Rational. Rhapsody

DoDAF Tutorial

DoDAF Tutorial for Rational Rhapsody

Before using the information in this manual, be sure to read the “Notices” section of the Help or the PDF available from **Help > List of Books**.

This edition applies to IBM® Rational® Rhapsody® 7.4 and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

DoDAF Tutorial for Rational Rhapsody

Welcome to the *DoDAF Tutorial for IBM Rational Rhapsody.* IBM® Rational® Rhapsody® is the Model-Driven Development environment of choice for systems engineers and software developers of either embedded or real-time systems.

While no prior knowledge of Rational Rhapsody is required to complete this tutorial, a basic understanding of DoDAF (Department of Defense Architectural Framework) allows better understanding of the architecture model.

DoDAF Tutorial Overview

This tutorial provides hands-on step-by-step instructions on how to create a DoDAF-compliant architecture model of the operational view for a simplified land and sea-based attack capability.

Before you do this tutorial you might find it helpful to review the *Getting Started Guide* for the Rational Rhapsody product. It provides a functional overview for the Rational Rhapsody product for system designers, system engineers, and software developers with more functions (meaning how to do something), explanations, and details than this tutorial provides.

In addition, references are made to other Rational Rhapsody documentation where appropriate in this tutorial. Note also that the *IBM Rational Rhapsody User Guide* has a Glossary section that you might find useful.

This tutorial is provided as part of the Rational Rhapsody DoDAF Add-on. This tutorial is intended to get you familiar with the Rational Rhapsody product and DoDAF. You should consider this tutorial as part of the Rational Rhapsody learning process, in addition, for example, to the *Rhapsody Essential Tool Training* class and the Rational Rhapsody eLearning courses, both of which are available at an additional cost.

Required Software

The Rational Rhapsody DoDAF Add-on can be used on Windows XP SP1 or newer. This tutorial requires the following application software to be properly installed and licensed on your system.

Required Application	Version
Rational Rhapsody	7.3 or later
Rational Rhapsody DoDAF Add-on	7.3 or later
Adobe Acrobat Reader	6.0 or later
Microsoft Word	2003 or later
Microsoft Excel	2003 or later
Microsoft Paint (or another bitmap image viewer)	N/A
Microsoft .NET Framework	1.1.4322 or later

Note

This tutorial does not include the Microsoft Excel, Microsoft Word, or Microsoft Paint software (these can be purchased from Microsoft Corporation). This tutorial does not include Adobe Acrobat Reader, which can be downloaded for free from the Adobe Web site.

About the Rational Rhapsody Product

The Rational Rhapsody product is a visual design tool for developing object-oriented embedded software, and performing structural and systems modeling. It enables you to perform these tasks:

- **Analyze**, during which you can define, analyze, and validate the system requirements.
- **Design**, during which you can specify and design the architecture.
- **Implement**, during which you can automatically generate code, then build and run it within the Rational Rhapsody product.
- **Model Execution**, during which you can animate the model on the local host or a remote target to perform design-level debugging within animated views.

UML Diagrams

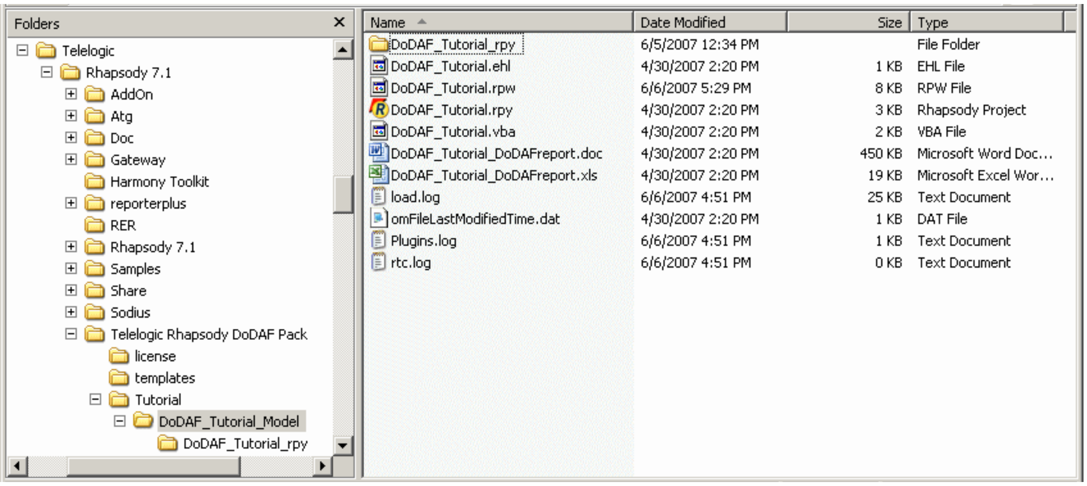
The following are the UML diagrams in Rational Rhapsody:

- **Use Case Diagrams** show the main functions of the system (use cases) and the entities (actors) outside the system.
- **Structure Diagrams** show the system structure and identify the organizational pieces of the system.
- **Object Model Diagrams** show the structure of the system in terms of classes, objects, and the relationships between these structural elements.
- **Sequence Diagrams** show sequences of steps and messages passed between structural elements when executing a particular instance of a use case.
- **Activity Diagrams** specify a flow for classifiers (classes, actors, use cases), objects, and operations.
- **Statecharts** show the behavior of a particular classifier (class, actor, use case) or object over its entire life cycle.
- **Collaboration Diagrams** provide the same information as sequence diagrams, emphasizing structure rather than time.
- **Component Diagrams** describe the organization of the software units and the dependencies among units.
- **Deployment Diagrams** show the nodes in the final system architecture and the connections between them.

In addition, **Flow Charts** are available in the Rational Rhapsody product. Flow charts are not in UML. They are a subset of activity diagrams with parts (of the functionality for activity diagrams) excluded. Flow charts have specifically event-driven behavior. You can use a flow chart to describe a function or class operation and for code generation.

About a Rational Rhapsody Project

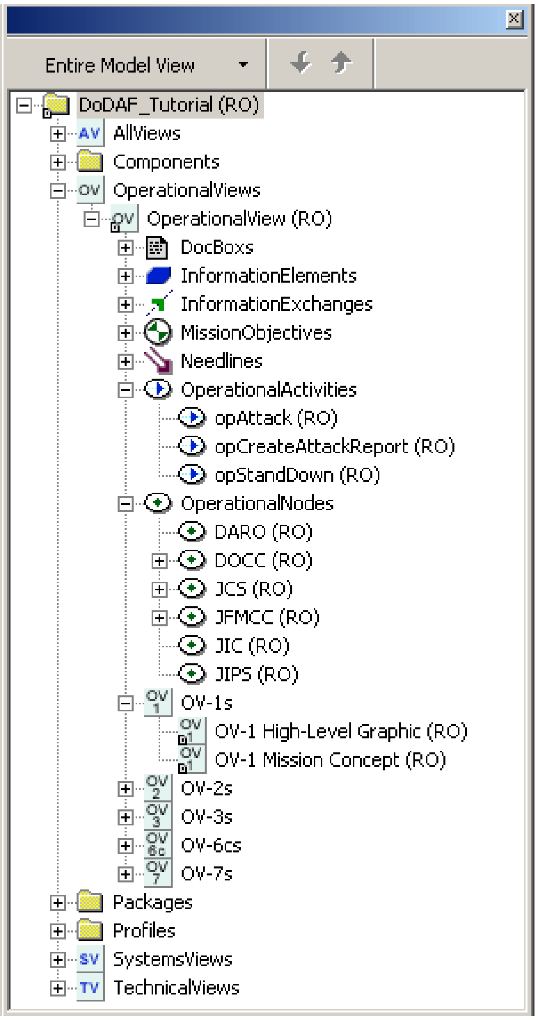
A Rational Rhapsody project includes the UML diagrams, packages, and code generation configurations that define the model and the code generated from it. When you create a new project, Rational Rhapsody creates a project folder that contains the *project files* in the specified location. The name you choose for your new project is used to name project files and folders, as shown in the following figure:



Name	Date Modified	Size	Type
DoDAF_Tutorial_rpy	6/5/2007 12:34 PM		File Folder
DoDAF_Tutorial.ehl	4/30/2007 2:20 PM	1 KB	EHL File
DoDAF_Tutorial.rpw	6/6/2007 5:29 PM	8 KB	RPW File
DoDAF_Tutorial.rpy	4/30/2007 2:20 PM	3 KB	Rhapsody Project
DoDAF_Tutorial.vba	4/30/2007 2:20 PM	2 KB	VBA File
DoDAF_Tutorial_DoDAFreport.doc	4/30/2007 2:20 PM	450 KB	Microsoft Word Doc...
DoDAF_Tutorial_DoDAFreport.xls	4/30/2007 2:20 PM	19 KB	Microsoft Excel Wor...
load.log	6/6/2007 4:51 PM	25 KB	Text Document
omFileLastModifiedTime.dat	4/30/2007 2:20 PM	1 KB	DAT File
Plugins.log	6/6/2007 4:51 PM	1 KB	Text Document
rtc.log	6/6/2007 4:51 PM	0 KB	Text Document

For more information about the folders and files that are part of a Rational Rhapsody model, see **About Project Files and Folders**.

In addition, the name appears at the top level of the project hierarchy on the *Rational Rhapsody browser.* The following figure shows the Rational Rhapsody browser for the DoDAF_Tutorial model.



- DoDAF_Tutorial (RO)
  - AllViews
  - Components
  - OperationalViews
    - OperationalView (RO)
      - DocBoxs
      - InformationElements
      - InformationExchanges
      - MissionObjectives
      - Needlines
      - OperationalActivities
        - opAttack (RO)
        - opCreateAttackReport (RO)
        - opStandDown (RO)
      - OperationalNodes
        - DARO (RO)
        - DOCC (RO)
        - JCS (RO)
        - JFMCC (RO)
        - JIC (RO)
        - JIPS (RO)
      - OV-1s
        - OV-1 High-Level Graphic (RO)
        - OV-1 Mission Concept (RO)
      - OV-2s
      - OV-3s
      - OV-6cs
      - OV-7s
  - Packages
  - Profiles
  - SystemsViews
  - TechnicalViews

About Project Files and Folders

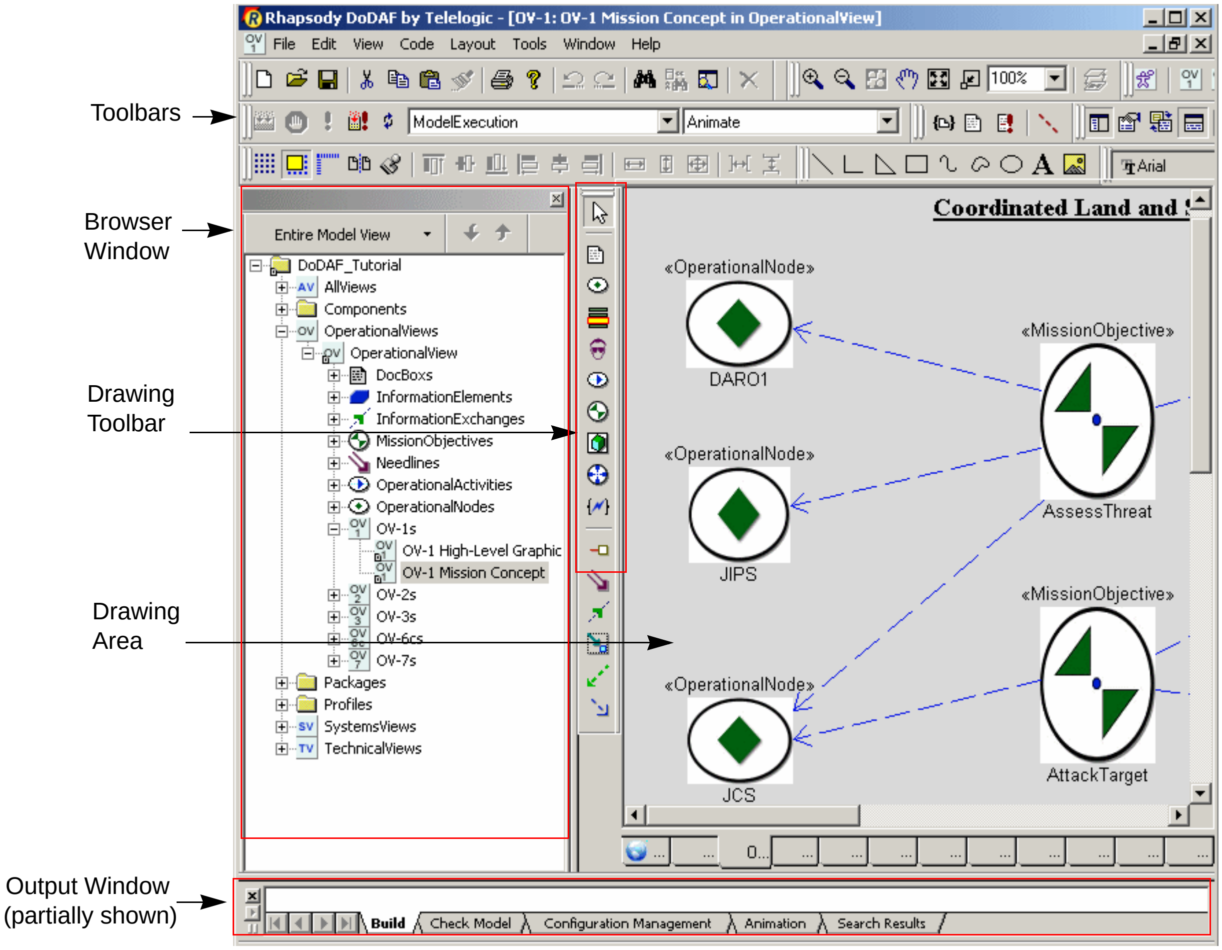
The Rational Rhapsody product creates the following files and subfolders in the project folder. Some folders and files are created when you initially create a project, others only when applicable.

- A project folder, called **<project_name>_rpy**, which contains the unit files for the project, including UML diagrams, packages, and code generation configurations.
- A project file, called **<project_name>.rpy**.
- A subfolder, called **<project_name>_auto_rpy**, which appears only when necessary (after ten minutes if a save has not been made) and disappears after you save.
- An event history file, called **<project_name>.ehl**, which contains a record of events injected during animation, and active and nonactive breakpoints. This file appears after your first save of a project.
- Log files, which record when projects were loaded and saved in the product; for example, **load.log** and **store.log**.
- A .vba file, called **<project_name_>.vba**, which contains macros or wizards.
- Backup project files and folders (**<project_name>_bak1_rpy**, **<project_name>_bak2_rpy**), which are optional, depending on project settings.
- An **_RTC** subfolder, when applicable, which holds any tests created using the Rational Rhapsody TestConductor™ add-on.

The **<project_name>.rpy** file and the **<project_name>_rpy** folder are necessary for the generation of source code.

Rational Rhapsody User Interface

Before proceeding with this tutorial, you should become familiar with the main features of the Rational Rhapsody graphical user interface (GUI). The Rational Rhapsody GUI is made up of three key windows and different toolbars for each of the UML diagram types. The following figure shows the Rational Rhapsody GUI.



Toolbars

The Rational Rhapsody toolbars provide quick access to the commonly used commands. These commands are also available from the menus. The Rational Rhapsody product has the following toolbars:

- **Standard** has icons for the frequently used options on the File, Edit, and Help menus. Examples: **New**, **Open**, **Save**; **Copy**, **Paste**, **Locate in Browser**; **About**.
- **Code** has icons for the frequently used options on the Code menu, such as **Make**, **Run executable** and **G/M/R** (for **Generate/Make/Run**).
- **Windows** has icons for the frequently used options on the View menu, such as **Show/Hide Browser** and **Show/Hide output window**.
- **Diagrams** has icons for the part of the Tools menu that give you quick access to the diagrams in the project, such as **Sequence Diagrams** and **Open Statechart**.
- **VBA** provides access to the VBA options, such as **VBA Editor** and **Show Macros Dialog**. Note that VBA is for Windows only.
- **Animation** has icons for the animation options during an animation session, such as **Go**, **Animation Break**, and **Quit Animation**.
- **Layout** has icons that help you with the layout of elements in your diagram, such as **Snap to Grid**, **Align Top**, and **Align Left**.
- **Drawing** has icons for the graphics editor used to create and edit diagrams. Each **Drawing** toolbar is unique to its particular diagram type. For example, the **Drawing** toolbar for a sequence diagram is different from that for a statechart.
- **Common Drawing** has icons to add requirements, comments, and other annotations to any diagram, such as **Note** and **Requirement**.
- **Free Shapes** has icons for basic drawing shapes, such as **Polyline** and **Polycurve**.
- **Zoom** has icons to zoom options, such as **Zoom In**, **Zoom Out**, and **Pan**.
- **Format** has icons for various text formatting options and line/fill options, such as **Italic** and **Font Color**.

Refer to the *IBM Rational Rhapsody User Guide* for detailed information about the toolbars.

Browser

The Rational Rhapsody browser shows the contents of the project in an expandable tree structure. By default, it is the upper, left-hand part of the Rational Rhapsody interface. The top-level folder, which contains the name of the project, is the *project folder* or *project node*. Although this folder contains no elements, the folders that reside under it contain elements that have similar characteristics. These folders are referred to as *categories*.

A project consists of at least one package in the **Packages** category. A package contains UML elements, such as classes, files, and diagrams. Rational Rhapsody automatically creates a default package called **Default**, which it uses to save model parts unless you specify a different package.

Opening the Rational Rhapsody Browser

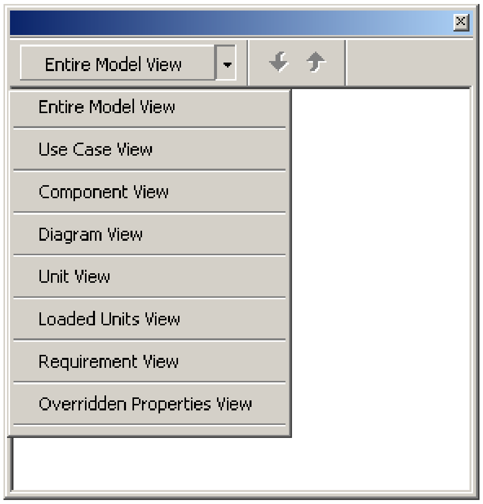
By default, the Rational Rhapsody browser is displayed the first time you open a project. In subsequent work sessions, Rational Rhapsody consults your workspace file (`<project name>.rpw`) to determine whether to open the browser when it opens a project. To open the Rational Rhapsody browser manually, follow these steps:

- Click the Show/Hide Browser icon on the Rational Rhapsody **Windows** toolbar.
- Select **View > Browser**.
- Press **Alt+0** (zero).

Filtering the Browser

The large size and nested hierarchy of a Rational Rhapsody project might complicate the process of locating and working with model elements. To help you navigate the Rational Rhapsody browser more easily, the browser has a filtering mechanism that you can use to display only the elements relevant to your current task.

To display the filter menu, as shown in the following figure, click the down arrow at the top of the browser. Whatever view you have selected is reflected in the label to the left of the arrow.



Refer to the *IBM Rational Rhapsody User Guide* for information on the view options.

Repositioning the Browser

To make more room for you to work on diagrams, you can move the browser outside of the Rational Rhapsody GUI and reposition it as a separate window on the desktop. To reposition the Rational Rhapsody browser, click the bar at the top of the browser and drag it to another desktop location.

Drawing Area

The drawing area displays the graphic editors and code editors, and it is the region for drawing diagrams. By default, it is the upper, right-hand section of the Rational Rhapsody interface. Rational Rhapsody displays each diagram with a tab that includes the name of the diagram and an icon that denotes the diagram type. When you make changes to a diagram, Rational Rhapsody displays an asterisk after the name of the diagram in the title bar to indicate that you must save your changes.

Output Window

The Output window is where Rational Rhapsody displays various output messages. Tabs on the Output window enable you to navigate easily among the different types of output messages:

- The **Log** tab shows all the messages from all the other tabs of the Output window (except for **Search Results**) in text—meaning non-tabular—format.
- The **Build** tab shows the messages related to building an application in tabular format.
- The **Check Model** tab shows the messages related to checking the code for a model in tabular format.
- The **Configuration Management** tab shows the messages related to configuration management actions for a model in text format.
- The **Animation** tab shows the message related to animating a model in text format.
- The **Search Results** tab shows the results from searches of your model in tabular format. Note that this tab might not appear until you perform a search.

By default, the Output window is located at the bottom portion of the main Rational Rhapsody window. Also by default, when you generate, build, or run an application; do a search, a configuration management action, or a check model, Rational Rhapsody opens the Output window. To open the Output window manually, select **View** > **Output Window**.

Drawing Toolbars

The **Drawing** toolbar provides access to tools (shown as icons) used in creating and editing diagrams in the graphic editors. Each graphic editor has a unique drawing toolbar. To display or hide the **Drawing** toolbar for the current diagram, select **View** > **Toolbars** > **Drawing**.

To view the complete set of drawing tools available for all the different diagrams, select **View** > **Toolbars** > **All Drawing**.

Features Dialog Box

The Features dialog box lets you view and edit the features of an element in the Rational Rhapsody product.

To open the Features dialog box, do one of the following:

- Double-click an element (for example, **attackStatusReport** [an information element])
- Right-click an element (for example, **OV-1 Mission Concept**), then select **Features**
- Select an element and press **Alt + Enter**
- Select an element and select **View** > **Features**

You can resize the Features dialog box and hide the tabs on it if you want. For more information about the Features dialog box, refer to the section on it in the *IBM Rational Rhapsody User Guide*.

Keeping Open the Features Dialog Box

Once you open the Features dialog box, you can leave it open and select other elements to view their features. This means that after you make changes to the Features dialog box for an element in your drawing or on the Rational Rhapsody browser, you can click **Apply**. Then, without closing the dialog box, you can select another element to view its features. Once you are done with the Features dialog box, you click **OK** to close it.

Note

Even though you clicked **Apply** or **OK** for your changes in the Features dialog box, you must still save your model to save all the changes you made. Clicking **Apply** or **OK** applies any changes to the currently opened model. However, to save the changes for the model so that they are in effect the next time you open it, you must save your model.

Note the following about the **Apply** and **OK** on the Features dialog box:

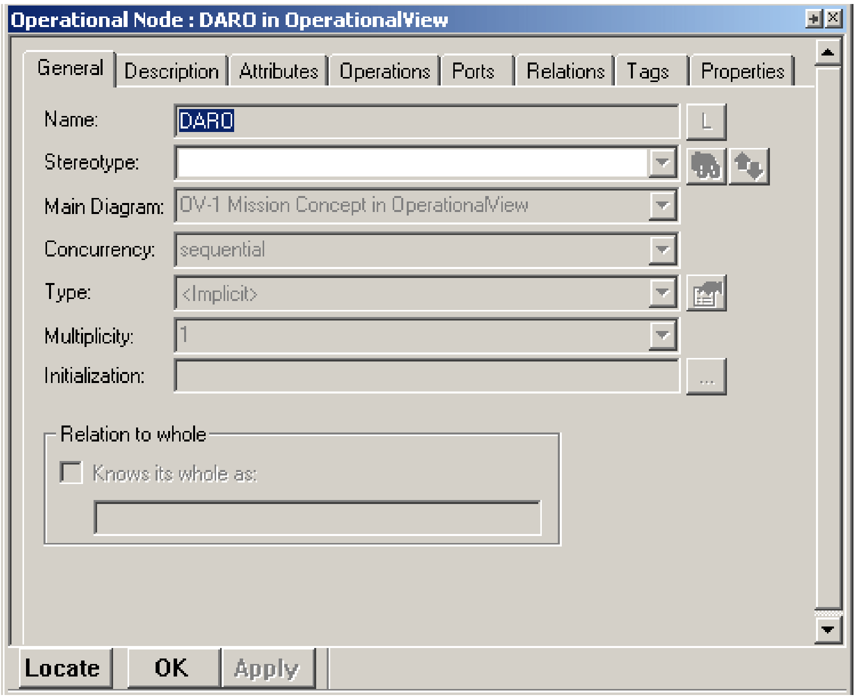
- Click **Apply** when you want to apply any changes you made to the Features dialog box but want keep it open. For example, you might need to apply a change before you can continue with using the Features dialog box, or you want to apply a change and see its effect before continuing making any more changes on the dialog box.
- Click **OK** when you want to apply your changes and close the Features dialog box at the same time.

Tabs for the Features Dialog Box

The Features dialog box has different tabs at the top of the dialog box and different boxes on the tabs depending on the element type.

The following tabs are common to all types of elements. For more information about these tabs, as well as the other tabs that you might see in the Features dialog box, refer to the section on it in the *IBM Rational Rhapsody User Guide*.

- **General** typically contains the name of the element and other general options, as shown in the following figure:



Operational Node : DARO in OperationalView

Name: DARO

Stereotype:

Main Diagram: OV-1 Mission Concept in OperationalView

Concurrency: sequential

Type: <Implicit>

Multiplicity: 1

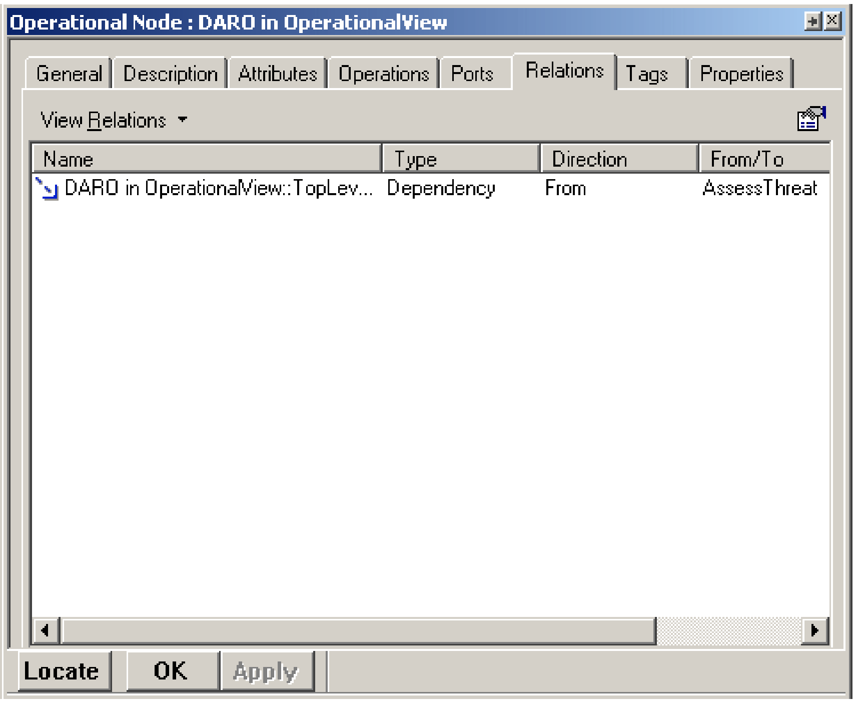
Initialization:

Relation to whole

□ Knows its whole as:

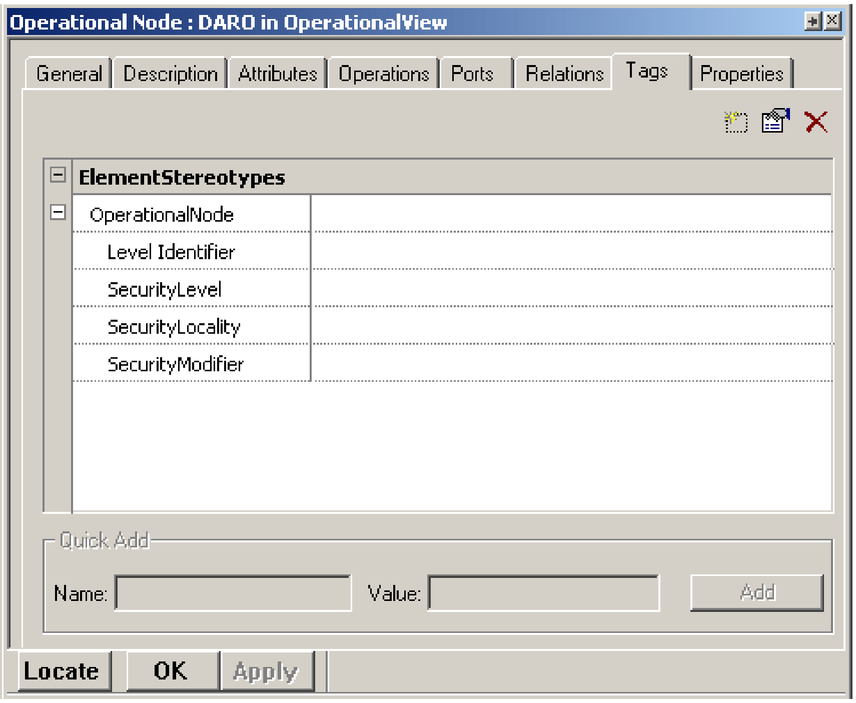
- **Description**, as its title implies, contains the description of the element, if it has been included.

- **Relations** lists all the relationships (dependencies, associations, and so on) an element is engaged with, as shown in the following figure:



Name	Type	Direction	From/To
DARO in OperationalView::TopLev...	Dependency	From	AssessThreat

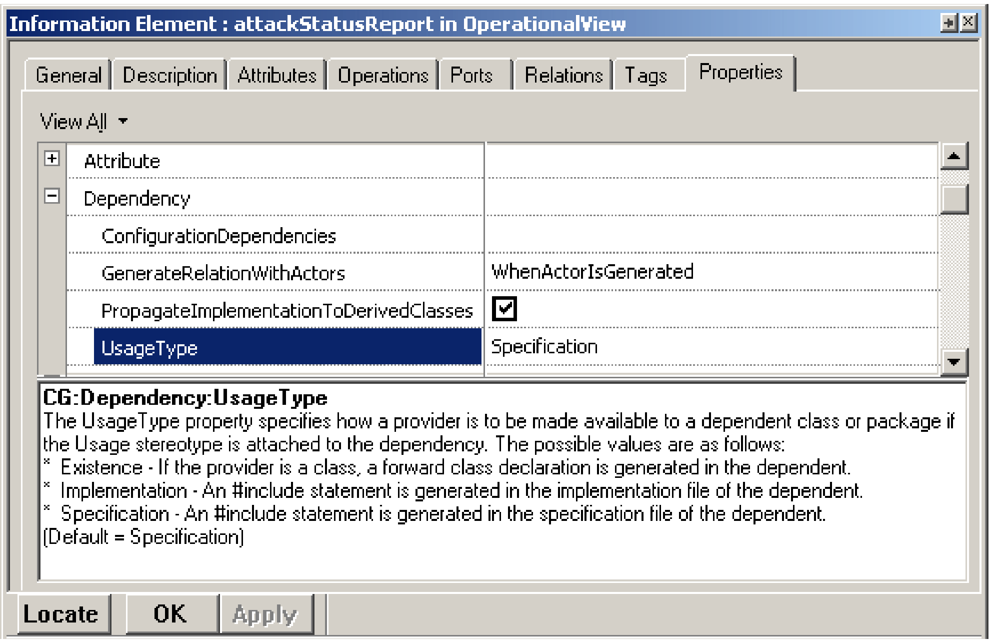
- **Tags** lists any tags available for an element. *Tags* enable you to add information to certain kinds of elements to reflect characteristics of the specific domain or platform for the modeled system. Refer to the *IBM Rational Rhapsody User Guide* for more information about tags.



ElementStereotypes

OperationalNode	
Level Identifier	
SecurityLevel	
SecurityLocality	
SecurityModifier	

- **Properties** lists the properties associated with the Rational Rhapsody element.
  - The top left column on this tab shows the metaclass and property (for example, **Dependency** and **UsageType**).
  - The top right column shows the default for the selected property, if there is one (for example, **Specification**).
  - The box at the bottom portion of the **Properties** tab shows the definition for the property selected in the upper left column of the tab. The definition display shows the names of the subject, metaclass, property, and the definition for the property, as shown in the following figure:



⊞ Attribute	
⊟ Dependency	
ConfigurationDependencies	
GenerateRelationWithActors	WhenActorIsGenerated
PropagateImplementationToDerivedClasses	☑
UsageType	Specification

CG:Dependency:UsageType
The UsageType property specifies how a provider is to be made available to a dependent class or package if the Usage stereotype is attached to the dependency. The possible values are as follows:
* Existence - If the provider is a class, a forward class declaration is generated in the dependent.
* Implementation - An #include statement is generated in the implementation file of the dependent.
* Specification - An #include statement is generated in the specification file of the dependent.
(Default = Specification)

Note: Rational Rhapsody documentation uses a notation method with double colons to identify the location of a specific property. For example, for the property in the above figure, the location is `CG::Dependency::UsageType` where `CG` is the subject, `Dependency` is the metaclass, and `UsageType` is the property.

Moving the Features Dialog Box

The Features dialog box is a floating window that can be positioned anywhere on the screen, or docked to the Rational Rhapsody GUI.

To dock the Features dialog box in the Rational Rhapsody window, do one of the following:

- Double-click the title bar. The dialog box docks. You can now drag it to another location if you want.
- Right-click the title bar and select **Docking by Drag**. Then drag the dialog box to another location.

To undock the Features dialog box, do one of the following:

- Double-click the title bar to undock it.
- Right-click the title bar and clear **Docking by Drag**, then drag the dialog box to another location.

DoDAF Basics

The model created in this tutorial uses the “out-of-the-box” version of the Rational Rhapsody DoDAF Add-on. The DoDAF Add-on is a template-driven solution that can be customized and extended to meet specific customer requirements and development processes. You can contact IBM Rational Rhapsody support for additional information on the Rational Rhapsody DoDAF Add-on.

DoDAF defines four views:

- Operational View
- Systems View
- Technical View
- All View

The architecture model created in this tutorial includes the architecture products listed in the following table:

Architecture Product	View	Product Name	Product Description
All Views	Package	AllViews	This optional stereotyped package allows you to add in AV products and other views and packages.
AV-1	All	Overview and Summary Information	This product is typically a text (Word, FrameMaker, HTML) document. You can add AV-1documents and launch them by clicking on them.
AV-2	All	Integrated Dictionary	This is a DoDAF-generated text product (report).
Operational View	Package		This optional stereotyped package is similar to the All View product. It supports all the operational products.
OV-1	Operational	High-Level Operational Concept Graphic	This High-level graphical/textual description of the operational concept allows you to import pictures and other operational elements, such as Operational Nodes, Human Operational Nodes, Operational Activities, and the relations among them.
OV-2	Operational	Operational Node Connectivity Description	This product shows the connections and flows among operational nodes and operational activities. If wanted, the behavior of operational nodes and operational activities can be shown by adding OV-5, OV-6a, OV-6b, and OV-6c diagrams. These diagrams are the primary source of information used by the DoDAF add-on to create the OV-3 diagram.

Architecture Product	View	Product Name	Product Description
OV-3	Operational	Operational Information Exchange Matrix	This product shows information exchanged between nodes and the relevant attributes of that exchange. OV-3 is generated from the information shown in OV-2 and other operational diagrams. This information is stored as a CSV file and can be added to any product.
OV-5	Operational	Operational Activity Model	This product details the behavior of operational nodes or more commonly, operational activities.
OV-6a	Operational	Operational Rules Model	This product is a textual description of “business rules” for the operation. It is a controlled file. One of three products used to describe the mission objective.
OV-6b	Operational	Operational State Transition Description	This product is a statechart that can be used to depict the behavior of an operational element (node or activity). One of three products used to describe the mission objective.
OV-6c	Operational	Operational Event Trace Description	This product is a sequence diagram that captures the behavioral interactions among and between operational elements and (in the Harmony process) captures the operational contracts among them. One of the three products used to describe the mission objective.
OV-7	Operational	Logical Data Model	This product is a class diagram that shows the relations among Informational Elements (data classes). This is similar to entity relationship diagrams, but is more powerful.
System View	Package		This optional stereotyped package is similar to other views, but contains system elements.
SV-1	Systems	Systems Interface Description	This product is a diagram that contains System nodes, systems, system parts and the connections between them (links). These can be used with or without ports.
SV-2	Systems	Systems Communications Description	This product is a diagram that shows the connections among systems via the communications systems and networks.
SV-3	Systems	Systems-Systems Matrix	This product is generated from the information in the other system views. SV-3 assumes that there are links between items stereotyped SystemNode, System, or System Part and represents these in an N^2 diagram.

Architecture Product	View	Product Name	Product Description
SV-4	Systems	Systems Functionality Description	This product represents the connection between System Functions and Operational Activities. The connection is made by drawing a Realize dependency line from the System Function to the Operational activity on the diagram. System Functions are mapped onto the system elements that support them by making System Functions parts of the system elements (that is, System Functions are drawn within the other system elements). System elements can also realize system functions. Note that here, as in almost all the other views, you can use Performance Parameters (bound to their constrained elements via anchors) to add performance data. This is summarized in SV--7.
SV-5	Systems	Operational Activity to Systems Function Traceability Matrix	This product is a spreadsheet-like generated view summarizing the relations among system elements (system nodes, systems and system parts), system functions that they support, and the mapping to operational activities.
SV-6	Systems	Systems Data Exchange Matrix	This product shows the information in the flows (information exchanges) between system elements. They might be embedded flows (bound to the links) or they might be flows independent of links. This is a spreadsheet-like generated product.
SV-7	Systems	Systems Performance Parameters Matrix	This is a generated spreadsheet-like product, showing all the performance parameters and the elements that they constrain.
SV-8	Systems	Systems Evolution Description	This product is the system evolution description. This is an activity diagram (there is a SystemProject element stereotype to serve as the “base” for this activity diagram). SV-8 depicts the workflow for system development, object nodes for products released, and performance parameters for things like start and end dates, slack time, and so forth.
SV-9	Systems	Systems Technology Forecast	This product is a text document - a stereotype of a Controlled File.
SV-10a, SV-10b, SV-10c	Systems	Systems Rules Model, Systems State Transition Description, Systems Event Trace Description	These products are similar to the OV-6a, OV-6b, and OV-6c products, but they are separately identified, even though they are structurally identical.
SV-11	Systems	Physical Schema	This product is similar to the OV-7 class diagram. This product uses a class diagram to show physical schema (data representation).

Getting Started with DoDAF Architecture

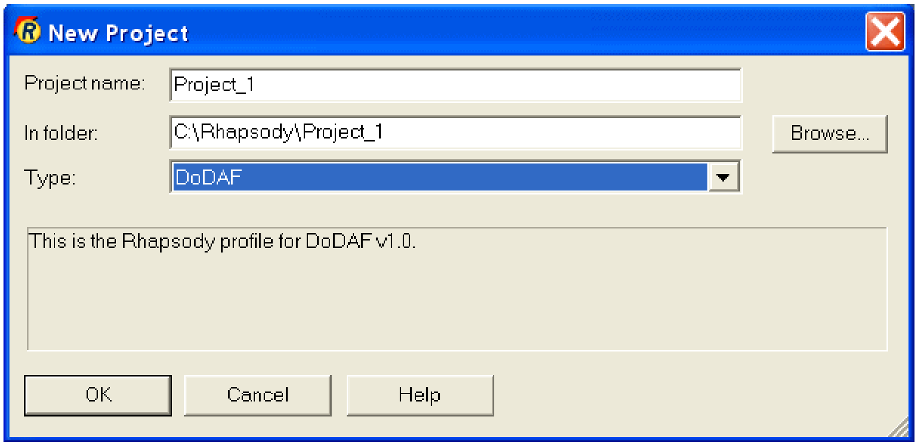
In this tutorial, you are provided with an AV-1 Word document and OV-1 bitmap image file to be included with the architecture model. You will create the following Operational View Products using UML diagrams based on the following mapping.

Architecture Product	Description
AV-1	Overview and Summary Information
AV-2	Integrated Dictionary Note: The Rational Rhapsody model is a dynamic integrated dictionary of the architecture. A snapshot of the AV-2 product is also included in the DoDAF report generated from the model.
OV-1	High Level Operational Concept Graphic
OV-2	Operational Node Connectivity Description
OV-3	Operational Information Exchange Matrix Note: The OV-3 and SV-5 products are not manually created, but rather are automatically derived from the architecture model. Since the creation of the Systems View has many parallels with the creation of the Operational View, this tutorial includes some hands-on work in this area, as well as a discussion on creating additional Systems View products.
OV-5	Mission Objective Model
OV-6b	Operational State Transition Description
OV-6c	Operational Event-Trace Description
OV-7	Logical Data Model
SV-1	Systems Interface Description
SV-4	Systems Functionality Description
SV-5	Operational Activity to Systems Function Traceability Matrix Note: The SV-5 and OV-3 products are not manually created, but rather are automatically derived from the architecture model. Since the creation of the Systems View has many parallels with the creation of the Operational View, this tutorial includes some hands-on work in this area, as well as a discussion on creating additional Systems View products.

Creating a New Rational Rhapsody Project

Start Rational Rhapsody and create a new project for your Coordinated Land and Sea Attack architecture model.

1. To start Rational Rhapsody, from the Windows Start menu, select **Programs** > **IBM Rational** > **IBM Rational Rhapsody *Version#*** > **Rhapsody Developer Edition** > **Rhapsody in C++**. The Rational Rhapsody graphical user interface (GUI) opens.
2. Select **File** > **New**. The New Project dialog box opens.
3. Type your **Project name** (`Project_1` in this example), specify a folder location (directory), and select **DoDAF** from the drop-down list as the project **Type** (profile) menu, as shown in the following example:



New Project

Project name: Project_1

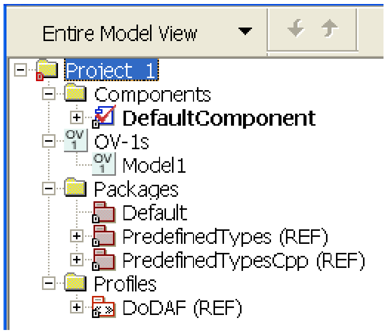
In folder: C:\Rhapsody\Project_1

Type: DoDAF

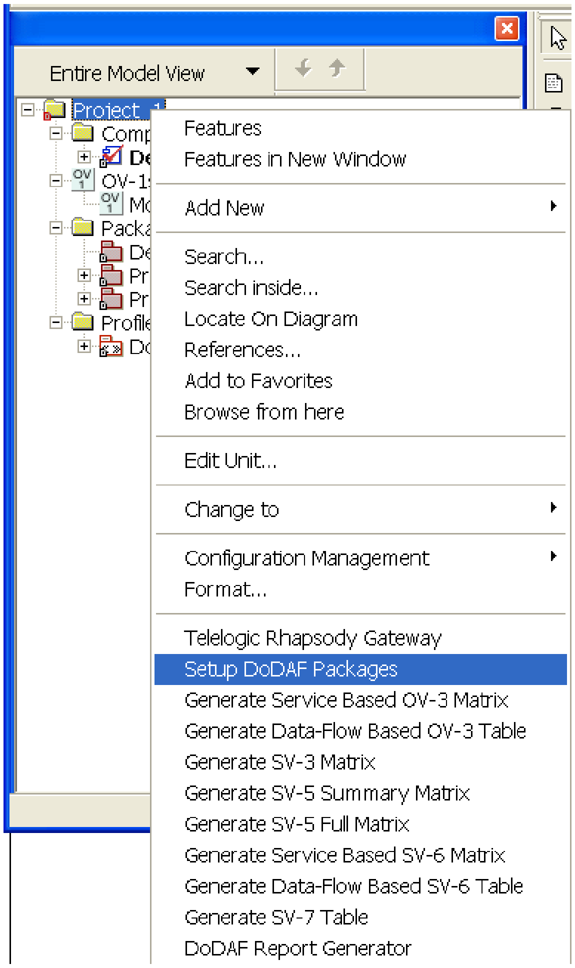
This is the Rhapsody profile for DoDAF v1.0.

4. Click **OK**.
5. If the folder you specified does not exist, you are asked if you want to create it. Click **Yes**. This new project is the standard starting point for a Rational Rhapsody project.

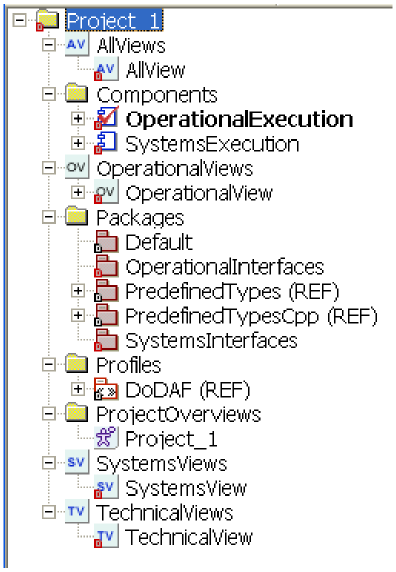
6. Expand **OV-1s**, **Packages** and **Profiles** by clicking the corresponding **+** sign in the Rational Rhapsody browser, as shown below. The packages and profiles, shown in your diagram, might vary depending on your site properties and product licensing.



7. To initialize the DoDAF project, right-click your project folder (named `Project_1` in this example) and select **Setup DoDAF Packages**. If you do not see this menu item, see **Manually Adding the DoDAF Helpers** in the “Troubleshooting” section of this tutorial.



8. The DoDAF helper adds some new packages, views, and overviews to the project (as shown in the example). Expand the folders to examine these project additions.



```
Project_1
  AllViews
    AllView
  Components
    OperationalExecution
    SystemsExecution
  OperationalViews
    OperationalView
  Packages
    Default
    OperationalInterfaces
    PredefinedTypes (REF)
    PredefinedTypesCpp (REF)
    SystemsInterfaces
  Profiles
    DoDAF (REF)
  ProjectOverviews
    Project_1
  SystemsViews
    SystemsView
  TechnicalViews
    TechnicalView
```

Example DoDAF Model Information

This tutorial guides you through the creation of an as-is architecture model for Coordinated Land and Sea attacks. All information used to create this architecture model is open-source.

This model assumes the viewpoint of battle commanders, providing visibility to the chain of command followed in Coordinated Land and Sea attacks. The model considers several missions in creating your architecture model, including reconnaissance, mobilization, covert entry team attacks, missile attacks, and damage assessment.

The operational nodes involved in this architecture include the following:

- DARO - Defense Airborne Reconnaissance Office, responsible for Airborne Reconnaissance
- DOCC - Deep Operations Coordination Cell, responsible for undercover operations
- JCS - Joint Chiefs of Staff, responsible for making decisions regarding multiple forces
- JFMCC - Joint Force Maritime Component Commander, responsible for commanding the naval component of an Attack
- JIC - Joint Intelligence Committee, responsible for assessing intelligence from the field
- JIPS - Joint Intelligence Image Processing Service, responsible for processing image data

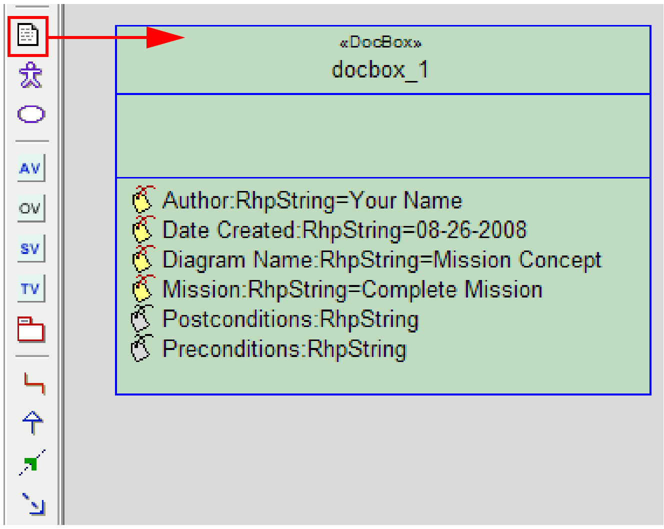
DoDAF Project Overview

The **ProjectOverviews** folder is listed after the **Profiles** folder. The **ProjectOverviews** folder contains the **Project_1** file which contains a high-level overview of the entire model. The **Project_1** file can be renamed by being clicking on twice.

To enter information into the project overview file, follow these steps:

1. Double-click the **Project_1** file in the browser.

2. Click the **Diagram (Documentation Box)** icon on the Drawing toolbar and click in the right pane to create a starting point <<DocBox>>, as shown in this example:

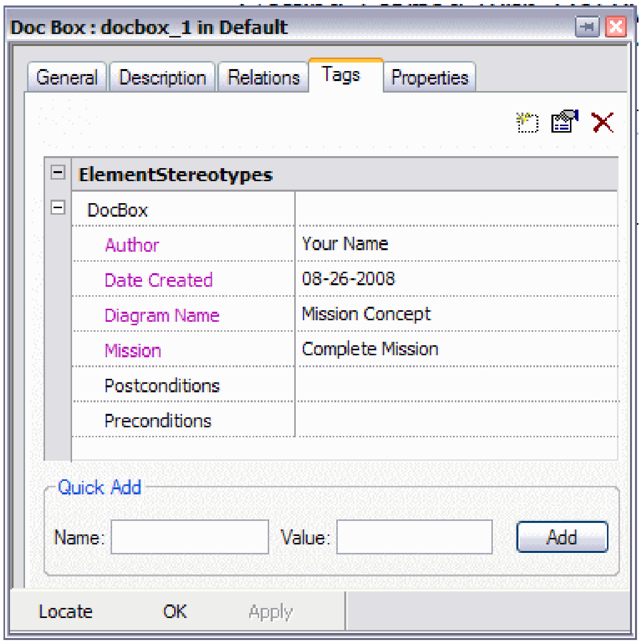


«DocBox»
docbox_1

Author:RhpString=Your Name
Date Created:RhpString=08-26-2008
Diagram Name:RhpString=Mission Concept
Mission:RhpString=Complete Mission
Postconditions:RhpString
Preconditions:RhpString

3. Right-click **<<DocBox>>** in your drawing area and select **Features**.

4. On the **Tags** tab, enter the appropriate information for the **Author**, **Date Created**, **Diagram Name**, and **Mission** you are documenting **DocBox** fields:



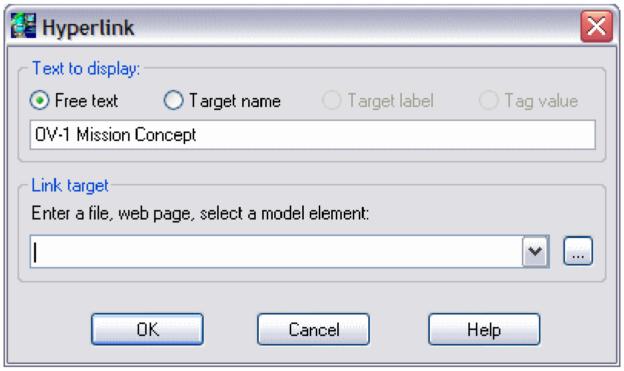
ElementStereotypes	
DocBox	
Author	Your Name
Date Created	08-26-2008
Diagram Name	Mission Concept
Mission	Complete Mission
Postconditions	
Preconditions	

5. Click **OK** when finished.

6. Return to the diagram and click twice in the middle cell of <<**DocBox**>> on your drawing area and type in the name of the object, element, reference, or note you want the <<**DocBox**>> to represent. This example uses `OV-1 Mission Concept`.

To create a hyperlink to the object, element, reference, or note you entered in <<**DocBox**>>, follow these steps:

1. Highlight the text in the middle cell <<**DocBox**>>, right-click, and select **Hyperlink**. A Hyperlink dialog box displays, as shown in the following figure:



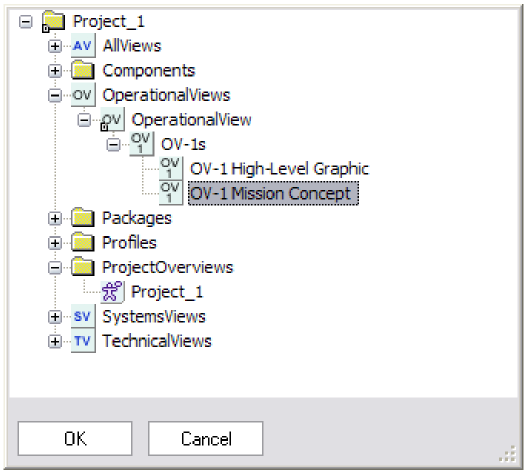
● Free text ○ Target name ○ Target label ○ Tag value

OV-1 Mission Concept

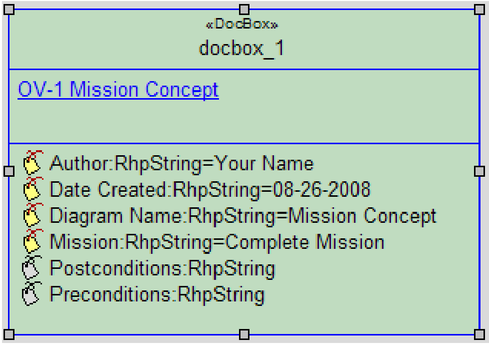
Link target

Enter a file, web page, select a model element:

2. Using the radio buttons, select either the **Free text** to display or the **Target name** and click the **Link target** drop-down menu (see the example below) to select the model object, element, or reference for this link.



3. Click **OK** twice and observe the added hyperlink, as shown in this example.



«DocBox»
docbox_1

OV-1 Mission Concept

Author:RhpString=Your Name
Date Created:RhpString=08-26-2008
Diagram Name:RhpString=Mission Concept
Mission:RhpString=Complete Mission
Postconditions:RhpString
Preconditions:RhpString

4. Click the new hyperlink and the **OV-1 Mission Concept** diagram displays.

Adding the AV-1 to the Rational Rhapsody Model

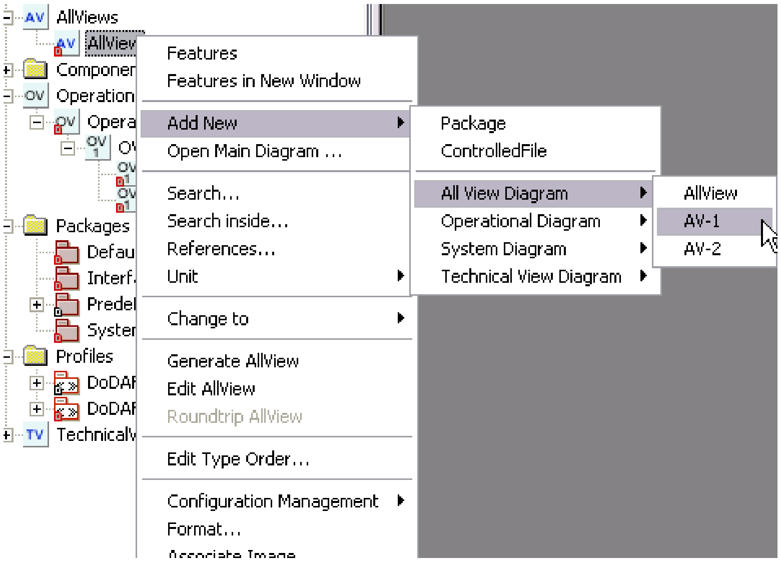
The AV-1 Overview and Summary Information product provides on overview of the architecture, including the purpose, context, scope, and mission scenarios. In this tutorial, you provide a Word Document to use as the AV-1 for your architecture model.

Note

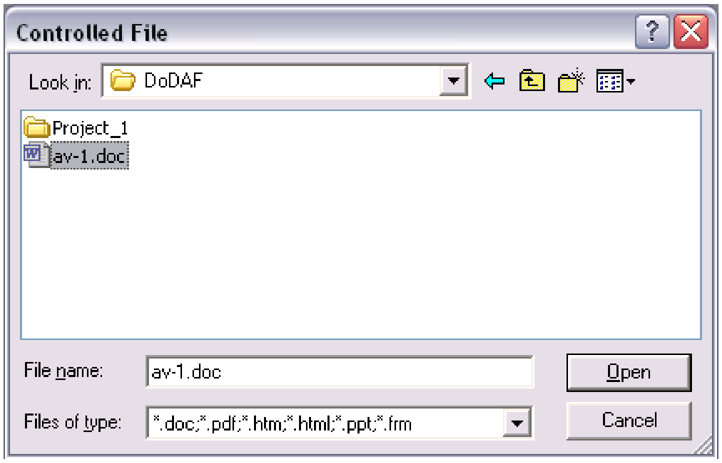
Before proceeding, locate the av-1.doc and ov-1.bmp files using Windows Explorer. If you are unsure about where the Rational Rhapsody DoDAF Add-on Tutorial is installed on your computer, see **Verifying the Rational Rhapsody DoDAF Add-on Installation** in the Troubleshooting section of this tutorial.

To add the AV-1 Overview and Summary Document to the model, follow these steps:

1. In the Rational Rhapsody browser, expand **AllViews**.
2. Right-click **AllView** and select **Add New > All View Diagram > AV-1**, as shown in the following figure:



3. The Controlled File dialog box appears. Select **av-1.doc** and click **Open**.



Controlled File

Look in: DoDAF

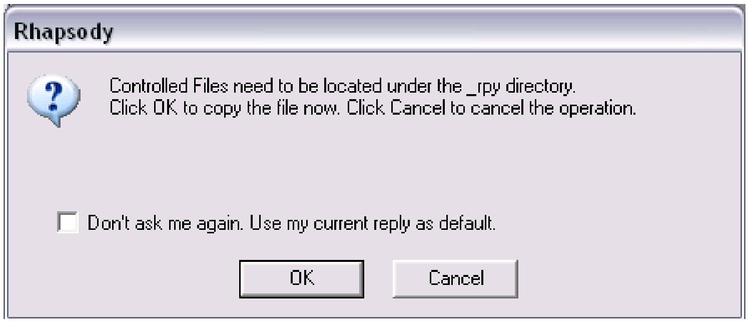
Project_1

av-1.doc

File name: av-1.doc

Files of type: *.doc;*.pdf;*.htm;*.html;*.ppt;*.frm

4. A message appears asking you to click **OK** to copy the AV-1.doc file to the _rpy directory, as shown in the following figure. Click **OK** to continue. By clicking **OK**, the AV-1.doc file is now maintained as part of the Rational Rhapsody model repository.

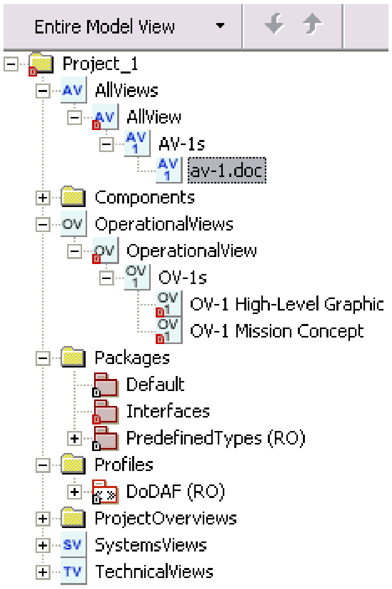


Rhapsody

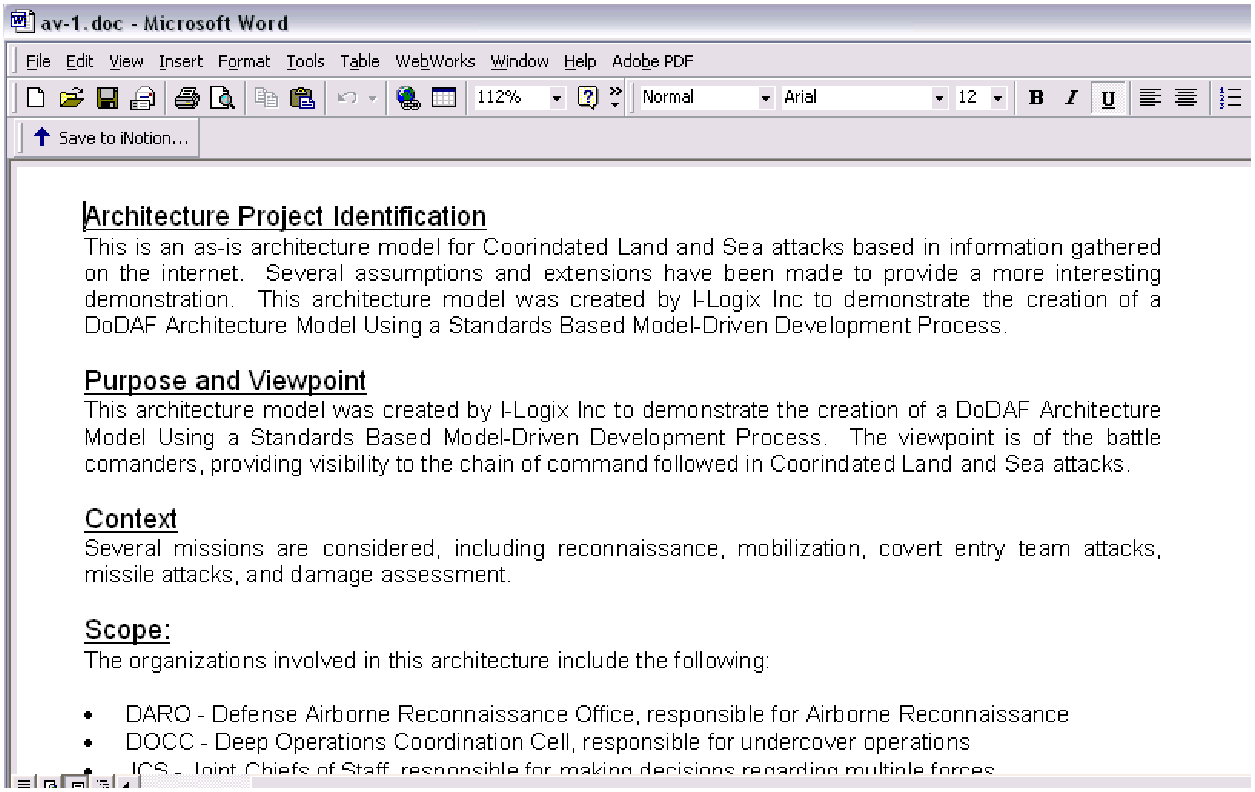
Controlled Files need to be located under the _rpy directory.
Click OK to copy the file now. Click Cancel to cancel the operation.

□ Don't ask me again. Use my current reply as default.

5. Your Rational Rhapsody browser should resemble the following figure:



6. Test the av-1 file by double-clicking it in the Rational Rhapsody browser and confirm the av-1.doc file is displayed, as shown in the following figure:


av-1.doc - Microsoft Word
File Edit View Insert Format Tools Table WebWorks Window Help Adobe PDF
112%
Normal
Arial
12
B I U
Save to iNotion...

Architecture Project Identification

This is an as-is architecture model for Coorindated Land and Sea attacks based in information gathered on the internet. Several assumptions and extensions have been made to provide a more interesting demonstration. This architecture model was created by I-Logix Inc to demonstrate the creation of a DoDAF Architecture Model Using a Standards Based Model-Driven Development Process.

Purpose and Viewpoint

This architecture model was created by I-Logix Inc to demonstrate the creation of a DoDAF Architecture Model Using a Standards Based Model-Driven Development Process. The viewpoint is of the battle comanders, providing visibility to the chain of command followed in Coorindated Land and Sea attacks.

Context

Several missions are considered, including reconnaissance, mobilization, covert entry team attacks, missile attacks, and damage assessment.

Scope:

The organizations involved in this architecture include the following:

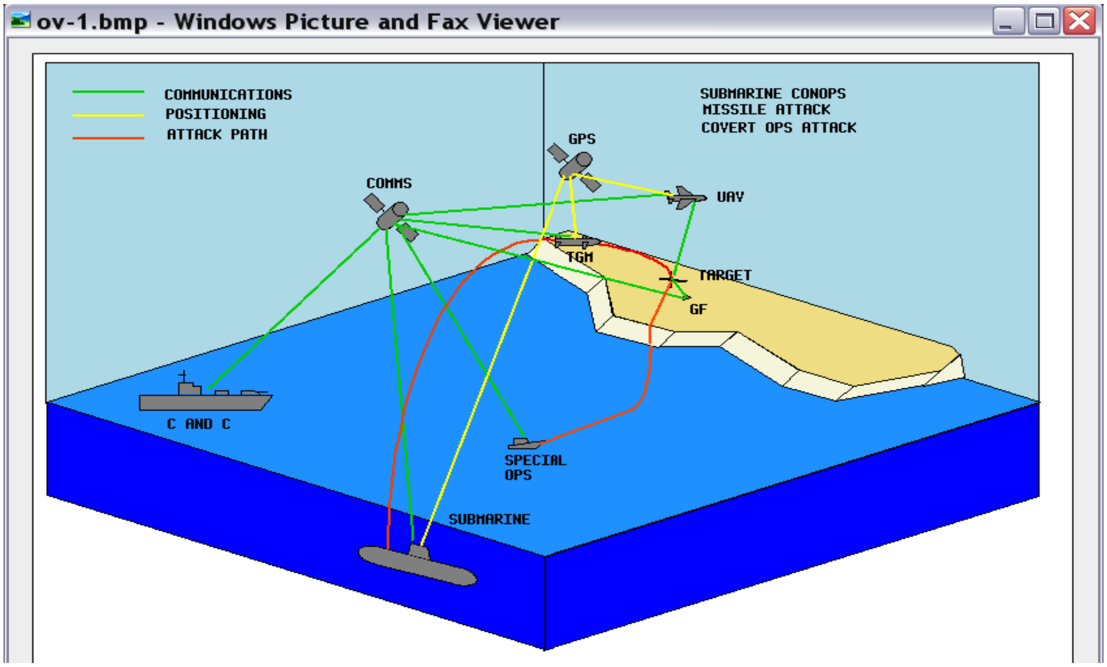
- DARO - Defense Airborne Reconnaissance Office, responsible for Airborne Reconnaissance
- DOCC - Deep Operations Coordination Cell, responsible for undercover operations
- JCS - Joint Chiefs of Staff, responsible for making decisions regarding multiple forces

Creating the Operational View Model

Now that you have associated AV-1 with the project, you can add OV-1, the High-Level Operational Concept Graphic.

Adding the OV-1 High Level Concept Graphic

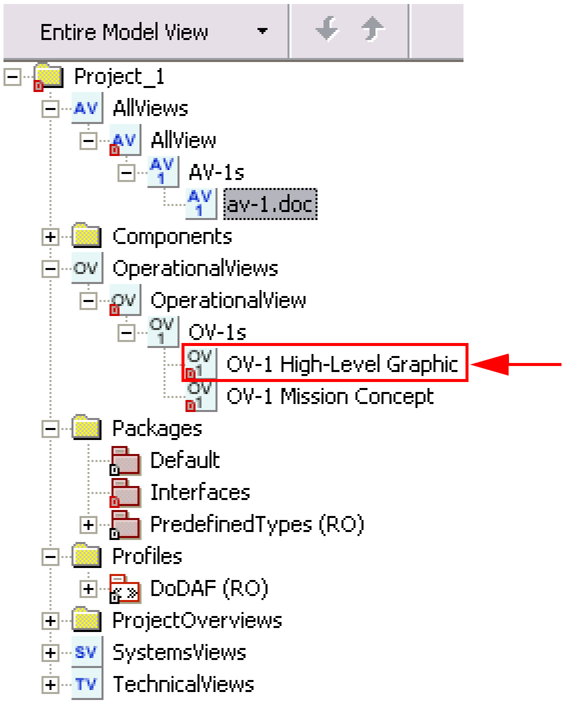
Traditionally, the OV-1 is a picture depicting the mission and the main operational nodes. In your architecture model, both a traditional picture as well as a diagram to communicate the high level mission concept are used, as shown in this example.



Begin by opening **OV-1 High-Level Graphic**, which was created when you set up your Rational Rhapsody DoDAF project.

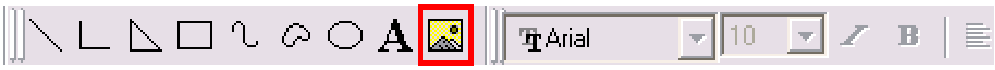
To do this, follow these steps:

1. In the Rational Rhapsody browser, double-click **OV-1 High-Level Graphic**, as shown in the following figure:



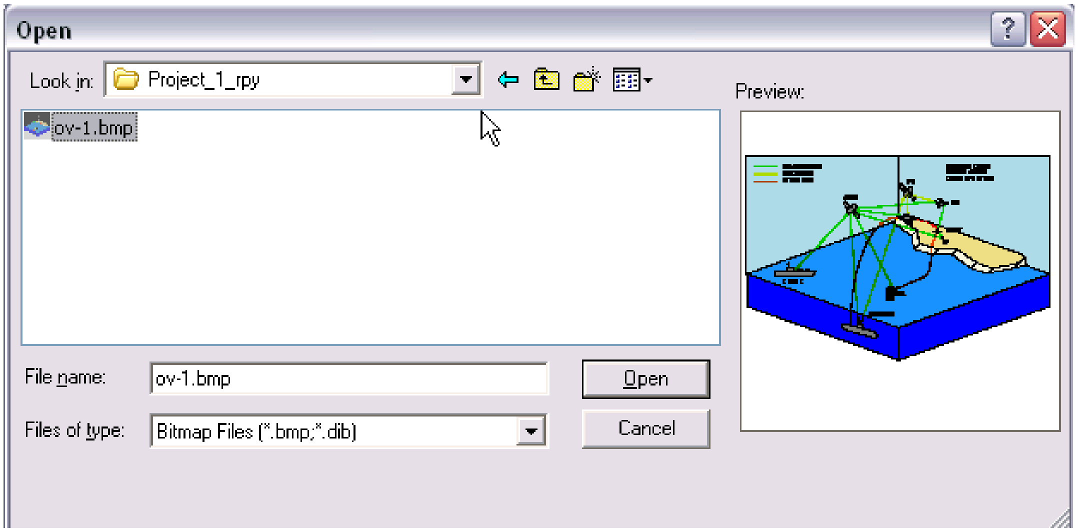
2. Maximize the diagram window using the □ icon if it is not maximized.

3. Click the **Image** icon on the **Free Shapes** toolbar, as shown in the following figure, to open the **Open** dialog box.



4. Select the ov-1.bmp image file, as shown in the following figure.

   Note: You might have to select **All Files** from the **Files of type** drop-down list to see .bmp files. You can use .jpg and image formats other than bitmaps.



Open

Look in: Project_1_rpy

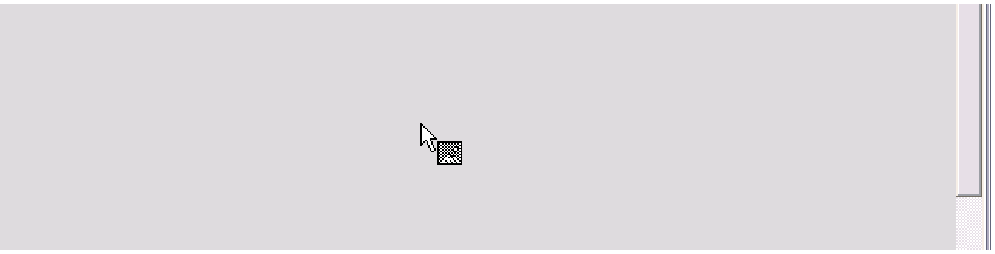
ov-1.bmp

File name: ov-1.bmp

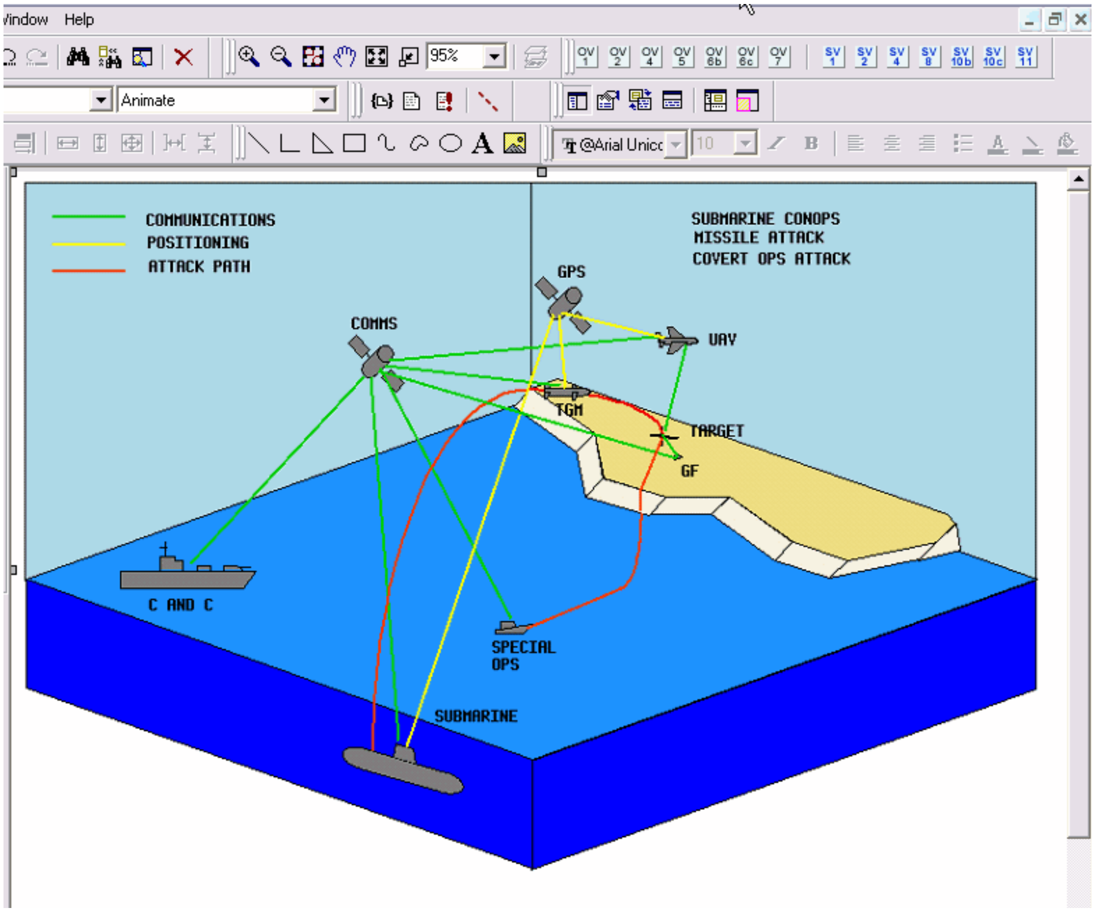
Files of type: Bitmap Files (*.bmp;*.dib)

5. Click **Open**.

6. A square graphic icon attached to the mouse pointer appears, as shown in the following figure:



7. Click the drawing area on the right side of the Rational Rhapsody window to place the image. The ov-1.bmp appears, as shown in the following figure:

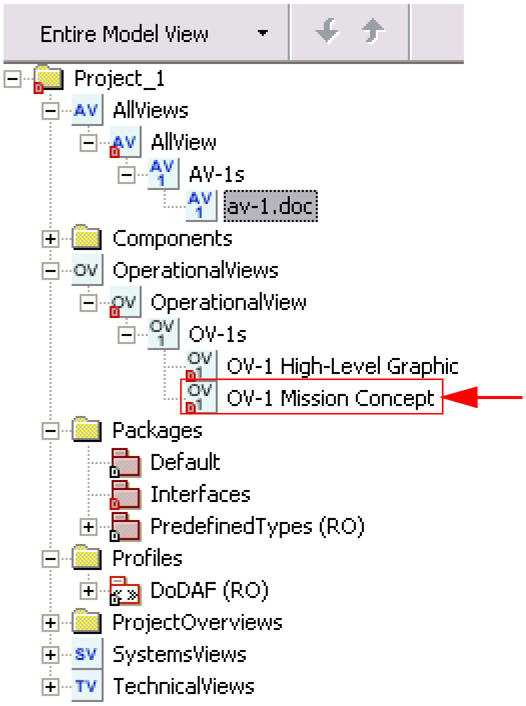


8. Use the Zoom to Fit icon , or the zoom setting 100% , as needed to view the entire image.

9. Now that the **OV-1 High-Level Graphic** is complete, save the architecture model by choosing **File** >**Save**.

Creating the OV-1 Mission Concept Diagram

In addition to the picture for the OV-1, you will add a mission concept diagram to depict the primary missions and operational nodes for the project. The **OV-1 Mission Concept** diagram was created when you set up your Rational Rhapsody project for DoDAF in the Rational Rhapsody browser, as shown in the following figure:



For this architecture model, you will define two operational nodes:

- AssessThreat Mission Objective
- AttackTarget Mission Objective

AssessThreat Mission Objective

The AssessThreat Operational Node contains the following operations in sequential order.

1. The Joint Intelligence Committee (JIC) detects threats.
2. The JIC then informs the Joint Chiefs of Staff (JCS) if JIC determines the threat might require action.
3. JCS can then request additional information such as reconnaissance from the Defense Airborne Reconnaissance Office (DARO).
4. DARO gathers reconnaissance data, and then sends it to the Joint Intelligence Image Processing Service (JIPS) for processing.
5. JIPS sends the results to JCS who then determines if an attack on the threat is necessary.

AttackTarget Mission Objective

The AttackTarget Operational Node contains the following operations in sequential order.

1. When a threat is assessed and determined to be a target, then an attack needs to be:
   a. Planned
   b. Executed
   c. Assessed
2. JCS plans the attack, including choosing an attack method.
3. JCS then issues attack orders to the Joint Force Maritime Component Commander (JFMCC).
4. JCS requests the Deep Operations Coordination Cell (DOCC) to perform damage assessment.
5. JFMCC then executes a missile or covert entry team attack on the target.
6. After JFMCC executes the attack on the target, DOCC reports damage to JCS.
7. JCS issues another attack order or ends the attack by ordering a stand-down.

From the two Mission Objectives above (**AssessThreat Mission Objective** and **AttackTarget Mission Objective**), you have enough information to create the OV-1 mission concept diagram.

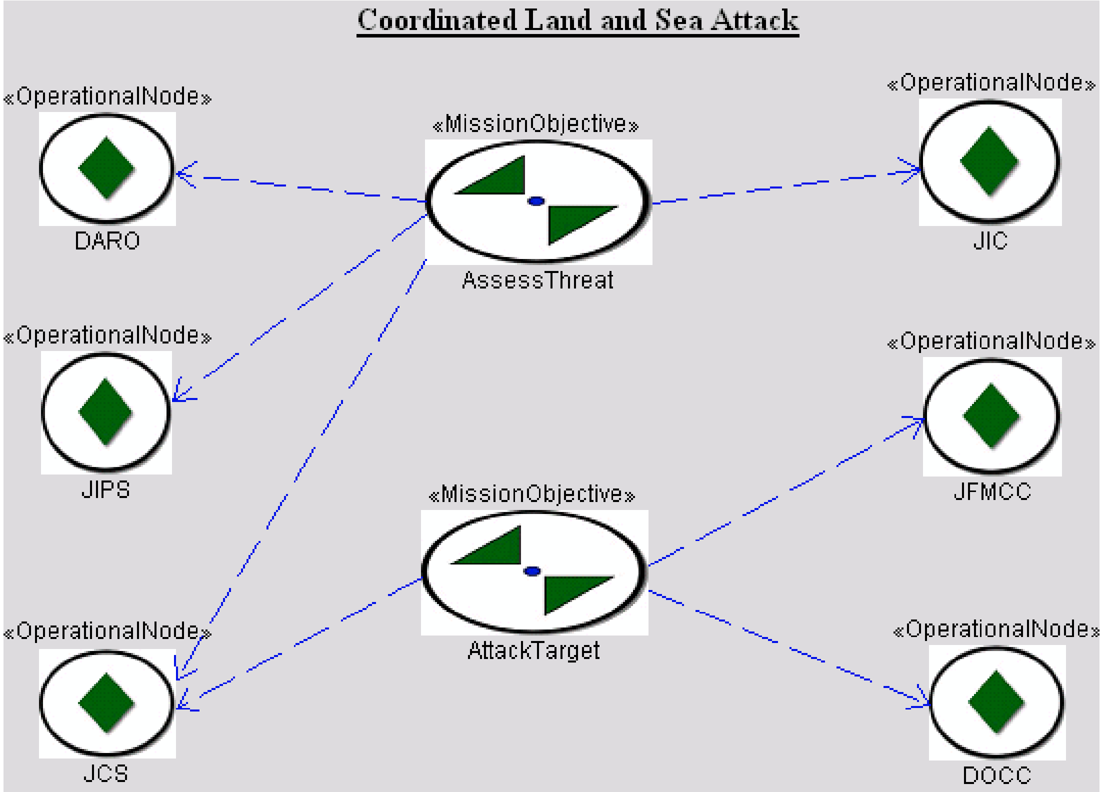
Note

Whenever you begin a new diagram, adding a docbox that includes the name and purpose (mission) of the diagram can be very helpful.

To draw the six Operational Nodes and show their association with the two Mission Objectives, follow these steps:

1. Double-click **OV-1 Mission Concept** in the Rational Rhapsody browser.

2. Use the **Operational Node** icon ⊙ on the **Drawing** toolbar to draw Operational Nodes called: `DARO`, `JIPS`, `JCS`, `JIC`, `JFMCC`, and `DOCC`, as shown in the following figure. As each node is drawn, the name is highlighted and can immediately be changed.



3. Use the **Mission Objective** icon on the **Drawing** toolbar to draw the two Mission Objectives: `AssessThreat` and `AttackTarget`. As with Operational Nodes, when each Mission Objective is drawn, the name is highlighted and can immediately be changed from the default name.

   Note: The Rational Rhapsody DoDAF add-on provides a military clipart image library that allows you to substitute appropriate images for various elements in the model. Right-click the element in the model, select **Display Options**, then select both **Enable Image View** and **Select an Image** and click the Ellipsis button ... to browse for images. The military clipart library is located at in the `<Rational Rhapsody installation path>\Share\Profiles\DoDAF\Images` subfolder.

4. Use the **Dependency** icon to draw lines from the AssessThreat Mission Objective to the Operational Nodes **DARO**, **JIC**, **JCS** and **JIPS**. Also draw dependency lines from the AttackTarget Mission Objective to the Operational Nodes **JCS**, **JFMCC**, and **DOCC**.

5. Use the **Text** icon A on the **Free Shapes** toolbar to add the title `Coordinated Land and Sea Attack` to the diagram.

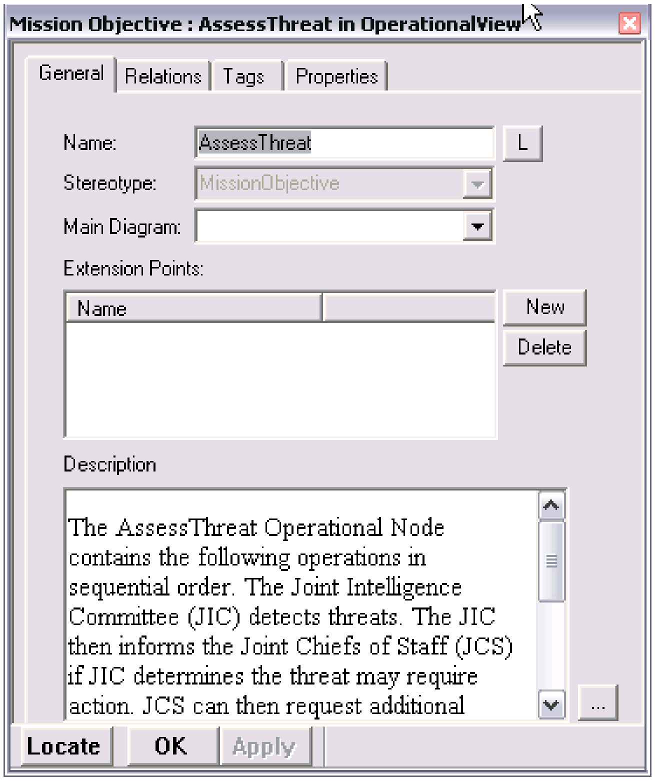
Note the following:

- You might find it helpful to turn on the grid and enable snap to grid when drawing the diagram. The grid is turned on using the icon, and snap to grid is enabled using the icon.
- In order to change the name of an operational node or activity, simply double-click the name and it will become editable. You can also double-click the node or activity to open the Features dialog box, and then change the name in the name field.
- If you want to completely delete anything you've drawn in the diagram, right-click the item and select **Delete from Model**.

Now that the **OV-1 Mission Concept** diagram is drawn, add descriptions to the Mission Objectives.

To add a description to a Mission Objective, follow these steps:

1. Double-click the Mission Objective in the **OV-1 Mission Objective** diagram to open the Features dialog box, as shown in the following figure:



Mission Objective : AssessThreat in OperationalView

Name: AssessThreat

Stereotype: MissionObjective

Main Diagram:

Extension Points:

Description

The AssessThreat Operational Node contains the following operations in sequential order. The Joint Intelligence Committee (JIC) detects threats. The JIC then informs the Joint Chiefs of Staff (JCS) if JIC determines the threat may require action. JCS can then request additional

2. Using the **Ctrl+C** (copy) and **Ctrl+V** (paste) keystrokes on the keyboard, copy and paste the Mission Objective descriptions (starting with **AssessThreat Mission Objective**) presented earlier in this section of the tutorial into the description area of the Features dialog box for each Mission Objective. You might have to touch-up the text for formatting purposes.

3. Click **OK** to close the Features dialog box.

4. Save the architecture model using **File** > **Save**.

Your **OV-1 Mission Concept** diagram is now complete. As with the **OV-1 High-Level Graphic** diagram, a link from the **OperationalView** package to this diagram was automatically added for you when the DoDAF project was created.

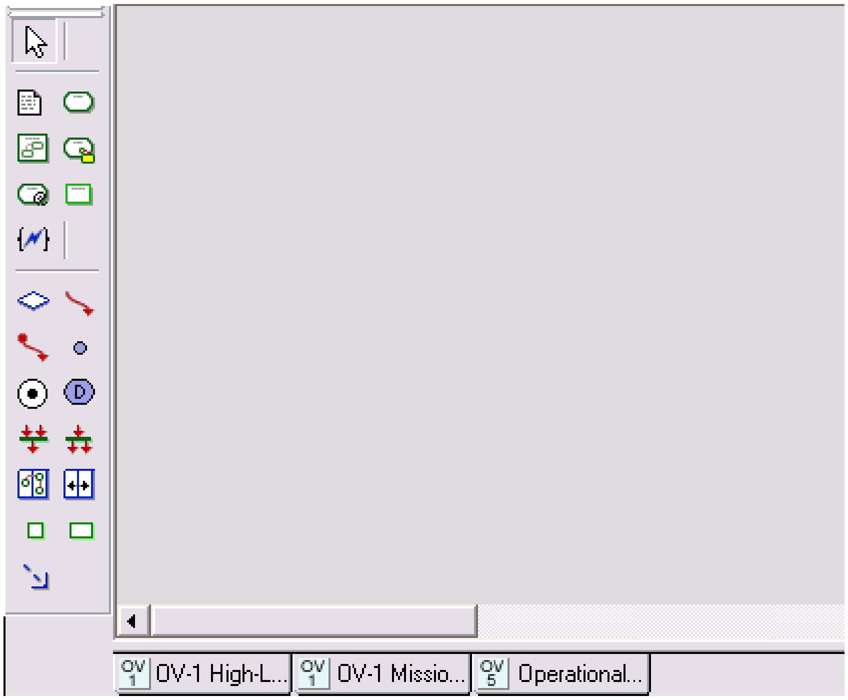
Creating the OV-5 Mission Objective Model Diagram

The **OV-5 Mission Objective Model** diagram is used to display the operations that are normally conducted in the course of achieving a mission. Based on the Mission Objective descriptions from the previous section, you have enough information to create the **OV-5 Mission Objective Model** diagram for the Mission Objectives in the model.

The OV-5 is closely associated to the Mission Objective, and in practice, each Mission Objective in the **OV-1 Mission Concept** diagram will have an OV-5 associated with it. Use an activity diagram to capture the OV-5 information. Focus only on the **AttackTarget** Mission Objective; you can create an OV-5 for the **AssessThreat** Mission Objective on your own as an exercise. The OV-5 you create for the **AttackTarget** Mission Objective will later serve as a guide for developing the Event Trace Descriptions (OV-6c) for the Mission Objective.

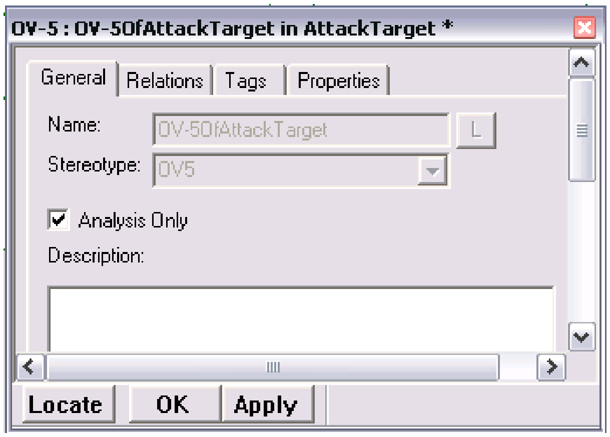
To create the OV-5, follow these steps:

1. Right-click the **AttackTarget** Mission Objective, and select **New OV-5**.
2. A new window appears, as shown in the following figure:



3. Right-click anywhere on the drawing area and select **Diagram Properties**. The Diagram Properties dialog box opens.

4. On the **General** tab, select the **Analysis Only** check box, as shown in the following figure:



OV-5 : OV-5OfAttackTarget in AttackTarget *

Name: OV-5OfAttackTarget

Stereotype: OV5

☑ Analysis Only

Description:

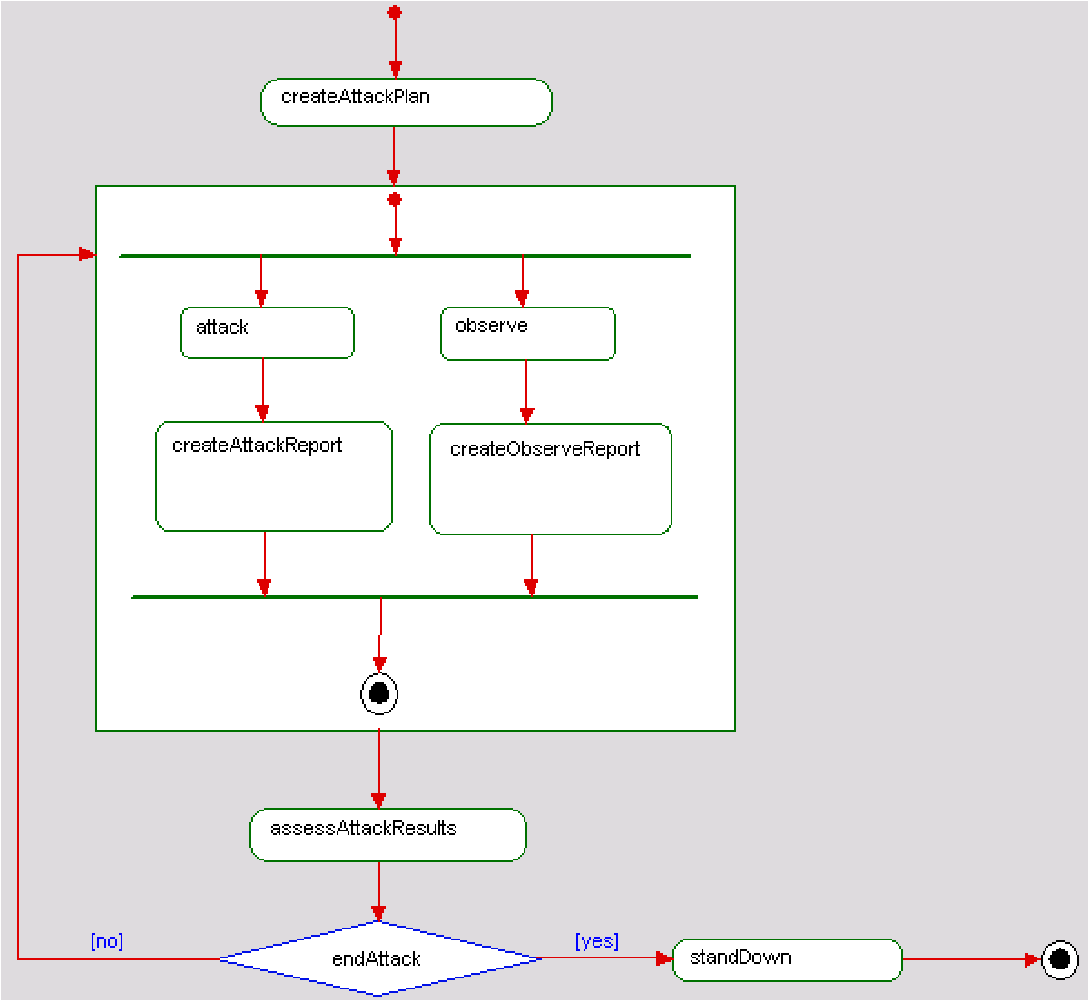
5. Click **OK**.

6. A prompt appears asking you to verify the operation. Click **OK**.

Based on the AttackTarget Mission Objective description, you know the basic flow of operations is to:

1. Create the attack plan.
2. JFMCC carries out the attack and reports to JCS.
3. DOCC performs damage observation and reports to JCS.
4. JCS starts another phase in the attack plan or decides to end the attack by standing down.

The completed OV-5 activity diagram is shown in the following figure.



In the activity diagram, the flow of execution is as follows:

Note

Whenever you begin a new diagram, adding a docbox that includes the name and purpose (mission) of the diagram can be very helpful.

1. Creating an attack plan (createAttackPlan in the diagram).

2. After the attack plan is created, a Fork Sync Bar is used to show that the:

   a. Attack (attack)

   b. Creation of the attack report (createAttackReport)

   c. Observation (observe)

   d. Creation of the observation report (createObserveReport)

      All happen simultaneously

   e. A termination state is added to show the end of the simultaneous actions.

   f. The attack results are assessed (assessAttackResults)

   g. A decision to continue the attack or stand down happens ("endAttack" in diamond)

   h. A "no" response to the "endAttack" question repeats the attack scenario (transition line)

   i. A "yes" response results in a stand down (standDown) and terminates the flow of execution (termination state ⊙).

To create the OV-5 diagram, follow these steps:

1. Use the **Action** icon on the **Drawing** toolbar to draw the operations: `createAttackPlan`, `attack`, `createAttackReport`, `observe`, `createObserveReport`, `assessAttackResults`, and `standDown`.

   Note: You can right-click any mission objective and select **Edit Text** to create or change the name.

2. Use the **Condition Connector** icon on the **Drawing** toolbar to draw the conditional branch below the **assessAttackResults** operation.

3. Use the **Termination State** icon on the **Drawing** toolbar to draw the termination state next to the **standDown** operation.

4. Use the **Fork Sync Bar** icon on the **Drawing** toolbar and click and drag horizontally from left to right to draw the fork below the **createAttackPlan** operation.

5. Use the **Join Sync Bar** icon on the **Drawing** toolbar and click and drag horizontally from left to right to draw the join line above the **assessAttackResults** operation.

6. Use the **Action Block** icon on the **Drawing** toolbar to draw the box around attack, **createAttackReport**, **observe**, and **createObserveReport**. You use the action block because there are multiple control paths that execute the actions within the block.

7. Draw a termination state below the join sync bar to indicate the end of the actions in the action block.

8. Use the **Default Flow** icon on the **Drawing** toolbar to draw the default arrow to the **createAttackPlan** action and the ForkSync Bar.

9. Use the **Activity Flow** icon on the **Drawing** toolbar to draw the remaining flows shown in the diagram.

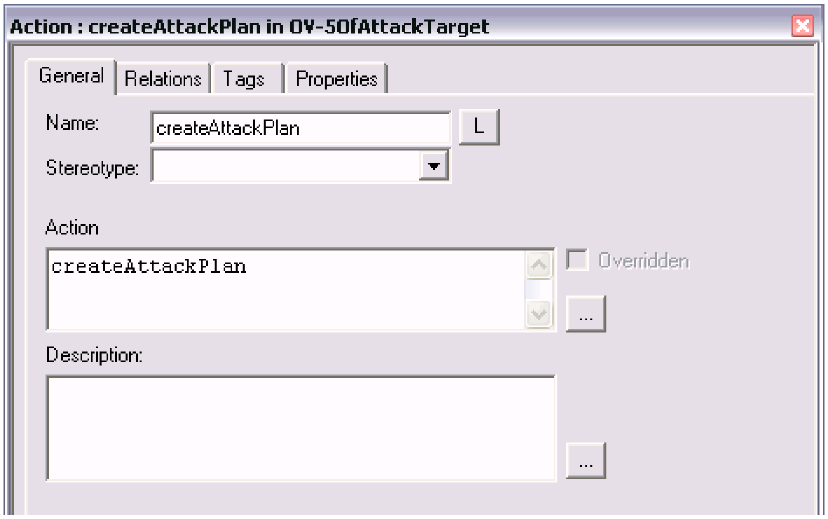
   Note: To change the Activity Flow line style (curved splines, straight, rectilinear, and so forth) right-click an Activity Flow line, and then select **Line Shape** and **Rectilinear**.

10. Double-click an Activity Flow to add the `yes` and `no` transition labels (located in the **Guard** field).

11. Save the completed architecture model using **File > Save**.

Note

To change the name of an operation, double-click the object (action box, condition connector, and so forth) to open the Features dialog box, as shown in the following figure. Change the name in the **Action** field of the Features dialog box, and then click **OK**.



Action : createAttackPlan in OV-5OfAttackTarget

Name: createAttackPlan

Stereotype:

Action

createAttackPlan

Description:

Although it is not required, it is possible to use swimlanes in the activity diagram to show which operational nodes are responsible for performing the operations. Swimlanes are added using the **Swimlanes Frame** icon and the **Swimlanes Divider** icon on the **Drawing** toolbar. You assign operations to operational nodes when you create the OV-6c Event Trace Descriptions.

Creating the OV-2 Operational Node Connectivity Diagram

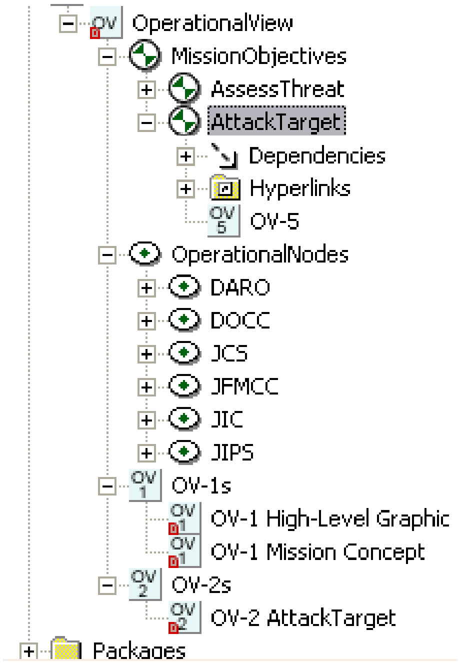
The **Operational Node Connectivity** diagram shows the operational nodes and the needlines (“communication pipes”) between them. A diagram captures the OV-2 information for the AttackTarget mission objective. You can create an OV-2 for the **AssessThreat** mission objective on your own as an exercise. Several steps in creating this architecture product are automated by helpers. In this section, you use a helper to create an initial OV-2 that includes the mission objectives (**AssessThreat Mission Objective**, **AttackTarget Mission Objective**).

Later, after creating the **OV-6c Event Trace Description** diagram, needline information is added to the OV-2 diagram.

To create the initial OV-2 diagram, follow these steps:

1. Open the **OV-1 Mission Concept** diagram from the Rational Rhapsody browser, or by selecting the OV-1 diagram tab from the bottom of the diagram pane if it is already open.
2. Right-click the **AttackTarget** Mission Objective and select **Create OV-2 from Mission Objective**. If you do not see **Create OV-2 from Mission Objective** menu item, see **Manually Adding the DoDAF Helpers** in the Troubleshooting section.
3. Click **OK** to dismiss the confirmation message.

In the browser, expand **MissionObjectives**, **OperationalNodes**, **OV-1s**, and **OV-2s**, as shown in the following figure.

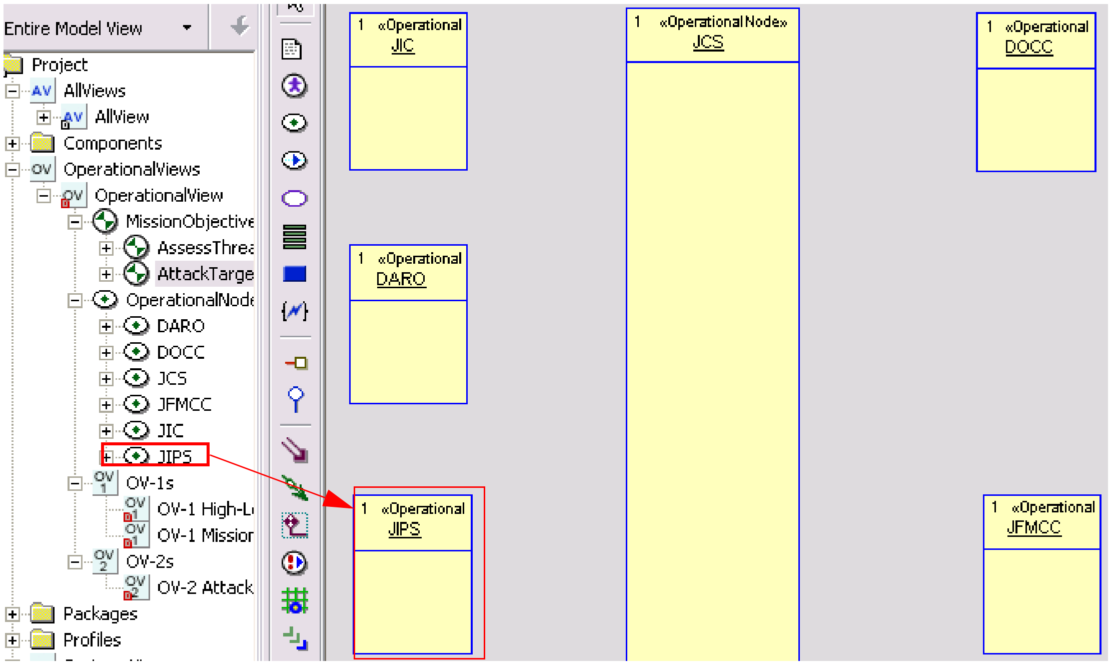


- OperationalView
  - MissionObjectives
    - AssessThreat
    - AttackTarget
      - Dependencies
      - Hyperlinks
      - OV-5
  - OperationalNodes
    - DARO
    - DOCC
    - JCS
    - JFMCC
    - JIC
    - JIPS
  - OV-1s
    - OV-1 High-Level Graphic
    - OV-1 Mission Concept
  - OV-2s
    - OV-2 AttackTarget
- Packages

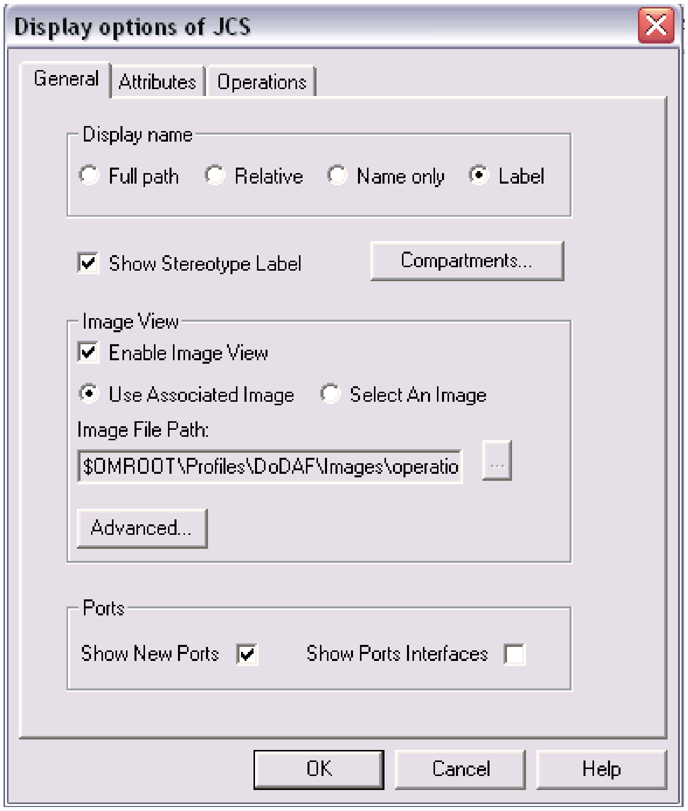
To complete this initial OV-2 diagram, follow these steps:

1. If the **OV-2 AttackTarget** diagram is not already open, open it by double-clicking on its icon in the Rational Rhapsody browser.

2. Drag and drop each of the blocks: **DOCC**, **JCS**, and **JFMCC** from the browser onto the diagram. The following figure shows the initial OV-2 diagram:



Note: To change the appearance of an operational node, select one or more nodes and right-click one. Select **Display Options** and clear the **Enable Image View** check box or select the **Select An Image** radio button and click the Ellipsis button ... to browse to the image, as shown in the following figure:


Display options of JCS

Display name
○ Full path ○ Relative ○ Name only ● Label

☑ Show Stereotype Label

Image View
☑ Enable Image View
● Use Associated Image ○ Select An Image
Image File Path:
$OMROOT\Profiles\DoDAF\Images\operatio

Ports
Show New Ports ☑ Show Ports Interfaces □

Note: Needlines connecting the operational nodes will be added later in the tutorial after the **OV-6c Operational Event Trace Description** diagram has been created.

3. Save the architecture model using **File** > **Save**.

Creating an OV-6c Operational Event Trace Description

The **Operational Event Trace Description** diagram is a vertical time-ordered diagram of the information exchanges between operational nodes during a particular scenario. Use a sequence diagram to capture the OV-6c information. Use the OV-5 diagram as a guide to create the OV-6c diagram. Each Event Trace Description focuses on an individual operational scenario.

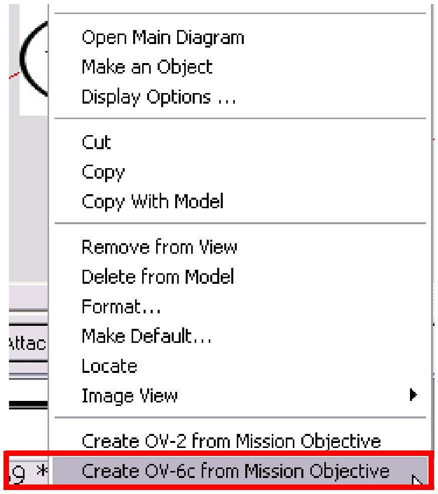
Several OV-6c diagrams are used to describe the various operational scenarios, including nominal operations as well as failures and exceptions. In this exercise, one scenario is added to your architecture model. This scenario shows the event trace for a two-phase attack and allows you to explore the complete functional flow shown in the OV-5 diagram. You can create an additional Event Trace Description on your own as practice.

Here, too, you can use a helper to assist you in creating the initial OV-6c diagram.

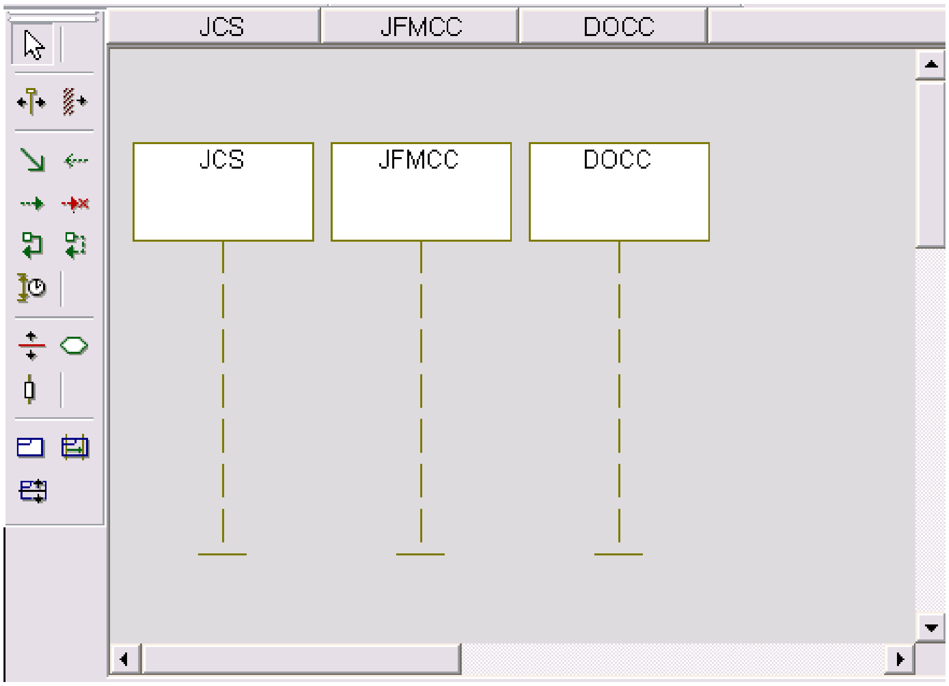
To create the initial OV-6c diagram, follow these steps:

1. Right-click the **AttackTarget** Mission Objective from either the Rational Rhapsody browser or in the **OV-1 Mission Concept** diagram.

2. Choose **Create OV-6c from Mission Objective**. If you do not see this menu item, see Manually Adding the DoDAF Helpers in the Troubleshooting section of this tutorial.

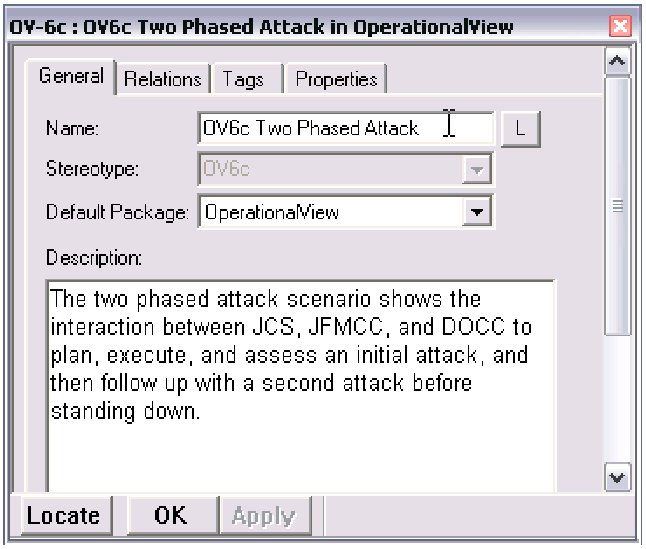


3. The new OV-6c diagram will be displayed, as shown in the following figure. Click **OK** to dismiss the confirmation message.



4. Open the Diagram Properties dialog box to rename the diagram, enter a description for the scenario, and change the display settings for the nodes. Right-click a blank part of the sequence diagram and select **Diagram Properties**.

5. On the **General** tab, change the diagram name to `OV6c Two Phased Attack` in the **Name** field, as shown in the following figure:



OV-6c : OV6c Two Phased Attack in OperationalView

Name: OV6c Two Phased Attack

Stereotype: OV6c

Default Package: OperationalView

Description:

The two phased attack scenario shows the interaction between JCS, JFMCC, and DOCC to plan, execute, and assess an initial attack, and then follow up with a second attack before standing down.

6. Add the following description in the **Description** field: `The two phased attack scenario shows the interaction between JCS, JFMCC, and DOCC to plan, execute, and assess an initial attack, and then follow up with a second attack before standing down.`

7. Click **OK**.

Before you begin adding the scenario messages to the diagram, refer to the event trace naming conventions for the messages that go between operational nodes in the following table.

Message Type	Message Prefix
Request Operational Activity	req
Notification and Reporting	ev

Review the OV-5 diagram and focus on the first attack sequence. You can identify six operational activities for the attack sequence:

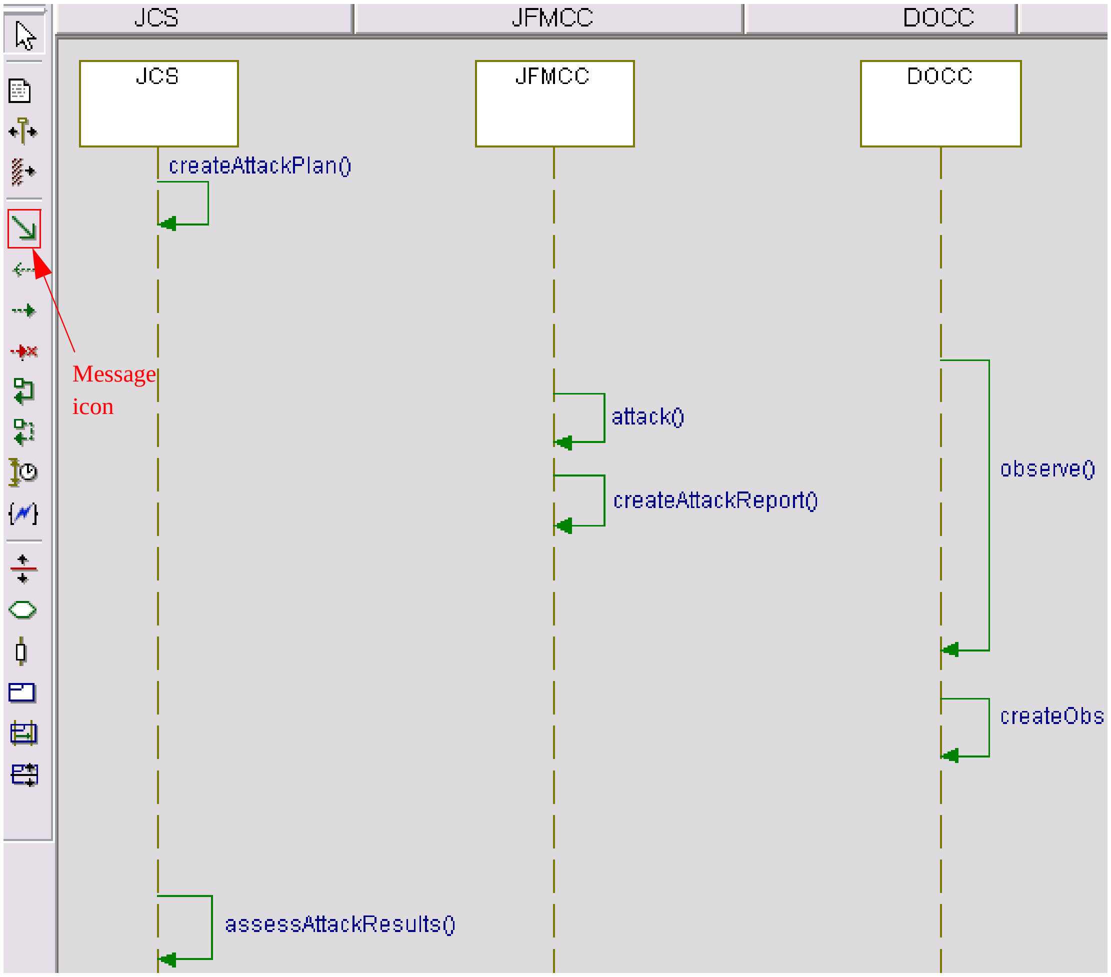
- **createAttackPlan**
- **attack**
- **createAttackReport**
- **observe**
- **createObserveReport**
- **assessAttackResults**

Keep in mind the operational event trace diagram is time ordered from top to bottom, and which node is responsible for what part of the mission objective is shown by attaching the appropriate operational activity to the appropriate node.

To add these operational activity messages to the Event Trace, follow these steps:

1. Space out the **JCS**, **JFMCC**, and **DOCC** nodes by dragging each of them horizontally across from left to right.

2. Use the **Message** icon ↘ on the Drawing toolbar, draw the **createAttackPlan** and **assessAttackResults** messages on the **JCS** node. Start drawing on the instance line and form a “box” that ends on the same instance line, as shown in the following figure. The message names can be edited as the messages are drawn on the diagram.



3. Draw the **attack** and **createAttackReport** messages to the **JFMCC** node, as shown in the figure above.

4. Draw the **observe** and **createObserveReport** messages to the **DOCC** node.

Note: To change a message name after the message has been drawn, double-click the message name.

You have now shown the six mission objectives for the attack sequence, identified the nodes that perform these activities, and provided the proper time ordering of these activities for this scenario. Next, you need to identify and draw the messages between the nodes where requests for each mission objective are made.

Also, JCS requests that DOCC carry out the observe mission objective by sending it the **reqObserve** command. By examining the sequence diagram, one can see that JCS requests JFMCC to carry out the attack mission objective by sending it the **reqAttack** message.

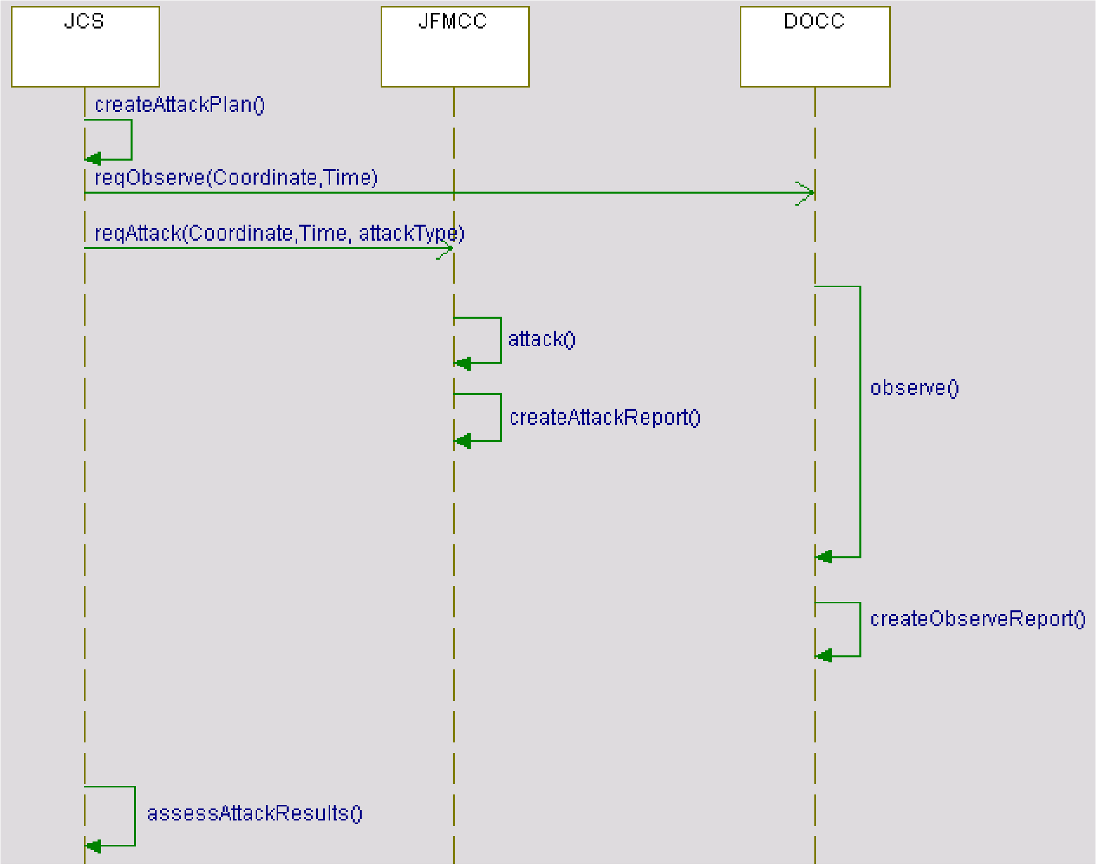
To add these request mission objective messages to the diagram, follow these steps:

1. Draw the **reqObserve()** message from JCS to DOCC. The **reqObserve** message contains data in the form of two parameters: **Coordinate** and **Time**. These parameters are used later in the tutorial to create the OV-7.

   Note: You do not have to type the "()" parentheses when naming a message without parameters. Rational Rhapsody does this automatically. Messages that contain parameters require you to type the parentheses.

2. Draw the **reqAttack** message from JCS to JFMCC. In the parenthesis, add the parameters `Coordinate`, `Time`, and `attackType` to indicate the data being sent to JFMCC, as shown in the following figure:

   Note: To move a group of messages up or down in the sequence diagram, first select the group of messages by clicking on them while pressing the **Ctrl** key. Then move the cursor over one of the selected messages, and while holding the **Shift** key, click and drag the selection.

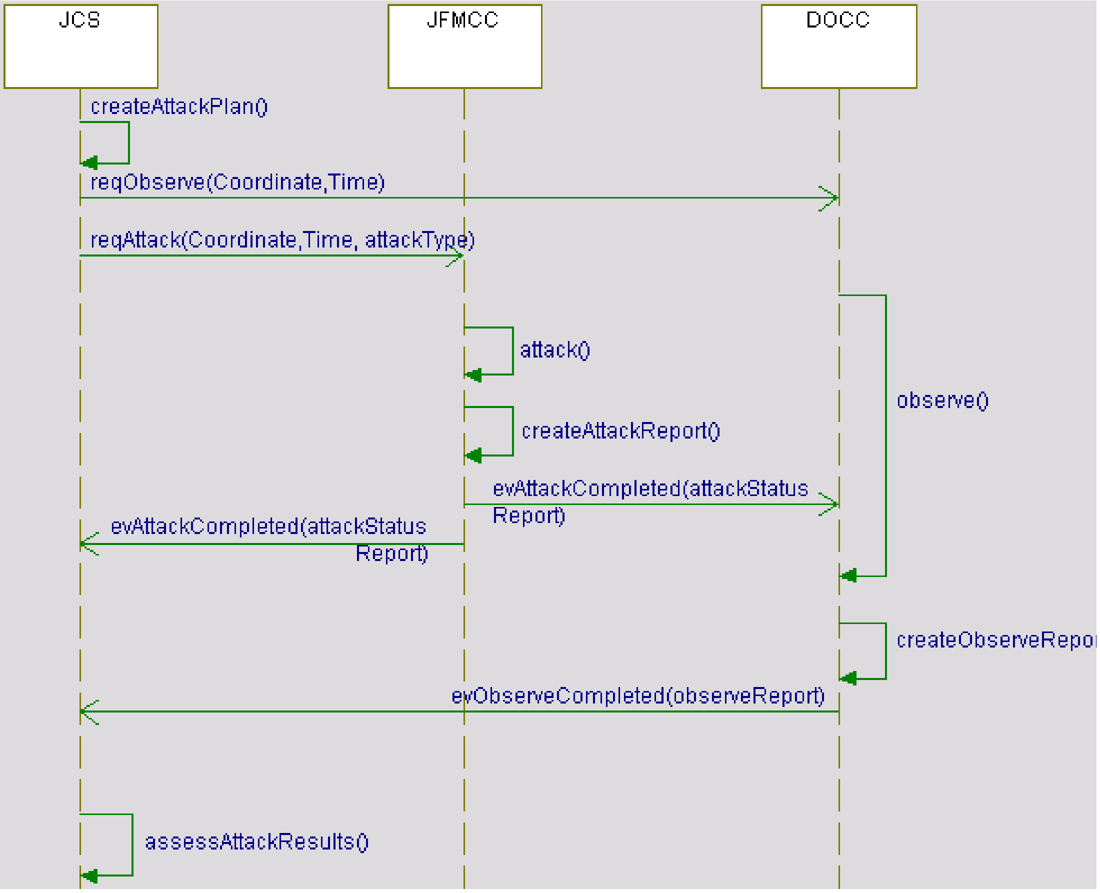


You have extended the sequence diagram to show how and when JCS requests JFMCC and DOCC carry out their mission objectives for the attack sequence. The last set of messages that need to be added are the notification messages, typically indicating the mission objective has been completed. Examination of the sequence diagram shows that JFMCC sends the **evAttackCompleted** message indicating when the attack is complete. JFMCC must report it has completed the attack to both JCS and DOCC. DOCC sends the **evObserveCompleted** message to JCS, indicating it has completed its observation.

To add these notification messages to the diagram, follow these steps:

1. Use the **Message** icon on the **Drawing** toolbar to draw the **evAttackCompleted** message from JFMCC to JCS and also from JFMCC to DOCC. In the parenthesis, add the parameter `attackStatusReport` to indicate a status report is being sent to both operational nodes.

2. Use the **Message** icon on the **Drawing** toolbar to draw the **evObserveCompleted** message from DOCC to JCS, as shown in the following figure. In the parenthesis, add the parameter `observeReport` to indicate a status report is being sent to JCS.

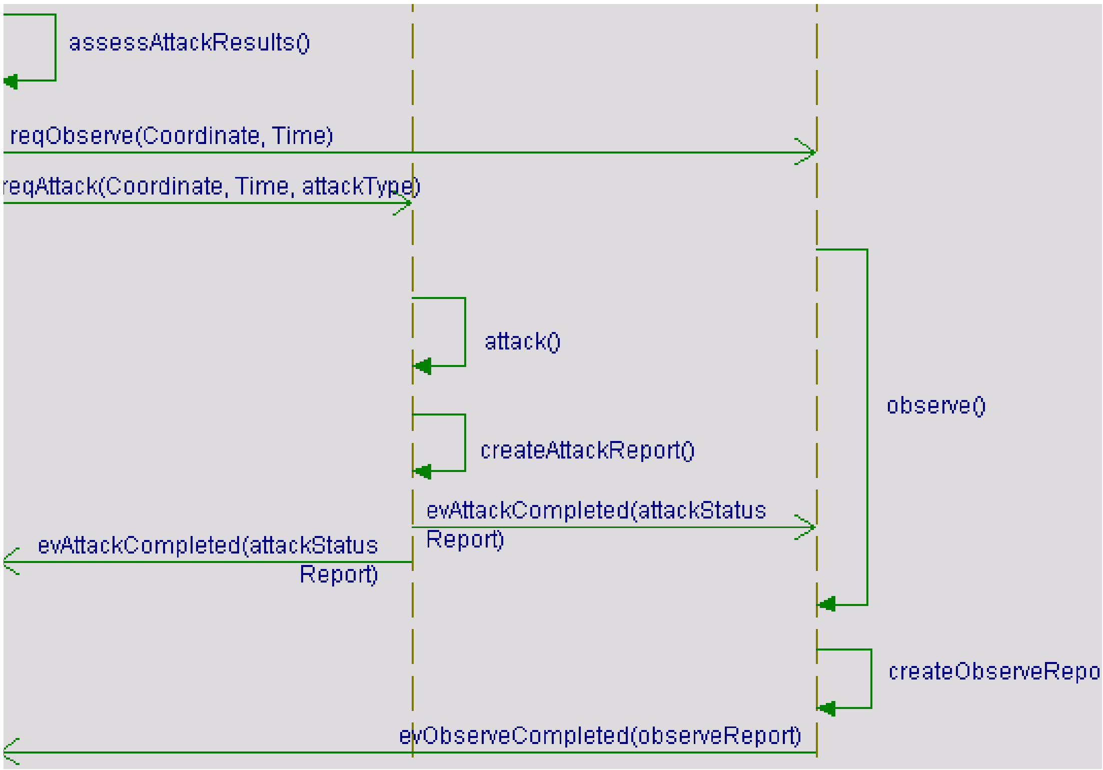

JCS
JFMCC
DOCC
createAttackPlan()
reqObserve(Coordinate,Time)
reqAttack(Coordinate,Time, attackType)
attack()
observe()
createAttackReport()
evAttackCompleted(attackStatus Report)
evAttackCompleted(attackStatus Report)
createObserveRepo
evObserveCompleted(observeReport)
assessAttackResults()

With these messages added, you have now completed the interaction between JCS and JFMCC and DOCC in carrying out the first phase of the attack sequence. You can now focus on the second phase of the attack sequence. The second phase of the attack is identical to the first, except that JCS does not create a new attack plan.

To add the messages for the second phase of the attack, follow these steps:

1. Using the vertical scroll bar, scroll down in the OV-6c sequence diagram.

2. Create the second phase of the attack sequence by repeating (redrawing) the first phase attack sequence, but omit the **createAttackPlan** mission objective, as shown in the following figure:

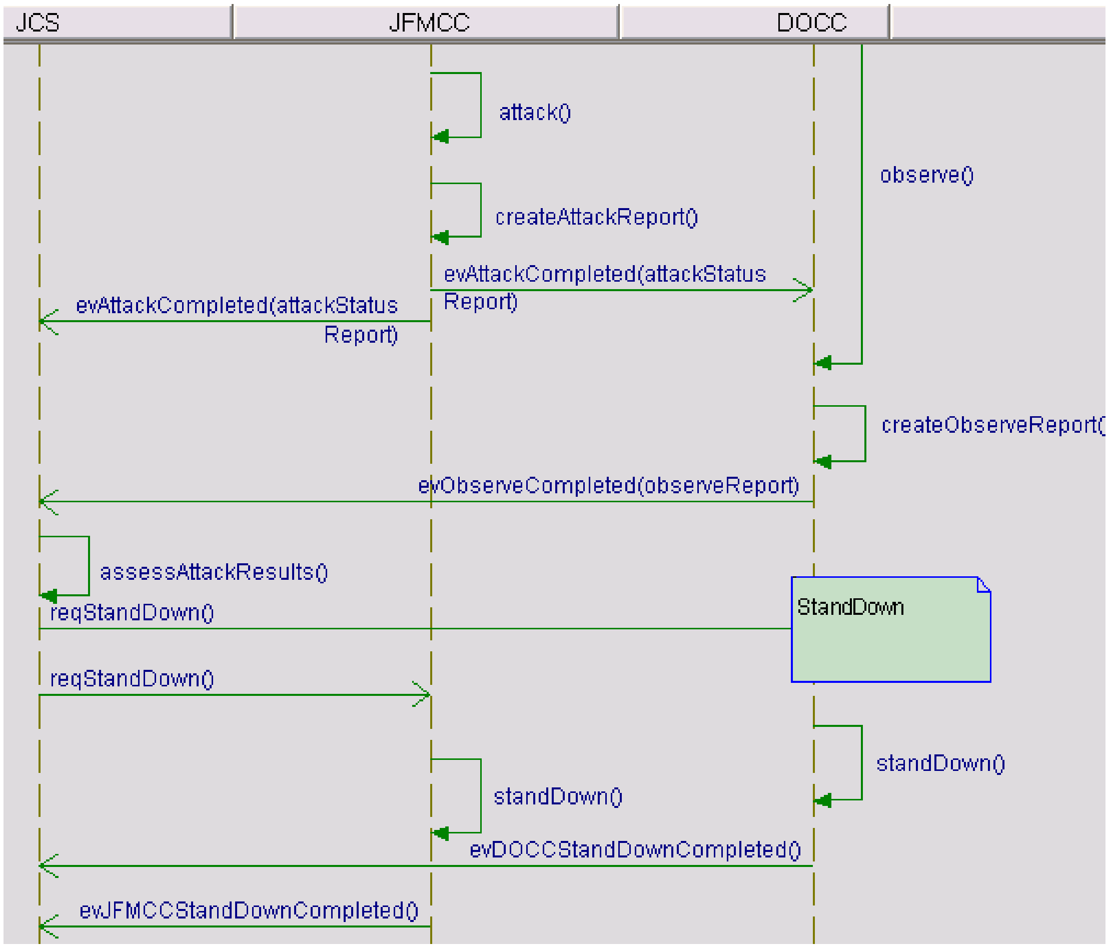

assessAttackResults()
reqObserve(Coordinate, Time)
reqAttack(Coordinate, Time, attackType)
attack()
observe()
createAttackReport()
evAttackCompleted(attackStatus Report)
evAttackCompleted(attackStatus Report)
createObserveRepo
evObserveCompleted(observeReport)

Sequences like this attack sequence that repeat can be captured by using what is called a referenced sequence diagram, which allows a sequence to be drawn once and then reused multiple times in other sequence diagrams or the same sequence diagram.

To complete the scenario, you need to add some additional messages after the second phase of the attack to address the stand down sequence. For stand down, JCS sends the **reqStandDown** message to JFMCC and DOCC, then both JFMCC and DOCC execute their **StandDown** mission objectives, and finally send a notification message back to JCS.

To complete the scenario for this OV-6c, follow these steps:

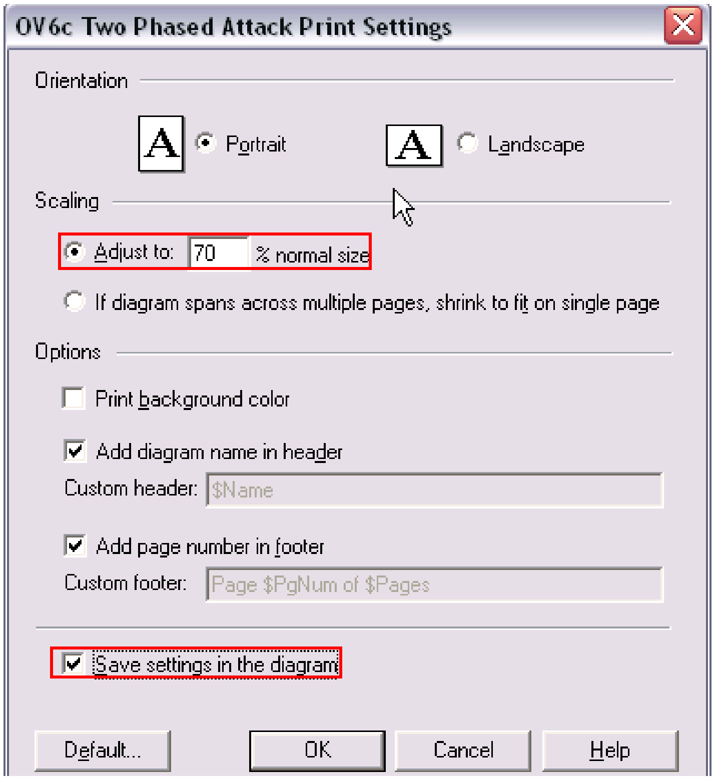
1. Using the vertical scroll bar, scroll down in the OV-6c sequence diagram.
2. Add the remaining sequence as shown in the following figure.
3. You can add the StandDown note shown in the diagram using the **Note** icon 📄 on the **Common Drawing** toolbar, as shown in the following figure:


JCS
JFMCC
DOCC
attack()
observe()
createAttackReport()
evAttackCompleted(attackStatus Report)
evAttackCompleted(attackStatus Report)
createObserveReport(
evObserveCompleted(observeReport)
assessAttackResults()
reqStandDown()
StandDown
reqStandDown()
standDown()
standDown()
evDOCCStandDownCompleted()
evJFMCCStandDownCompleted()

The last step in creating the OV-6c is to change some of its printing settings. Since this diagram is rather tall and narrow, you need to change the print settings to make it look better for when it appears in a generated DoDAF report.

To change the print settings, follow these steps:

1. Right-click an empty area of the diagram and choose **Printing** > **Print Settings**. The Print Settings dialog box opens, as shown in the following figure:



OV6c Two Phased Attack Print Settings

Orientation

● Portrait ○ Landscape

Scaling

● Adjust to: 70 % normal size

○ If diagram spans across multiple pages, shrink to fit on single page

Options

□ Print background color

☑ Add diagram name in header

Custom header: $Name

☑ Add page number in footer

Custom footer: Page $PgNum of $Pages

☑ Save settings in the diagram

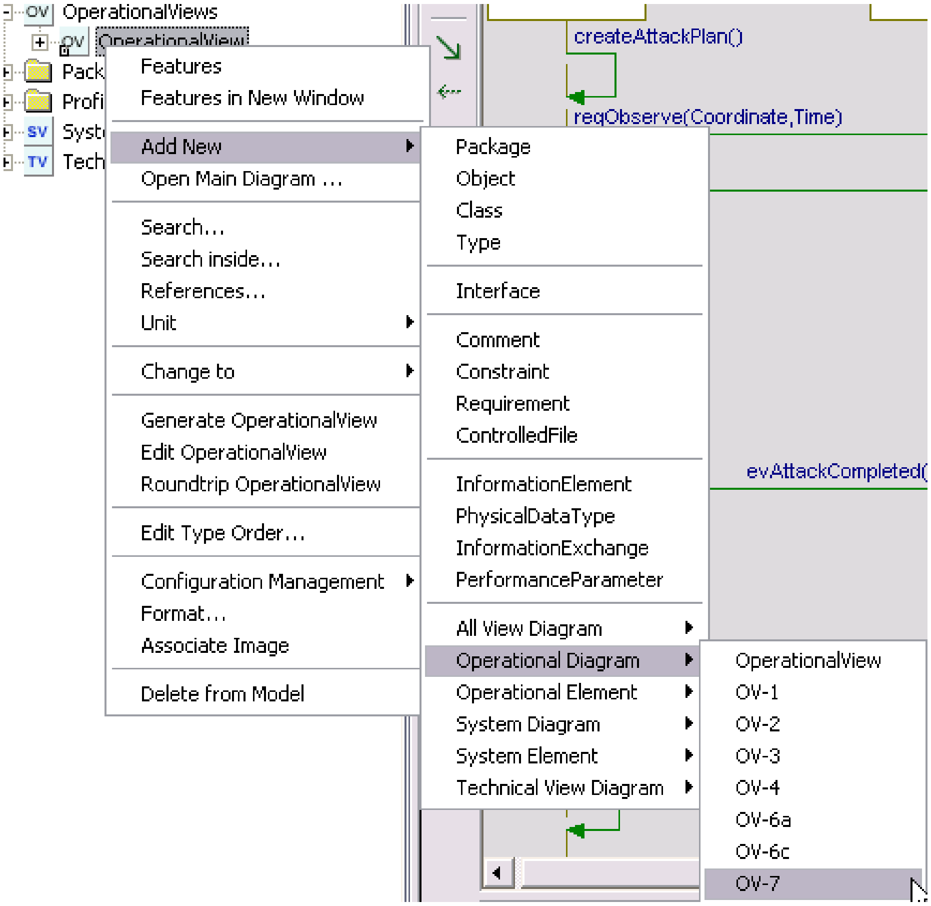
2. Set the Scaling to 70% of normal size.
3. Select **Save settings in the diagram**.
4. Click **OK** to close the Print Settings dialog box.
5. Move your cursor to an empty area on the diagram, right-click, and choose **Printing** > **Print Preview**. The Print Settings dialog box opens.
6. If necessary, repeat these steps until you are satisfied with the scaling of the diagram.
7. Be sure to save the architecture model using **File** > **Save**.

Creating an OV-7 Logical Data Model

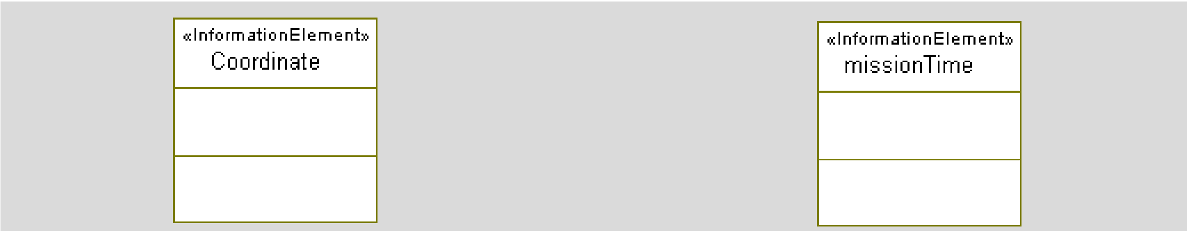
OV-7 Logical Data Model is a class diagram that shows the relations among Informational Elements (data classes).

To create the OV-7 informational elements `Coordinate` and `missionTime`, follow these steps:

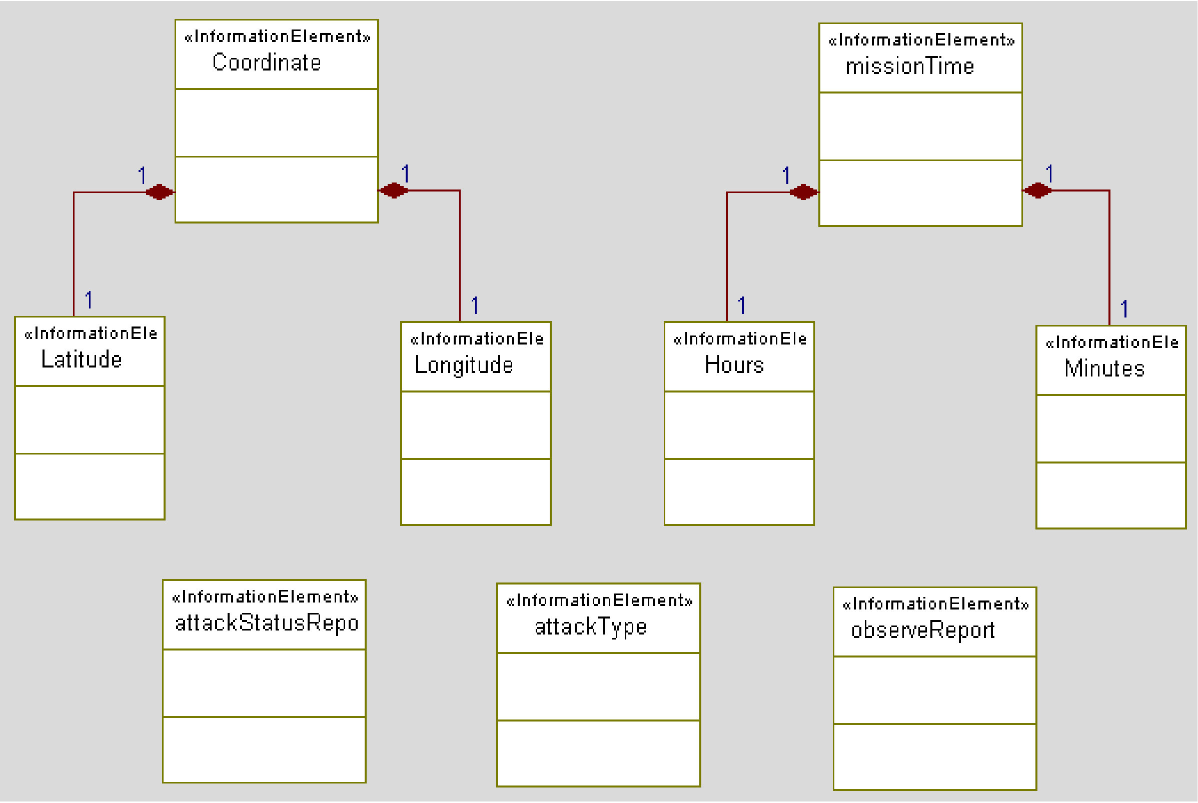
1. Right-click **OperationalView** in the browser and select Add **New** > **Operational Diagram** > **OV-7**, as shown in the following figure:



2. Now open the OV-7 diagram and use the **Informational (Data) Element** icon on the **Drawing** toolbar to draw the informational elements `Coordinate` and `missionTime`, as shown in the following figure:



3. Use the **Informational (Data) Element** icon to draw the informational elements `Latitude`, `Longitude`, `Hours` and `Minutes` and use the **Composition Line** icon to draw connecting lines between the informational elements. Next, draw the informational elements `attackStatus`, `attackType`, and `observeReport`, as shown in the following figure:



Adding Needline Information and Mission Objectives to the OV-2

The OV-2 created earlier did not include needlines. The reason for this is that the information needed to create the needlines is contained in the **Operational Event Trace Description** diagram (OV-6c) which also includes the information necessary to assign mission objectives to each of the operational nodes in the OV-2 diagram.

In the following steps, one of the DoDAF helpers is used to assist you in adding the needlines so that you do not have to enter the information manually. Before you can update the OV-2 diagram, you must change the type (**PrimitiveOperation**, **TriggeredOperation**, **Event**) of some of the messages in the OV-6c diagram.

Messages are represented by arrows between the operational nodes in the OV-6c diagram. Different messages show different things. **PrimitiveOperation** is the default type of message and shows a mission objective being executed, such as attacking a target. **PrimitiveOperation** messages occur on one operational node only and are not sent to others. All the messages drawn in the OV-6c diagram were **PrimitiveOperation** messages and some must be changed.

Communication between operational nodes are called ‘Event’ messages. All Messages sent between operational nodes must be changed from type **PrimitiveOperation** to type **Event**. Afterwards the diagram must ‘realize’ all of the message type changes.

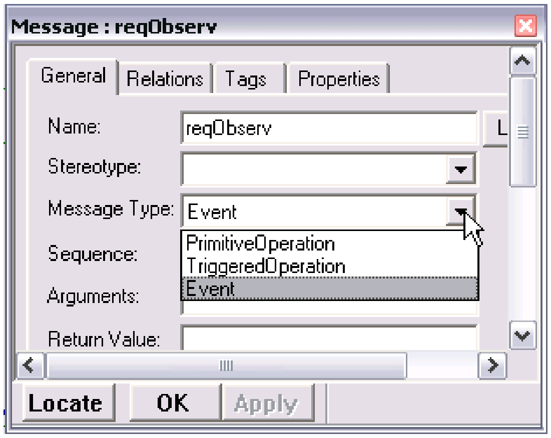
By realizing these messages, you're indicating that you want the messages to be formally recognized as elements in the architecture model. Without realizing the messages, they are treated as informal notes rather than actual elements of the architecture model.

To change the appropriate message types in the OV-6c diagram, follow these steps:

1. Open the **OV-6c Two-Phased Attack** sequence diagram from the browser if it is not already open.

2. Right-click the **reqObserve** message and select **Features**. The Features dialog box opens.

3. Change the Message Type from **Primitive Operation** to **Event**, as shown in the following figure:



Message : reqObserv

Name: reqObserv

Stereotype:

Message Type: Event

Sequence:

Arguments:

Return Value:

4. Repeat the above steps for ALL occurrences of the following messages:

   a. **reqObserve**

   b. **reqAttack**

   c. **evAttackCompleted**

   d. **evObserveCompleted**

   e. **reqStandDown**

   f. **evDOCCStandDownCompleted**

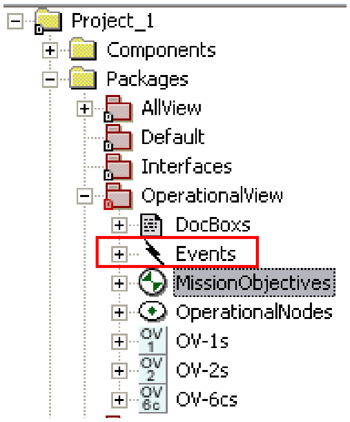
   g. **evJFMCCStandDownCompleted**

Note

Before proceeding, review the OV-6c diagram and make sure all the messages going from one operational node to another have a hollow arrowhead, and the mission objective messages (starting and ending at the same node) have a solid arrowhead.

To realize the messages in the OV-6c diagram, follow these steps:

1. Select all the elements in the diagram using **Edit > Select > Select All**.
2. Realize the messages using **Edit > Auto Realize**. After realizing, you will see a folder in the OperationalView named **Events**, as shown in the following figure:



- Project_1
  - Components
  - Packages
    - AllView
    - Default
    - Interfaces
    - OperationalView
      - DocBoxs
      - Events
      - MissionObjectives
      - OperationalNodes
      - OV-1s
      - OV-2s
      - OV-6cs

3. Save the architecture model using **File > Save**.

Note

If you create additional OV-6c diagrams, Rational Rhapsody provides a shortcut for typing in the names of messages you have previously realized. After drawing the arrow and leaving the default message name (message_0), right-click the message name and choose **Select Message** to see a list of previously realized message names.

Now that the messages have been realized, you can use the helper to update the OV-2 diagram with the needline and mission objective information in the OV-6c.

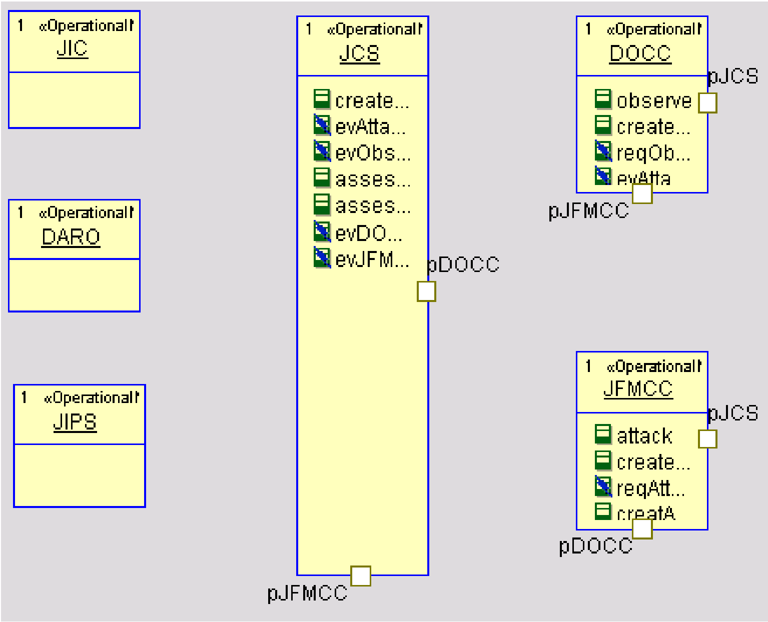
To update the OV-2, follow these steps:

1. Select the **OV-6c Two Phased Attack** sequence diagram from the Rational Rhapsody browser.
2. With the mouse over the OV-6c icon, right-click and choose **Update OV-2 from OV-6c**. If you do not see this menu item, see **Manually Adding the DoDAF Helpers** in the Troubleshooting section of this tutorial.

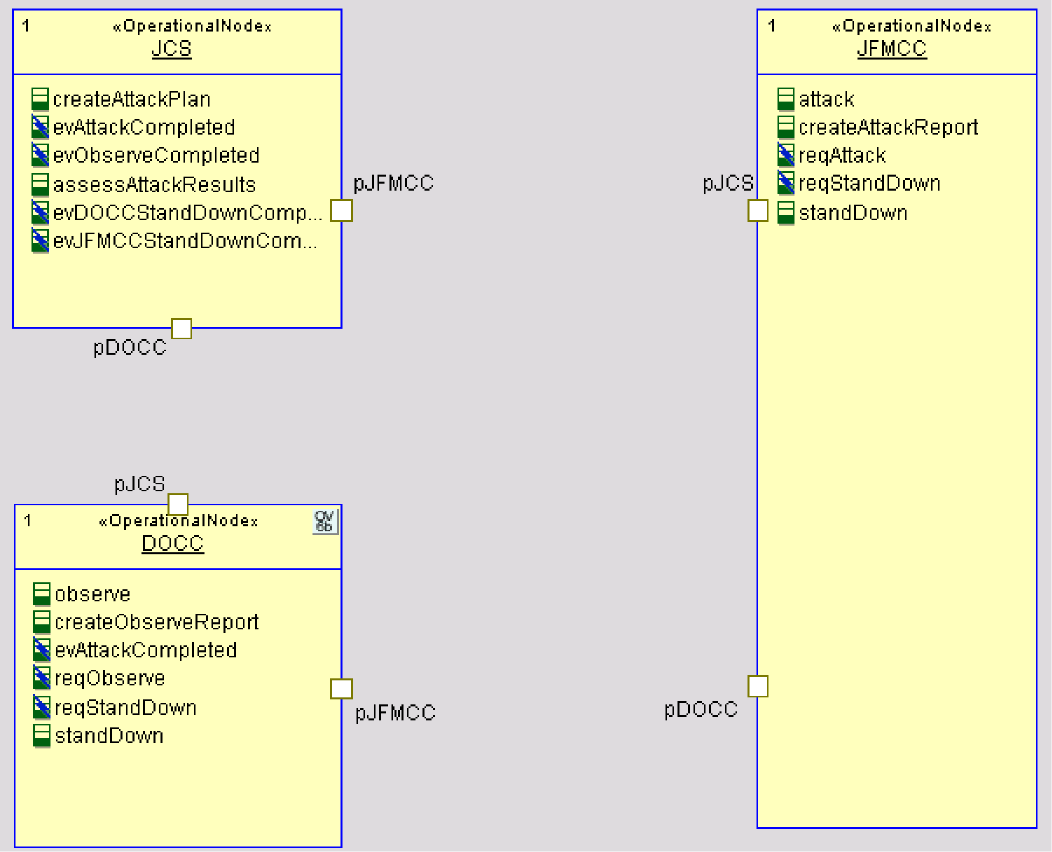
After a few moments, you will be prompted with a message dialog box indicating the OV-2 diagram was updated. You can now look at the OV-2 diagram and see what has been added to it.

To see what has changed, follow these steps:

1. Open the **OV-2 AttackTarget** diagram by double-clicking on its icon in the Rational Rhapsody browser, or select it from among the diagram tabs at the bottom of the screen if it is already open.



2. Re-arrange and stretch the nodes so that the diagram resembles the following figure:



3. To move a node, simply click it and drag it (while holding down the mouse button).

4. To stretch a node, select it by clicking on it and then move the cursor over any of the eight control points and drag the point while holding down the mouse button.

5. To move a port, which are the small boxes on the edges of the nodes, simply click it and drag it as you would a node.

As you look inside the OV-2 nodes, you will see the mission objectives have been added to the appropriate node, and are marked with the operation symbol ▤. Other message labels you added to the OV-6c are visible, and these are related to the needline and information exchange between nodes.

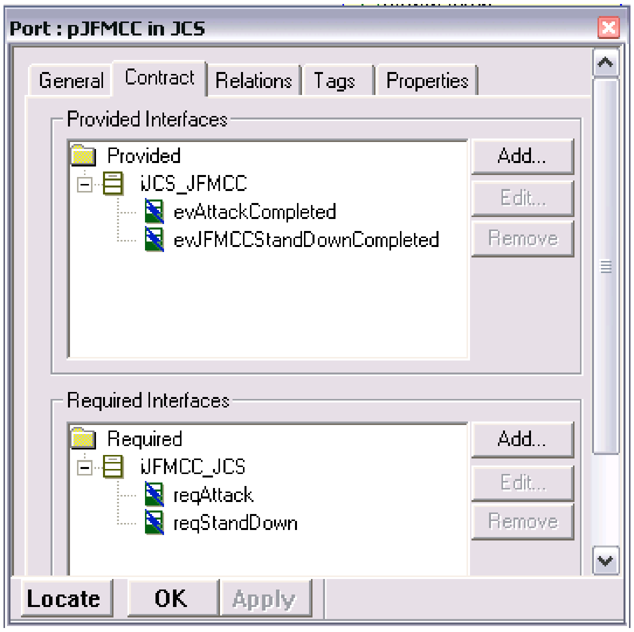
Note

If any of the information exchange messages (**reqObserv**, **reqStandDown**) appear as mission objectives (**observe**, **standDown**) in your model, see **Fixing the Model if Messages Appear as Mission Objectives** in the Troubleshooting section of this tutorial.

Focus on the **JCS** node. Inside the **JCS** node, you see the **createAttackPlan** and **assessAttack** mission objectives. Along with the mission objectives, you see four notification messages: **evAttackCompleted**, **evObserveCompleted**, **evDOCCStandDownCompleted**, and **evJFMCCStandDownCompleted**. While the mission objectives are performed at the **JCS** node, the notification messages imply an information exchange with the DOCC or **JFMCC** node.

Notice the two small boxes on the edge of **JCS**. The boxes are the **JCS** ports; one is labeled **pJFMCC** and the other **pDOCC**. The **pJFMCC** port was created to contain the information exchange between **JCS** and **JFMCC**. The naming convention for ports is `p<OperationalNodeClassName>`.

To see what information flows through the **pJFMCC** port, double-click the port and look at the **Contract** tab in the Features dialog box, as shown in the following figure. The contract describes the type of information exchange.



Provided Interfaces

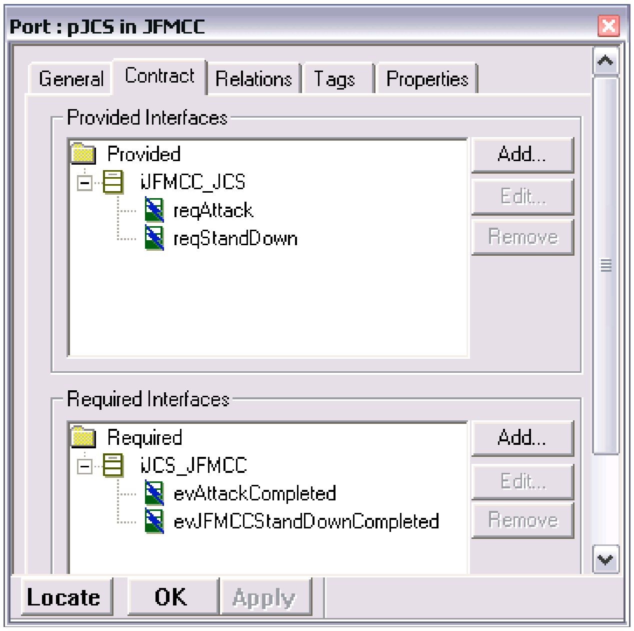
- Provided
  - iJCS_JFMCC
    - evAttackCompleted
    - evJFMCCStandDownCompleted

Required Interfaces

- Required
  - iJFMCC_JCS
    - reqAttack
    - reqStandDown

JCS has one provided interface with JFMCC named **iJCS_JFMCC**. The naming convention for interfaces is `i<ReceiverClassName>_<SenderClassName>`. The **iJCS_JFMCC** interface contains the two messages **evAttackCompleted** and **evJFMCCStandDownCompleted**. By providing this interface, JCS provides the ability to receive these two messages sent from JFMCC. JCS also has a required interface with JFMCC named **iJFMCC_JCS**. This interface contains the two messages **reqAttack** and **reqStandDown**. By requiring this interface, JCS requires JFMCC to be able to receive these two messages. These contracts are consistent with the manner in which you created the OV-6c diagram.

Examine the **pJCS** port on the **JFMCC** node. Notice that the provided and required interfaces on this port are the reverse of those on the **pJFMCC** port. The figure **pJCS** Port Contracts shows the port contracts for the **pJCS** port. The complementary nature of the contracts of these two ports shows the interfaces are consistent on both ends. Examine the ports on the diagram to ensure they are also complementary.



Port : pJCS in JFMCC

Provided Interfaces
- Provided
  - iJFMCC_JCS
    - reqAttack
    - reqStandDown

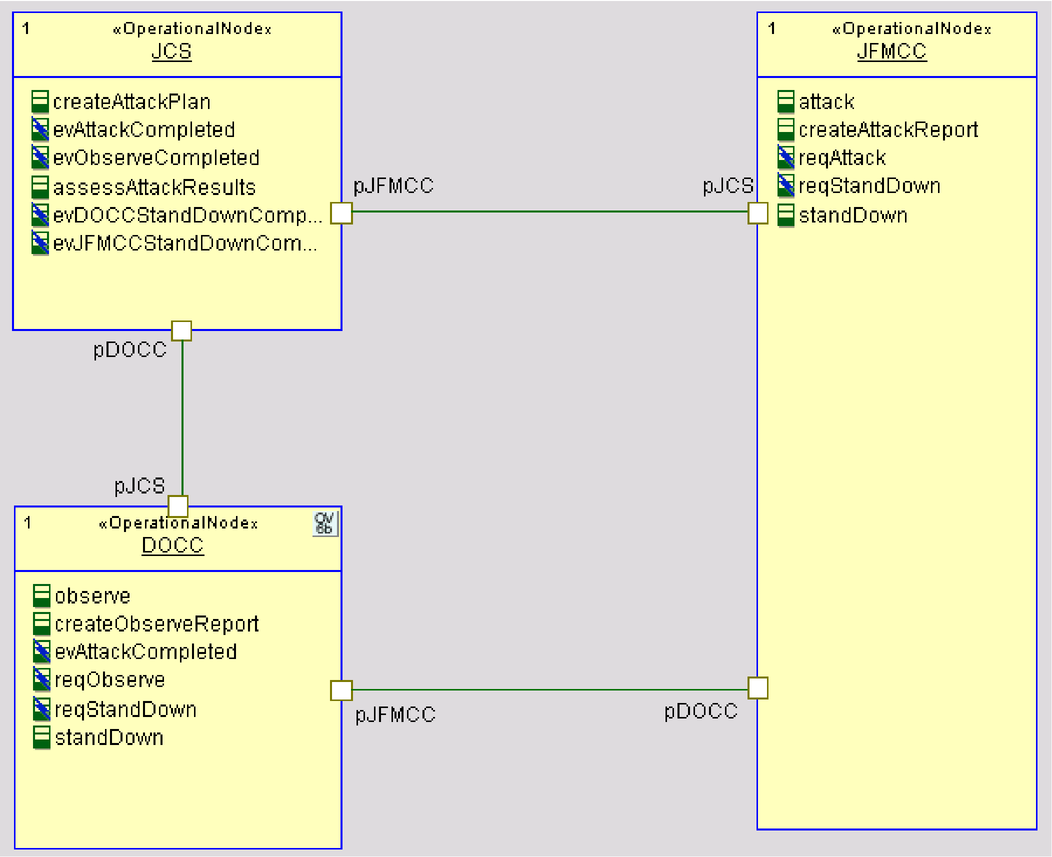
Required Interfaces
- Required
  - iJCS_JFMCC
    - evAttackCompleted
    - evJFMCCStandDownCompleted

Connecting the Ports

Now that you have examined the changes to the OV-2, and seen how the mission objective information and needline information have been added, draw the links between the nodes and connect the ports.

To draw the links, follow these steps:

1. Select the **Needline** icon on the **Drawing** toolbar.
2. Connect the ports as shown in the figure by clicking on one port, dragging the line to the connecting port, and releasing the mouse button, as shown on the following figure:



3. Save the architecture model using **File** > **Save** in the Rational Rhapsody main window.

Note

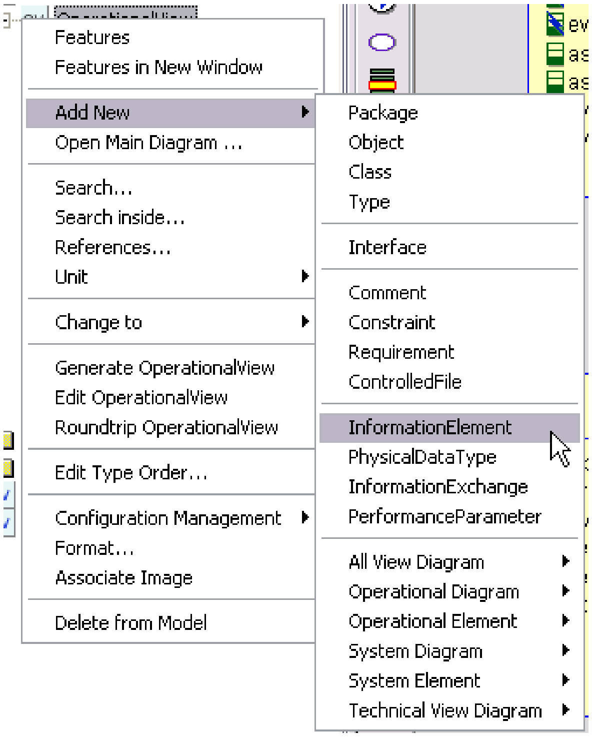
You must repeat the steps in this section of the tutorial for any other OV-6c diagrams you create.

Adding Information Exchanges

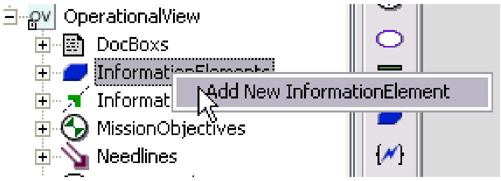
Now it is time to add the information exchanges into the OV-2 diagram that represent the **missionTime** and **Coordinates** parameters.

To create the information elements you plan to use, follow these steps:

1. For the first information element, you must right-click **OperationalView** in the Rational Rhapsody browser and select **Add New > InformationElement**, as shown in the following figure:



2. For all additional information elements, right-click **InformationElements** and select **Add New InformationElement**, as shown in the following figure:



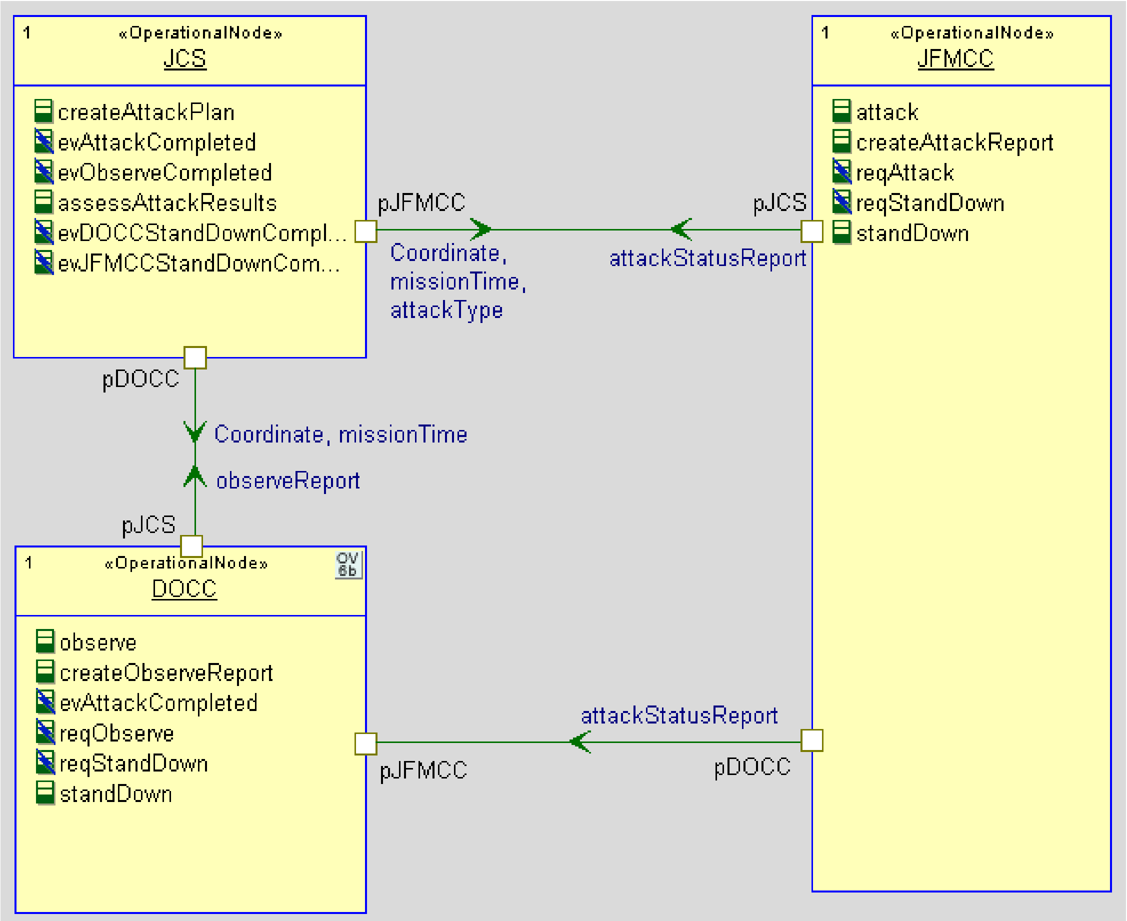
3. Create the following information elements: `attackStatus`, `attackType`, `Coordinate`, `Hours`, `Latitude`, `Longitude`, `Minutes`, `observeReport`, and `missionTime`.

4. Using the information exchange tool , move the mouse pointer over the needline in the diagram between the JCS and JFMCC Operational Nodes, until the mouse pointer assumes the appearance of a bullseye and then click the needline.

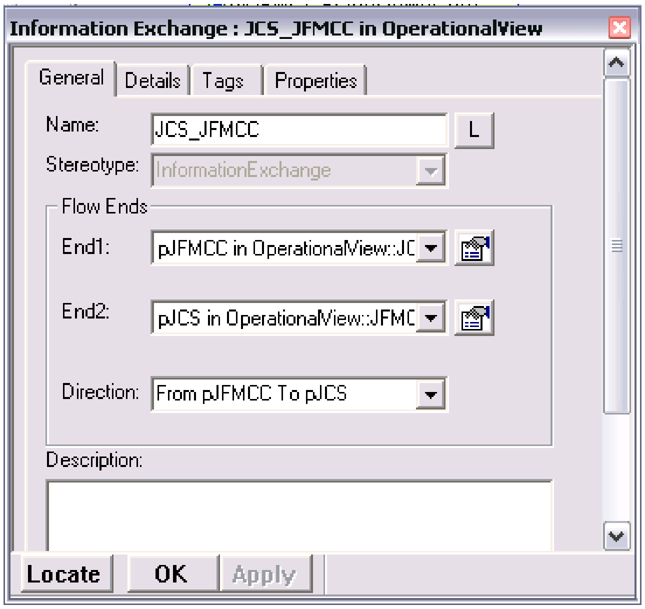
5. An arrow appears accompanied by a blinking cursor. Type the words `Coordinate, missionTime, attackType`, as shown on the following figure.

   Note: To remove the <<**InformationExchange**>> label, right-click it and select **Display Options**. Clear the <<**flow**>> keyword or **Stereotype** check box.

6. Repeat the previous two steps to create two information exchanges between the **JCS** and **DOCC** operational nodes, and another information exchange between the **JFMCC** and **JCS** nodes, as shown in the following figure:



7. To change the direction of the information exchange (shown by arrows), double-click the information exchange arrow, and in the dialog box that appears, select the appropriate direction from the **Direction** drop-down list, as shown in the following figure:


Information Exchange : JCS_JFMCC in OperationalView

Name: JCS_JFMCC

Stereotype: InformationExchange

Flow Ends

End1: pJFMCC in OperationalView::JC

End2: pJCS in OperationalView::JFMC

Direction: From pJFMCC To pJCS

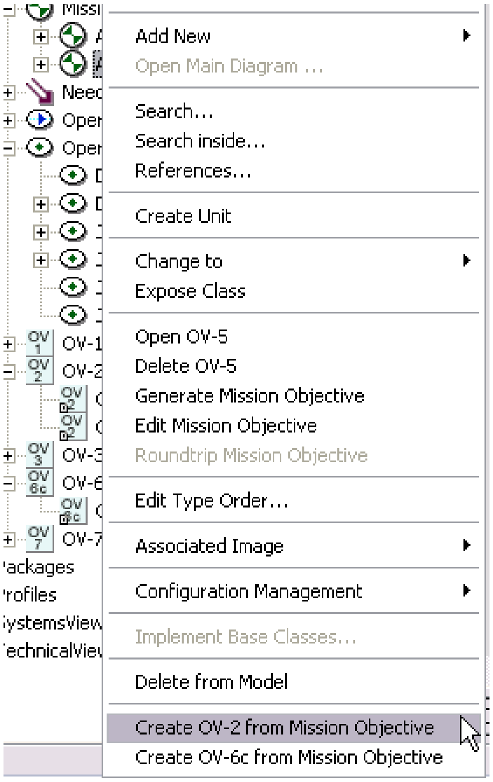
Description:

Adding Another OV-2 Diagram

At this point you will now add another OV-2 diagram. Parts of this diagram are used for developing the SV-4 Systems Functionality Description at a later point.

1. In the browser, expand **MissionObjectives**.

2. Right-click **attackTarget** and select **Create OV-2 from Mission Objective**, as shown on the following figure:



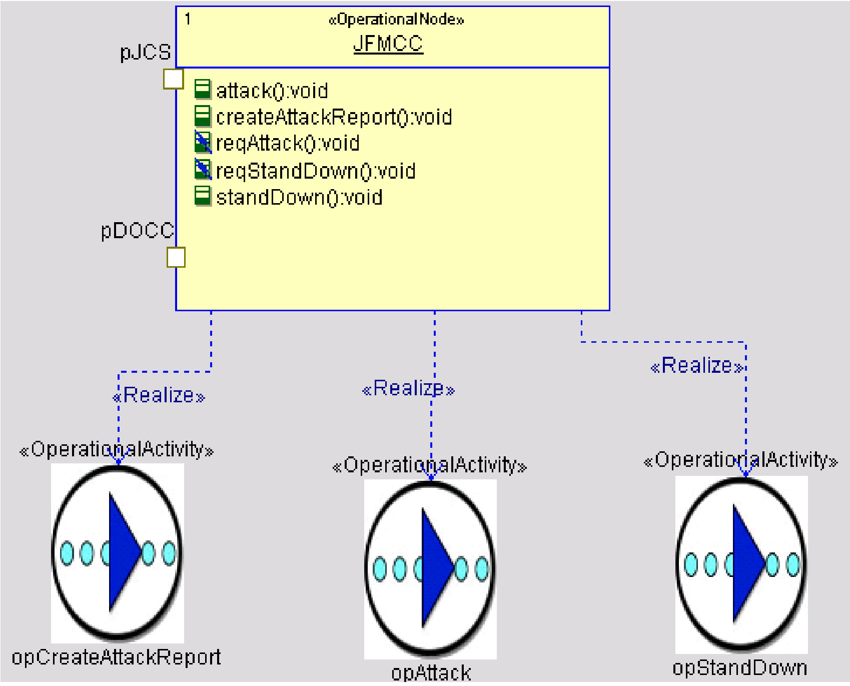
3. In the browser, expand **OperationalNodes** and drag the **JFMCC** operational node into the drawing area.

4. Customize the **JFMCC** operational node as it exists in the drawing pane to look as you want. To achieve the view, right-click the **JFMCC** node and select **Display Options**. Clear the **Enable Image View** check box and click **OK**. This view allows you to see all the operations the node contains.

5. Using the Operational Activity tool , insert three operational activities into the diagram.

6. Name the operational activities `opAttack`, `opCreateAttackReport`, and `opStandDown`, as shown in the following figure. The “op” prefix stands for “operation,” and keeps Rational Rhapsody from confusing like system function and operation names when generating code.

7. Using the **Realize** icon on the **Drawing** toolbar, draw a realize line from the **JFMCC** Operational Node to each operational activity.



Note

The above example was described for informational purposes. If you want to build a correct and working model, you must repeat the steps above for all of the other operational nodes in your model.

Creating the OV-6b Operational State Transition Description

The last diagram created in this tutorial is the Operational State Transition Description diagram (OV-6b). The Operational State Transition Description diagram uses multiple state machine diagrams to show how an operational node responds to various input events. Statecharts are used to represent the Operational State Transition Description. In this tutorial, the **DOCC** node is the focus point. You can create Operational State Transition Descriptions for other operational nodes on your own as an exercise.

Before you create the statechart for **DOCC** node, think about what the modes of operation are for the **DOCC** node. A good way to identify the modes of operation is to look at the OV-6c diagrams, paying attention to the mission objectives, which often indicate a mode of operation. Also, the messages going between nodes typically indicate when the message receiver should enter a new mode.

Note

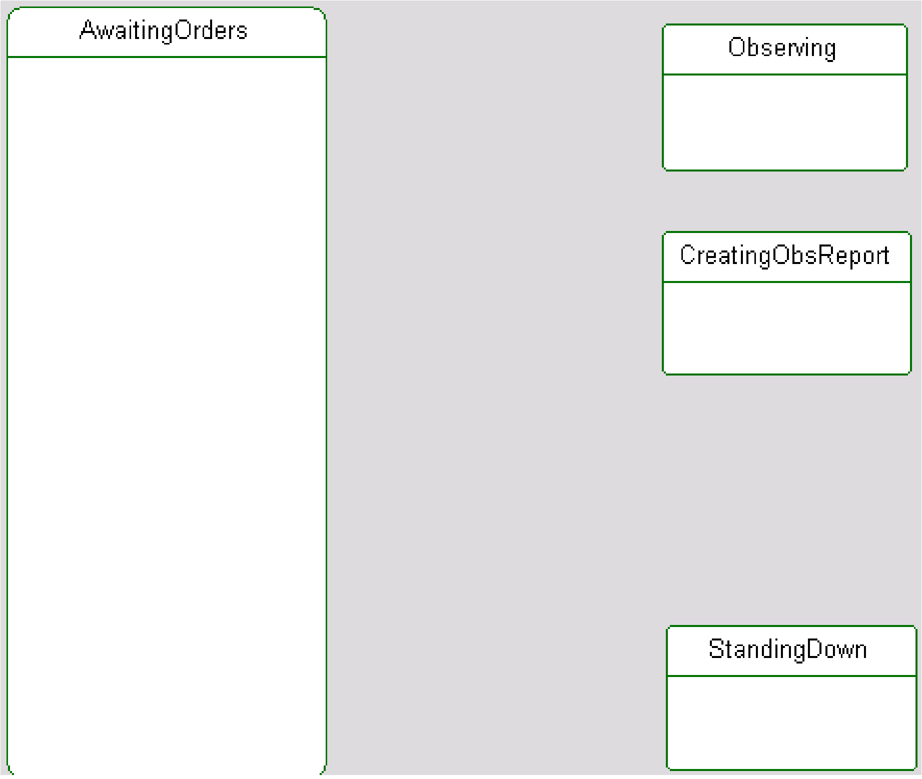
Unlike OV-6c, which reflects many nodes across one mission objective, the OV-6b reflects one node across all mission objectives.

From viewing the OV-6c diagram, it can be seen that **DOCC** might be awaiting orders, observing, preparing an observation report, or standing down. This means there are four main modes (states) in your statechart, one for each of the modes you identified.

To create the statechart and draw the modes, follow these steps:

1. Open the **OV-2 AttackTarget** diagram by double-clicking on its icon in the Rational Rhapsody browser, or by selecting it from among the diagram tabs at the bottom of the window if it is already open.
2. Click the **DOCC** node to select it.
3. Right-click the **DOCC** node and select **New OV-6b**.

4. Using the **State** icon ▢ on the **Drawing** toolbar, draw the four states: `AwaitingOrders`, `Observing`, `CreatingObsReport`, and `StandingDown`, as shown in the following figure. Type in the names of each state as you draw them.



With the states drawn, consider the logic for the transition labels. The logic that moves you from state to state can include an event trigger and a guard condition. The operations that must be performed as you change states are called actions. The logic clearly starts in the **AwaitingOrders** state.

From **AwaitingOrders**, **DOCC** moves to the **Observing** state when it receives the **reqObserve** message. To reflect this you will add the label `reqObserve` to the transition that connects the **AwaitingOrders** state to the **Observing** state.

Upon entering the **Observing** state, you will transition to the **CreatingObsReport** state when the **evAttackCompleted** message is received. So, you will add the label `evAttackCompleted` to the transition that connects the **Observing** state to the **PreparingObsReport** state.

From **PreparingObsReport**, **DOCC** can only change to the **AwaitingOrders** state. **DOCC** changes to the **AwaitingOrders** state when it has completed creating the observation report, and it will also send the **evObservationCompleted** message to **JCS**.

The following transition label will be added to the transition from the **PreparingObsReport** state to the **AwaitingOrders** state:

```
/OPORT(pJCS)->GEN(evObserveCompleted(observeReport))
```

which translates to `'generate the evObserveCompleted() message on the JCS port and send the data observeReport with the message'`. By sending the **evObserveCompleted()** out of the **pJCS** port, it is certain **JCS** will receive this message.

You must now consider the transitions between the **AwaitingOrders** state and the **StandingDown** state. **DOCC** will transition to **StandingDown** when it receives the **reqStandDown** message. So, you will add the label `reqStandDown` to the transition that connects the **AwaitingOrders** state to the **StandingDown** state.

The final transition is from **StandingDown** back to **AwaitingOrders**. **DOCC** will make this transition when it has completed standing down. So, you will use the following label on the transition that connects the **StandingDown** state to the **AwaitingOrders** state:

```
/OPORT(pJCS)->GEN(evDOCCStandDownCompleted())
```

which translates to `'generate the evDOCCStandDownCompleted() message on the JCS port'`. Recall that by sending the evDOCCStandDownCompleted() message out of the **pJCS** port, it is certain **JCS** will receive this message.

With the transition labels identified, you can add them to the OV-6b statechart. To do this, follow these steps:

1. Using the **Default Connector** icon on the **Drawing** toolbar, draw the default transitions to the **AwaitingOrders** state.

2. Using the **Transition** icon on the **Drawing** toolbar, draw the transitions listed below in order, as shown in the following diagram. Label the transitions by selecting the **Transition Label** icon on the **Drawing** toolbar and clicking on them.

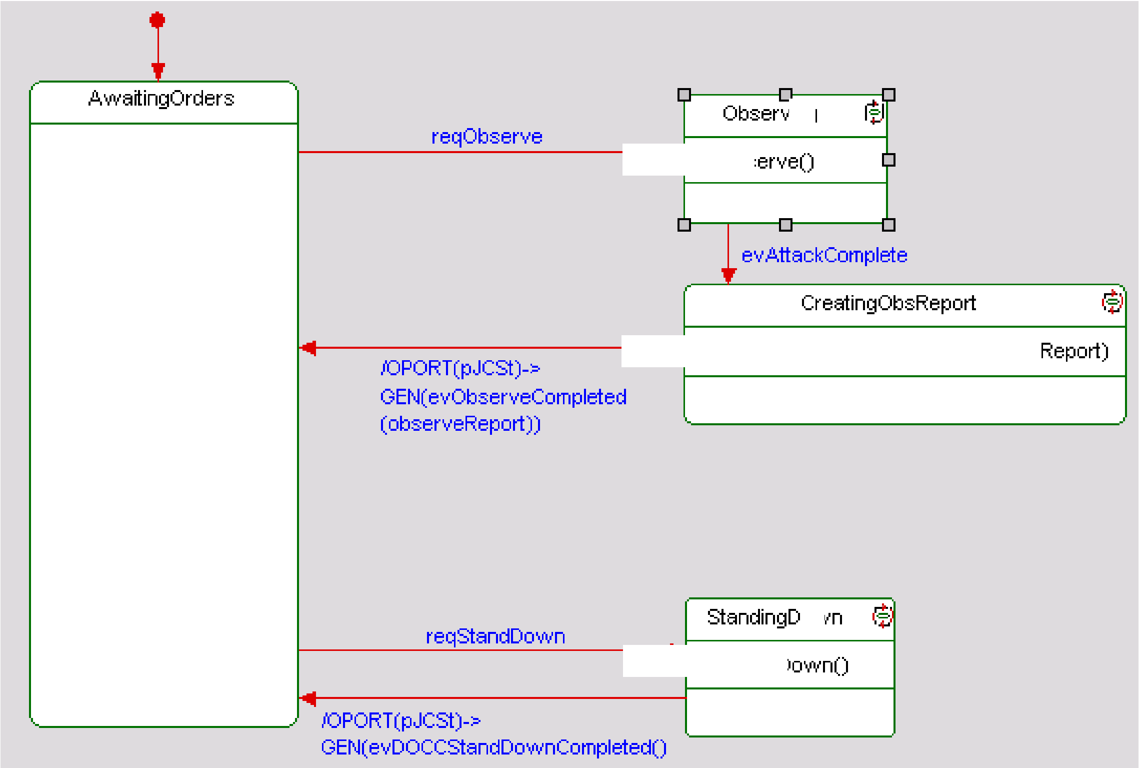
   a. **reqObserve**: Draw from **AwaitingOrders** to **Observing**

   b. **evAttackCompleted**: Draw from **Observing** to **CreatingObsReport**

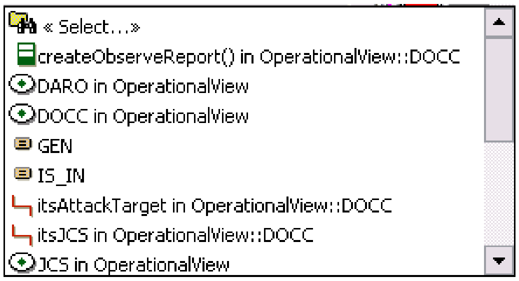
   c. **/OPORT(pJCS)->GEN(evObserveCompleted(observeReport))**: Draw from **CreatingObsReport** to **Awaiting Orders**

   d. **reqStandDown**: Draw from **AwaitingOrders** to **StandingDown**

   e. **/OPORT(pJCS)->GEN(evDOCCStandDownCompleted())**: Draw from **StandingDown** to **AwaitingOrders**

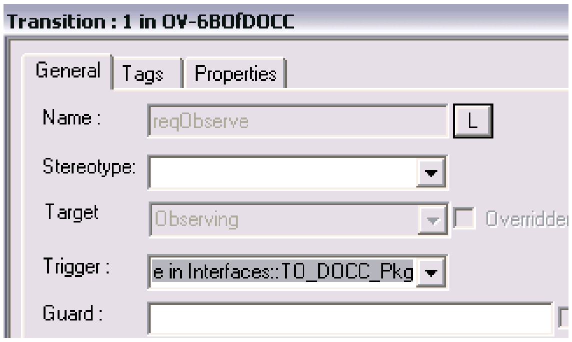


Note: Rational Rhapsody provides assistance in selecting the names of defined messages when labeling transitions. To activate this feature, follow the steps above, but use the **Ctrl+Spacebar** key sequence before, or at any point while typing in the label. You will be presented with a subwindow showing allowable message names you can select from. See the figure.



« Select…»
createObserveReport() in OperationalView::DOCC
DARO in OperationalView
DOCC in OperationalView
GEN
IS_IN
itsAttackTarget in OperationalView::DOCC
itsJCS in OperationalView::DOCC
JCS in OperationalView

3. Double-click the **reqObserve** transition. The dialog box that appears should closely resemble the following figure. If the dialog box looks different, make sure you selected the correct name to the appropriate transition.



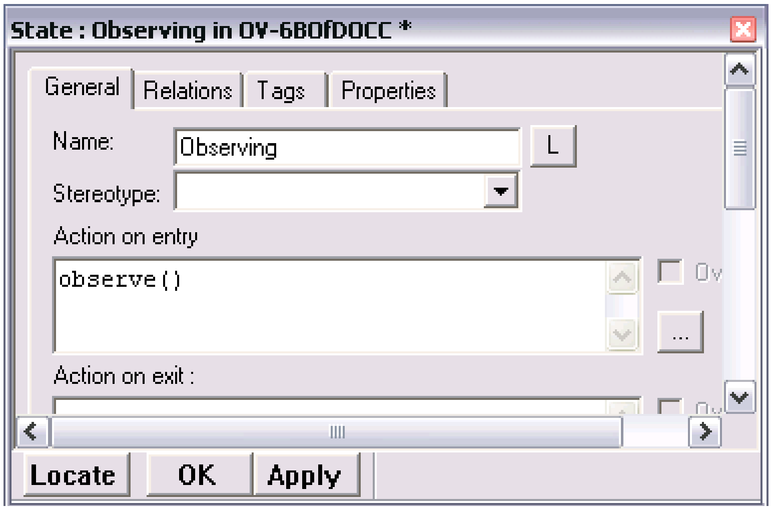
Transition : 1 in OV-6BOfDOCC

Name : reqObserve
Stereotype:
Target Observing □ Overridde
Trigger : e in Interfaces::TO_DOCC_Pkg
Guard :

With the transition labels in place, add the execution of the mission objectives to the **Observe**, **PreparingObservationReport**, and **StandingDown** states.

To add these mission objectives to be performed in a state, follow these steps:

1. Right-click the **Observing** state, and select **Features**.
2. In the Features dialog box, enter `observe()` in the **Action on entry** box, as shown in the following figure:



State : Observing in OV-6B0fDOCC *

Name: Observing

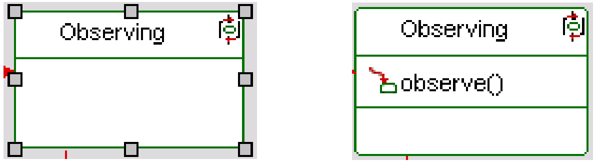
Stereotype:

Action on entry

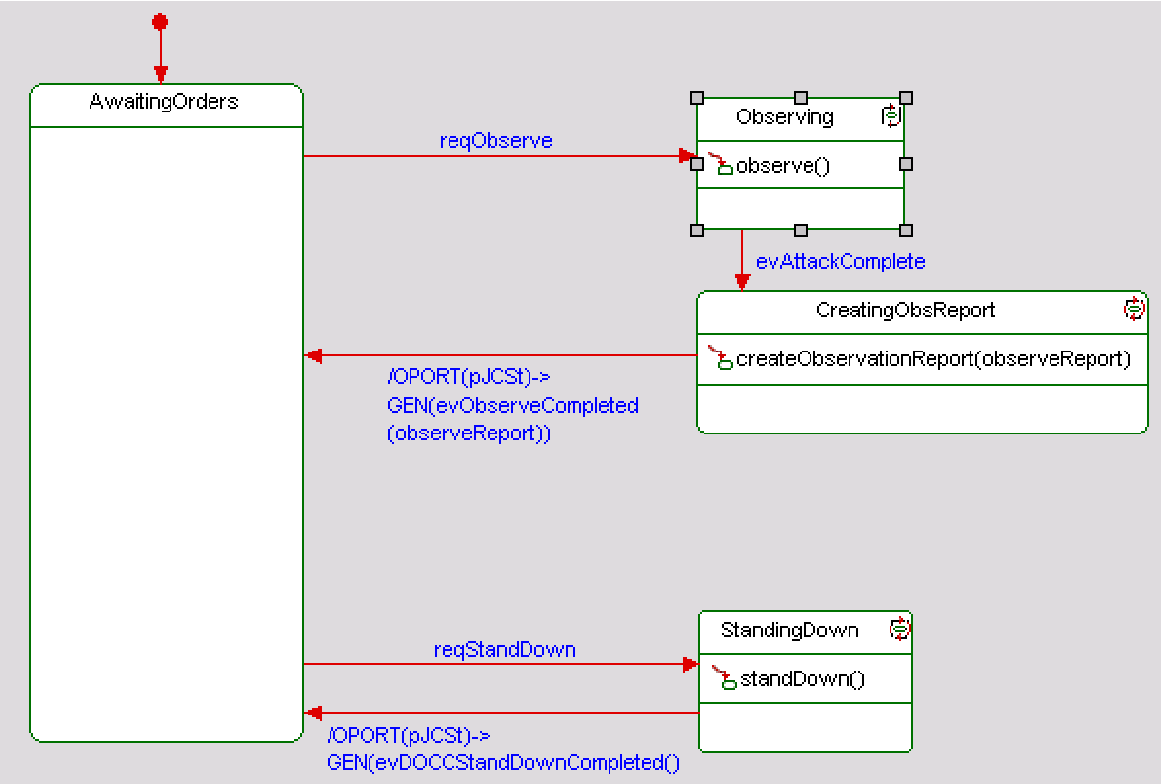
observe()

Action on exit :

3. You will see the state reaction icon in the top right corner of the **Observing** state. Click the icon to toggle the display of the observe() mission objective on the diagram, as shown in the following figure:



4. Repeat steps the above steps to add the **createObservationReport()** mission objective to the **CreatingObsReport** state, and the **standDown()** mission objective to the **StandingDown** state, as shown in the following figure:



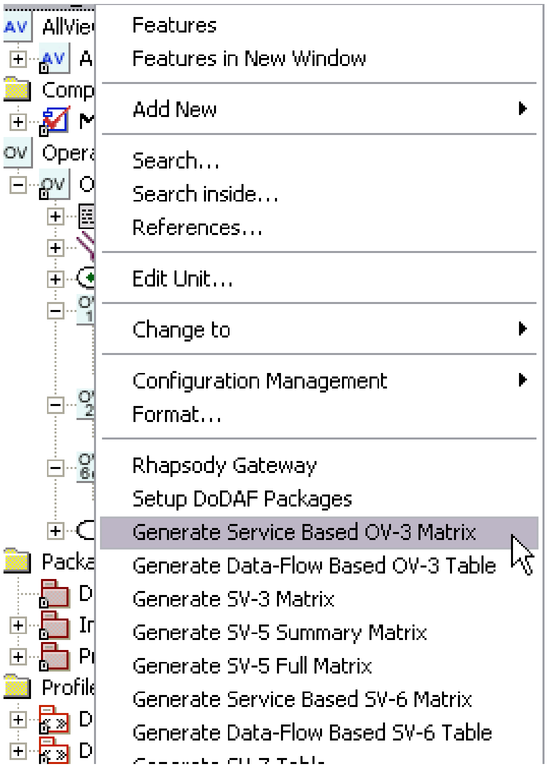
5. Save the architecture model, select **File** > **Save**.

Generating the OV-3 Operational Information Exchange Matrix

The **OV-3 Operational Information Exchange Matrix** provides a detailed report of the information exchange between operational nodes.

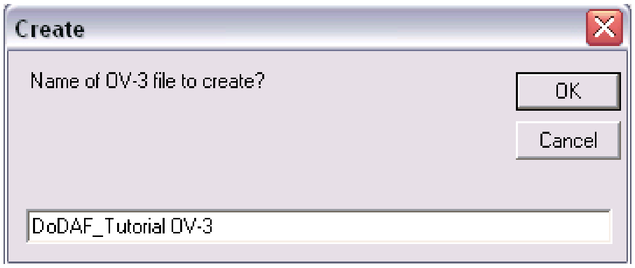
To automatically generate the OV-3 Matrix, follow these steps:

1. Right-click the Project folder and select **Generate Service Based OV-3 Matrix**, as shown in the following figure. The **Create** dialog box opens.



Note: Given the content of the model in this tutorial, you can also generate the **Data-Flow Based OV-3 Table**.

2. Enter the name of the OV-3 file and click **OK**.



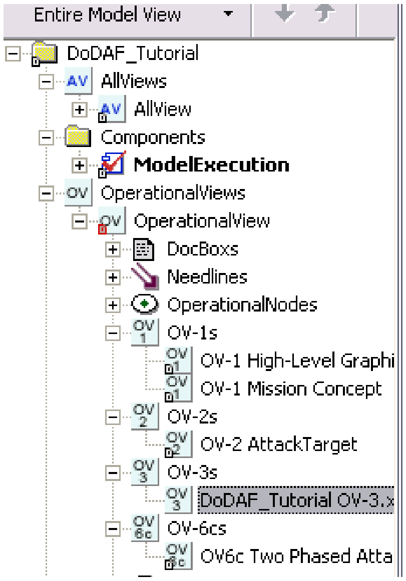
Create

Name of OV-3 file to create?

DoDAF_Tutorial OV-3

3. A message appears stating that the OV-3 has been created successfully. Click **OK**.

4. The browser now reflects the changes made, as shown in the following figure:

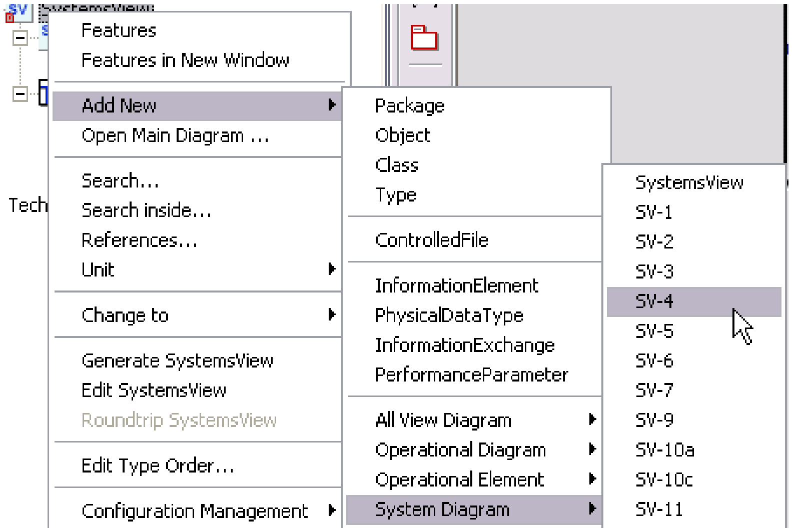


Creating the SV-4 System Function Description Diagram

In the Systems View, the Operational Activities are closely related to the SV-4 Systems Functionality Description. The SV-4 is a refinement of the OV-5 Mission Objective Model where the mission objective flow is translated to a flow of systems functions.

To create an SV-4 System Function Description Diagram, follow these steps:

1. Expand **SystemsView** in the Rational Rhapsody browser.
2. Right-click **SystemsView** and select **Add New** > **System Diagram** >**SV-4**, as shown in the following figure.
3. Name the diagram `SV-4 Attack` and press the **Enter** key. The SV-4 icon appears in the browser.



4. Expand **OperationalView** and **OperationalActivities** in the Rational Rhapsody browser.
5. Drag-and-drop the attack operational activity onto the drawing area.

For each operational activity, the SV-4 diagram is the main connection point between the operational view and the systems view.

The “attack” operational activity is the center of this diagram. You must now display the systems which are needed to perform, or “execute” the “attack” operational activity. You have dragged the “attack” operational activity into the drawing area. You must now show which system functions must be performed to realize the “attack” operational activity.

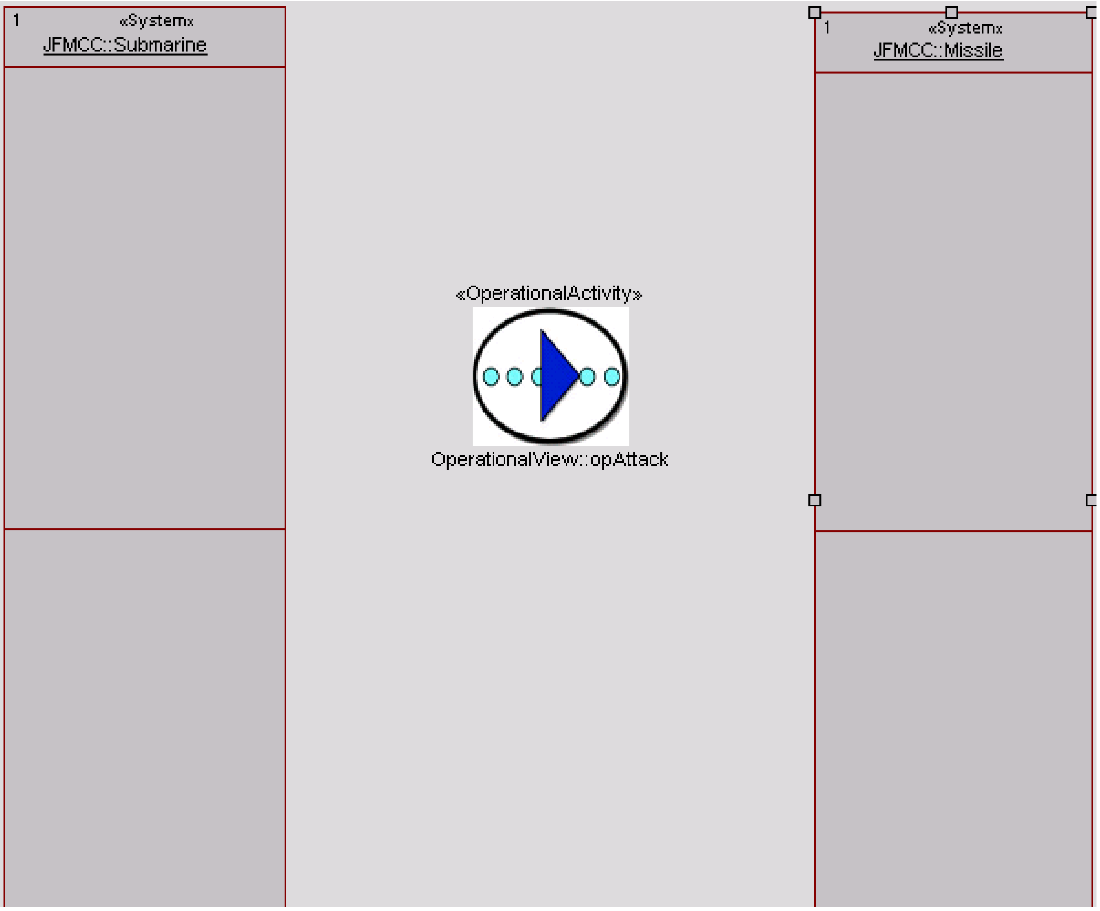
The system functions reside within unique Systems. Each System resides in a SystemsNode.

The Systems involved in this SV-4 diagram, are the Submarine and Missile Systems. The Submarine system performs the following functions: select silo, open silo door, and extend launch tube. The Missile system performs the following functions: arm, launch, and guidance.

To add the Submarine and Missile Systems and their corresponding functions, follow these steps:

1. Using the **System** icon ⊞ on the **Drawing** toolbar, draw two narrow rectangles (systems) on either side of the attack operational activity. Name the left system `Submarine` and the right system `Missile`, as shown in the following figure:

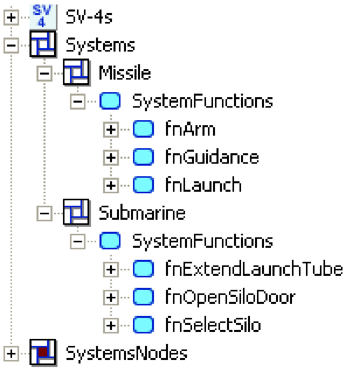
2. For functionality purposes, change the appearance of each System. Right-click a system "rectangle" and select **Display Options**. Clear the **Enable Image View** check box. Click **OK** to close the **Display Options** dialog box. Repeat this step for the other System in the diagram.



3. Select each system and click the **Specification/Structured View** icon to enter “Structured” mode. This enables Rational Rhapsody to “assign” the system functions to their respective systems.

   If Rational Rhapsody is in “Specification” mode, Rational Rhapsody does not associate the system functions with their respective Systems, and the browser displays the system functions as being independent of each system.

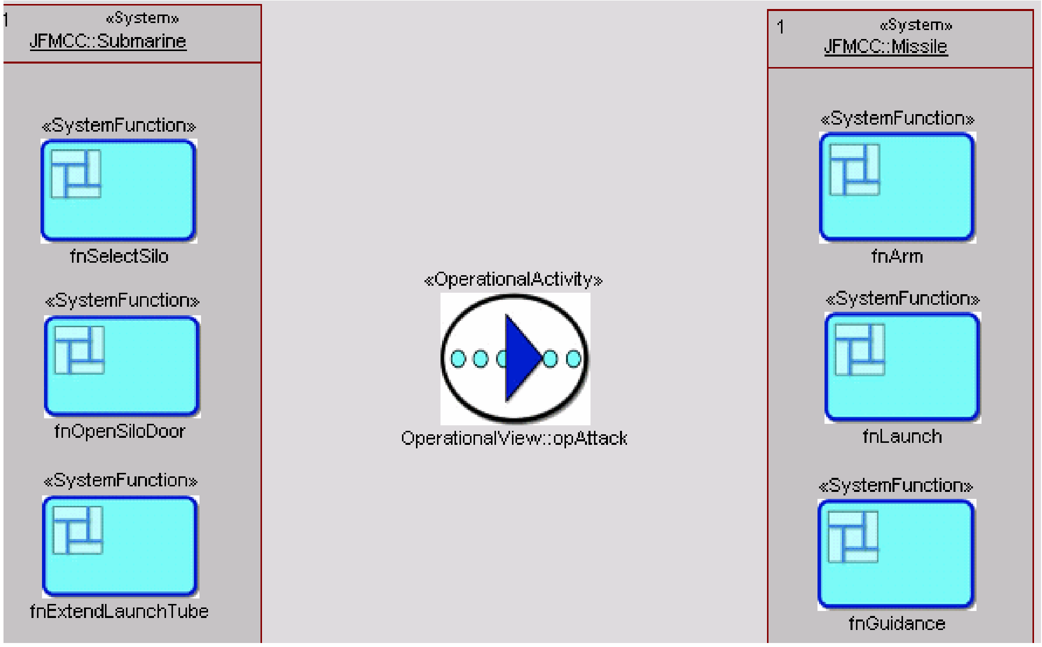
   The figure below displays the browser with the system functions assigned to their respective systems.



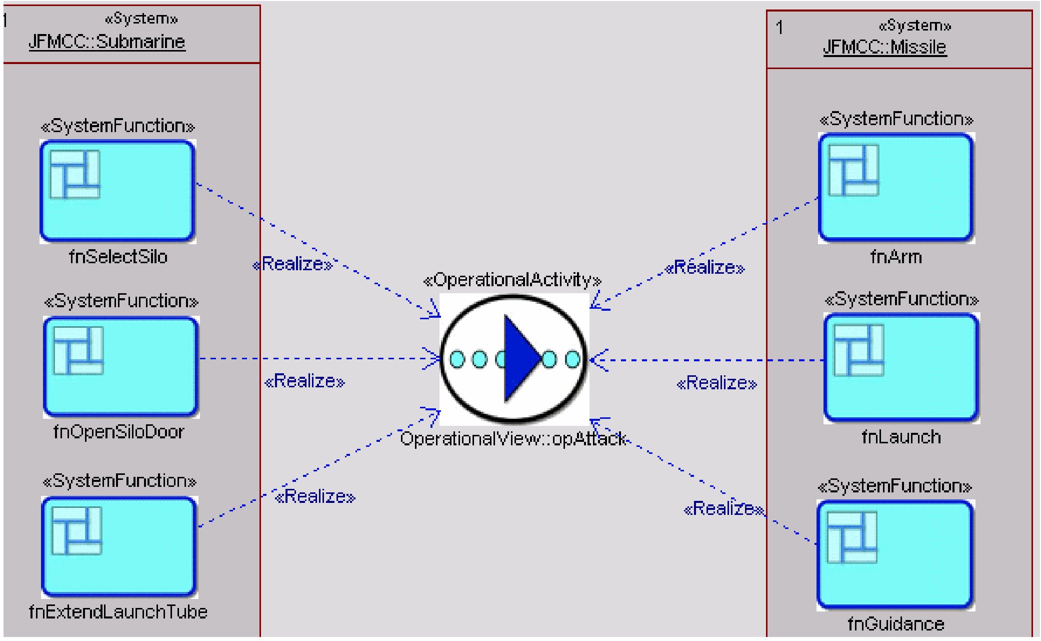
4. Using the **System Function** icon ▢ on the **Drawing** toolbar, draw the system functions `fnSelectSilo`, `fnOpenSiloDoor`, and `fnExtendLaunchTube` in the Submarine System rectangle.

   Draw the system functions `fnArm`, `fnLaunch`, and `fnGuidance` in the Missile System.

   The "fn" prefix on each name stands for "function" and keeps Rational Rhapsody from confusing like system function and operation names when generating code.



5. Using the realize tool ⊹, draw realize lines from each system function to the attack operational activity, as shown in the following figure:

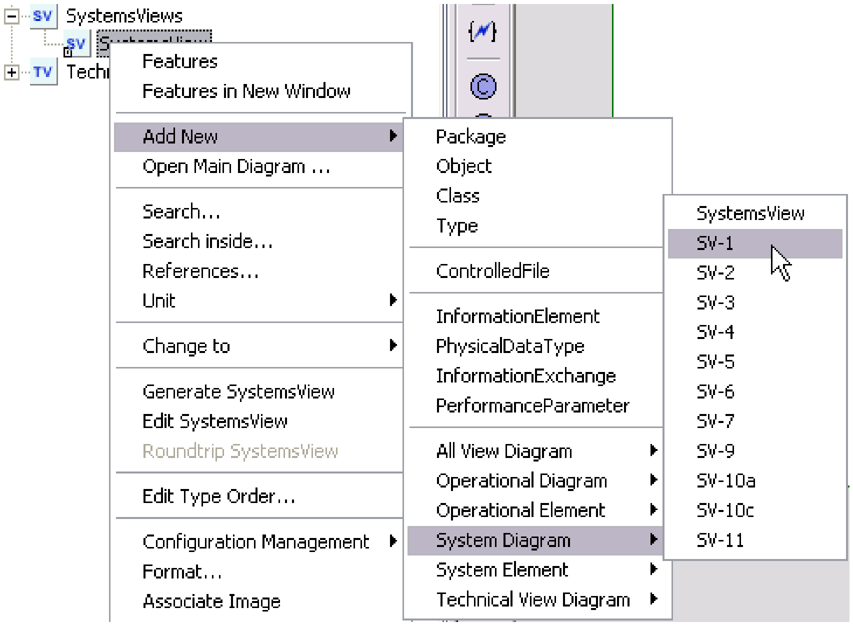


Creating the SV-1 System Interface Description Diagram

The starting point for the **SV-1 Systems Interface Description** is the **OV-2 Operational Node Connectivity** diagram. The OV-2 diagram provides a guide for developing the SV-1, where the Operational Nodes can be detailed with the systems contained by the nodes (recall the systems have been identified in the mission objective diagram as described above).

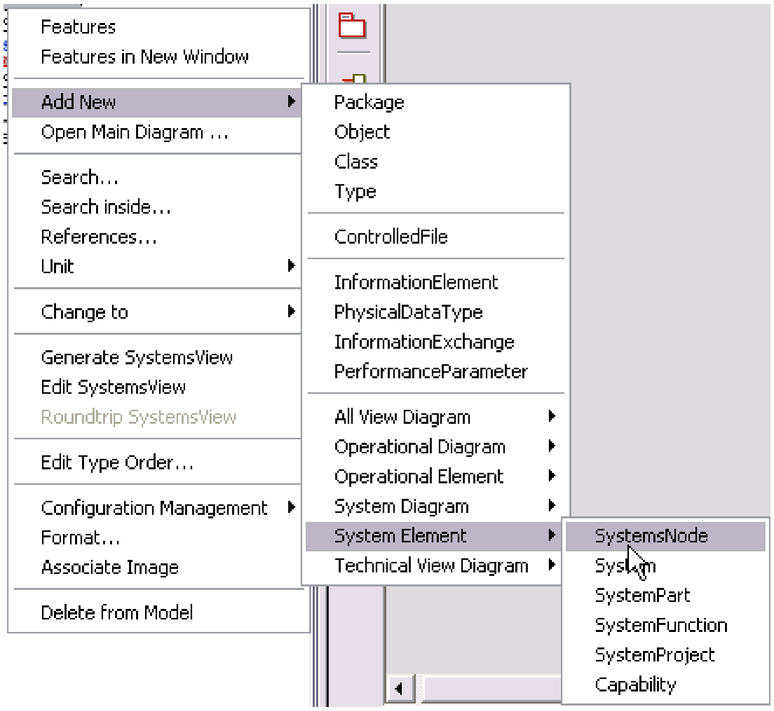
To create an **SV-1 System Interface Description** that is consistent with the **attackTarget** mission objective, follow these steps:

1. Expand the **SystemsView** package in the Rational Rhapsody browser.
2. Right-click **SystemsView** and select **Add New** > **System Diagram** > **SV-1**, as shown in the following figure:



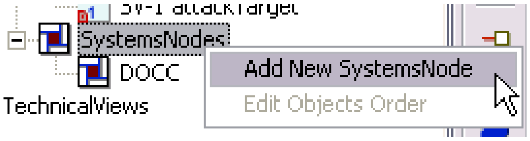
3. Name the diagram `SV-1 attackTarget`.

4. To add the first system nodes onto the diagram, right-click **SystemsView** and select **Add New** > **System Element** > **SystemsNode**, as shown in the following figure. Name the node `DOCC` and press **Enter**.



5. The **DOCC** system node appears in the browser. Drag-and-drop the **DOCC** system node icon from the browser onto the drawing area.

6. For the next system node, right-click **SystemsNodes** and select **Add New SystemsNode**, as shown in the following figure:



7. Name the next system node `JFMCC` and press **Enter**. The **JFMCC** system node appears in the browser under the **DOCC** system node icon. Drag the **JFMCC** system node icon from the browser onto the drawing area.

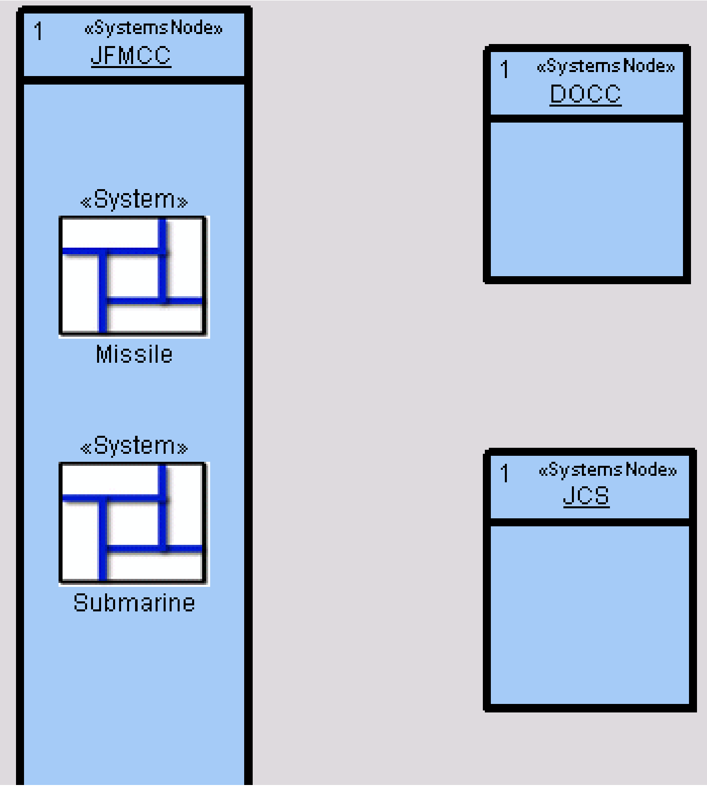
8. To create the JCS system node, repeat the previous two steps.

9. Select all three system node and right-click one of them. Select **Display Options** and clear the **Enable Image View** check box.

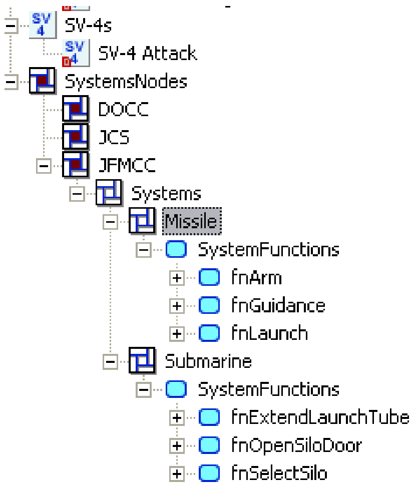
10. With all three system nodes still selected, click the Specification/Structured View icon to enter “Structured” mode.

11. Enlarge the **JFMCC** system node.

12. Drag-and-drop the Missile and Submarine systems from the browser into the **JFMCC** system node rectangle in the drawing area, as shown in the following figure:



13. Your Rational Rhapsody browser should now appear as shown in the following figure:

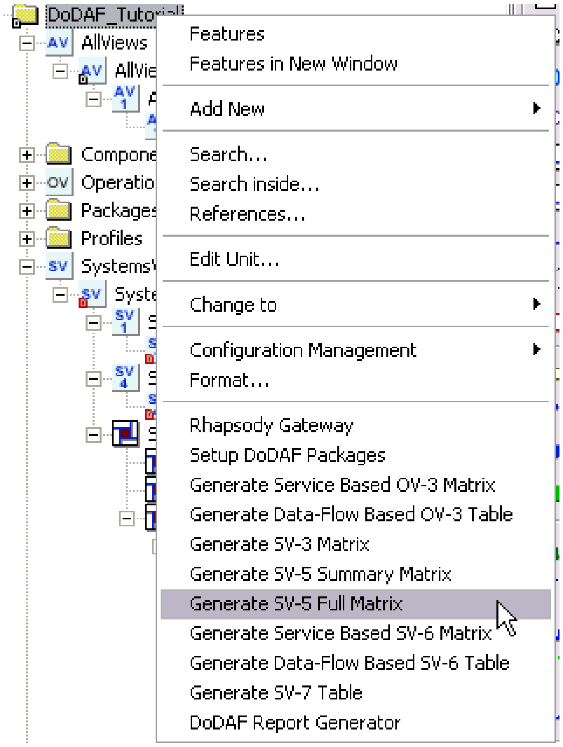


Creating the SV-5 Operational Activity to Systems Function Traceability Matrix

The SV-5 Matrix summarizes the mapping between system functions and operational activities based on the information contained in the SV-4 diagram(s).

To generate the **SV-5 Operational Activity to Systems Function Traceability Matrix**, use the following steps:

1. In the Rational Rhapsody browser, right-click the project folder (in this example **DoDAF_Tutorial**), and select **Generate SV-5 Full Matrix**, as shown in the following figure:



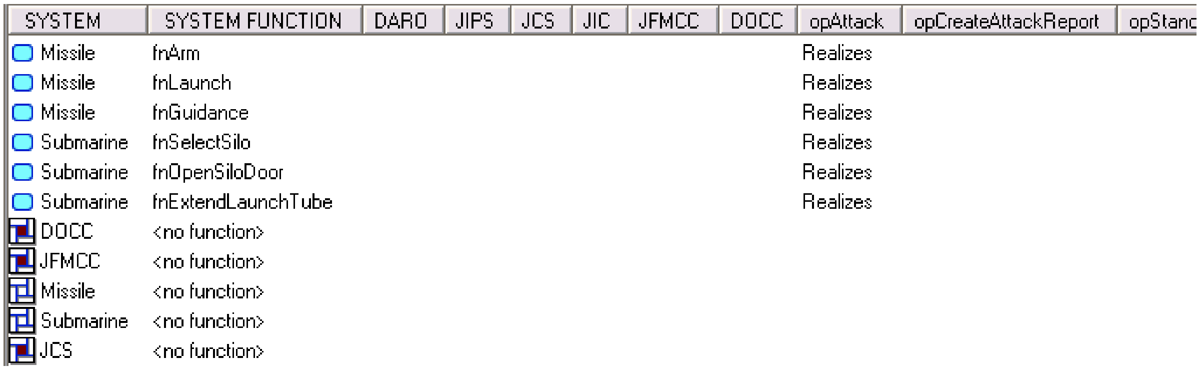
2. Click **OK** at the prompt.

3. Name the SV-5 at the next prompt. For this example, name the Matrix `Coordinated Land and Sea Attack SV5`.

Note

Do not use a dash in your SV-5 name, as conflicts might arise with some configuration management tools.

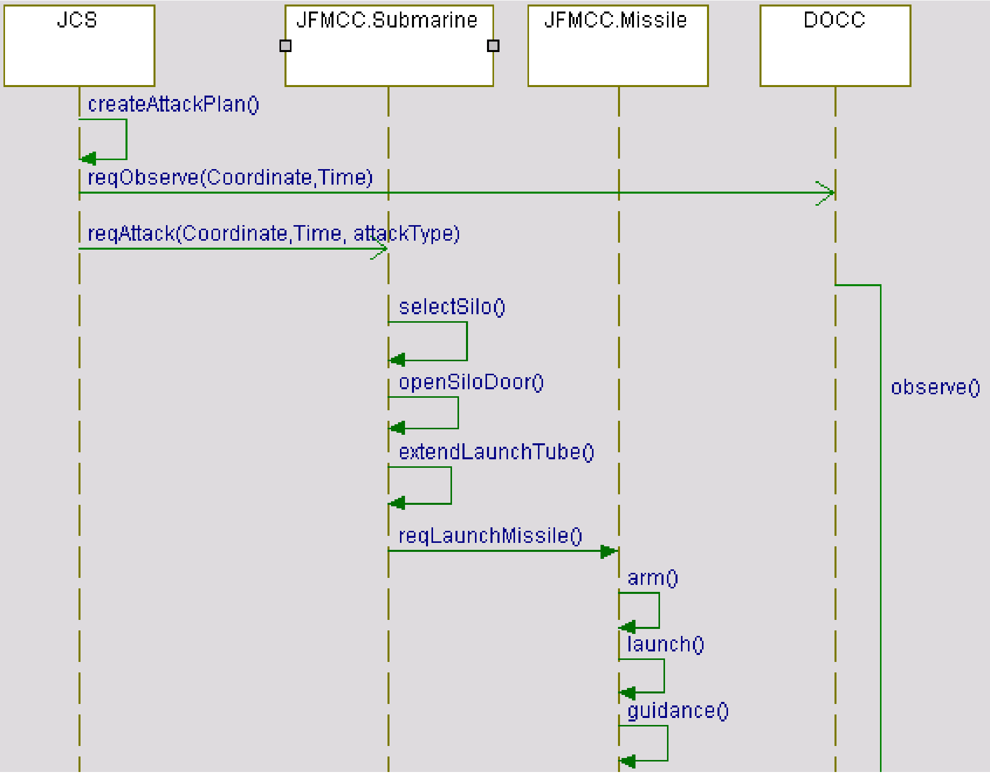
4. The SV-5 window appears, as shown in the following figure:



SYSTEM	SYSTEM FUNCTION	DARO	JIPS	JCS	JIC	JFMCC	DOCC	opAttack	opCreateAttackReport	opStanc
Missile	fnArm							Realizes		
Missile	fnLaunch							Realizes		
Missile	fnGuidance							Realizes		
Submarine	fnSelectSilo							Realizes		
Submarine	fnOpenSiloDoor							Realizes		
Submarine	fnExtendLaunchTube							Realizes		
DOCC	<no function>									
JFMCC	<no function>									
Missile	<no function>									
Submarine	<no function>									
JCS	<no function>									

Information on Creating the SV-10b and SV-10c Descriptions

This section gives high-level guidance on creating the SV-10b and SV-10c. Given that these products are very similar to their OV-6 counterparts, this tutorial does not give explicit step-by-step instruction on how to create these products. The SV-10c Systems Event Trace Description is a refinement of the OV-6c Operational Event Trace Description. To create the SV-10c, copy the OV-6c sequence diagram(s) into the SystemsView package, and then replace the operational node lifelines (the vertical lines) with the set of lifelines showing all the systems at the system node that participate in each specific event trace. Mission objectives are broken into one or more system functions, and additional messaging (communication) from system to system is added as needed. See the following figure for an example of the attack operational activity being performed by the Submarine and Missile systems within the JFMCC system node.

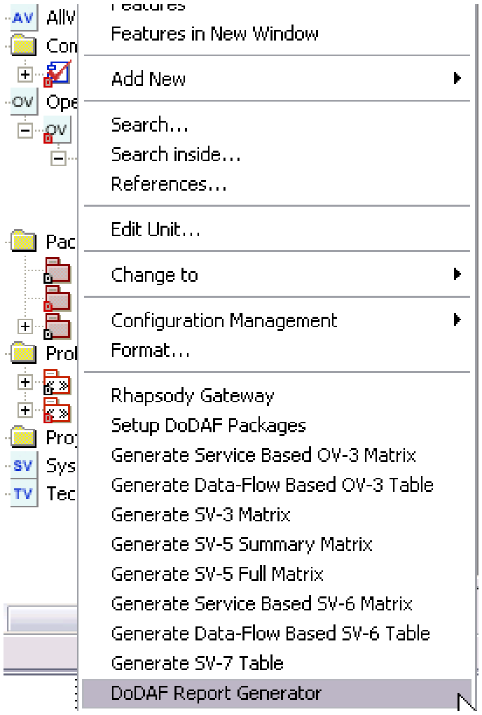


The SV-10b System State Transition Description can be based on the OV-6b Operational State Transition Descriptions if the systems within an operational node are loosely coupled. In this case, the OV-6b statechart diagrams can be copied into the **SystemsView** package and then modified as needed. In other cases where these systems are tightly coupled, it is often preferable to create new statecharts for the systems.

Generating the DoDAF Report from the Architecture Model

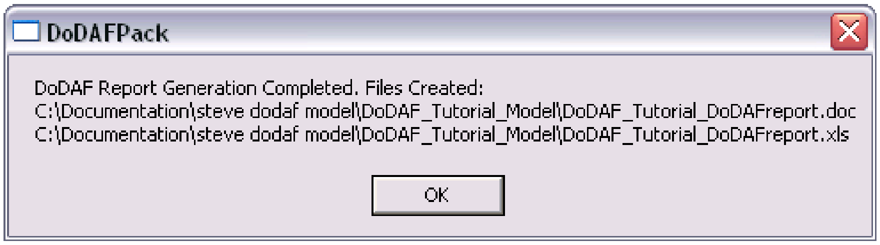
The last procedure in this tutorial is to generate a DoDAF report from the architecture model. Using another helper, you can generate a DoDAF report from the architecture model that includes all of the products utilized in the model. The AV-2 is generated for you automatically from the architecture model data. Keep in mind that the Rational Rhapsody model itself is a dynamic and searchable AV-2 including all elements of the architecture model.

To generate the DoDAF report, right-click your project folder in the Rational Rhapsody browser and select **DoDAF Report Generator**, as shown in the following figure:



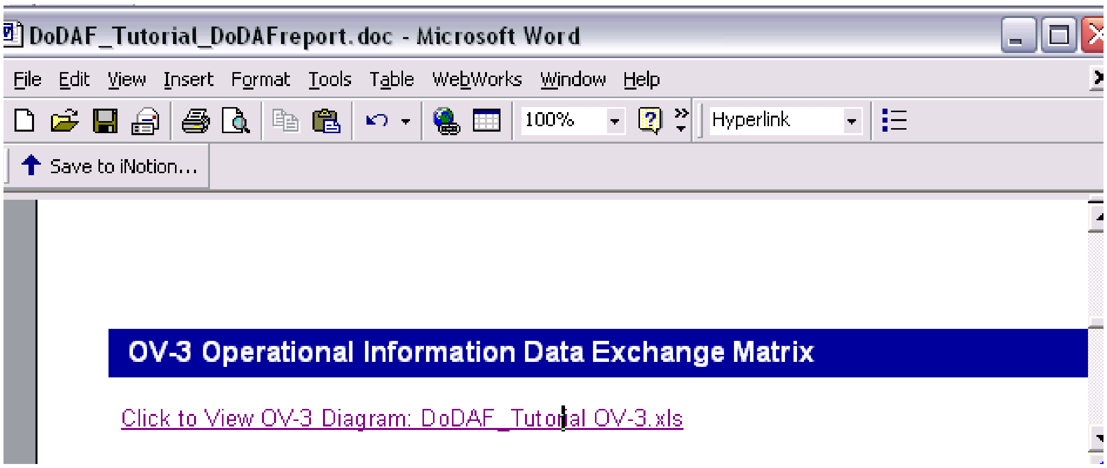
The DoDAF Report Generator window appears, and ReporterPLUS starts to load the model and generate the document. This process might take a few minutes. After the document has been created, it is displayed, and the document is formatted.

As the last step in generating the report, Microsoft Word displays a message indicating where the files have been saved in the Rational Rhapsody project directory, as shown in the following figure. The OV-3 spreadsheet is saved as a worksheet in the Excel workbook file in the Rational Rhapsody project directory with name <projectName>_DoDAFreport.xls. Click **OK** to complete the report generation.



DoDAF Report Generation Completed. Files Created:
C:\Documentation\steve dodaf model\DoDAF_Tutorial_Model\DoDAF_Tutorial_DoDAFreport.doc
C:\Documentation\steve dodaf model\DoDAF_Tutorial_Model\DoDAF_Tutorial_DoDAFreport.xls

The generated .doc Word file contains a hyperlink to the excel workbook containing the OV-3 Matrix, as shown in the following figure:


DoDAF_Tutorial_DoDAFreport.doc - Microsoft Word
File Edit View Insert Format Tools Table WebWorks Window Help
100%
Hyperlink
Save to iNotion...

OV-3 Operational Information Data Exchange Matrix

Click to View OV-3 Diagram: DoDAF_Tutorial OV-3.xls

It is possible to navigate to the definition of any interfaces displayed in the OV-3 Matrix. Double-clicking an interface name in the OV-3 Excel file or in the OV-3 tables in the Word document will bring you to the corresponding definition of the interface in the AV-2 Data Dictionary. The Back Button on the Word Web Toolbar can be used to return to the OV-3.

Note

The Word document can be converted to other formats including HTML and PDF using third party software (not provided by IBM).

Summary

Congratulations on completing the Rational Rhapsody DoDAF Add-on Tutorial! In this tutorial, you used Rational Rhapsody to create a DoDAF-compliant architecture model of the operational view for a simplified land and sea-based attack capability.

The tutorial started with the creation of a new DoDAF project using one of the helpers in the Rational Rhapsody DoDAF Add-on. Next, you were introduced to the AV-1 Overview and Summary for the architecture model

The AV-1, which was created for you, was associated to the architecture model using a hyperlink.

After working with the AV-1, you began the work of building the architecture model using UML diagrams in Rational Rhapsody. Starting with two mission objective diagrams, you created a traditional **OV-1 High Level Operational Concept Graphic**, as well as an **OV-1 High Level Mission Objective** diagram. Both diagrams helped you describe the mission and main operational nodes in the architecture. In the mission objective diagram, you identified the two primary mission objectives for this architecture: Assess Threat and Attack Target. In the remainder off the tutorial, you focused your attention on the Attack Target mission objective.

From the OV-1, you moved to the **OV-5 Mission Objective Model**, used to describe the operations that are normally conducted in the course of achieving a mission. You used an Activity diagram to capture the OV-5 information. The Activity diagram shows the functional flow of operations including control logic for achieving the Attack Target mission objective.

You then started your **OV-2 Operational Node Connectivity** diagram. Using another helper from the Rational Rhapsody DoDAF Add-on, you created the initial OV-2 showing the operational nodes you documented in the OV-1 mission objective diagram. You used a diagram to capture the OV-2 information. Next, you created the OV-6c, which you later used to add the mission objective and needline information to the OV-2.

The **Operational Event Trace Description** is used to show a sequence-ordered graph of the information exchanges between operational nodes as a result of a particular operational scenario. In constructing the **OV-6c Event Trace Description**, you referred to the OV-5 to guide you along. While there are many event traces you can construct for the Attack Target mission objective, you focused on one that covered all the operations you documented in the OV-5. In constructing the **Two Phased Attack Event Trace**, you used a helper in the DoDAF Add-on to create the initial sequence diagram showing the Operational Node lifelines. Next, you added the appropriate mission objectives. You then added messages that are sent between operational nodes to request a node carry out an mission objective. Last, you added messages that are sent between operational nodes to provide notification and reporting information.

With the OV-6c completed, you were then able to complete the OV-2. You used another helper to update the OV-2 nodes with mission objectives and needline information. You examined the ports and interfaces created by the helper to see how the messages you created in the OV-6c were translated into needlines and information exchanges. All that you had to do manually was layout the nodes and add the links to connect the ports.

After completing the OV-2, you moved on to the OV-7 Logical Data Model. In the OV-7, you documented the data exchanged in the OV-6c event traces. Some of this data were further decomposed in the OV-7 diagram.

You then created the **OV-6b Operational State Transition Description**. The OV-6b shows how an operational node responds to input messages. You started your statechart by reviewing the OV-6c and identifying the modes of operation. You added the state transitions, and discussed the trigger, guards, and actions used on the transition labels.

You created the **OV-3 Operational Information Exchange Matrix**. This matrix documents the information exchanged among the operational nodes. This was the last operational view product created. The OV-3 is generated using a DoDAF helper.

At this point you transitioned to the Systems View products. You started by creating the **SV-4 Systems Functionality Description** where operational activities were mapped to system functions and system functions allocated to systems.

You created the **SV-1 System Interface Description**. This product contained the system nodes which corresponded to operational nodes. These system nodes house the individual systems identified in the SV-4 product. These individual systems in turn contain system functions.

The last systems view product created was the **SV-5 Operational Activity to Systems Function Traceability Matrix**. This product showed the mapping of operational activities to system functions. The SV-5 is generated using a DoDAF helper.

The last step in the tutorial was to generate a DoDAF report from the architecture model. Here too, you used a helper from the Rational Rhapsody DoDAF Add-on. The generated report documents all of the architecture products you created in building your architecture model. In addition, the report includes the hyperlinked AV-1, and the AV-2 and OV-3, which are generated from the architecture model data. Keep in mind that the Rational Rhapsody model itself is a dynamic and searchable AV-2 including all elements of the architecture model.

With just a little more effort, you can simulate your architecture model with a user-friendly Web-based interface that provides stimulus to the model and displays the model's response. You can use the Rational Rhapsody Sequence diagram Compare capability to ensure the model's actual behavior matches what you captured in your **OV-6c Event Trace Descriptions**. You can use Rational Rhapsody Test Conductor to run regressions tests as you modify the architecture model and ensure you do not break any of it's capabilities, or to compare the behavioral performance of alternative architectures. You can also use the Rational Rhapsody Gateway to provide full requirements traceability and impact analysis as you consider alternative architectures. This tutorial just scratches the surface of what you can do when you build a DoDAF compliant architecture model with Rational Rhapsody.

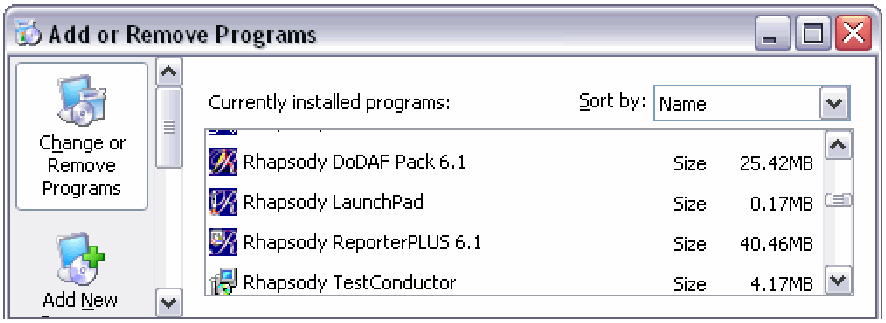
The solution presented in this tutorial is based on the Rational Rhapsody Harmony Process for Systems and Software Engineering, and is immediately available when purchasing the Rational Rhapsody DoDAF Add-on. You have created this solution to be customizable and extensible to meet the needs of a wide variety of customers, including those needing to develop models compliant with architecture frameworks other than DoDAF. All of the automation provided in this solution is based on the Rational Rhapsody COM API and the ReporterPLUS document generator, making it easy to tailor to other approaches for architecture modeling.

Troubleshooting

The following examples are possible problems users might encounter when using the DoDAF Add-on.

Verifying the Rational Rhapsody DoDAF Add-on Installation

The first step in verifying the installation is to examine the targets for the shortcuts in your start menu under **Start** > **All Programs** > **Rhapsody DoDAF Add-on**. If your start menu does not include an entry for the Rational Rhapsody DoDAF Add-on, then use the Add or Remove Programs utility in the Windows Control Panel to see if the DoDAF Add-on is installed, as shown in the following figure:



Rhapsody DoDAF Pack 6.1 Size 25.42MB
Rhapsody LaunchPad Size 0.17MB
Rhapsody ReporterPLUS 6.1 Size 40.46MB
Rhapsody TestConductor Size 4.17MB

If you find the Rational Rhapsody DoDAF Add-on is not installed on your system, install it. If the Rational Rhapsody DoDAF Add-on is installed on your system, then the next step is to check your system’s registry to find where the DoDAF Add-on is installed. To examine your system’s registry, you use the RegEdit program.

Note

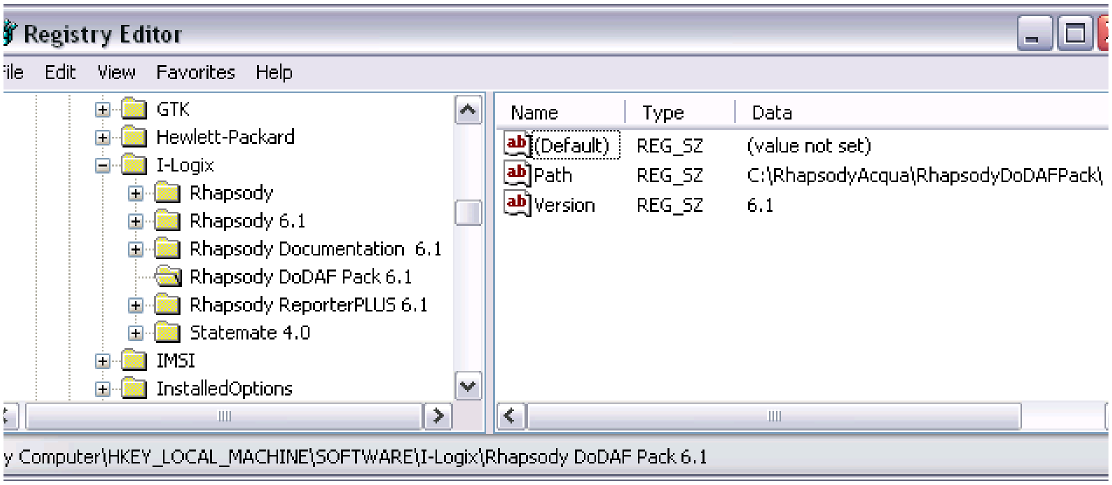
Use care whenever you access your system’s registry as inadvertent changes can render your system unusable. For this tutorial, you are not making any changes to your system’s registry, just examining one of the registry settings.

To launch RegEdit and check your system’s settings, follow these steps:

1. Select **Start** > **Run** from the Windows Taskbar.

2. Type in `regedit` for the program name, and click **OK**.

3. In Registry Editor window, locate the `HKEY_LOCAL_MACHINE\SOFTWARE\I-Logix\Rhapsody DoDAF Add-on` key, as shown in the following figure:



Name	Type	Data
ab (Default)	REG_SZ	(value not set)
ab Path	REG_SZ	C:\RhapsodyAcqua\RhapsodyDoDAFPack\
ab Version	REG_SZ	6.1

y Computer\HKEY_LOCAL_MACHINE\SOFTWARE\I-Logix\Rhapsody DoDAF Pack 6.1

4. Make sure the Version setting shows you are using version 1.4.0 or newer. If you are using an older version of the Rational Rhapsody DoDAF Add-on, please upgrade to the latest version.

5. Make note of the Path setting for the Rational Rhapsody DoDAF Add-on.

6. Close Regedit using **File** > **Exit** from the drop-down menus.

Using Windows Explorer, locate the folder specified by the path setting, and verify this tutorial, the av-1.doc and ov-1.bmp files are located in the Tutorial subfolder in this path. If these files are not found, or if you cannot find the key in your registry, then your Rational Rhapsody DoDAF Add-on installation has been damaged. Uninstall and then re-install the Rational Rhapsody DoDAF Add-on to repair your installation if it has been damaged.

Manually Adding the DoDAF Helpers

Typically, the Rational Rhapsody DoDAF Add-on installer will automatically configure the Rational Rhapsody DoDAF Add-on helpers for you.

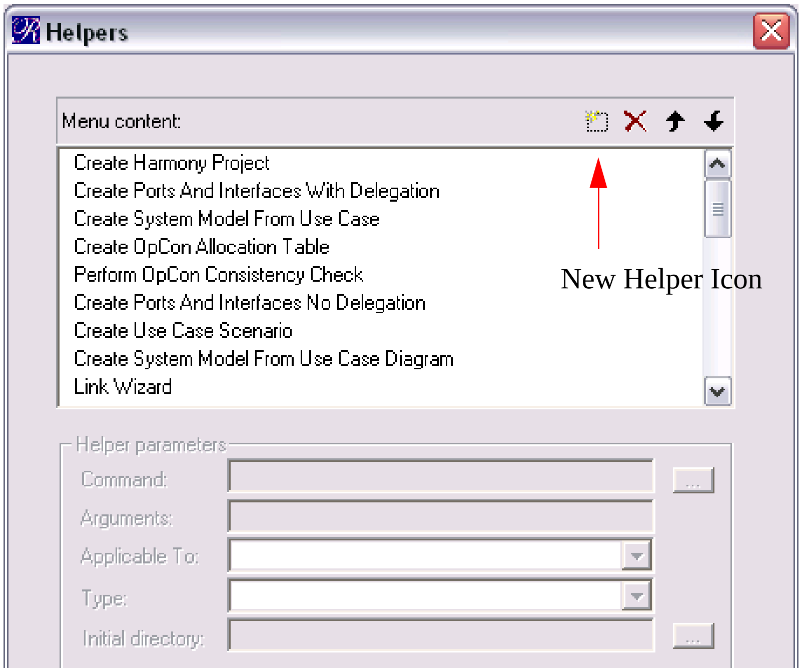
There are thirteen helpers provided with the Rational Rhapsody DoDAF Add-on. The table below summarizes the helpers and their settings:

Helper Name	Command	Arguments	Applicable To
Setup DoDAF Packages	<Path>\DoDAFPack.exe	-dodaf -i	DoDAF
Create OV-2 from Mission Objective	<Path>\DoDAFPack.exe	-dodaf -uc	MissionObjective
Create OV-6c from Mission Objective	<Path>\DoDAFPack.exe	-dodaf -sd	MissionObjective
Update OV-2 from OV-6c	<Path>\DoDAFPack.exe	-dodaf -d	OV-6c
Generate Service Based OV-3 Matrix	<Path>\DoDAFPack.exe	-dodaf -ov3m	DoDAF
Generate Data-Flow Based OV-3 Table	<Path>\DoDAFPack.exe	-dodaf -ov3t	DoDAF
Generate SV-3 Matrix	<Path>\DoDAFPack.exe	-dodaf -sv3	DoDAF
Generate SV-5 Summary Matrix	<Path>\DoDAFPack.exe	-dodaf -sv5short	DoDAF
Generate SV-5 Full Matrix	<Path>\DoDAFPack.exe	-dodaf -sv5long	DoDAF
Generate Service Based SV-6 Matrix	<Path>\DoDAFPack.exe	-dodaf -sv6m	DoDAF
Generate Data-Flow Based SV-6 Table	<Path>\DoDAFPack.exe	-dodaf -sv6t	DoDAF
Generate SV-7 Table	<Path>\DoDAFPack.exe	-dodaf -sv7	DoDAF
DoDAF Report Generator	<Path>\DoDAFPack.exe	-dodaf -report	Project

If you find that any of the DoDAF helpers are missing from the submenus in Rational Rhapsody, first make sure you have verified the Rational Rhapsody DoDAF Add-on installation as described in the previous section. Once you have verified the installation, manually configure the helpers using the instructions that follow. While verifying the installation, make sure you have noted the path where the Rational Rhapsody DoDAF Add-on is installed.

To install the Rational Rhapsody DoDAF Add-on helpers, follow these steps:

1. Select **Tools** > **Customize** from the Rational Rhapsody main window to open the Helpers dialog box, as shown in the following figure:


Helpers
Menu content:
Create Harmony Project
Create Ports And Interfaces With Delegation
Create System Model From Use Case
Create OpCon Allocation Table
Perform OpCon Consistency Check
Create Ports And Interfaces No Delegation
Create Use Case Scenario
Create System Model From Use Case Diagram
Link Wizard
New Helper Icon
Helper parameters
Command:
Arguments:
Applicable To:
Type:
Initial directory:
...

2. Use the Create New Helper Icon to create a new helper, and enter the appropriate helper name from the There are thirteen helpers provided with the Rational Rhapsody DoDAF Add-on. The table below summarizes the helpers and their settings: table.

3. Set the Command, Arguments, and Applicable To: as set in the There are thirteen helpers provided with the Rational Rhapsody DoDAF Add-on. The table below summarizes the helpers and their settings: table on the previous page. For the command, replace <Path> with the path where the Rational Rhapsody DoDAF Add-on is installed.

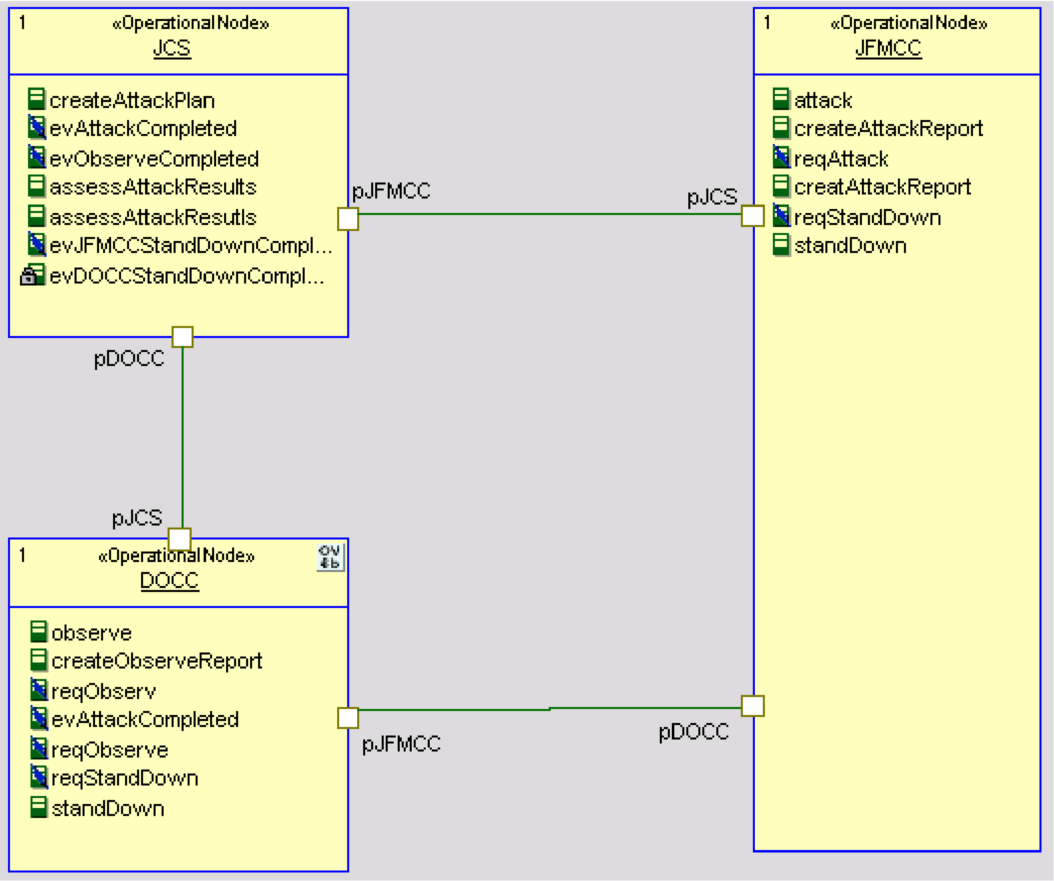
4. Set Type to External Program and select the Show in popup menu box.

5. Click **Apply** to apply the changes.

6. If needed, configure another helper by repeating steps 2-5, or click **OK** to close the Helpers dialog box.

Fixing the Model if Messages Appear as Mission Objectives

After updating the OV-2 from an OV-6c, you should check the OV-2 and make sure the messages between nodes are correctly appearing as events. If you find a message going between operational nodes appears in the OV-2 as an operation rather than an event, this indicates the message type was not changed to Event in the OV-6c diagram.The following figure shows an example of evJFMCCStandDownCompleted appearing with the private operation symbol 🔒, but should have the event symbol ▧.



To fix this problem, follow these steps:

1. Expand the Rational Rhapsody browser to see the operation under the appropriate operational node. In this case, you need to expand the folder `Project_1\OperationalViews\Operational View\Operational Nodes\JCS\Operations\evDOCCStandDownCompleted`.

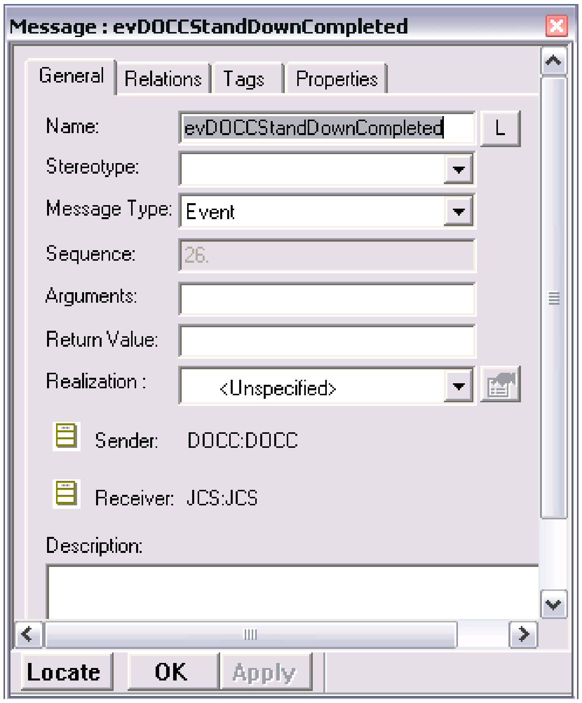
2. Delete the operation by right-clicking on the incorrect operation and selecting **Delete from Model**.

3. Open the OV-6c diagram and check all the messages with the message name in question.

Note

To spot the problematic message, make sure messages that connect operational nodes have a hollow arrowhead ←, and mission objective messages (messages that start and end on the same node) have a solid arrowhead ◀.

4. To change the incorrect message, change the message type by double-clicking the message and selecting **Event** from the **Message Type** drop-down list, as shown in the following figure. Click **OK**.



Message : evDOCCStandDownCompleted

Name: evDOCCStandDownCompleted

Stereotype:

Message Type: Event

Sequence: 26.

Arguments:

Return Value:

Realization : <Unspecified>

Sender: DOCC:DOCC

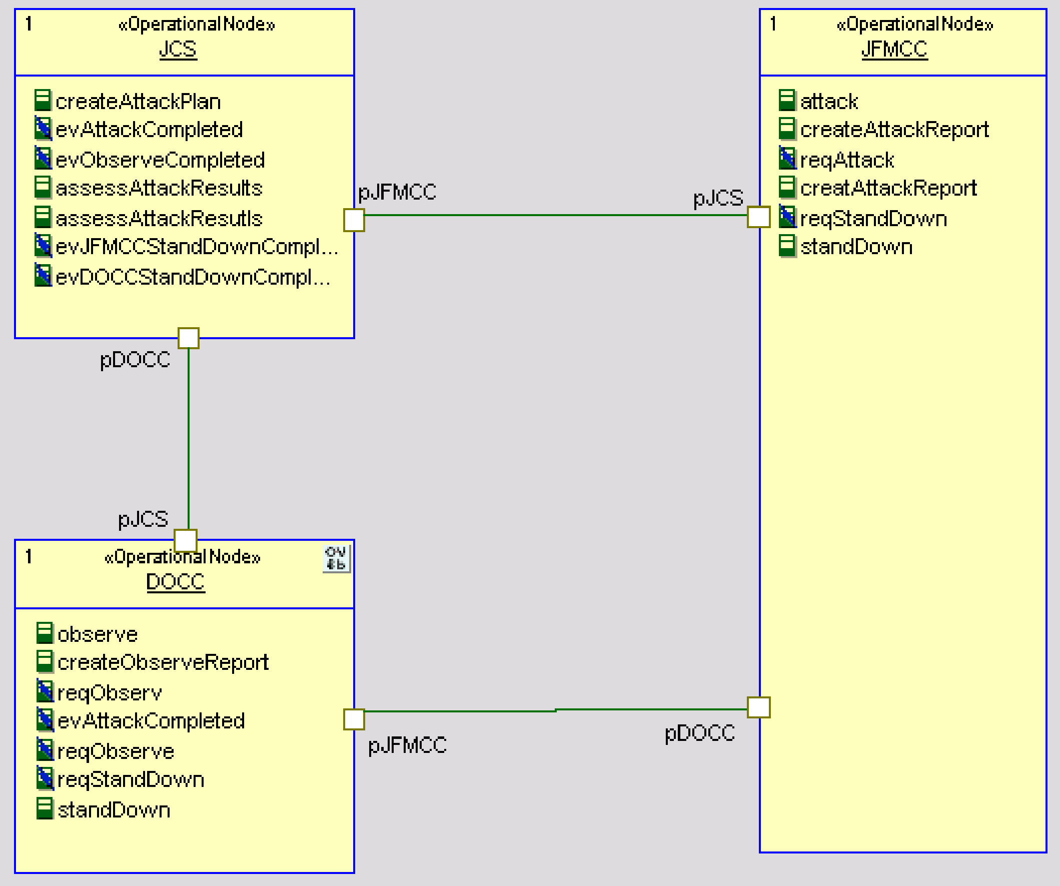
Receiver: JCS:JCS

Description:

5. With the message selected, use **Edit > Auto Realize** from the pull-down menus to realize the message.

6. Select **OV-6c Two Phased Attack** from the Rational Rhapsody browser.

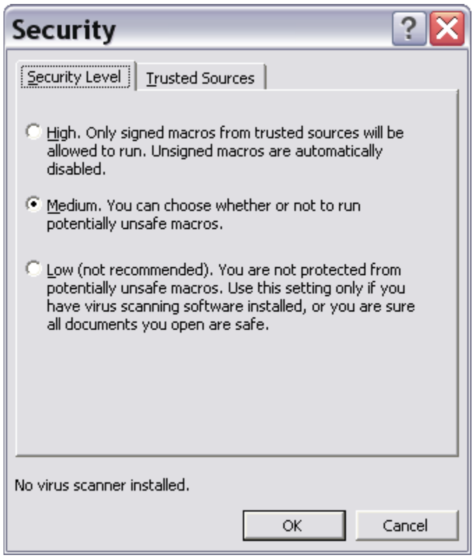
7. Right-click the OV-6c icon and select **Update OV-2 from OV-6c**. See the figure.



8. You have repaired the OV-6c and OV-2 diagrams.

OV-3, SV-3, SV-5, SV-6, Captions, or Table of Figures are Not Visible in the Document

If the final document does not include an OV-3 matrix, figure captions, or table of figures, the macros in the DoDAFReportRTF.dot Word Document Template file are not being executed. Make sure the security settings for Word are set to medium security to allow macros to be run. You can change the security setting using **Tools** > **Macro** > **Security** (see the figure).



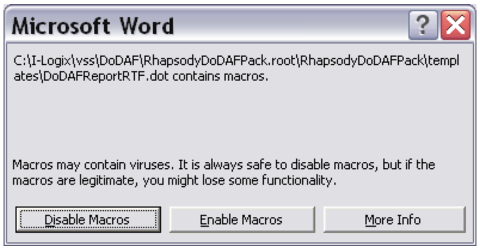
○ High. Only signed macros from trusted sources will be allowed to run. Unsigned macros are automatically disabled.

● Medium. You can choose whether or not to run potentially unsafe macros.

○ Low (not recommended). You are not protected from potentially unsafe macros. Use this setting only if you have virus scanning software installed, or you are sure all documents you open are safe.

No virus scanner installed.

Microsoft Word might prompt you to enable macros in the generated DoDAF report when it is opened. You will need to select Enable Macros in this dialog box when opening the DoDAF report in order for the macros to run (see the figure).



C:\I-Logix\vss\DoDAF\RhapsodyDoDAFPack.root\RhapsodyDoDAFPack\templates\DoDAFReportRTF.dot contains macros.

Macros may contain viruses. It is always safe to disable macros, but if the macros are legitimate, you might lose some functionality.

Index