoDAFv1 04] *DoD Architecture Framework: Volume I: Definitions and Guidelines* Version 1.0, DoD Architecture Framework Working Group.

[DoDAFv2 04] *DoD Architecture Framework: Volume II: Product Description* Version 1.0, DoD Architecture Framework Working Group.

[DoDAFv3 04] *DoD Architecture Framework: Volume III: Desktop* Version 1.0, DoD Architecture Framework Working Group.

[DoDacq 04] “SoS and FoS FAQ,” Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics, [http://www.acq.osd.mil/dpap/Docs/FAQs%20--%20SoS%20&%20FoS.doc], accessed April 16, 2004.

[DoDmemo 04] “Subject: The Department of Defense Architecture Framework (DoDAF),” Memorandum from Office of the Secretary of Defense, February 9, 2004.

[MDE 04] *Model-Driven Engineering with UML 2*, workshop training materials, PivotPoint Technology Corp., 2004.

[UML2 03] *UML Superstructure, version 2.0*, Final Adopted Specification, OMG document ad/03-08-02.

Web Resources

The latest DoDAF specifications are available from the Document Archive page of the DoD’s *Network & Information Integration* Web:

http://www.defenselink.mil/nii/

Information about the UML 2.0 Final Adopted Specifications is available from the following Web:

http://www.uml.org

Information about Telelogic's solution for defining and constructing DoDAF-compliant architectures can be found at:

http://www.telelogic.com/dodaf

ABOUT THE AUTHORS

Cris Kobryn is Telelogic's representative to the Object Management Group (OMG) and the International Council of Systems Engineers (INCOSE). Cris is an internationally recognized expert in software and systems modeling, and has successfully applied advanced technologies to diverse industries ranging from financial services and healthcare to telecom and aerospace. He has broad international experience leading high-performance software development teams, and has architected custom applications and commercial products. Cris formerly held senior technical positions at Telelogic, EDS, MCI Systemhouse, and SAIC.

As an Object Management Group representative, Cris has been a major contributor to the Unified Modeling Language (UML) specification, which is the industry standard for specifying software architectures. He chaired large international standardization teams to specify UML 1.1 and UML 2, and serves as the co-chair of the OMG's Analysis and Design Task Force. Cris is a member of the IEEE, ACM, INCOSE and AAAI. Contact him at cris.kobryn@telelogic.com.

Chris Sibbald is the Director of Application Engineers for Telelogic, North American Inc. Dr. Sibbald holds a Ph.D. in electrical engineering from the University of Ottawa. He has over 10 years of systems engineering experience.

Chris was the Payload Systems Engineer for the Canadian Space Agency's RADARSAT spacecraft. Following the launch of RADARSAT in 1995 Chris ran an Electromagnetic Compatibility consultancy that provided expert services to military and aerospace customers. From 1997 to 2000 Chris was the President of Objective SST an engineering consultancy in Ottawa that provides software and systems development process mentoring and consulting.

Chris has worked with over thirty companies in the military, aerospace, telecommunications, financial, and software segments, providing expert training and guidance on requirements management process, systems and software engineering, and Telelogic DOORS®. He lives in Ottawa, Ontario, Canada with his wife Ann Marie, daughter Stephanie and son Liam. Contact him at chris.sibbald@telelogic.com.