I-Logix

DODAF Architectures in UML

Bruce Powel Douglass, PhD

What is DODAF?

The DODAF Architecture Framework is a semantic framework for developing, representing, and integrating architectures in a consistent way for the Department of Defense applications [1] *DoD Architectural Framework Version 1.0: Deskbook* DoD Architectural Framework Working Group . The DoDAF specification is a recent upgrade to the 1997 C^4ISR-AF specification. It was conceived as a way of providing a common means to specify systems for the Department of Defense (DoD) in its many facets and programs. The DODAF specification defines architecture to be:

> An architecture description is a representation of a defined domain, as of a current or future point in time, in terms of its component parts, what those parts do, how the parts relate to each other, and the rules and constraints under which the parts function. What constitutes each of the elements of this definition depends on the degree of detail of interest. For example, domains can be at any level, from DoD as a whole down to individual functional areas or groups of functional areas. Component parts can be anything from "U.S. Air Force" as a component of DoD, down to a "satellite ground station" as a component part of a communications network, or "workstation A" as a component part of system "x." What those parts do can be as general as their high- level operational concept or as specific as the lowest- level action they perform. How the parts relate to each other can be as general as how organizations fit into a very high- level command structure or as specific as what frequency one unit uses in communicating with another. The rules and constraints under which they work can be as general as high- level doctrine or as specific as the e-mail standard they must use.

The term *architecture* is generally used both to refer to an architecture description and an architecture implementation. Hereafter in this document, the term *architecture* will be used as a shortened reference to *architecture description*. Occasionally the term *architecture description* is used for emphasis. References to architecture implementations will use the term *architecture implementation*. An architecture description is a representation of a current or postulated "real-world" configuration of resources, rules, and relationships. Once the representation enters the design, development, and acquisition portion of the system development life-cycle process, the architecture description is transformed into a real implementation of capabilities and assets in the field. The Framework itself does not address this

representation-to-implementation transformation process but references policies that are relevant to that process.

The changes implemented in the creation of the DODAF specification include:

- Inclusion of comments and suggestions from the development community
- Reorganization into three sections:
  - Volume 1 includes guidelines
  - Volume 2 include product (artifact) definitions
  - Deskbook contains supplementary information
- Product descriptions have been revamped and suggestions for UML representations have been included in Volume 2
- Processes and techniques for representing architectures have been included
- Analytic techniques for using the architectural products to support DoD processes are included in the Deskguide, including:
  - Air Force's Task Force capability-based analysis
  - Navy's Mission Capability Package analysis approach
  - Office of the Assistant Secretary of Defense for Networks and Information Integration/J6 Key Interface process for addressing interoperability at interfaces
  - Architecture input to C4I Support Plans
  - The role of architectures in Capital Planning and Investment Control

The purpose of the DODAF is to provide assistance in the specification of architectures. Architecture itself has a number of definitions. The DODAF uses the definition of IEEE 610.12[2] *IEEE Standard Glossary of Software Engineering Terminology*, 1990, IEEE STD 610.12-1990, Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.:

> The structure of components, their relationships, and the principles and guidelines governing their design and evolution over time.

Architectures in the DODAF have three fundamental views – operational, systems, and technical. The emphasis in each of these views is, of course, different and distinct, but they overlap to a significant degree.

The *operational view* is a description of the tasks and activities, operational elements, and information flows required to accomplish or support a military operation. This view includes doctrine (which in another environment might be called "business rules"), activities, assignment of these activities to operational elements, as well as the sequences and time frames of the execution of the activities. Operational architectures are usually independent of the systems used to implement them.

The *systems view* is a description of the systems and their interconnections providing for, or supporting, warfighting functions. The systems view includes the large-scale elements and objects that interact to achieve the operational goals as well as their locations, interconnections, etc. The systems involved may include key nodes (including

material), networks (as well as interconnections and interfaces), war fighting platforms, weapons systems, and so on, as well as their various qualities of service such as MTBF, maintainability, speed, capacity, availability, etc. Systems described in the systems view can be used to achieve many different operational architectures, organizations and missions. The systems view does depend on the underlying technology described in the technical view and are constrained by their limitations.

The *technical view* provides the minimal set of rules governing the arrangement, interaction, and interdependence of system parts or elements, whose purpose is to ensure that a conforming system satisfies a specified set of requirements. The technical view provides the basis for engineering specification of the systems in the systems view and includes technical standards. In other words, the technical view is the engineering infrastructure that supports the systems view. Figure 1 shows the high-level relationships between the three architectural views.

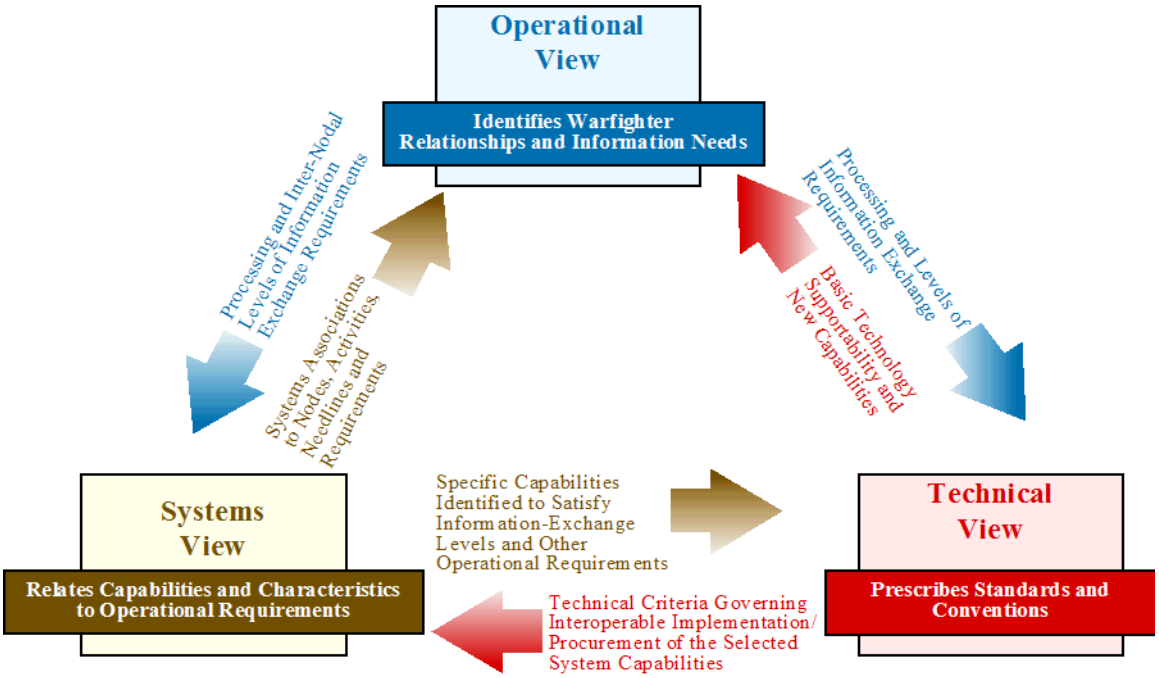

Figure 1: Relationships Between the Views

Within each of these architectural areas, the standard defines work products. The list of these products is given in Table 1. Each of these products will be discussed and in most cases a UML view that meets both the needs and intent of the product will be shown. The products shown in shaded background are “required for an integrated architecture.”

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	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, operational activities performed at each node, 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	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 the three products used to describe operational activity sequence and timing - identifies business rules that constrain operation
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
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
Operational	OV-7	Logical Data Model	Documentation of the data requirements and structural business process rules of the Operational View.
Systems	SV-1	Systems Interface Description	Identification of systems and system components and their interconnections, within and between nodes
Systems	SV-2	Systems Communications Description	Systems nodes 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 interfaces, planned vs. existing interfaces, etc.
Systems	SV-4	Systems Functionality Description	Functions performed by systems and the information flow among system functions
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
Systems	SV-6	Systems Data Exchange Matrix	Provides details of systems data being exchanged between systems
Systems	SV-7	Systems Performance Parameters Matrix	Performance characteristics of each system(s) hardware and software elements, for the appropriate timeframe(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 timeframes, and that will affect future development of the architecture
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
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
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
Systems	SV-11	Physical Schema	Physical implementation of the information of the Logical Data Model, e.g., message formats, file structures, physical schema
Technical	TV-1	Technical Standards Profile	Extraction of standards that apply to the given architecture
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

Table 1: DODAF Work Products

[6] *Real-Time UML: Advances in the UML for Real-Time Systems* Douglass, Bruce Powel; Addison-Wesley, 2004 discusses the specification of C4ISR and DODAF products with the UML. That information is summarized here.

Products of DODAF

The DODAF identifies a large number of artifacts, called *products* that are used in the description of the various architecture views. Some of these are required for compliance to the DODAF but most are optional and can be used when appropriate. These are referred to as "supporting products." In this section, we will discuss the required products from Table 1.

AV-1 Overview and Summary Information

The Overview and Summary Information product is an essential artifact for projects to be compliant with the DODAF. [1] *DoD Architectural Framework Version 1.0: Deskbook* DoD Architectural Framework Working Group provides a list of the required contents of this artifact as well as a sample format. This information may be directly entered into the model in the model description field as shown in Figure 2, or may be entered in a separate textual document and a hyperlink to that document may be placed in the Rhapsody model.

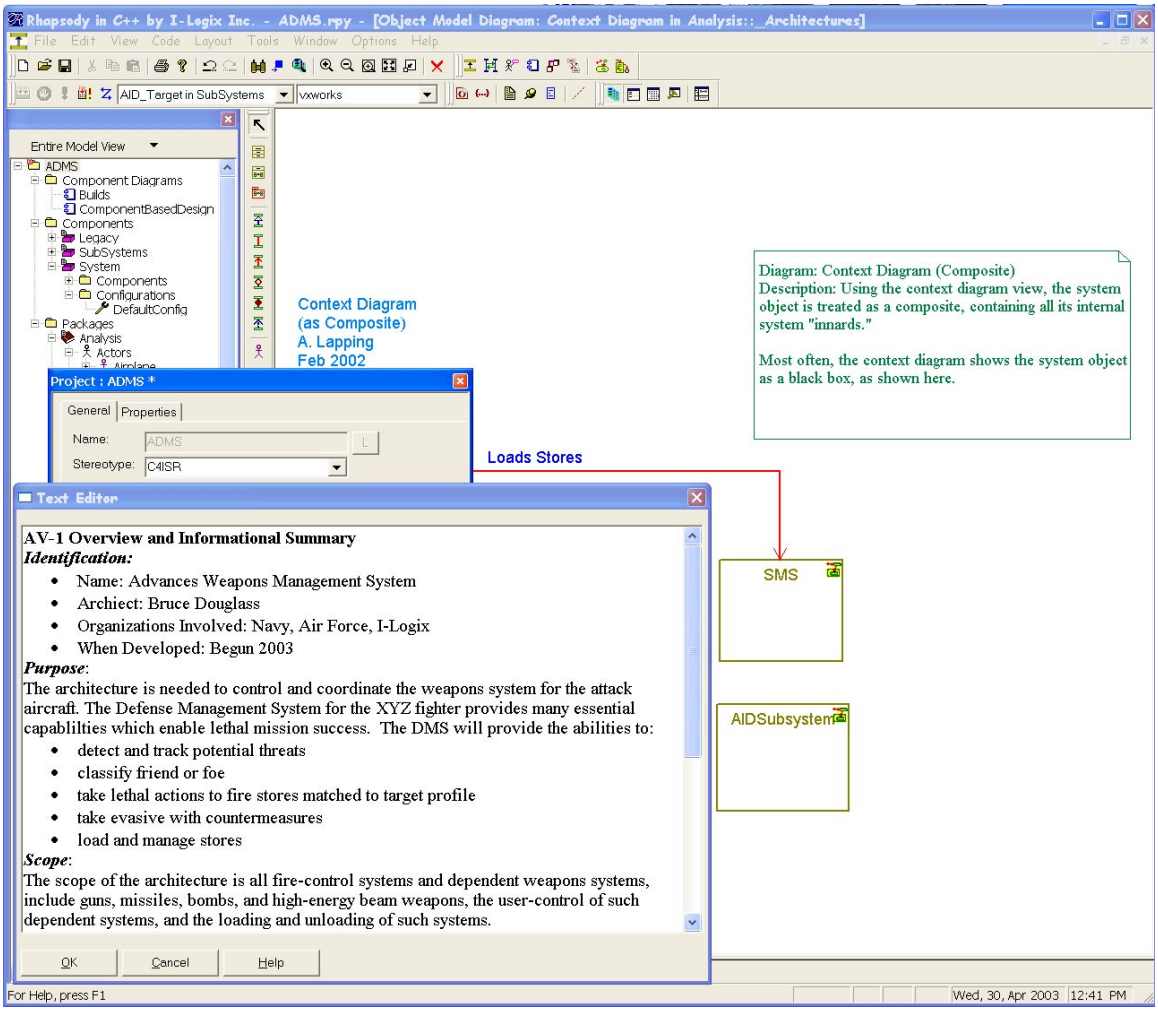

Figure 2: Report on Model for AV-1 Overview

AV-2 Integrated Dictionary

The Integrated Dictionary product defines the metadata of the product and is normally provided in a textual output. This metadata is maintained for you automatically by Rhapsody and may be viewed in the Rhapsody Browser or may be used to generate reports of various kinds and desired details, as illustrated in Figure 3. Besides the built-in “report on model” which can generate reports, Rhapsody has a powerful reporting facility known as ReporterPLUS that allows reports to be generated in customizable formats and templates.

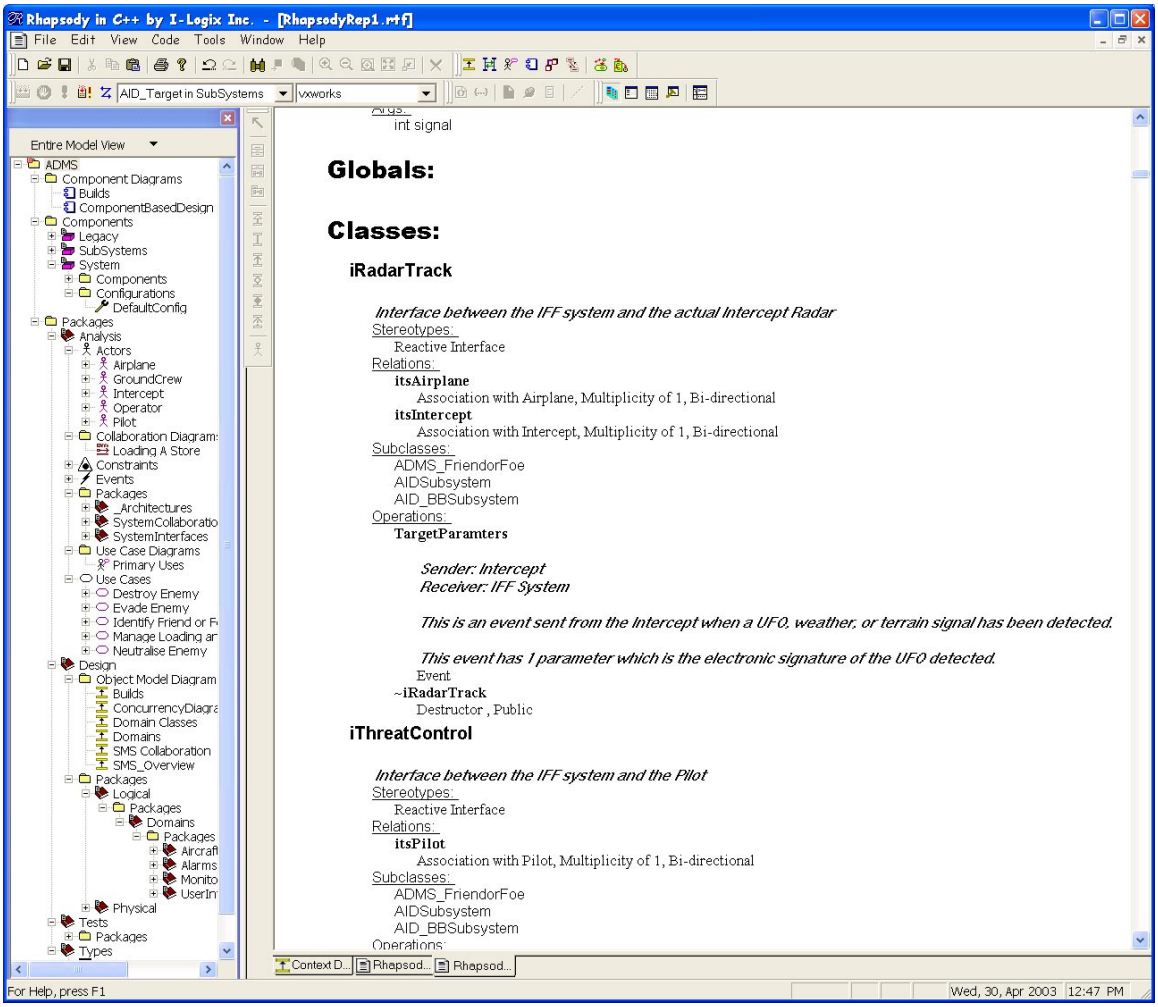

Figure 3: AV-2 Integrated Dictionary

OV-1 High-Level Operational Concept Graphic

This is a very general architectural picture of the architecture-description products. It is used to facilitate human-human communication, especially high-level decision makers. Commonly, it graphically depicts the coordinated deployment of systems to achieve the operational objectives. This is easily done in a class or deployment diagram, using stereotypes to identify the various kinds of systems involved in the operational concept. Figure 4 shows an example operational concept diagram as a class diagram using standard UML elements – classes with appropriate stereotypes and dependency relations among them.

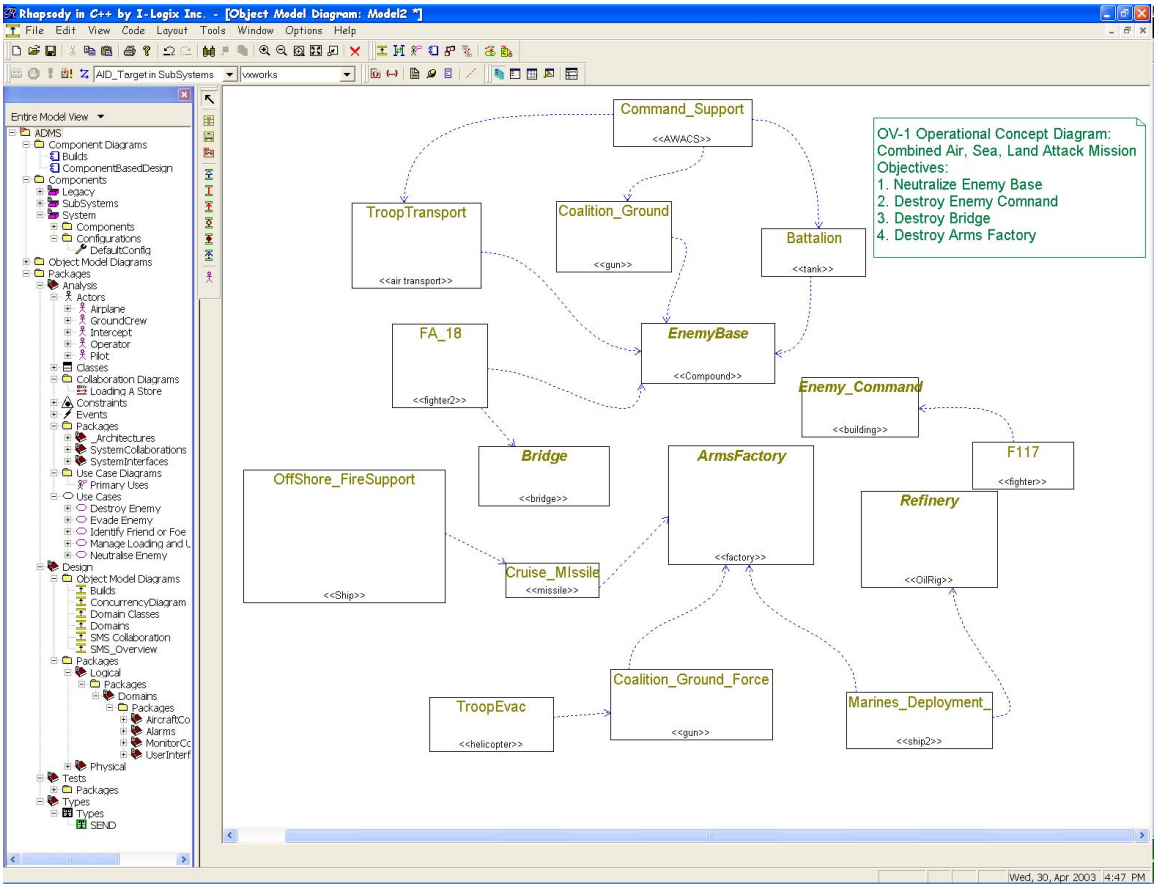

Figure 4: OV-1 Operation Concept Diagram with Standard Notation

Figure 5 shows the very same diagram but using meaningful bitmap icons to represent the operational elements. This is very simple to do with Rhapsody and simplifies the interpretation of the graphic.

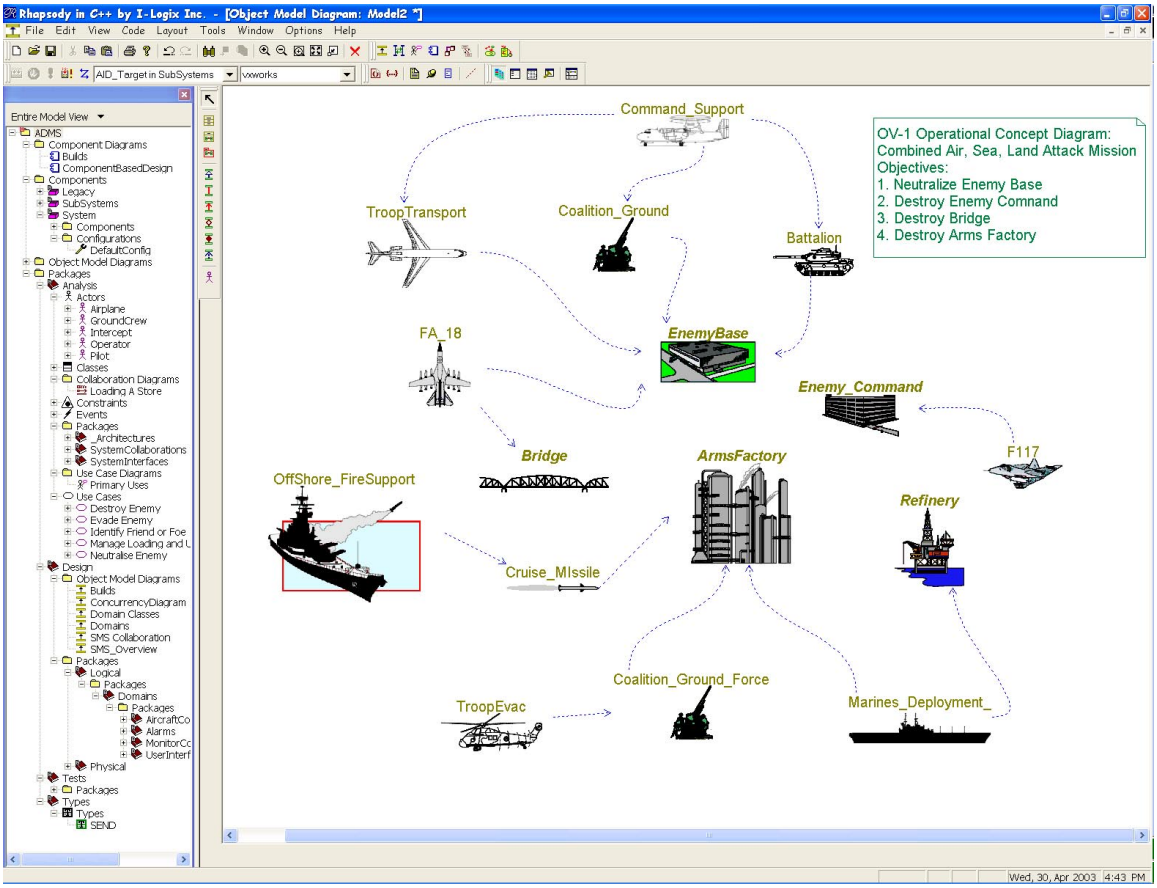

Figure 5: OV-1 Operational Concept Diagram in Rhapsody with Icons

OV-2 Operational Node Connectivity Description

The Operation Node Connectivity Description identifies the operational nodes, their connections, and the information shared among them. In the UML, operational nodes may be represented as classes on class diagrams or as nodes on deployment diagrams. In Figure 6[1], operational nodes and subnodes are shown as classes; the interfaces among the operational nodes are mediated via the associations. The actual information content transmitted along these associations is captured in constraints (the notes with the curly braces), and supporting information is shown in either free text or comments within note boxes.

[1] The example is taken from [1] and recast into UML notation and semantics.

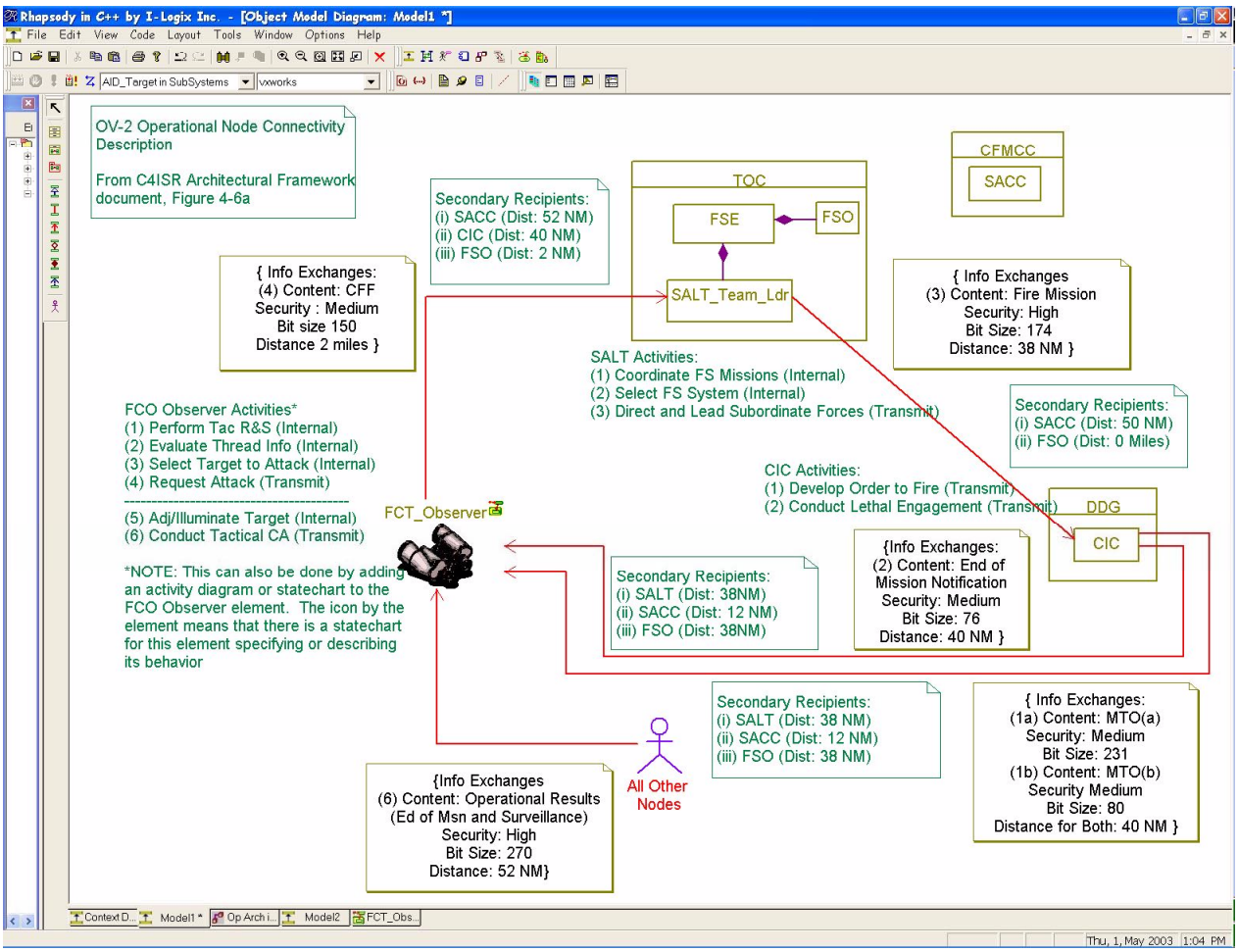

Figure 6: OV-2 Operational Node Connectivity with Classes

The same information is shown in the following diagram, except that the operational nodes are instead shown as nodes on a deployment diagram. In general, classes have richer semantics than nodes on deployment diagrams, and are usually preferred for that reason.

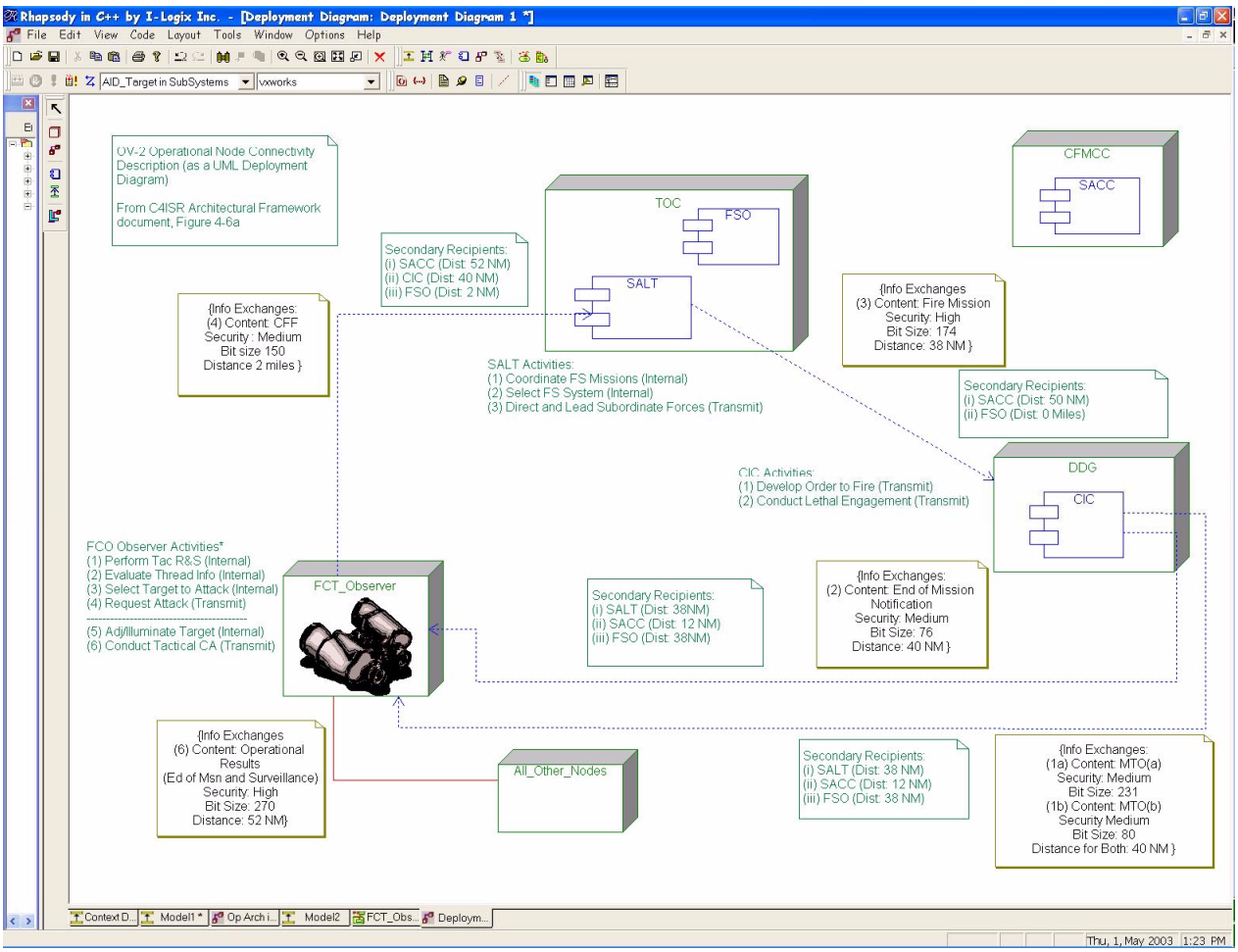

Figure 7: OV-2 Operational Node Connectivity with Deployment Diagram

OV-3 Operational Information Exchange Matrix

The Operational Information Exchange Matrix expresses the relationship between activities, operational elements and information flow, focusing on the latter. The UML doesn't have a "matrix" notation, but this can be cast as a specialized format of a report constructed from the model repository held by Rhapsody.

Alternatively, the data flow notation of the UML 2.0 can easily depict the information exchange among operational elements. In the UML diagram in Figure 8, icons are used to represent operational elements (modeled with UML classes) with information flows among these elements.

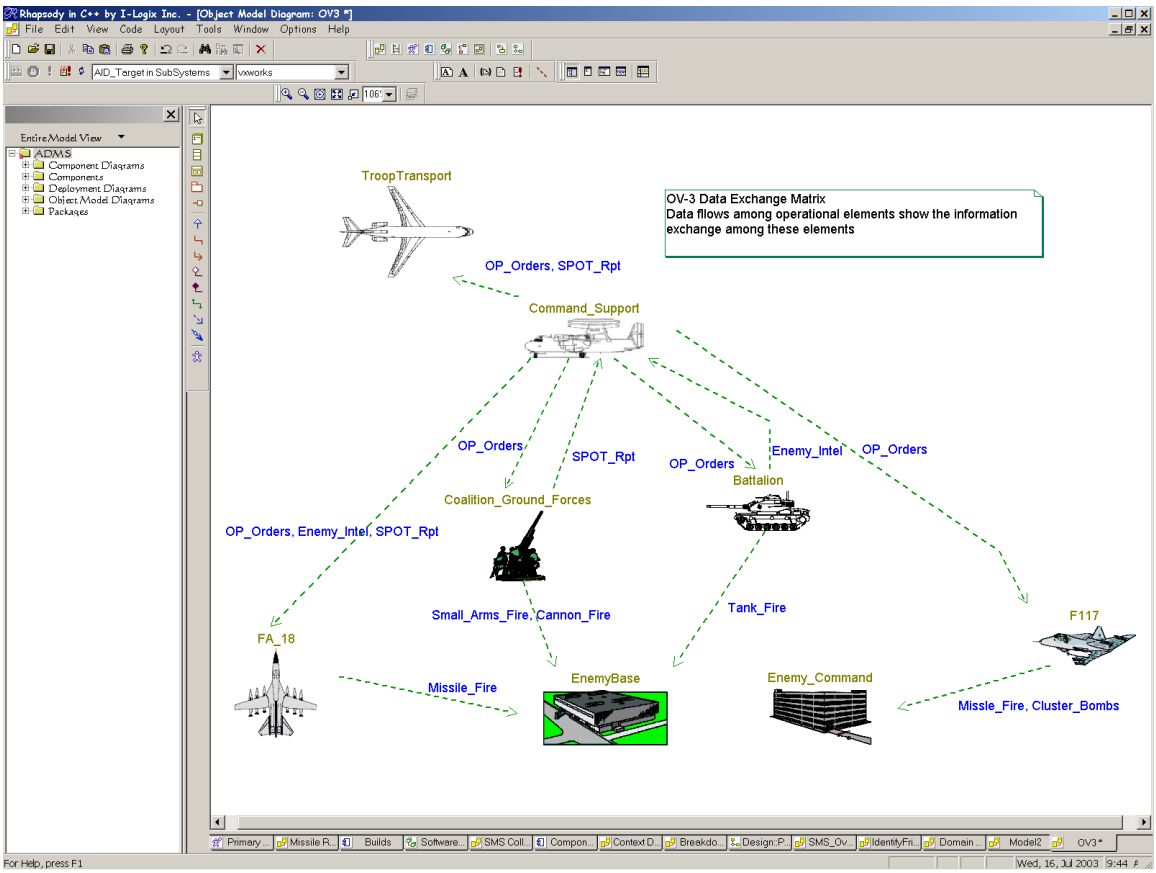

Figure 8: OV-3 Data Information Exchange

OV-4 Command Relationships Chart

The OV-4 diagram is used primarily as the command structure of some unit or units of battle. The UML view for this is a class diagram, using aggregation to show “ownership.” Other relations may be added; for significant communications between subunits or individuals, associations should be used. For less obvious relations, dependencies may be used. Figure 9 shows a relation set including ownership but also assignment relations, when a person is assigned to one unit or individual but commanded by another.

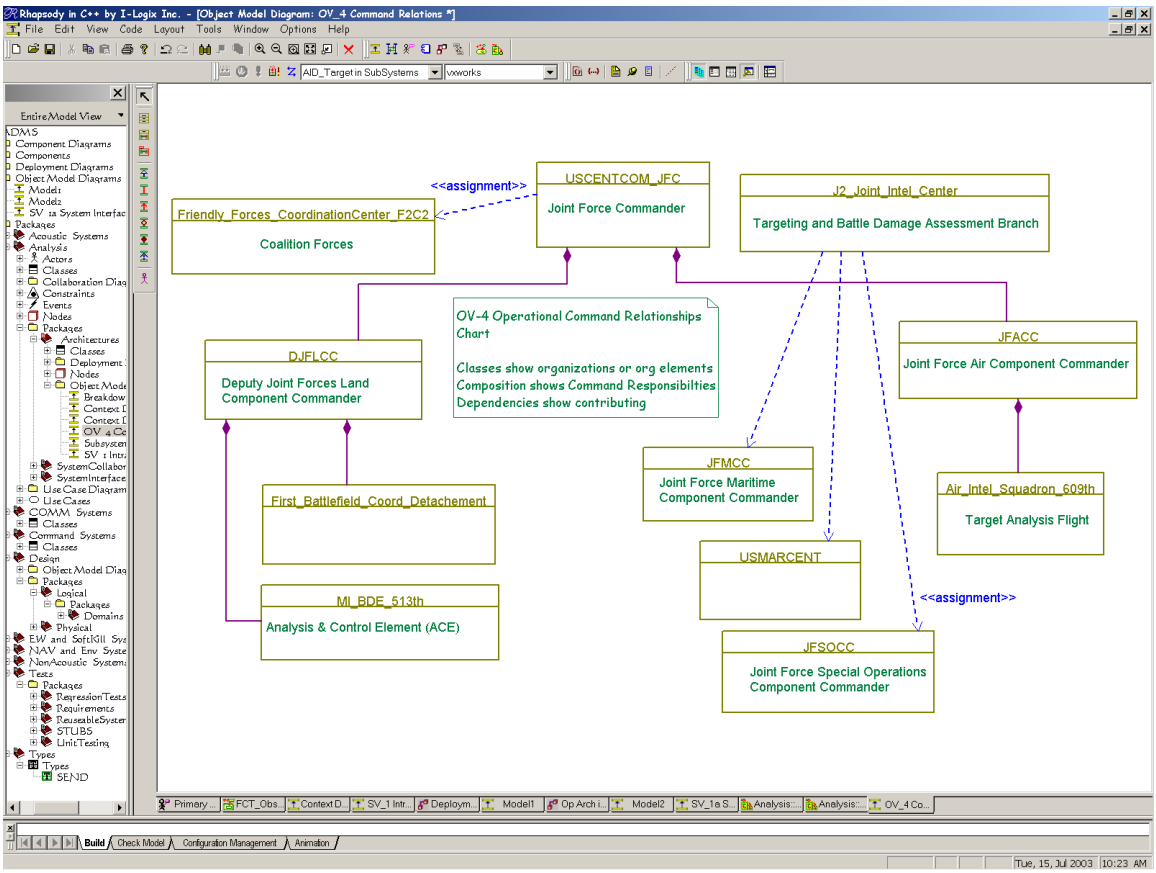

Figure 9: OV-4 Command Relationship Chart

OV-5 Operational Activity Model

OV-5, the Operational Activity Model specifies the flow of execution among activities, possibly with the generation of artifacts such as reports and OP Orders. Figure 10 shows a simple model that uses the UML Activity Diagram to show the primary activities and their sub-activities to achieve a Joint Force Targeting activity. The nesting depicts the whole-part nature of the activities (i.e. activities that can be divided into smaller sub-activities). The arrows depict the flow of execution as the activities complete over time. The bars indicate forks or joins; a fork indicates a branching into concurrent (simultaneously executing) activities, while a join indicates a merging together of previously concurrent activities.

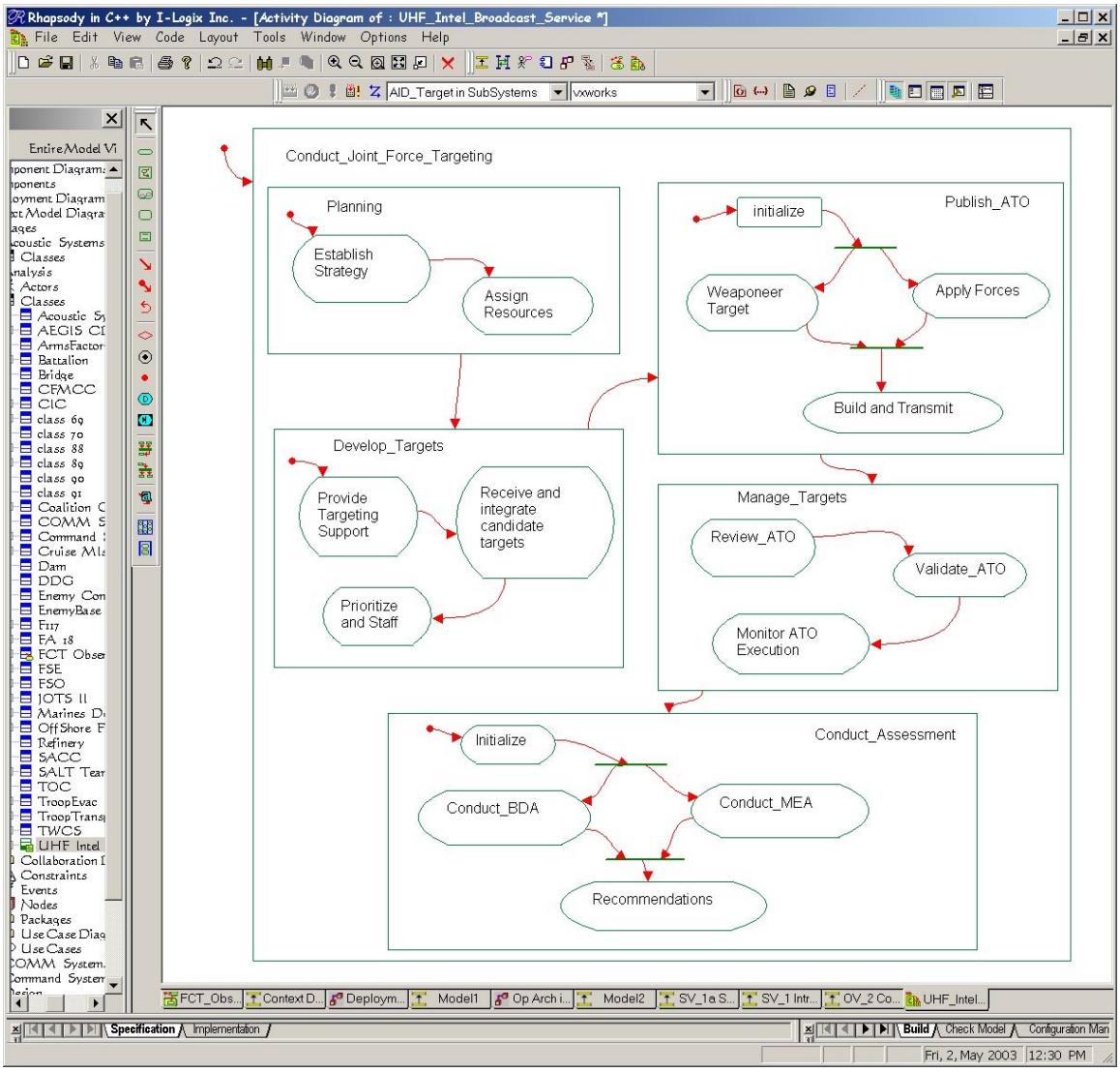

Figure 10: OV-5 Operational Activity Model

Often, activities are performed by a set of operational or system elements collaborating together. This can be indicated using swimlanes to divide the activities into activities done by a single operational or systems entity. An abstract example is shown in Figure 11, with swimlanes for External Elements, Special Ops, CIC, Batallion Commander, and Air Support. This diagram shows forks and joins as well as branch points (indicated with a diamond). A fork differs from a branch in that all transitions from a fork become simultaneously active while only a single transition from a branch is taken.

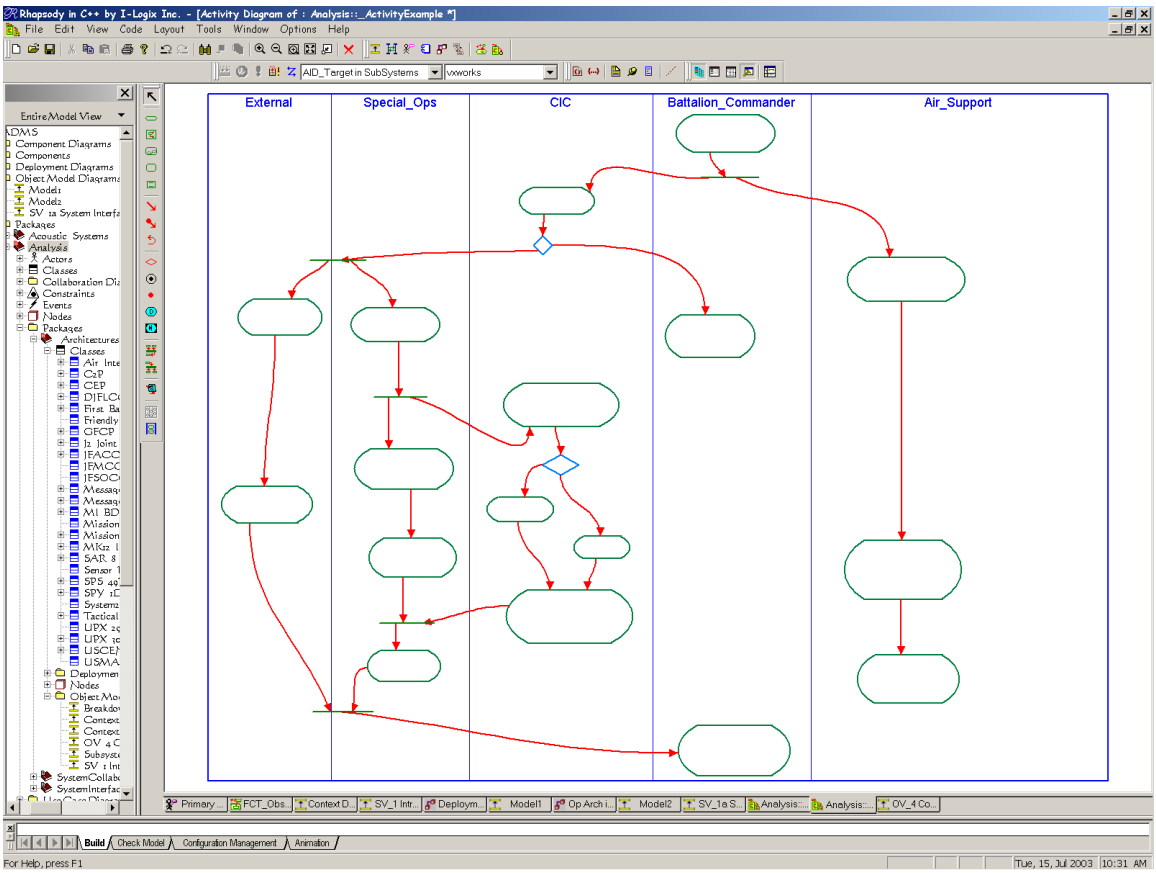

Figure 11: OV-5 Operational Activity Model with Swimlanes

The next figure shows a concrete example. Two entities, OTC CWC and FLTCINC are shown, executing a set of activities. The structure of their collaboration becomes clear in this diagram. Diagram connectors are used to beautify the diagram.

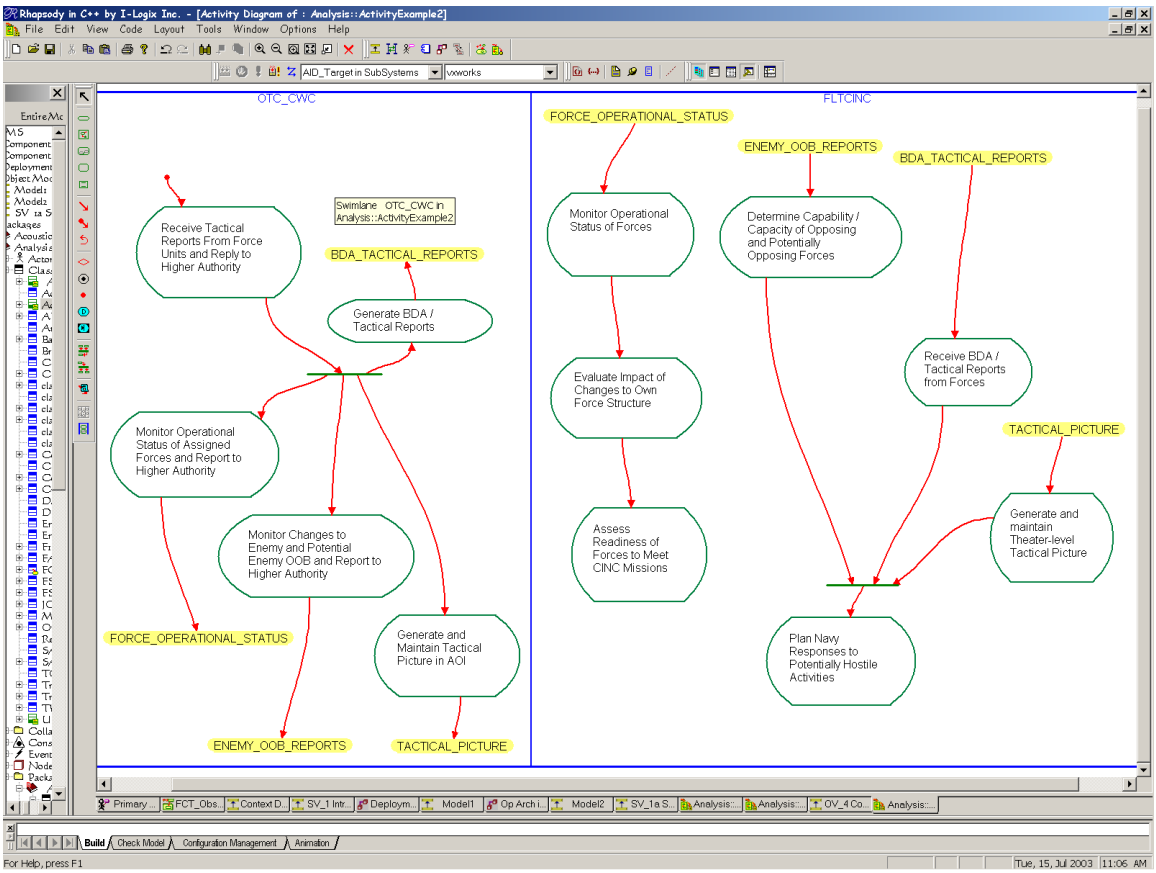

Figure 12: OV-5 Operational Activity Model with Two Agencies

OV-6a Operational Rules Model, SV-10a Systems Rules Model

The operational rules model shows the relationship among the activities. This can be done using any of the behavioral diagrams of the UML:

- An activity diagram is preferred when the activities flow from one to the next upon completion, but may have forks and joins (concurrently executing activities), branching (alternatives), or may be assigned to different system or operational elements (via swimlanes).
- A statechart is preferred when the states or conditions of existence persist until some event of interest occurs, such as an incoming asynchronous event (e.g. hostile missile launch), a synchronous service request (e.g. the requester waits until the service is completed before moving forward), or a timeout (e.g. “At 23:00, launch attack”). Behaviors are executed during the change of state or within a state when such an event is received.
- A sequence diagram is preferred when a particular operational scenario is to be shown – i.e. a collaborative behavior of system or operational elements, beginning with specific preconditions and with a specific event sequence, ignoring other possibilities.

In this case, we will show only a class diagram for the “logical data model” and describe some business rules. In the later OV-6 diagrams we’ll present a statechart for the OV-6b example and a sequence diagram for OV-6c example.

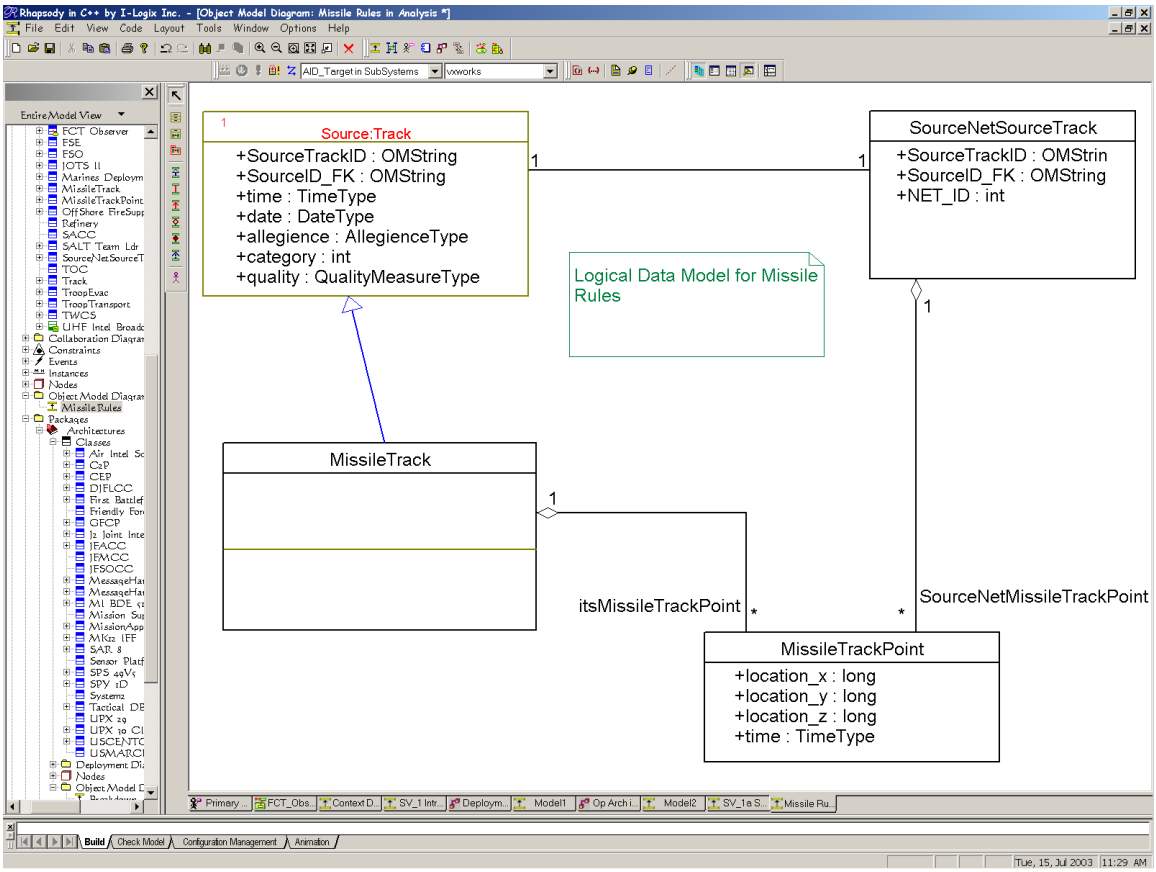

Figure 13: OV-6a Logical Data Model for Operational Rules

Business rules can be applied to the logical data model (taken from [1] *DoD Architectural Framework Version 1.0: Deskbook* DoD Architectural Framework Working Group :

For Each MISSILE TRACK entity instance
If MISSILE TRACK boost phase code > 0,
Then MISSILE TRACK acceleration rate *is non-null*
Else MISSILE TRACK drag effect rate *is non-null*
And
There Exists a MISSILE TRACK POINT entity instance *Such That*
MISSILE TRACK.SOURCE TRACK identifier =
MISSILE TRACK POINT.SOURCE TRACK identifier
And
MISSILE TRACK POINT.SOURCE identifier
End If
End For

OV-6b Operational State Transition Description, SV-10b Systems State Transition Description

The Operational State Transition is best described using a statechart. In Figure 14, the AIDSystem shows the states it goes through in acquiring and destroying a target.

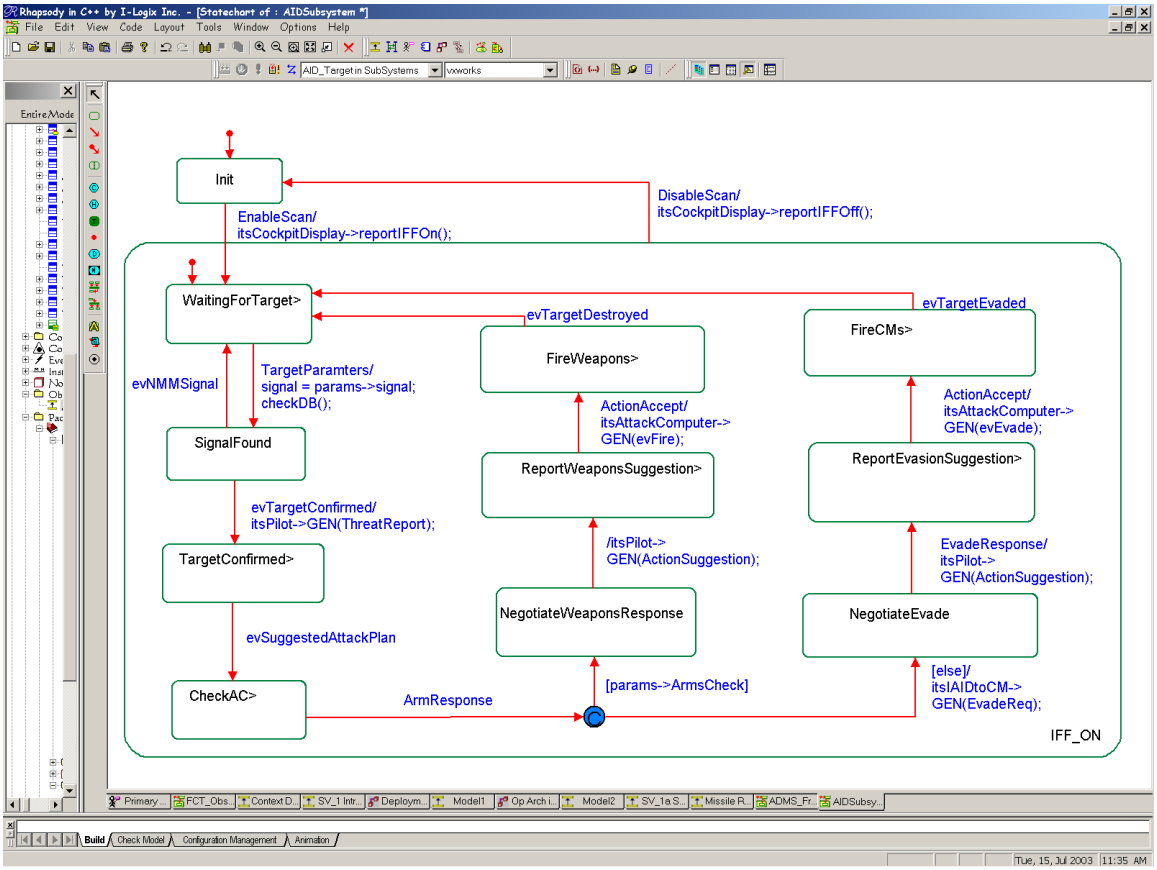

Figure 14: OV-6b Statechart for Operation State Transition Description

OV-6c Operational Event-Trace Description, SV-10c Systems Event Trace Description

OV-6c is the Event Trace Description, and is used to show specific flow or sequence of messages and events during a very specific scenario or example execution of the system. Branch points are typically not shown. Emphasis is on a specific execution given a specific set of preconditions and a specific sequencing of incoming events and messages. It is common to produce dozens of such operational scenarios, each on a separate diagram for the purpose of exploring variations of system behavior.

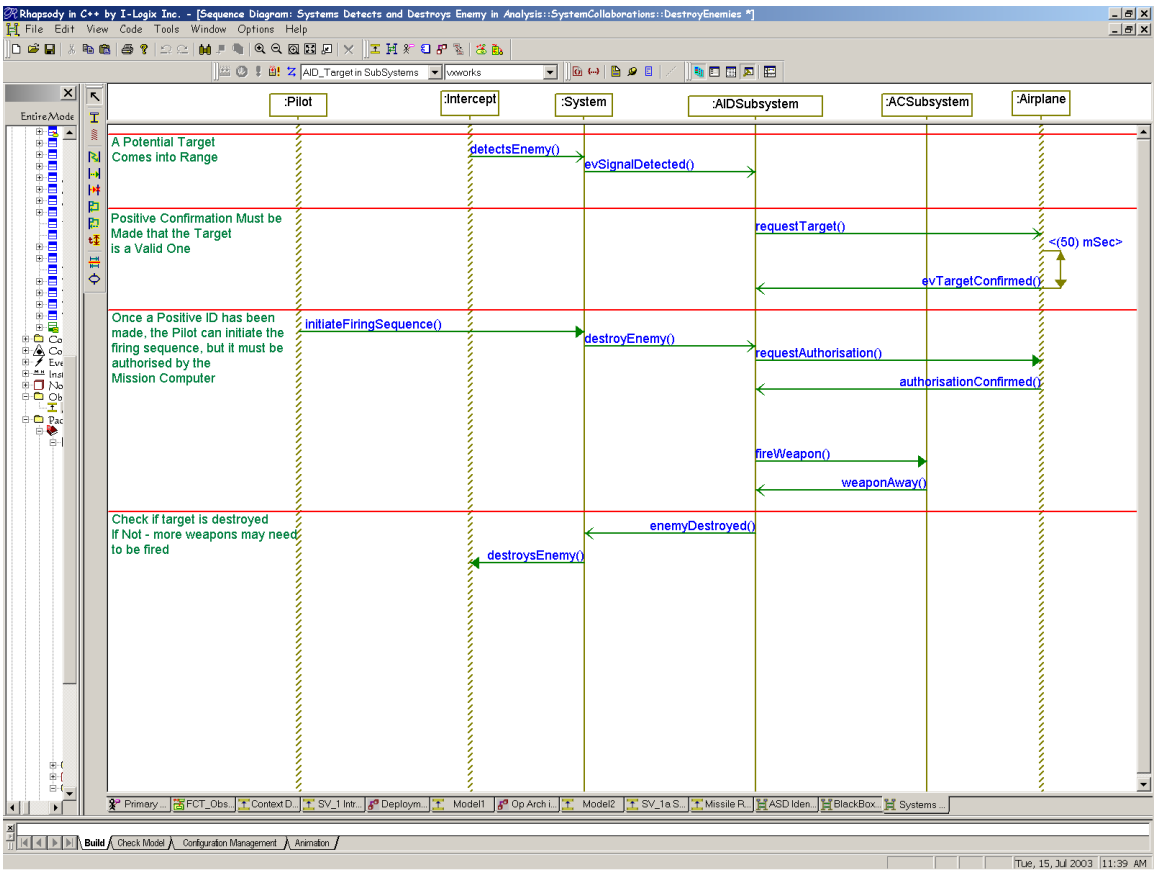

Figure 15: OV-5c Event-Trace Description with Sequence Diagram

OV-7 Logical Data Model

The logical data model focuses on the information known to, or processed by, the system during its execution. In the UML, this is modeled with class diagrams. Class diagrams can depict system structure as well as information content, so to achieve the OV-7 goals, the class diagrams used focus instead on the informational content of the system and the relationships among those data, and not on the processing of this information.

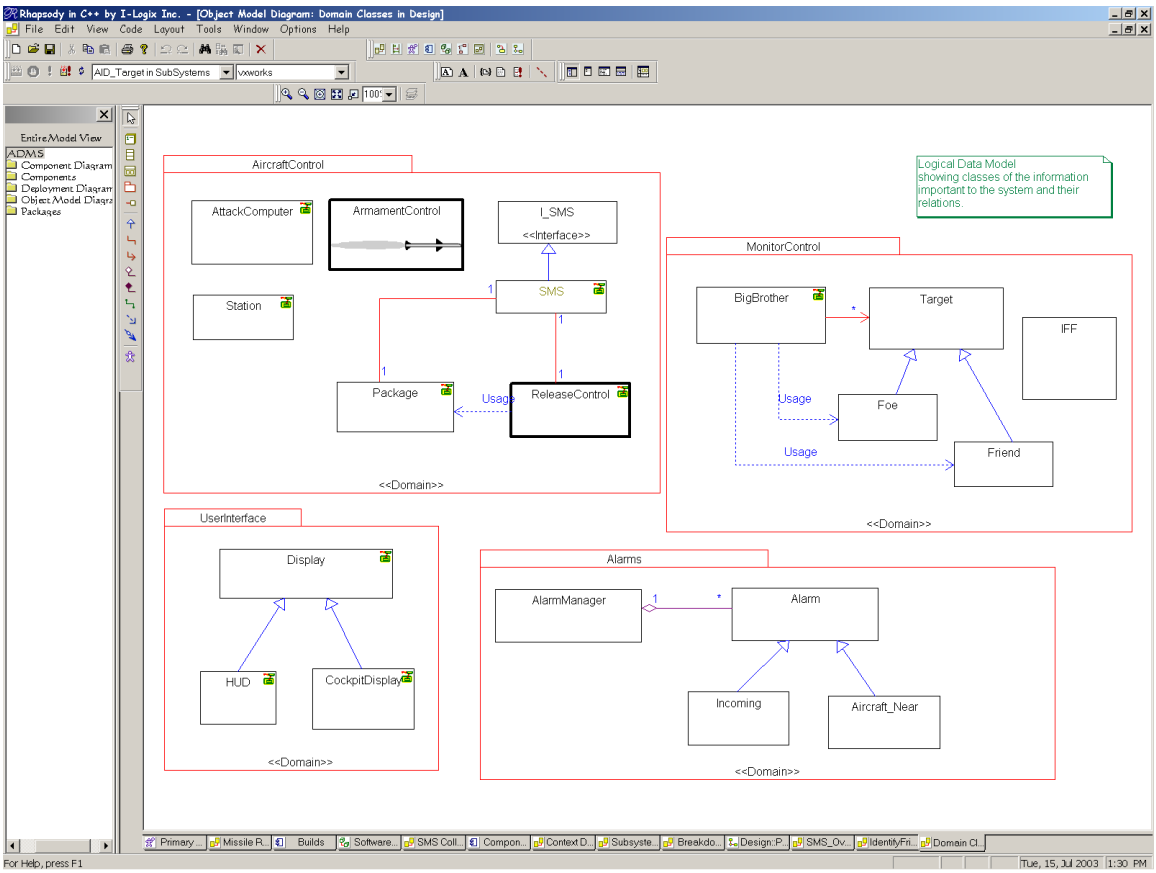

Figure 16: OV-7 Logical Data Model

SV-1 System Interface Description

The System Interface Description shows the structural elements of the systems architecture and the informational interfaces among them.

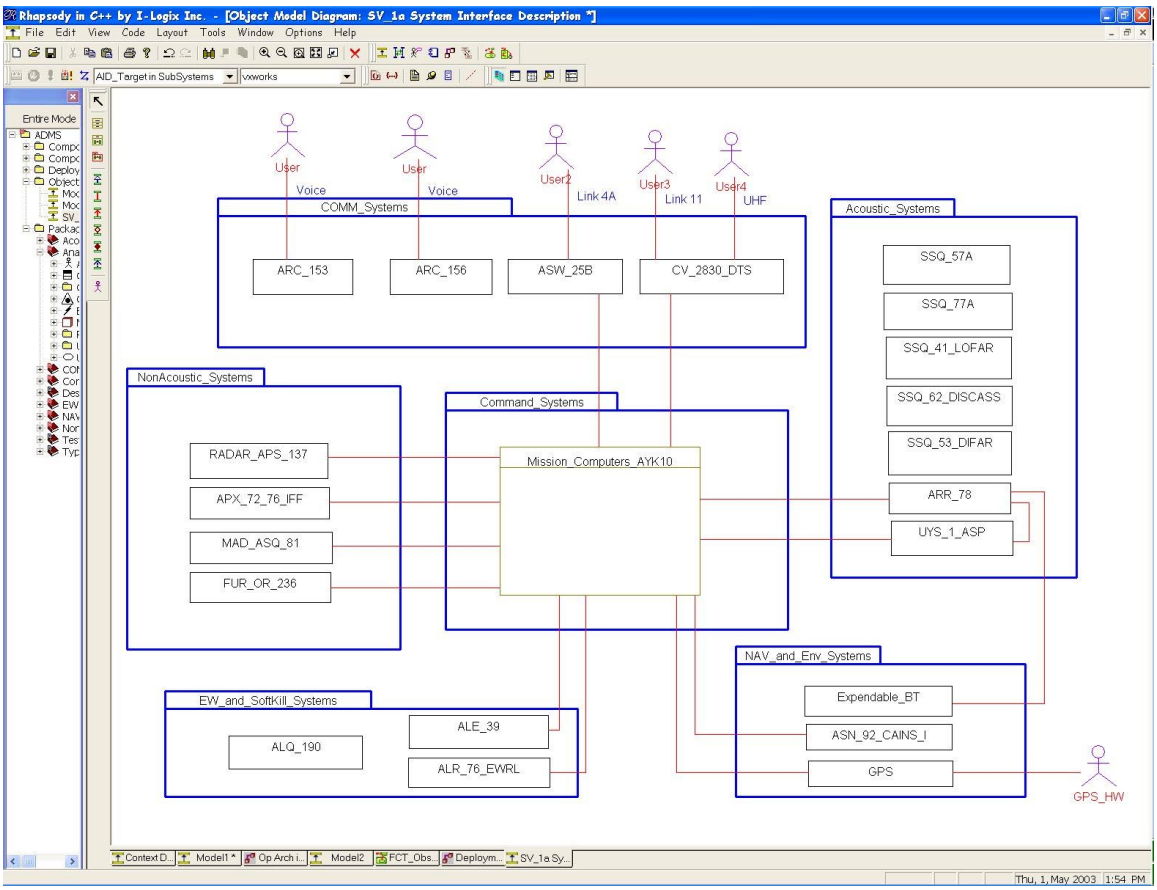

Figure 17: SV-1 System Interface Description

Sometimes it is important to show the large-scale deployment of the elements into the systems rather than their categories, as is shown in Figure 17. The next figure, Figure 18 shows the deployed systems as structured classes with the internal subsystems as classes. These elements may have stereotypes attached, if desired. This more detailed view shows the subsystems and their interconnections within the systems as well as between systems.

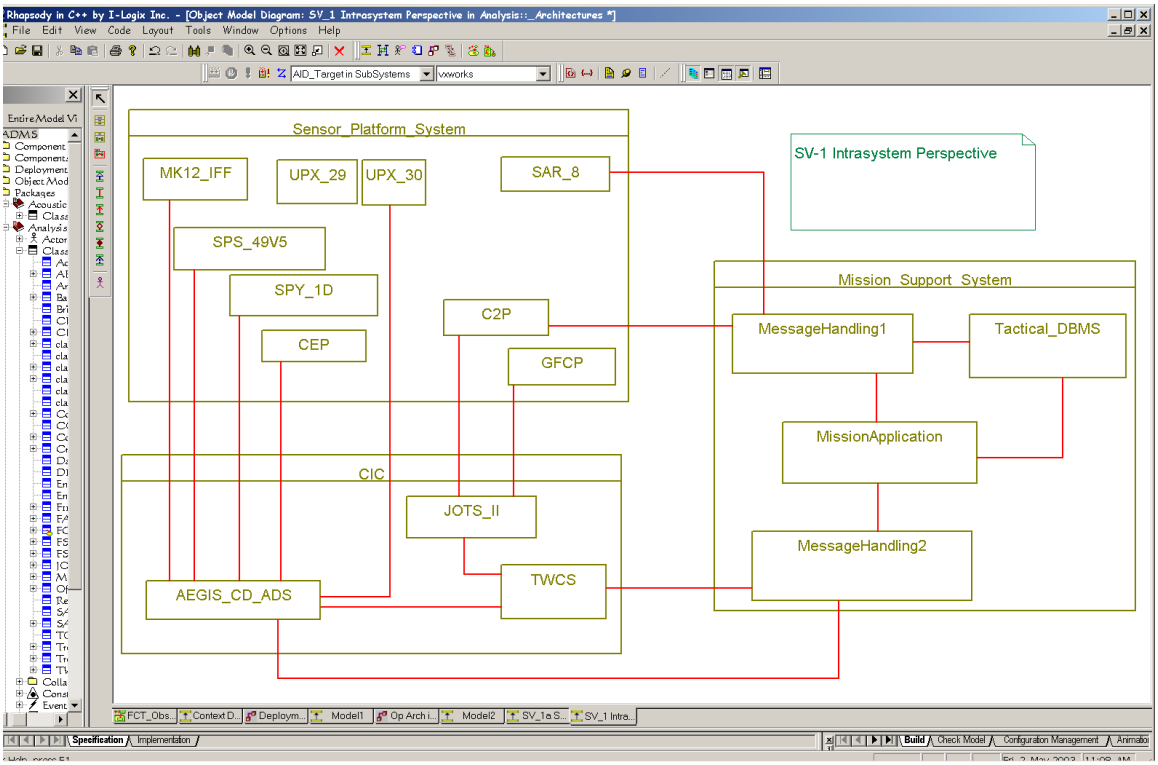

Figure 18: SV-1 Intrasystem perspective

In UML 2.0 and Rhapsody 5, it is possible to represent explicit connection points (ports) that are typed by a set of synchronous and asynchronous services (interfaces). This offers a very clear way to specify exactly how systems interconnect and collaborate. Figure 19 shows how this works for an aircraft guidance and control system.

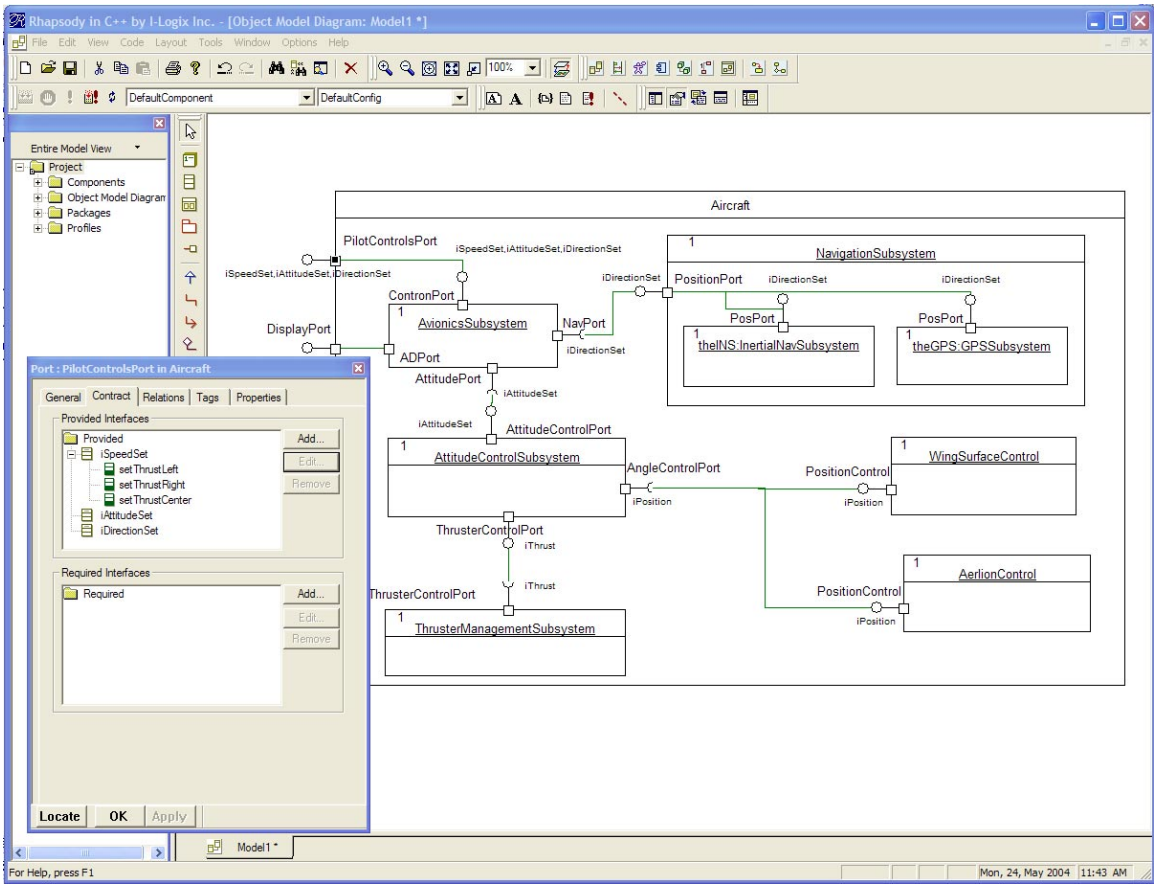

Figure 19: SV-1 With System Interfaces and Ports

SV-2 Systems Communications Description

The Systems Communications Description depicts the communications “lay downs” – the nodes, links and connections among the systems in the system architecture. The physical media that supports the interfaces described in SV-1 are emphasized in SV-2. The more common UML depiction is with a deployment diagram. Figure 20 shows such a deployment diagram.

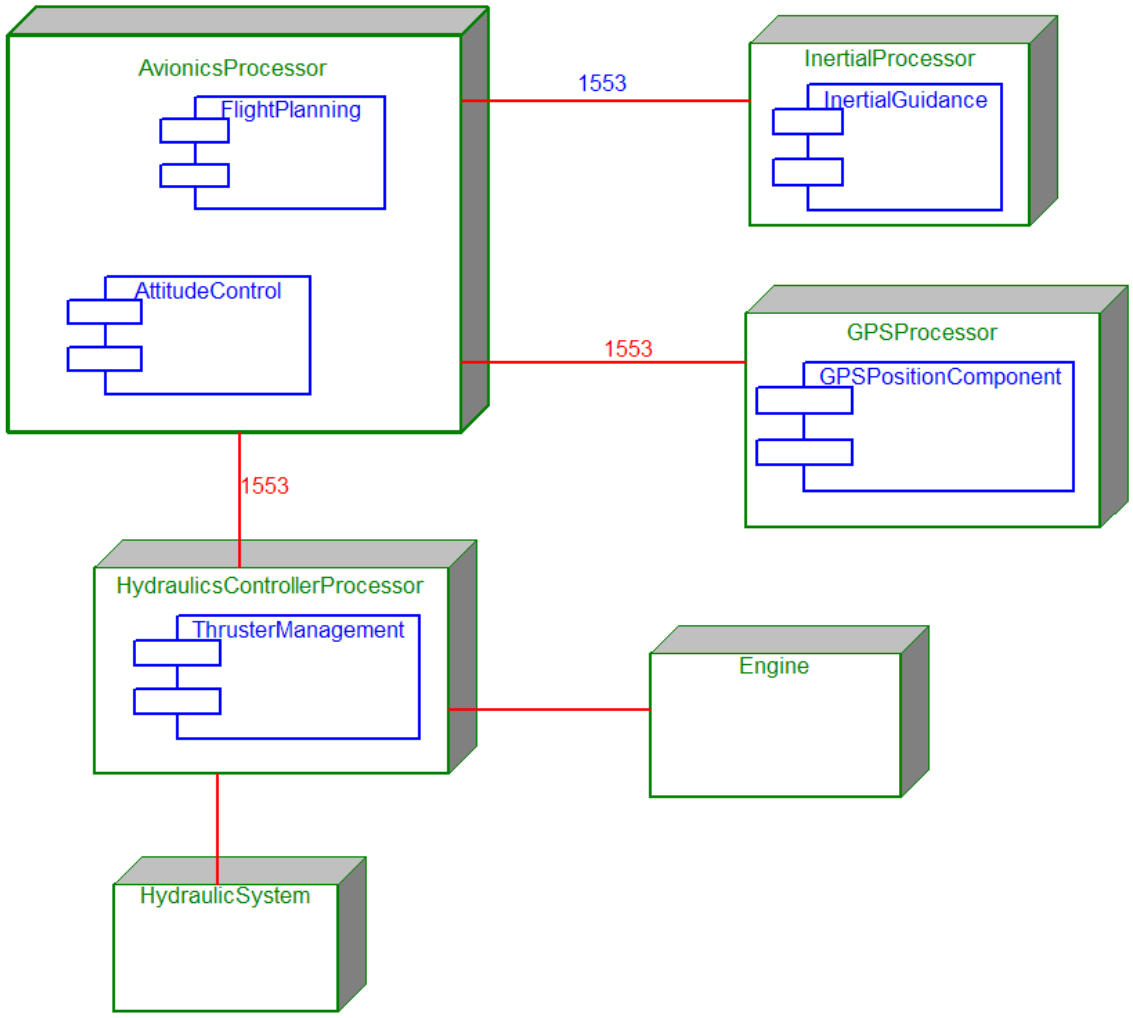

Figure 20: SV-2 Systems Communication Description

SV-3 Systems-Systems Matrix

The systems-systems view is to show the relationships among the set of large-scale entities (systems). As mentioned previously, the UML does not have a matrix view. However, the information can be shown easily on a class diagram (also a two-dimensional view), as shown in Figure 21. It is also possible to construct a matrix from the data held in the model repository.

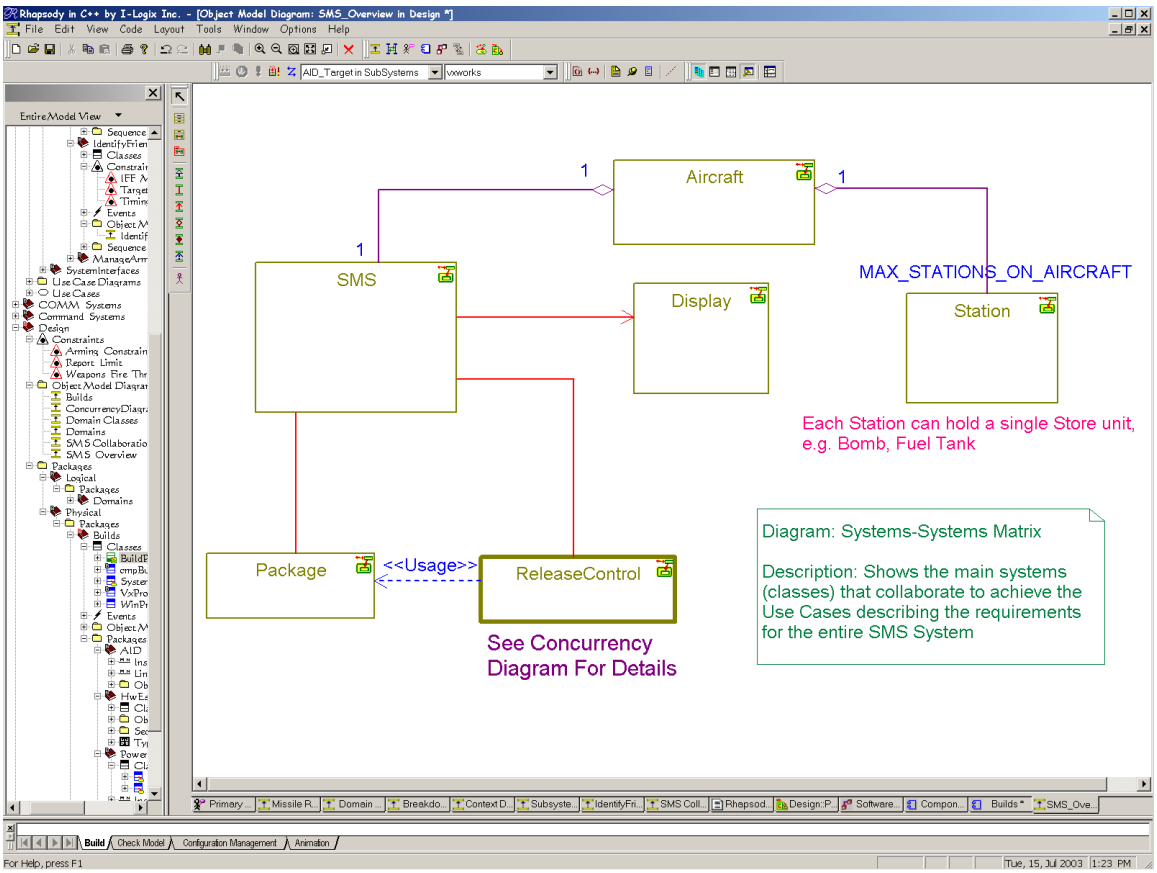

Figure 21: SV-3 Systems-Systems Matrix with Class Diagram

SV-4 Systems Functionality Description

Systems functionality (SV-4) is to show the functionality of a system or set of systems at a high-level view. The natural view in the UML is the Use Case diagram. Of course, just the names of the use cases are insufficient, so Rhapsody has description fields that can hold text (and/or hyperlinks to other documents in other tools such as Word or Framemaker) to more fully explain the functionality represented by the use case. It is common to "detail" the use case with a combination of text, statecharts, activity diagrams, and sequence diagrams. Figure 22 shows a use case diagram with the description field for one of the use cases.

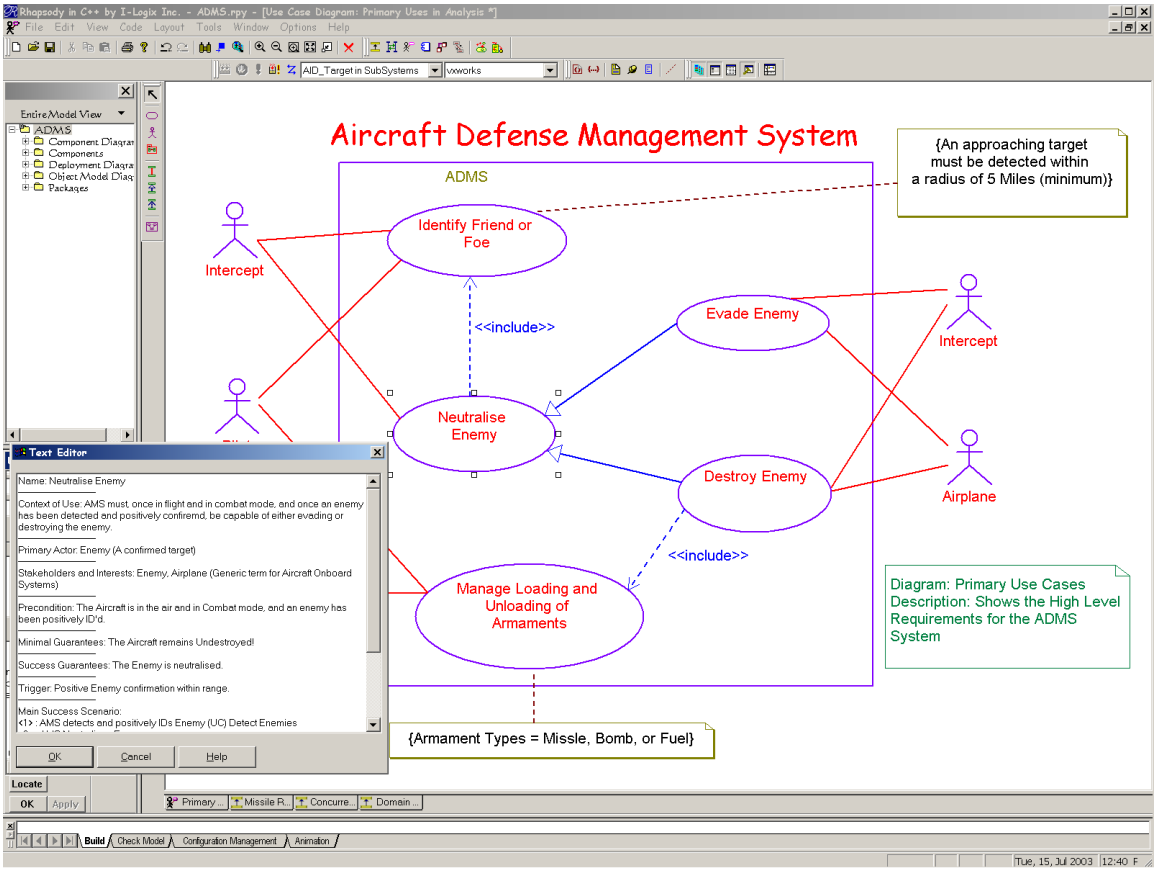

Figure 22: SV-4 Systems Functionality Description

SV-5 Operational Activity to Systems Function Traceability Matrix

The purpose of SV-5 is to allow mapping from operational activities defined in OV-5, OV-6a, PV-6b, and OV-6c into systems functions. This can be done in UML using dependencies if desired, but the most common way in the UML to achieve this goal is with the swim lanes in the activity diagrams. The swimlanes can represent system elements or functions while the activities in the diagram represent the operational activities. See Figure 11 and Figure 12 for examples.

Another common approach to traceability is the use of third party requirements traceability tools, such as DOORS, to define the mapping between the operational activities and the system use cases or system elements.

SV-6 Systems Data Exchange Matrix

SV-6 shows how information is exchanged among system elements. Although the name of the product includes the word “matrix”, it may be met by any visualization that shows the information flow among system elements. The most natural view in the UML is to show the system elements as classes connected with data flows, as defined in the UML

2.0 specification. Figure 23 shows elements of an aircraft weapons management system with information flows among the elements. An information flow is shown as a «flow» stereotype of dependency, with information items attached. These information items may be rich and complex, and so may themselves be classes with their structure depicted on the diagram(s).

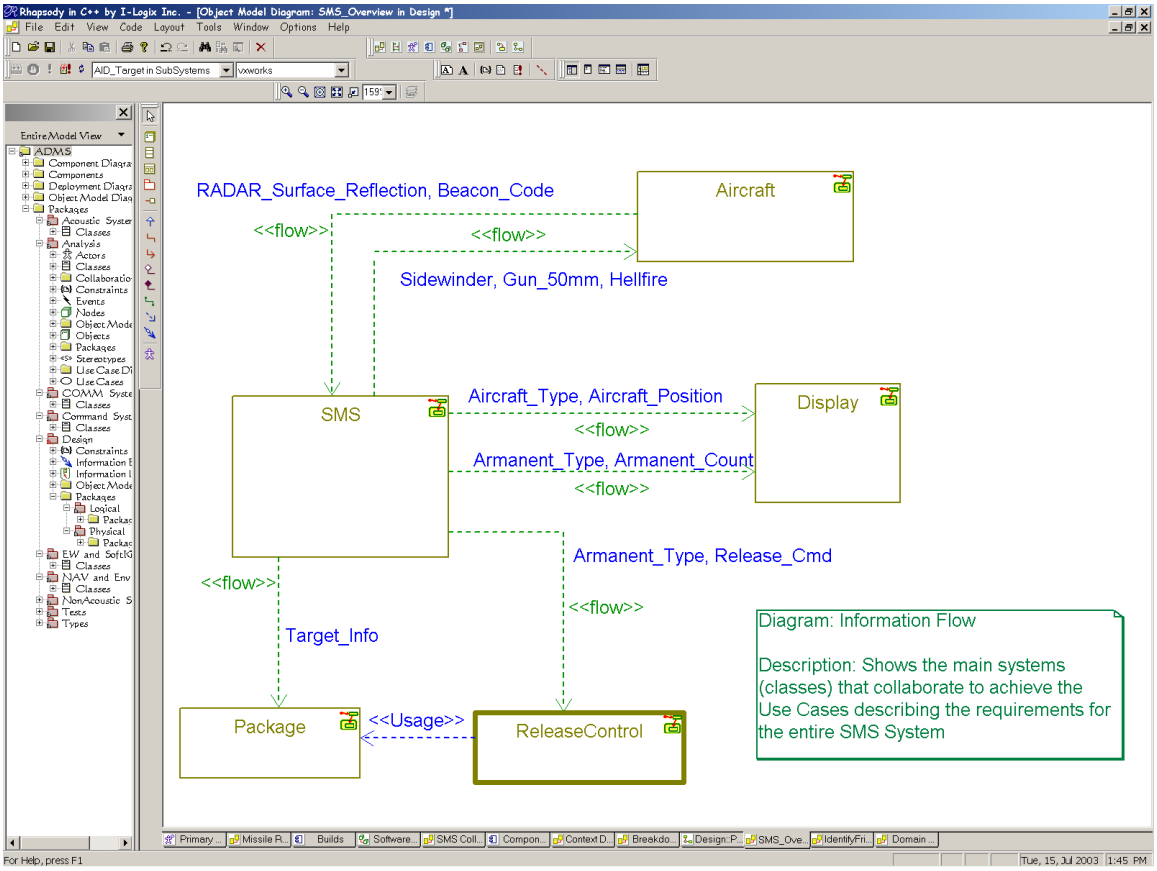

Figure 23: SV-6 Data Flow on Class Diagram

SV-7 Systems Performance Parameters Matrix

Performance is, of course, a crucial aspect of military and other real-time and embedded systems. In the UML, performance is a rather large subclass of the more general concept of Quality of Service (QoS). Performance-related QoS properties include worst-case execution time, throughput, average execution time, capacity, and so on. The OMG provides a standard set of performance-related property tags in [5] *UML Profile for Schedulability, Performance and Time* ptc2003-03-02, Object Management Group. In the UML, QoS properties may be attached to just about any element. However, whether or not these "standard" properties or custom properties are used, the usage model is the same. The system elements are stereotyped to have the appropriate properties and then these properties are assigned values in constraints. These constraints can be shown on the class diagrams with the elements they

constrain, such as in Figure 24 or they may be view in the model browser and in reports, as shown in Figure 25.

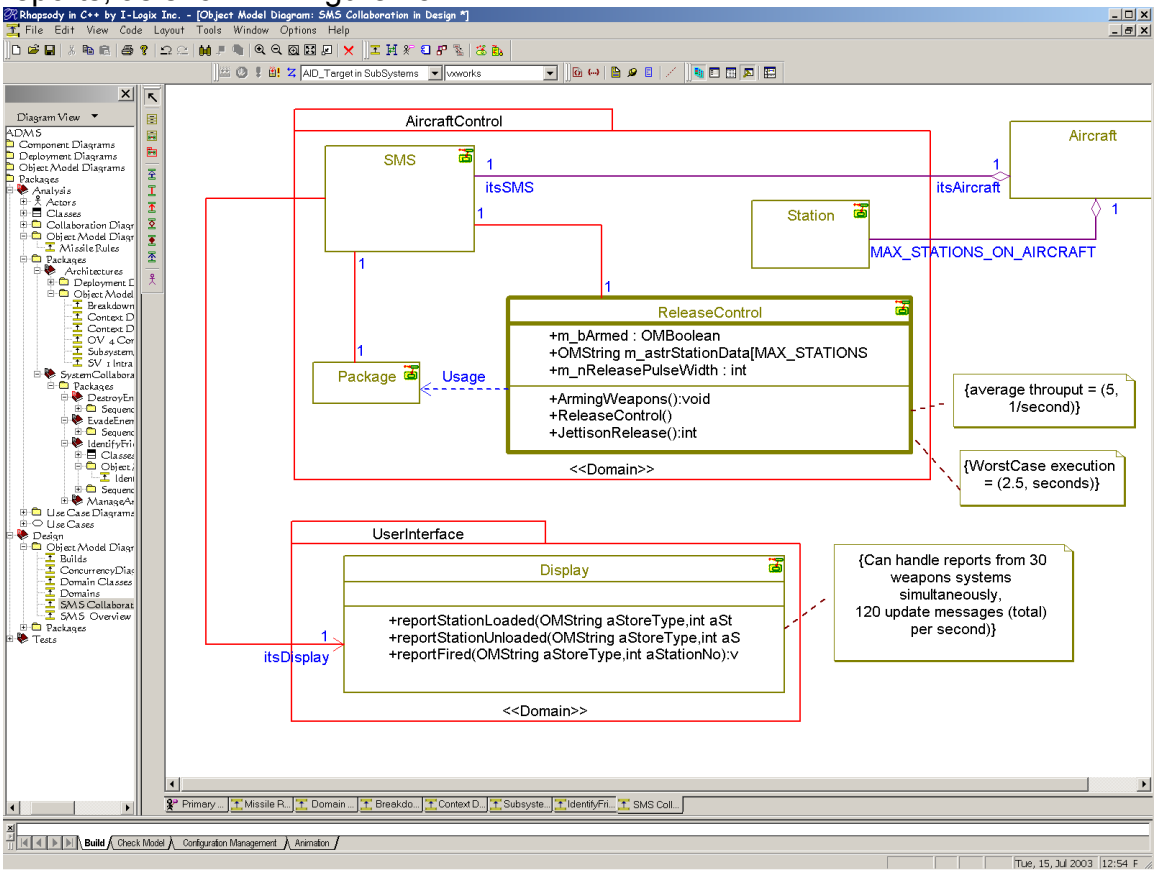

Figure 24: SV-7 Systems Performance on Class Diagram

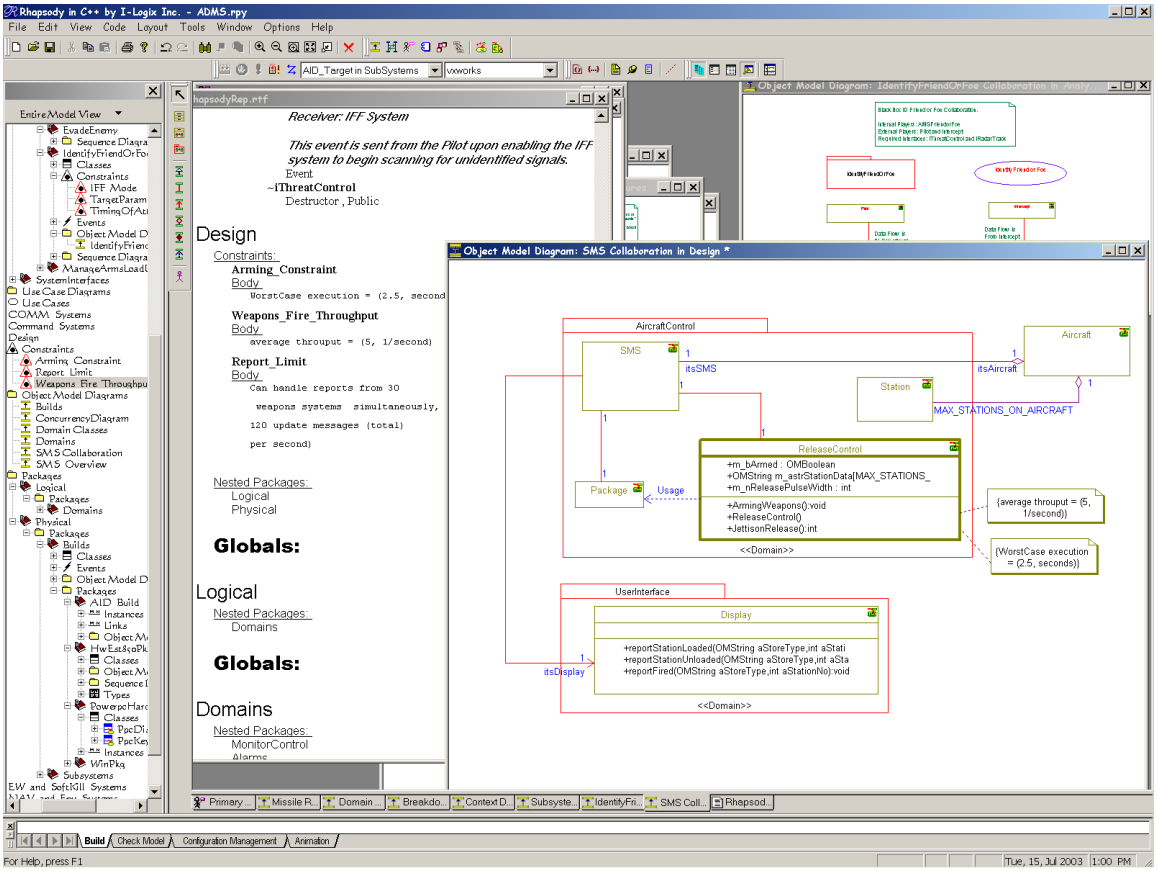

Figure 25: SV-7 Systems Performance in Reports and Browser

SV-8 Systems Evolution Description

SV-8, Systems Evolution Description, is a map of development activities leading to successive releases and versions of one or more systems. The UML activity diagram represents such processes clearly and distinctly. In Figure 26, the development activities are shown in the rounded rectangles while the artifacts are shown in the rectangles. Such evolution descriptions may be as complex as needed to describe the development plans for any system element or set of system elements.

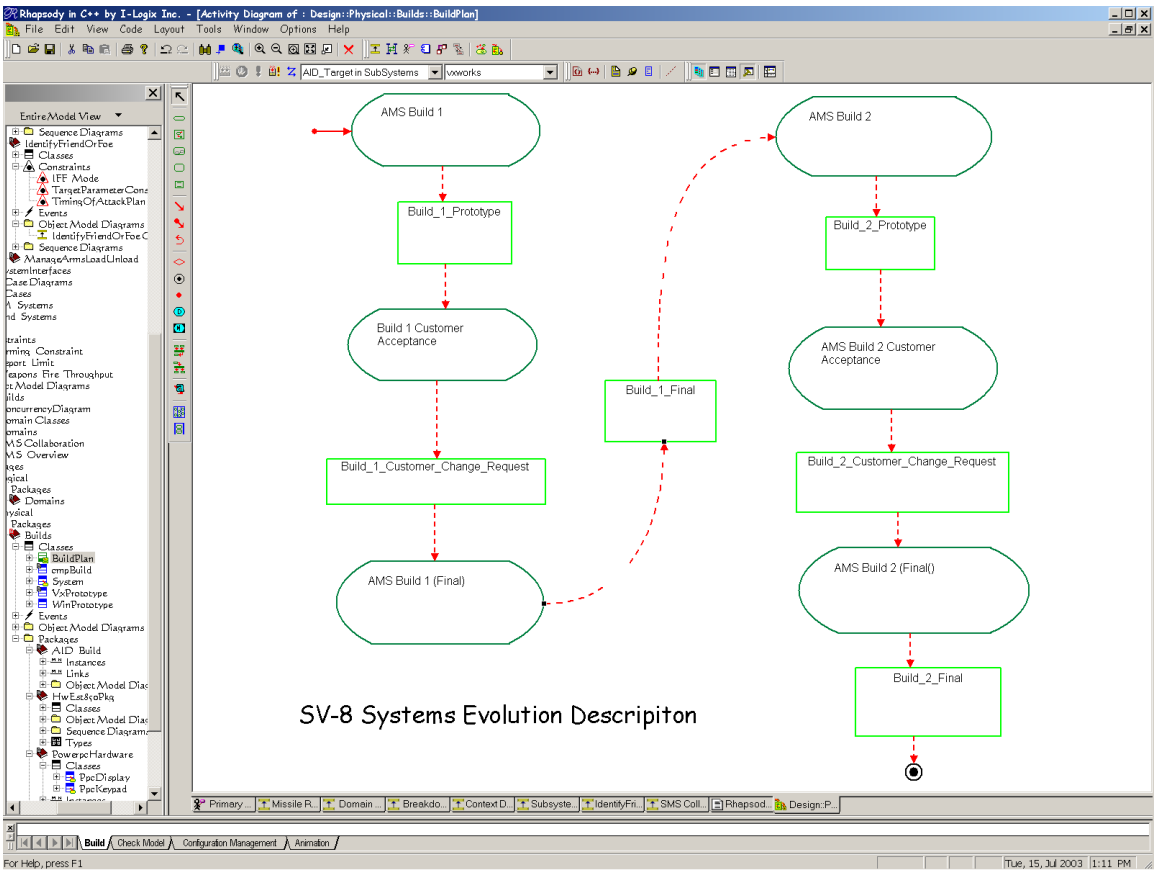

Figure 26: SV-8 Systems Evolution Description

SV-9 Systems Technology Forecast

As with TV-2, SV-9 is almost exclusively a textual document. There is no UML representation although text can be represented in the description fields of a tool such as Rhapsody. Additionally, external documents may be linked to with hyperlinks in Rhapsody so that selecting them executes the appropriate application to read, view or modify the document.

SV-11 Physical Schema

The Physical Schema shows how the Logical Data Model (OV-7) is to be physically realized – that is, how the information maps onto artifacts, devices and other elements in the systems architecture view. This is most commonly done with a deployment diagram where the nodes represent the hardware devices and the components represent the software or informational elements. It can also be done in a component diagram in which the logical elements (classes) are mapped into the components. A class diagram is yet another possibility, where some classes represent the physical items and others represent the logical elements.

Figure 27 shows the mapping of components onto the deployment nodes (hardware elements) using a standard UML deployment diagram.

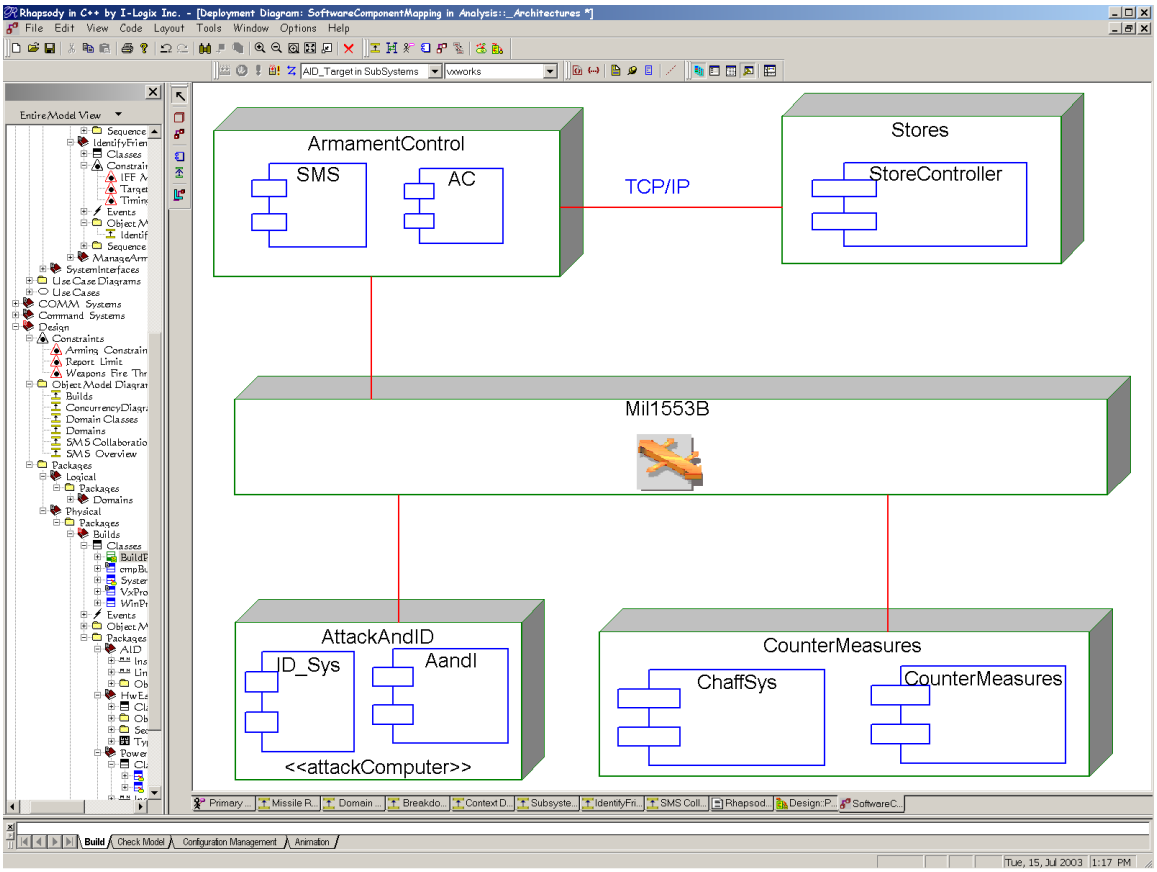

Figure 27: SV-11 Physical Schema with Deployment

Physical schema may also refer entirely to the organization of software elements into component artifacts, such as documents, link libraries, executables and so on. Figure 28 shows an example of this. The stereotypes «Executable», «Library», and «Hardware» show the kind of component being represented.

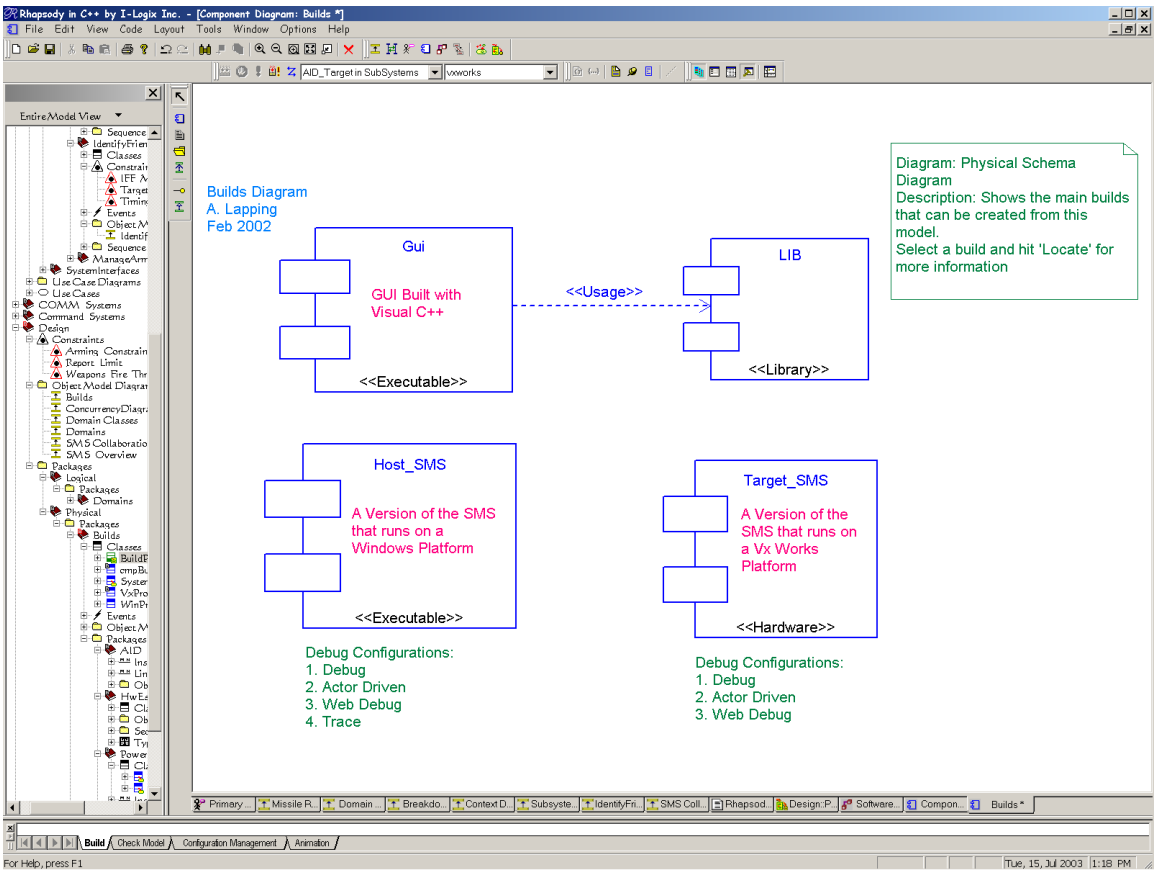

Figure 28: SV-11 Physical Schema with Components

TV-1 Technical Architecture Profile

The Technical Architecture Profile is a description of the technologies to be included in the system, normally as references to application standards documents, such as [1] *DoD Architectural Framework Version 1.0: Deskbook* DoD Architectural Framework Working Group or the POSIX standard, for example. These are most commonly specified in a textual document or they may be added as constraints to the model elements.

TV-2 Technical Standards Forecast

The Technical Standards Forecast identifies expected changes and evolutions of the standards and conventions identified in the TV-1 standard. This is also a textual specification and has no direct UML representation.

Summary

The DODAF standard defines a number of work products, some of which are identified as required for a system to be compliant, others are identified as supporting and are recommended but not required. The 1997 C4ISR standard came out at about the same time as the UML standard and therefore doesn't take advantage of the rich semantics and clear notation provided by the UML. Since then, the UML has been widely adopted in the specification of software and systems within the DoD and other environments. Additionally, the new DODAF standard provides recommended representations of the DODAF products using UML notation and semantics. This paper discusses each product required by the DODAF and shows how standard UML views and semantics can be used to represent them. Clearly, the unifying nature of the UML meets the needs of the DODAF very well, now and into the future.

Acknowledgements

The aircraft weapon systems diagrams were taken or adapted from the ADMS model by Andy Lapping, of I-Logix. Some of the other examples were adapted from [1] *DoD Architectural Framework Version 1.0: Deskbook* DoD Architectural Framework Working Group .

References

[1] *DoD Architectural Framework Version 1.0: Deskbook* DoD Architectural Framework Working Group

[2] *IEEE Standard Glossary of Software Engineering Terminolog*y, 1990, IEEE STD 610.12-1990, Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

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

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

[5] *UML Profile for Schedulability, Performance and Time* ptc2003-03-02, Object Management Group

[6] *Real-Time UML: Advances in the UML for Real-Time Systems* Douglass, Bruce Powel; Addison-Wesley, 2004

About the Author

Bruce Powel Douglass has over 20 years experience designing safety-critical real-time applications in a variety of hard real-time environments. He has designed and taught

courses in object-orientation, real-time, and safety-critical systems development. He is an advisory board member for the Embedded Systems Conference, UML World Conference, and Software Development magazine. He is a co-chair for the Real-Time Analysis and Design Working Group in the OMG standards organization. He is the Chief Evangelist at I-Logix, a leading real-time object-oriented and structured systems design automation tool vendor. He can be reached at bpd@ilogix.com.