REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	2/12/96	5/18/95-11/18/96 Final Report

4. TITLE AND SUBTITLE
Structured Essential Model for Mine Warfare

5. FUNDING NUMBERS
Contract # N60921-95-C-0027

6. AUTHOR(S)
Dr. Joseph H. Discenza and
Mr. Peter P. Haglich

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Daniel H. Wagner Associates, Inc.
2 Eaton St., Suite 500
Hampton VA 23669

8. PERFORMING ORGANIZATION REPORT NUMBER
Case: 6230

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Commanding Officer
Coastal Systems Station
Cmr. Victor S. Newman, Code 2610)
6703 West Hwy 98
Panama City FL 32407-7001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT (see Section 5.3b of this solicitation)
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

Report developed under SBIR contract for N94-217. This report describes the proposed system, "Structured Essential Model for Mine Warfare," (SEMMIW). The report provides class diagrams, class specifications, object-scenario diagrams, and descriptions, all in the Booch notation. It also provides a mathematical foundation for predicting the expected number of remaining mines, given INTEL estimates and given the results of mine search operations.

14. SUBJECT TERMS	15. NUMBER OF PAGES
SBIR Report, Countermeasures, Environmental Effects, Booch Method, Object-Oriented Analysis	371
	16. PRICE CODE

17. SECURITY CLASSIFICTION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UNLIMITED

Case: 6230

STRUCTURED ESSENTIAL MODEL FOR MINE WARFARE

PHASE I FINAL REPORT

Contract Number N60921-95-C-0027

February 12, 1996

Prepared for

Coastal Systems Station
Naval SurfaceWarfare Center - Dahlgren Division

By

Dr. Joseph H. Discenza
Mr. Peter P. Haglich

Daniel H. Wagner Associates, Inc.
2 Eaton Street, Suite 500
Hampton, Virginia 23669

Table of Contents

List of Figures

Chapter 2 Diagrams

Class Category Optimization Specifications

Chapter 3 Diagrams

Legend

Use Case: Specify the Force

Scenario: Add New MCM-1 Ship
Scenario: Add New MHC-51 Ship
Scenario: Add New MH-53 Helicopter
Scenario: Add New EOD Detachment
Scenario: Add New Surface Minelayer
Scenario: Add New Air Minelayer
Scenario: Add New Submarine Minelayer
Scenario: Delete Asset From Force
Scenario: Edit MCM-1 Member of the Force
Scenario: Edit MHC-51 Member of the Force
Scenario: Edit MH-53E Member of the Force
Scenario: Edit EOD Detachment Member of the Force
Scenario: Edit Surface Minelayer Member of the Force
Scenario: Edit Air Minelayer of the Force
Scenario: Edit Submarine Minelayer Member of the Force

Use Case: Specify the Mission and MOE

Scenario: Specify Breakthrough Mission and MOE
Scenario: Specify Exploratory MCM Mission and MOE
Scenario: Specify Clearance Mission and MOE
Scenario: Specify Damage High Value Unit Mission and MOE
Scenario: Specify Anti-tonnage Mission and MOE

Use Case: Specify Mine Area

Scenario: Specify Mine Area

Use Case: Plan MCM Mission

Scenario: Plan Breakthrough Mission
Scenario: Plan Exploratory MCM Mission
Scenario: Plan Clearance Mission

Use Case: Specify Equipment Characteristics

Add a new side scan sonar
Add a New Surface Sonar
Add a New Air Sonar
Add a New Visual Sensor
Add a New Laser Sensor
Add a New Magnetometer
Add a New Metal Detector
Add a New Air-Towed Mechanical Sweep Gear

Add a New Surface-Towed Mechanical Sweep Gear
Add a New Air-Towed Influence Sweep Gear
Add a New Surface-Towed Influence Sweep Gear
Add a New MNV
Add a New UUV
Add a New ROV
Delete an MCM Equipment
Edit a Side Scan Sonar
Edit a Surface Sonar
Edit an Air Sonar
Edit a Visual Sensor
Edit a Laser Sensor
Edit a Magnetometer
Edit a Metal Detector
Edit an Air-Towed Mechanical Sweep Gear
Edit a Surface-Towed Mechanical Sweep Gear
Edit an Air-Towed Influence Sweep Gear
Edit a Surface-Towed Influence Sweep Gear
Edit a MNV
Edit an UUV
Edit a ROV

Use Case: Specify Equipment Settings

Scenario: Specify Equipment Settings

Use Case: Define Characteristics for C4I Systems

Scenario: Add a Receiving Node
Scenario: Add a Transmit Node
Scenario: Delete a Node
Scenario: Edit a Receiving Node
Scenario: Edit a Transmit Node

Use Case: Define Characteristics for Navigation Systems

Scenario: Define Characteristics for Commercial GPS
Scenario: Define Characteristics for Military GPS
Scenario: Define Characteristics for Differential GPS
Scenario: Define Characteristics for PINS
Scenario: Define Characteristics for Visual Navigation Systems
Scenario: Define Characteristics for Electronic Navigation Systems

Use Case: Specify Environmental Data

Scenario: Add Bathymetry Data for an Area
Scenario: Add Current Data
Scenario: Add Underwater Visibility Data

Scenario: Add SVP Data
Scenario: Add NOMBO Location Data
Scenario: Add NOMBO Density Data for an Area
Scenario: Add Wind Data
Scenario: Add Sea State Data
Scenario: Add Surface Visibility Data

Use Case: Determine MCM System Effectiveness

Scenario: Determine if Side Scan Sonar Detects Mine
Scenario: Determine if Surface Sonar Detects Mine
Scenario: Determine if Air Sonar Detects Mine
Scenario: Determine if UUV Detects Mine
Scenario: Determine if Influence Sweep is Effective
Scenario: Determine if Mechanical Sweep is Effective
Scenario: Determine if MNV Neutralizes Mine
Scenario: Determine if Diver Neutralizes Mine

Use Case: Specify Mine Characteristics

Scenario: Edit a Mine Type
Scenario: Delete a Mine Type
Scenario: Specify Mine Settings

Use Case: Plan a Minefield

Scenario: Plan a Minefield

Use Case: Determine Mine Effectiveness

Scenario: Create a Non-Firing Actuation Event.
Scenario: Determine If Mine Fires

Use Case: Specify Mine Distribution

Scenario: Build Mine Distribution Prior
Scenario: Update Mine Distribution Based on Found Mine
Scenario: Update Mine Distribution Based on Search
Scenario: Update Mine Distribution Based on Sweeping

Chapter 1. Introduction and Executive Summary

This document comprises the Phase I Final Report for the Structured Essential Model for Mine Warfare contract number N60921-95-C-0027. The overall goal of this project is to define, prototype, and develop a comprehensive data system to support all phases of mine warfare (MIW). In this first chapter we describe the Phase I effort and summarize the major findings from Phase I. We also describe the Booch method and notation. In Chapter 2 we present the SEMMIW class diagrams and class specifications. In Chapter 3 we present the SEMMIW object-scenario diagrams and descriptions. In Chapter 4 we present some mathematical findings on estimating the number of mines remaining in an area after MCM efforts have been conducted. In Appendix A we present an example of the operational benefits that can be realized by using mine distributions in MCM planning. In Appendix B we provide C++ header code for a small sample of the classes defined in Phase I. C++ header code for all of the classes defined in Phase I has been provided to the COTR.

1.1. Background

Mines are the most dangerous threat facing naval expeditionary forces in littoral warfare, according to Major General Harry Jenkins, USMC, the Director for Expeditionary Warfare on the staff of the Chief of Naval Operations (OPNAV N85). Mines are likely to be a problem in many of the potential conflicts in which naval forces will be involved in the naval "From the Sea" doctrine. In these scenarios the mine threat would significantly complicate maneuver from the sea and power projection ashore by expeditionary forces. For this reason, mine warfare (MIW) has become one of the most important areas of naval research and development. Development of intelligence systems, C^4I systems, precision localization systems, and MIW equipment has received increased emphasis.

Unfortunately, the mine warfare development initiatives are spread across a number of laboratories and organizations. With this diversification it is difficult to measure the relative contribution of various development systems to operational effectiveness in littoral warfare scenarios. This Final Report presents the first step in the development of an abstract, systems-independent, detailed structured essential model of Mine Warfare called "Structured Essential Model for Mine Warfare (SEMMIW)."

Benefits of the System. SEMMIW will provide many benefits to the sponsor and the Government. SEMMIW will be used to determine quantitative requirements for the MIW areas of mine countermeasures (MCM) and mining. The quantitative requirements can then

be mapped to candidate solutions for meeting littoral warfare needs for the purpose of evaluating among these candidate solutions. In particular, the appropriate emphasis of the synergistic contributions of intelligence, C^4I, precision localization, and MIW equipments can be assessed independently.

This innovative capability is especially valuable in an atmosphere of reduced defense spending. SEMMIW will provide a means to determine the contribution to overall warfighting effectiveness of new MIW-related systems. In particular, SEMMIW will be capable of providing resource sponsors and decision makers with the information to make program decisions such as choosing to develop MCM sensor A instead of MCM navigation system B. A maxim of naval R&D is that the last 20% improvement in performance typically costs as least 50% of the program costs. It is typically difficult to assess the impact of this 10% increase in performance on overall warfare effectiveness. If the marginal gain in warfare effectiveness is small, then a great deal of money may be spent which could be better used in other programs. On the other hand, if the decision is made to accept reduced performance to save money, but it turns out that the marginal gain in warfighting effectiveness would have been great, then we may not be able to meet military needs. SEMMIW will provide background to resource sponsors to make correct decision of this type.

1.2. Phase I Objectives

The Phase I effort had the following technical objectives:

1. *Complete Requirements Analysis*. Develop a system charter and system function statement.

2. *Complete Domain Analysis*. Identify key classes and their relationships. Define the operations associated with SEMMIW use cases. Assign attributes to classes. Define inheritance among classes. Validated the domain model. Develop complete class diagrams, class specifications, object-scenario diagrams, and object specifications.

Both of these technical objectives were met in Phase I and are described in this report.

1.3. Booch Method of Object Oriented Analysis and Design as Applied to SEMMIW

System Development Using Booch Method. Our development approach is based on the Booch method of software development and uses the Computer Aided Software Engineering

(CASE) tool Rational Rose. The Booch method is an object-oriented software-development process that provides a model to support solid analysis and design of a software intensive system. A system developed using an object-oriented method has the following advantages over systems developed using more traditional methods:

- The software system is easily changed as the real-world situation changes. For example, the model for a new MIW equipment can be developed as an object and incorporated into the model easily.
- Software units fit the development environment rather than a specific computer system and therefore can be easily ported to other computer systems. In particular, the units developed on a Windows PC can be ported to a UNIX workstation and vice versa.
- The software system is easily understood and easily maintained by users, developers, and maintainers.

This report presents a comprehensive set of models in the Booch notation. In Phase II we will directly implement these models in units of working code, or executable releases. Each executable release is a true subset of the final working code. Each portion of the Booch method (diagram, specification, or executable release) is a view of an underlying fully-integrated model of the system as it finally will be implemented.

The fundamental sub-processes in the Booch method are *Requirements Analysis*, *Domain Analysis*, and *System Design*. Requirements analysis is the process of determining what the system is to do and is performed by systems analysts in conjunction with end users and domain experts. The requirements analysis phase consisted of developing a charter for the structured essential MIW model and a statement of system function. Domain analysis provides the key logical structure for the system and defines an object-oriented model of the real-world system that is to be reflected in the software system. The domain analysis phase identifies the major MIW classes such as mission objectives, MIW systems, and MIW tactics as well as the relationships between them. The products of the domain analysis phase include class diagrams, class specifications, and object-scenario diagrams. System design implements the logical structure developed in domain analysis in a specific computer system architecture.

The CASE tool that we are using in this research and development effort, Rational Rose, automates the Booch method. Rational Rose can be used in forward-engineering to produce

C++ or Ada outline and header code. Rational Rose also incorporates reverse-engineering capabilities to support iterative development. During the domain analysis phase Rational Rose was used to produce the following system deliverables:

- A complete class diagram which illustrates the major classes and relationships.
- A class-category diagram showing the categories to be used.
- Class specifications for all classes defining the class, its relationships, its attributes, and the interface of its operations.
- Object-scenario diagrams showing the use of operations in the use cases of the system.

During the system design phase Rational Rose will be used to produce the following:

- A description of the architectural services of the system.
- Updated class diagrams showing the implementation of relationships, additional operations, and access control.
- Updated class specifications showing operation specifications.
- Design object-scenario diagrams for nontrivial operations.

The Booch Notation. Here we will describe the major aspects of the Booch notation as used in this report. Examples are shown in the Legends for Chapters 2 and 3. We adopt the definitions given by Booch in [1].

Objects. An *object* has state, behavior, and identity. Typically, an object models some part of the real-world domain. The Booch symbol for an object is a cloud with the name of the object.

Classes. A *class* defines a structure for objects of that class. The class defines the structure and behavior for the objects. In this sense the terms instance and object are interchangeable. The Booch symbol for a class is a cloud with the name of the class and with a dotted boundary.

Class Categories. A class category is a cluster of classes that is internally cohesive and loosely coupled relative to other classes. The Booch symbol for a class category is a rectangle with a solid boundary and the name of the class category inside.

Association. An association relationship is a general relationship that shows some semantic dependency between two classes. The symbol for an association relationship is a solid line connecting the two classes.

Inheritance. An inheritance relationship occurs when one class, known as the subclass, inherits structure and behavior from a more generic class, known as the superclass. An inheritance relationship is also known as an "is a" relationship. The symbol for an inheritance relationship is an arrowed line pointing from the subclass to the superclass.

Aggregation. An aggregation relationship occurs when one class contains another class as part of it. An aggregation relationship is also known as a "has" relationship. The symbol for an aggregation relationship is a line connecting the aggregate to the member, with a solid circle on the aggregate.

Using. A using relationship occurs when one class accesses the structure or operations of another class. The symbol for a "using" relationship is a line connecting the using class to the class being used, with an open circle on the user class.

Links. A link denotes the specific association through which one object applies the services of another object or through which one object may navigate to another. The symbol for a link is a solid line connecting the objects. The link may be adorned with arrows showing the directions of message flow.

1.4. Requirements Analysis Results

1.4.1. System Charter

The Structured Essential Model for Mine Warfare (SEMMIW) will be able to:

- Model the operational effectiveness of mine countermeasures forces in littoral warfare scenarios.
- Model the operational effectiveness of mining missions in littoral warfare scenarios.
- Compute operational measures of effectiveness (MOEs) for mine warfare missions using specified systems and tactics in a specific environment against a specific threat.

- Provide quantitative results that support the determination of the contributions of intelligence, C^4I, precision localization, MCM equipment, and mining equipment to mine warfare operational effectiveness.

1.4.2. System Function Statement

The Structured Essential Model for Mine Warfare (SEMMIW) will have three types of users. These are the commander of mine warfare forces (Commander), the mine countermeasures system and tactics analyst (MCM Analyst) and the mining systems and tactics analyst (Mining Analyst). The use cases for each user type are specified below:

The Commander will use SEMMIW to:

- Specify the mine warfare force by adding, deleting, or modifying the assets in the force.
- Specify the mine warfare mission and the measure of effectiveness (MOE).
- Specify the minefield areas of interest by adding, deleting, or modifying mine danger areas.
- Compute the predicted value of the MOE and related statistics.
- Assign operating areas to assets in the force.

The MCM Analyst will use SEMMIW to:

- Define the characteristics for MCM equipment.
- Specify the settings for MCM equipment in use.
- Define the characteristics for C^4I systems in use.
- Define the characteristics for navigation and precision localization systems in use by MCM assets.
- Specify environmental and oceanographic parameters for the area of interest.
- Determine whether a minehunting sensor detects a mine.
- Determine whether minesweeping gear is effective.
- Determine whether a mine has been neutralized.
- Determine the expected number of undetected mines in the mine danger areas.

- Determine the expected time to complete the MCM mission.

The Mining Analyst will use SEMMIW to:

- Define the characteristics for threat mines.
- Specify the settings for threat mines.
- Define the characteristics for navigation and precision delivery systems in use by minelayers.
- Plan the placement of minefields.
- Plan the composition of minefields.
- Determine whether a mine is actuated by a ship.
- Determine whether a mine causes damage to a ship.

1.5. Top Level Logical Structure

The top level class diagram for SEMMIW is shown in Figure 1. In our analysis of the mine warfare domain we identified the following key class categories:

- Command
- Forces
- MIW Mission
- Mine Danger Area
- Planning/Optimization
- Platforms
- Mining Analysis
- MIW MOE
- Navigation
- MCM Equipment
- Threat
- MCM Analysis
- MIW Model

- MOE Statistics
- C4I
- Oceanography
- User Interface
- Geography

In Chapter 2 we show the class diagrams for each of the class categories and provide the specifications for the classes, relationships, and operations. In Chapter 3 we show the object-scenario diagrams and provide descriptions of them.

1.6. Estimated Number of Mines Remaining

An ancillary objective of the Phase I effort was to provide a mathematical formulation for the notion of a mine distribution. In order to do this one must have some estimate of the number of mines in the area of interest. The quality of this estimate is based on the quality of the prior intelligence concerning the number of mines in the area. The estimated number of remaining mines can then be updated based on the number of mines detected or swept and the effectiveness of the search/sweep effort. Chapter 4 provides the mathematical details for different conditions of intelligence quality. In Appendix A we provide sample calculations to show how accounting for mine distributions can provide a 53% reduction in the time to verify a lane is clear in an example.

1.7. Future SEMMIW Development

We outline the expected progress of continued SEMMIW development.

1.7.1. Phase I Option Period

In the Phase I Option Period we will develop a prototype SEMMIW that will demonstrate the idea of the mine distribution and updates to it from intelligence, detection or sweeping of mines, and minehunting or minesweeping effort. The prototype will be based on our commercial MELIAN II underwater search and salvage system and will run on a portable UNIX system.

1.7.2. Phase II

In the Phase II effort we will develop a fully functional SEMMIW prototype. To accomplish this we will complete the SEMMIW design based on the logical model completed in Phase I. We will fully specify the structure and behavior of all SEMMIW

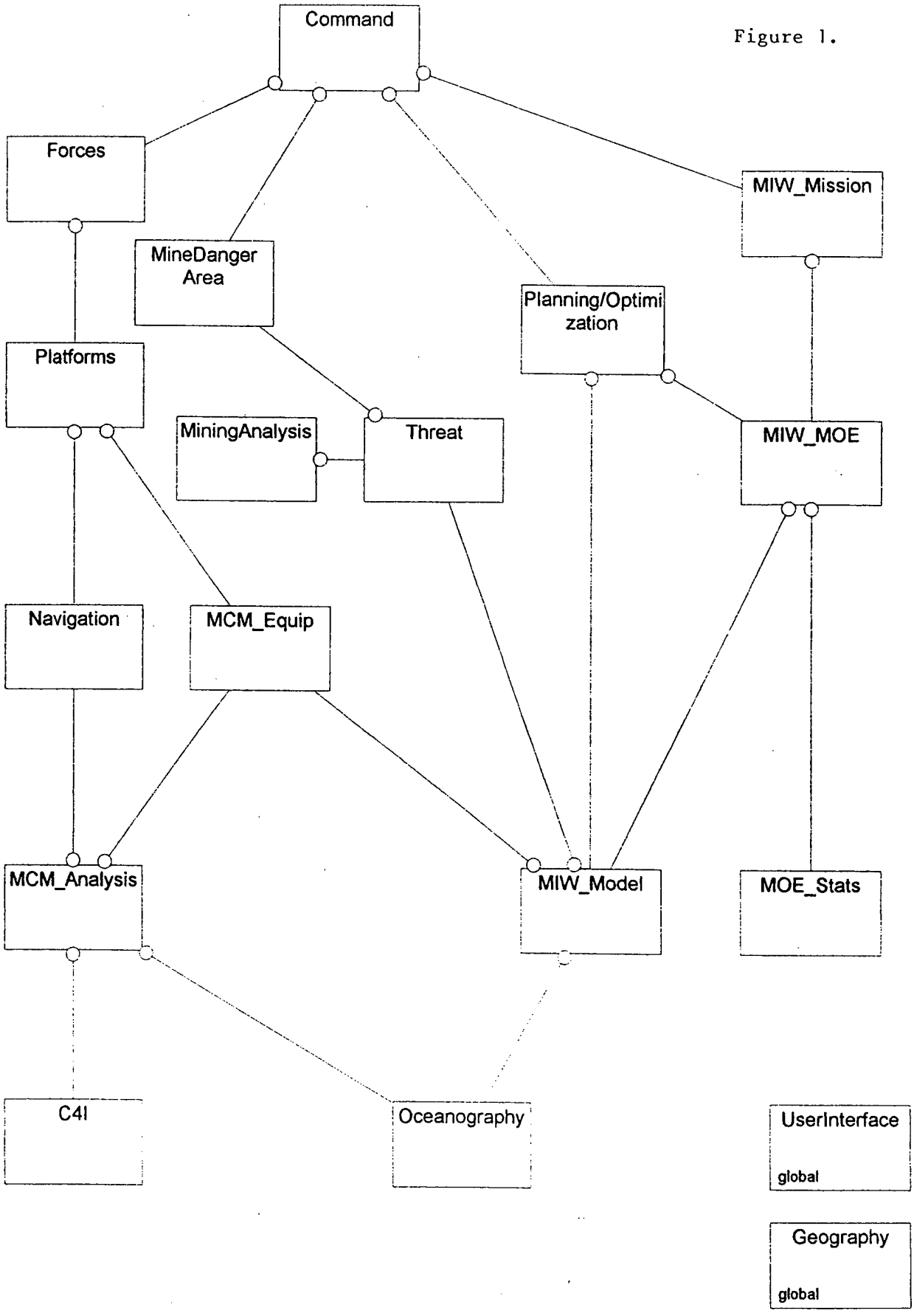
classes. We will develop the algorithms to implement the operations defined in the class specifications.

The Phase II prototype will be delivered to the Coastal Systems Station for further test and evaluation. We will also demonstrate the Phase II prototype to Commander, Mine Warfare Command, and other members of the mine warfare community.

1.7.3. Phase III

In the Phase III effort we will develop deployable versions of SEMMIW for use by mine warfare commanders and analysts. We expect that there will be two major SEMMIW variants, a tactical variant and an analytical variant. These variants will both be based on the Phase II SEMMIW prototype and will have many core functions in common with the Phase II prototype.

Figure 1.



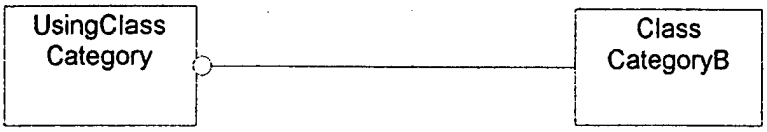
File: c:\rose\semmiw\semmiw.mdl Sun Nov 26 09:28:27 1995 Class Diagram: <Top Level> / Main

Chapter 2: Class Diagrams and Specifications

In this chapter we present the top level class category diagram, class diagrams for all SEMMIW class categories, and specifications for all SEMMIW classes and the operations that are presently defined for those classes. The specifications are listed immediately following the class diagram for each class category.

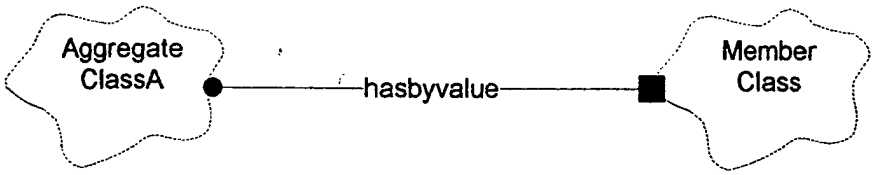
Legend for Class Diagrams in the Booch notation:

A. Class categories with a "using" relationship.



B. Aggregation By Value

1. The solid circle is on the aggregate's end.
2. The solid box on the member's end indicates containment "by value", meaning the member has no existence independent of the aggregate.



C. Aggregation by Reference

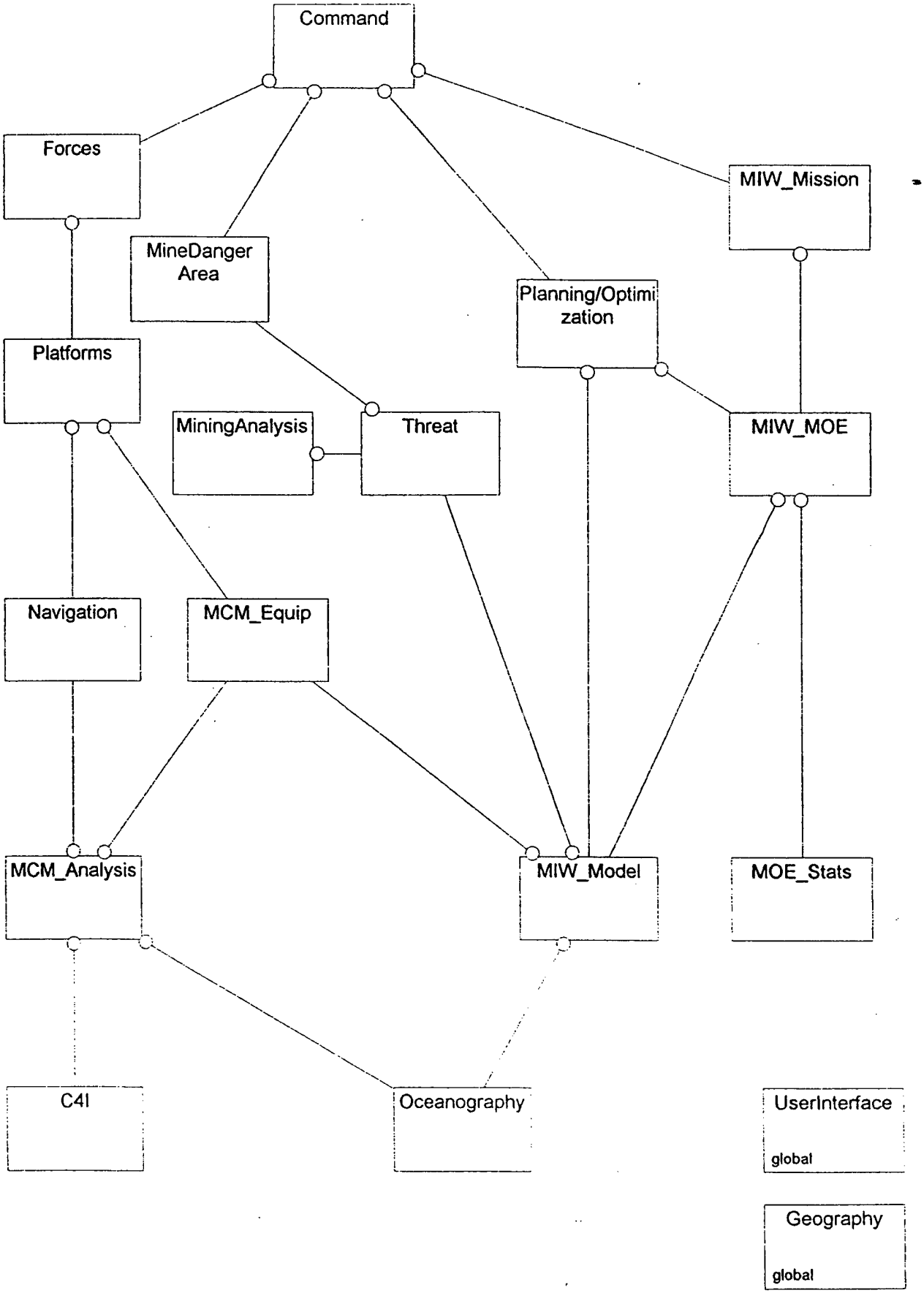
1. The solid circle is on the aggregate's end.
2. The open box on the member's end indicates containment "by reference", indicating the member has significance independent of the aggregate.

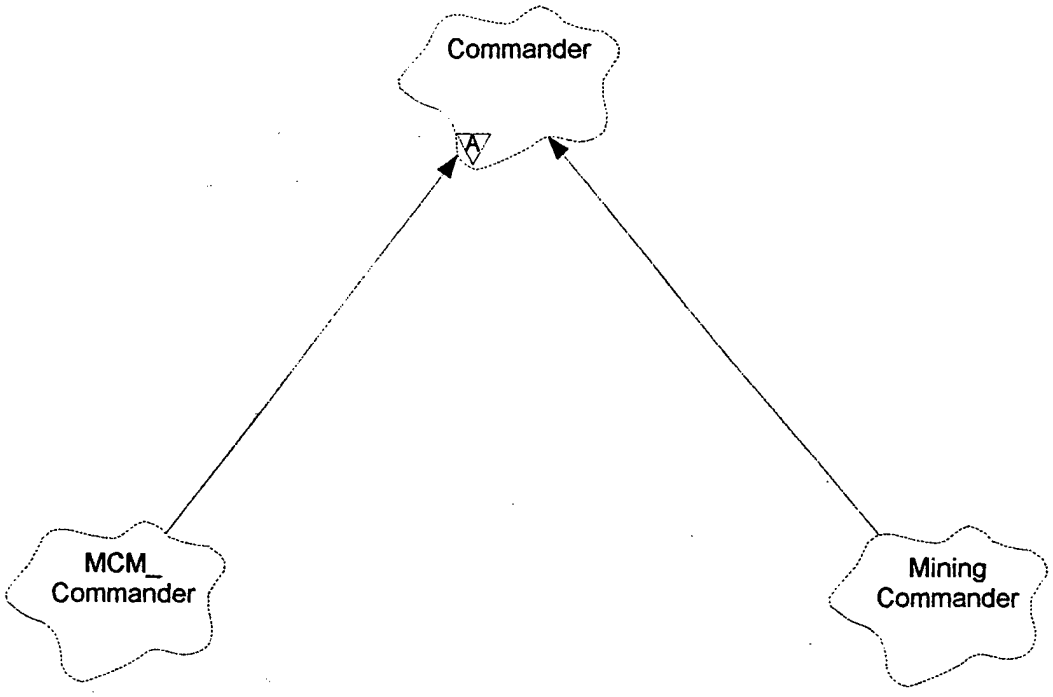
D. Using Relationship Between Classes.

The client class "uses" the server class.

E. Inheritance

The subclass inherits structure and behavior from the superclass but is more specialized.


Command
Forces
MIW_Mission
MineDanger Area
Planning/Optimi zation
Platforms
MiningAnalysis
Threat
MIW_MOE
Navigation
MCM_Equip
MCM_Analysis
MIW_Model
MOE_Stats
C4I
Oceanography
UserInterface
global
Geography
global


Commander
MCM_
Commander
Mining
Commander

Class name:

Commander

Category: Command
Documentation:
Surrogate for the commander of forces

Export Control: Public
Cardinality: 1
Hierarchy:
Superclasses: none
Public Uses:

Force	commands
Mission	orders

Public Interface:
Has-A Relationships:

Mildesig	Designator

Unique identifier (e.g. CTU 29.2.3) for the commander

State machine: No
Concurrency: Sequential
Persistence: Persistent

Class name:

MCM_Commander

Category: Command
Documentation:
Surrogate for the Commander of MCM Forces

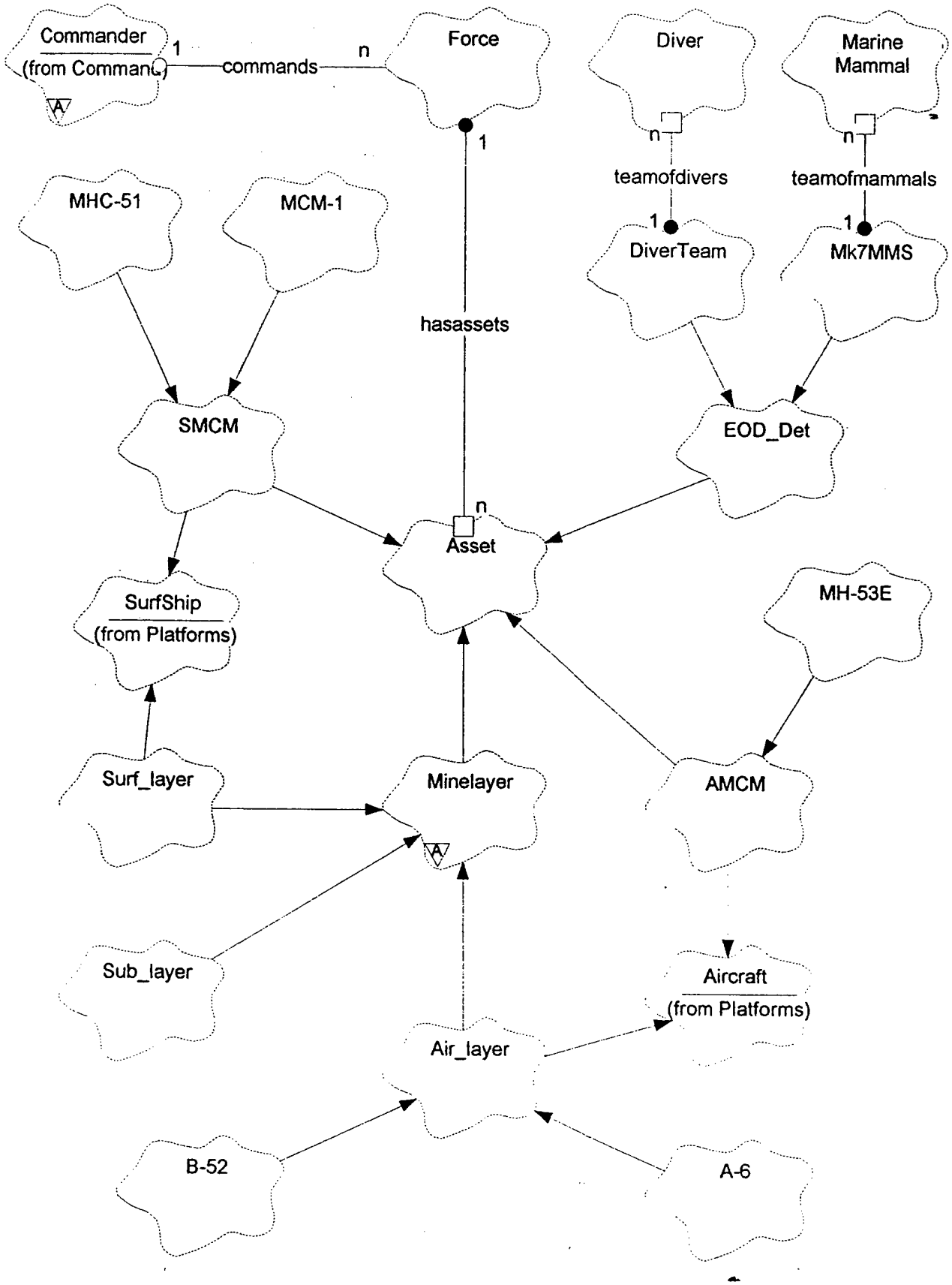
Export Control: Public
Cardinality: 1
Hierarchy:
Superclasses: Commander
State machine: No
Concurrency: Sequential
Persistence: Persistent

Class name:

MiningCommander

Category: Command
Documentation:
Surrogate for the commander of mining forces

Export Control:	Public
Cardinality:	1
Hierarchy:	
Superclasses:	Commander
State machine:	No
Concurrency:	Sequential
Persistence:	Persistent



File: c:\rose\semmiw\semmiw.mdl Wed Nov 22 10:03:53 1995 Class Diagram: Forces / Main

Class name:

DiverTeam

Category: Forces
Documentation:
Represents a team of EOD divers

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: EOD_Det
Public Interface:
Has-A Relationships:

Integer Divercount
Number of divers in the team.

Diver teamofdivers

Operations:

add_a_diver ( )
subtract_a_diver ( )

State machine: No
Concurrency: Sequential
Persistence: Transient

Operation name:

add_a_diver

Public member of: DiverTeam
Return Class: Integer
Documentation:
Add 1 to the diver count

Semantics:
Divercount = Divercount + 1

Object diagram: (Unspecified)

Concurrency: Sequential

Operation name:

subtract_a_diver

Public member of: DiverTeam
Return Class: Integer
Documentation:
Decrement the number of divers in the team by one.

Semantics:
divercount = divercount - 1

Object diagram: (Unspecified)

Concurrency: Sequential

Class name:

Force

Category: Forces
Documentation:
Represents the Mine Warfare military force

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:
Asset hasassets

Protected Interface:
Has-A Relationships:
Mildesig Designator
The military designation of the particular mine warfare force.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Asset

Category: Forces
Documentation:
Represents the mine warfare military asset

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Protected Interface:
Has-A Relationships:

Mildesig assetdesig
The military designation of the asset

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

SMCM

Category: Forces
Documentation:
Class of surface mine-countermeasures ships

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Asset, SurfShip
Public Interface:
Has-A Relationships:
float shockhardness
Maximum shock factor the SMCM ship is certified for.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

MCM-1

Category: Forces
Documentation:
Class of MCM-1 Mine Countermeasures Ships and sisters

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: SMCM
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

MHC-51

Category: Forces
Documentation:
Class of MHC-51 coastal mine countermeasures craft and sister ships.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: SMCM
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:
EOD_Det

Category: Forces
Documentation:
Represents an EOD MCM detachment.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Asset
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:
Mk7MMS

Category: Forces
Documentation:
A team of mammals that is trained in EOD operations.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: EOD_Det
Public Interface:
Has-A Relationships:
integer Mammalcount
The number of mammals in the team

MarineMammal teamofmammals

Operations:

addamammal ( )
subtractamammal ( )

State machine: No
Concurrency: Sequential
Persistence: Transient

Operation name:

addamammal

Public member of: Mk7MMS
Documentation:
Increment the mammal count by 1.

Concurrency: Sequential

Operation name:

subtractamammal

Public member of: Mk7MMS
Documentation:
Decrement the mammal count by one

Concurrency: Sequential

Class name:

Minelayer

Category: Forces
Documentation:
Represents the abstraction of a minelaying vessel.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Asset
Public Interface:
Has-A Relationships:

integer Minecapacity
The number of mines that the minelayer can carry

State machine: No
Concurrency: Sequential

Persistence: Transient

Class name:

Surf_layer

Category: Forces
Documentation:
Represents a surface ship minelayer.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Minelayer, SurfShip
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Sub_layer

Category: Forces
Documentation:
Represents a submarine minelayer.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Minelayer
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Air_layer

Category: Forces
Documentation:
Represents an aircraft minelayer.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Minelayer, Aircraft
State machine: No
Concurrency: Sequential

Persistence: Transient

Class name:

B-52

Category: Forces
Documentation:
Represents a B-52 in a minelaying role.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Air_layer
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

A-6

Category: Forces
Documentation:
Represents an A-6 in a minelaying role.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Air_layer
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Commander

Category: Command
Documentation:
Surrogate for the commander of forces

Export Control: Public
Cardinality: 1
Hierarchy:
Superclasses: none
Public Uses:
Force commands

Mission orders

Public Interface:

Has-A Relationships:

Mildesig Designator
Unique identifier (e.g. CTU 29.2.3) for the commander

State machine: No
Concurrency: Sequential
Persistence: Persistent

Class name:

Diver

Category: Forces
Documentation:

Represents an EOD diver

Export Control: Public
Cardinality: n
Hierarchy:

Superclasses: none

Public Interface:

Has-A Relationships:

Integer Cycletime
Allowable time at depth.

Float Maxswimspeed
Maximum speed the diver can swim, usually ranges between 0 and 2 knots.

Operations:

adjcycletime (integer)
adjmaxswimspd (float)

State machine: No
Concurrency: Sequential
Persistence: Transient

Operation name:

adjcycletime

Public member of: Diver
Arguments:

integer cycleadjtime

Documentation:
Time in minutes the cycle time is to be adjusted by. Can be positive or negative.

Concurrency: Sequential

Operation name:
adjmaxswimspd

Public member of: Diver
Arguments:
float Speedadj
Documentation:
Adjust swim speed by the specified amount

Concurrency: Sequential

Class name:
MarineMammal

Category: Forces
Documentation:
A marine mammal trained in EOD operations.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:
integer endurance
Time the mammal can operate, measured in minutes.

integer maxdepth

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:
SurfShip

Category: Platforms
Documentation:
Represents a vessel that travels on the surface of the water.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Ship
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

AMCM

Category: Forces
Documentation:
Represents airborne mine-countermeasures assets.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Asset, Aircraft
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

MH-53E

Category: Forces
Documentation:
Represents the MH-53E airborne mine countermeasures helicopter.

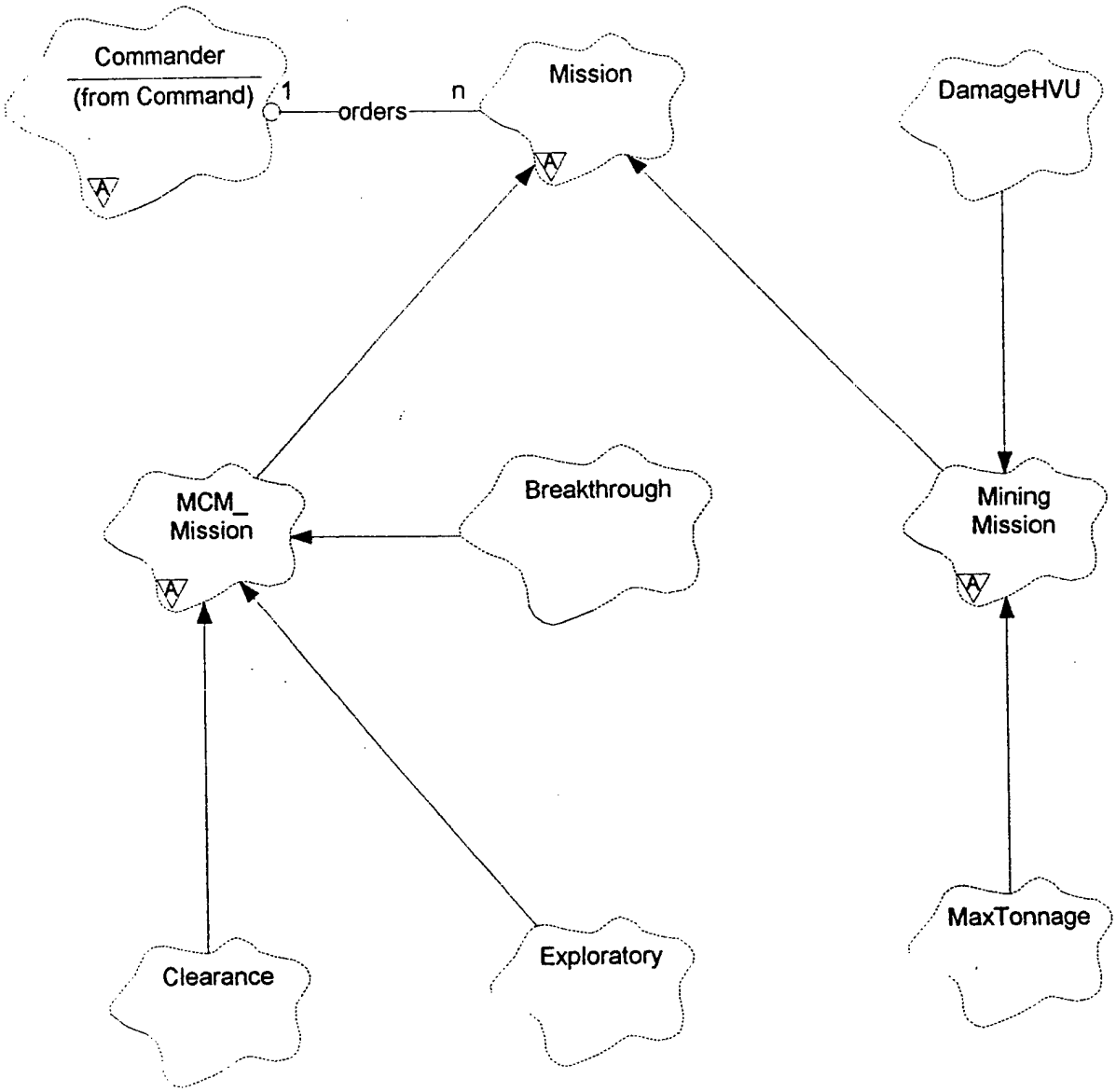
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: AMCM
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Aircraft

Category: Platforms
Documentation:
Represents an airplane or helicopter.

Export Control:	Public	
Cardinality:	n	
Hierarchy:		
Superclasses:	Platform	
Public Interface:		
Has-A Relationships:		
	int	Altitude
State machine:	No	
Concurrency:	Sequential	
Persistence:	Transient	


Commander
(from Command)
1
orders
n
Mission
DamageHVU
MCM_
Mission
Breakthrough
Mining
Mission
Clearance
Exploratory
MaxTonnage

Class name:

Mission

Category: MIW_Mission
Documentation:
Represents the abstraction of a military mission.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

MCM_Mission

Category: MIW_Mission
Documentation:
Represents the abstraction of a mine countermeasures mission.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Mission
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

MiningMission

Category: MIW_Mission
Documentation:
Represents the abstraction of a mining mission.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Mission
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

DamageHVU

Category: MIW_Mission
Documentation:
Represents a mininig mission to damage high value units.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: MiningMission
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

MaxTonnage

Category: MIW_Mission
Documentation:
Represents a mining mission where the objective is to maximize the amount of tonnage sunk.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: MiningMission
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Clearance

Category: MIW_Mission
Documentation:
Represents a mission to clear an area of mines.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: MCM_Mission
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Exploratory

Category: MIW_Mission
Documentation:
Represents a MCM mission that is exploratory, i.e. to determine if an area has been mined.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: MCM_Mission
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Breakthrough

Category: MIW_Mission
Documentation:
Represents a breakthrough MCM mission.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: MCM_Mission
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Commander

Category: Command
Documentation:
Surrogate for the commander of forces

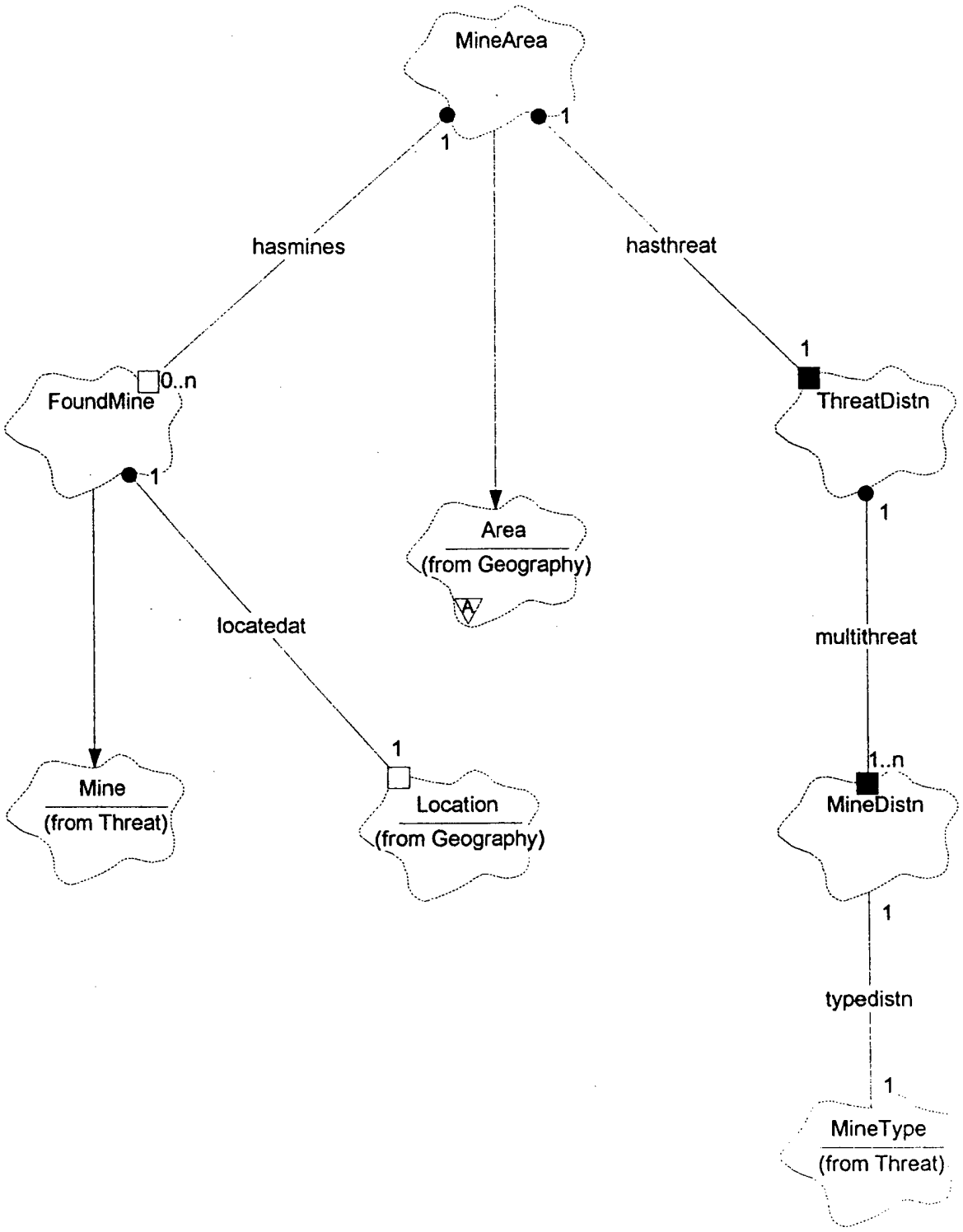
Export Control: Public
Cardinality: 1
Hierarchy:
Superclasses: none
Public Uses:

Force	commands
Mission	orders

Public Interface:
Has-A Relationships:

Mildesig Designator
Unique identifier (e.g. CTU 29.2.3) for the commander

State machine: No
Concurrency: Sequential
Persistence: Persistent


MineArea
1
1
hasmines
hasthreat
1
0..n
FoundMine
ThreatDistn
1
1
Area
(from Geography)
locatedat
multithreat
Mine
(from Threat)
1
Location
(from Geography)
1..n
MineDistn
1
typedistn
1
MineType
(from Threat)

Class name:

MineArea

Category: MineDangerArea
Documentation:
Represents an area that either is known to be mined or is suspected of being mined.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Area
Public Interface:
Has-A Relationships:

FoundMine hasmines
A mine area will have some (0 to N) found mines in it.

ThreatDistn hasthreat
A mine area has a composite mine threat distribution associated with it.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

FoundMine

Category: MineDangerArea
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Mine
Public Interface:
Has-A Relationships:

Location locatedat
A mine that has been found is at some geographical location.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

ThreatDistn

Category: MineDangerArea
Documentation:
Represents the composite distribution for threat mines in the mine area.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:

MineDistn multithreat
Each threat distribution consists of a composite threat distribution for multiple minetypes.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

MineDistn

Category: MineDangerArea
Documentation:
Represents the two dimensional probability distribution for mines of the specified type.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Associations:

MineType typedistn
Links the distribution for the mine type with the mine type.

State machine: No
Concurrency: Sequential
Persistence: Transient
Each threat distribution consists of a composite threat distribution for multiple minetypes.

A mine area will have some (0 to N) found mines in it.

A mine area has a composite mine threat distribution associated with it.

Class name:

Area

Category: Geography
Documentation:
Represents a geographic area.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:
Location hasIocations
Relates an area to locations contained in it.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Mine

Category: Threat
Documentation:
Represents a naval mine.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:
MineType hastype
Relates a mine to its type.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Location

Category: Geography
Documentation:
Represents a physical location on the globe.

Export Control: Public

Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:

int depth
Depth of the location, referenced to sea level.

lattype latitude
The latitude of the location

longtype longitude
The longitude of the location.

State machine: No
Concurrency: Sequential
Persistence: Transient
A mine that has been found is at some geographical location.

Class name:

MineType

Category: Threat
Documentation:
Template for a given type of mine

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:

ArmingMechanism armedby
Relates the mine to its arming mechanism.

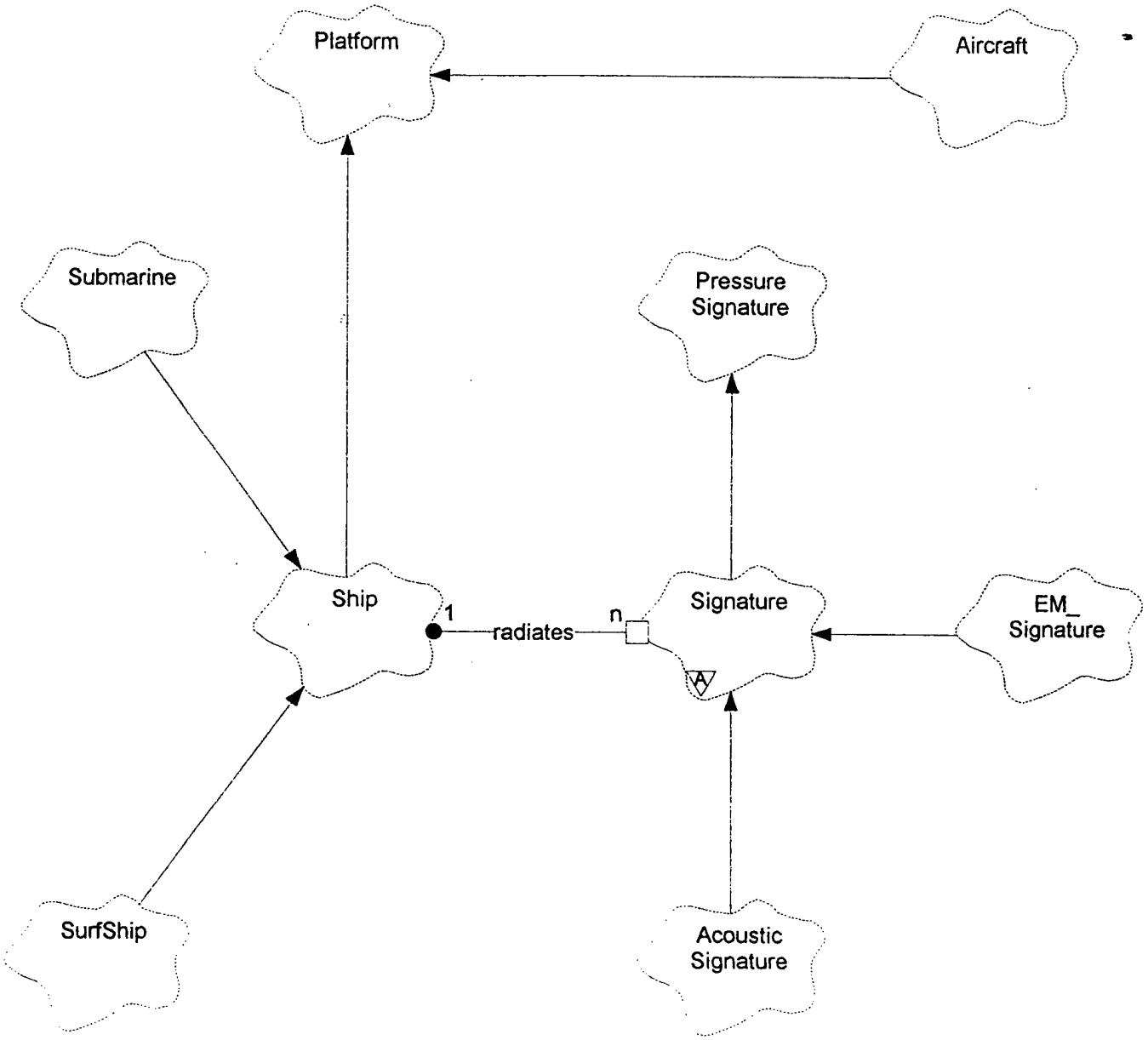
Explosive detonates
Relates mine type to its explosive.

ActuatingMechanism firedby
Relates the mine type to the actuating mechanism.

SterilizingMechanism hassteril
Relates mine type to the mechanism that sterilizes it.

char minetypename
The name of the mine type.

State machine: No
Concurrency: Sequential
Persistence: Transient
Links the distribution for the mine type with the mine type.


Platform
Aircraft
Submarine
Pressure Signature
Ship
1
radiates
n
Signature
EM_ Signature
SurfShip
Acoustic Signature

Class name:

Platform

Category: Platforms
Documentation:
Represents a moving thing such as a ship or aircraft.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:

Equipment carries
Represents the association between a platform and the equipment it carries.

NavigationSystem navigates

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Aircraft

Category: Platforms
Documentation:
Represents an airplane or helicopter.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Platform
Public Interface:
Has-A Relationships:

int Altitude

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Ship

Category: Platforms
Documentation:
Represents a vessel that travels on or below the surface of the water.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Platform
Public Interface:
Has-A Relationships:
Signature radiates

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:
SurfShip

Category: Platforms
Documentation:
Represents a vessel that travels on the surface of the water.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Ship
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:
Submarine

Category: Platforms
Documentation:
Represents a submarine vessel.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Ship
Public Interface:
Has-A Relationships:
int Depth

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Signature

Category: Platforms
Documentation:
Represents the radiation pattern of a ship.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: PressureSignature
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

PressureSignature

Category: Platforms
Documentation:
Represents the radiated pressure waves caused by a ship's motion.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

EM_Signature

Category: Platforms
Documentation:
Represents the electromagnetic fields generated by the motion of a ship and the operation of its machinery.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Signature
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

AcousticSignature

Category: Platforms
Documentation:
Represents the radiated noise caused by the motion of a ship or the operation of its machinery.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Signature
State machine: No
Concurrency: Sequential
Persistence: Transient

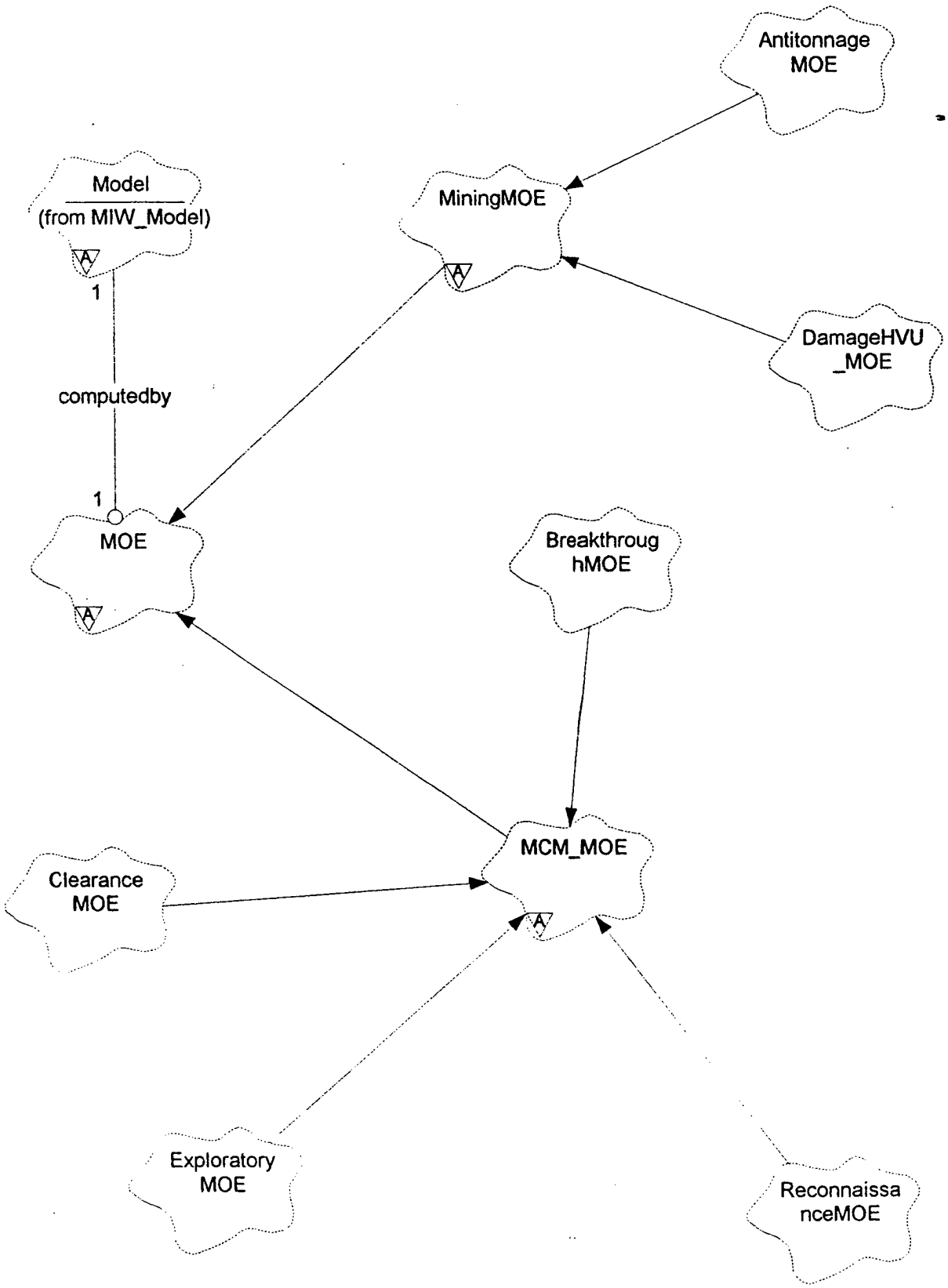

Mining
Analyst

Class name:

MiningAnalyst

Category: MiningAnalysis
Documentation:
Surrogate for the analyst concerned with minefield and mine effectiveness.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
State machine: No
Concurrency: Sequential
Persistence: Transient


Antitonnage MOE
Model
(from MIW_Model)
MiningMOE
1
computedby
1
DamageHVU _MOE
MOE
Breakthroug hMOE
MCM_MOE
Clearance MOE
Exploratory MOE
Reconnaissa nceMOE

Class name:

MOE

Category: MIW_MOE
Documentation:
Represents a measure of effectiveness.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Uses:
Model computedby

Public Interface:
Has-A Relationships:
float MOE_value
Represents the value of the MOE.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

MiningMOE

Category: MIW_MOE
Documentation:
Represents MOEs for mining.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: MOE
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

AntitonnageMOE

Category: MIW_MOE
Documentation:
Represents MOEs for antitonnage missions.

Export Control: Public

Cardinality: n
Hierarchy:
Superclasses: MiningMOE
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

DamageHVU_MOE

Category: MIW_MOE
Documentation:
Represents MOEs for missions against high value units.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: MiningMOE
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

MCM_MOE

Category: MIW_MOE
Documentation:
Represents minecountermeasures measures of effectiveness.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: MOE
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

BreakthroughMOE

Category: MIW_MOE
Documentation:
Represents MOEs for breakthrough MCM operations.

Export Control: Public

Cardinality: n
Hierarchy:
Superclasses: MCM_MOE
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

ClearanceMOE

Category: MIW_MOE
Documentation:
Represents MOEs for MCM Clearance operations.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: MCM_MOE
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

ExploratoryMOE

Category: MIW_MOE
Documentation:
Represents MOEs for exploratory MCM missions.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: MCM_MOE
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

ReconnaissanceMOE

Category: MIW_MOE
Documentation:
Represents MOEs for MCM reconnaissance missions.

Export Control: Public

Cardinality: n
Hierarchy:
Superclasses: MCM_MOE
State machine: No
Concurrency: Sequential
Persistence: Transient

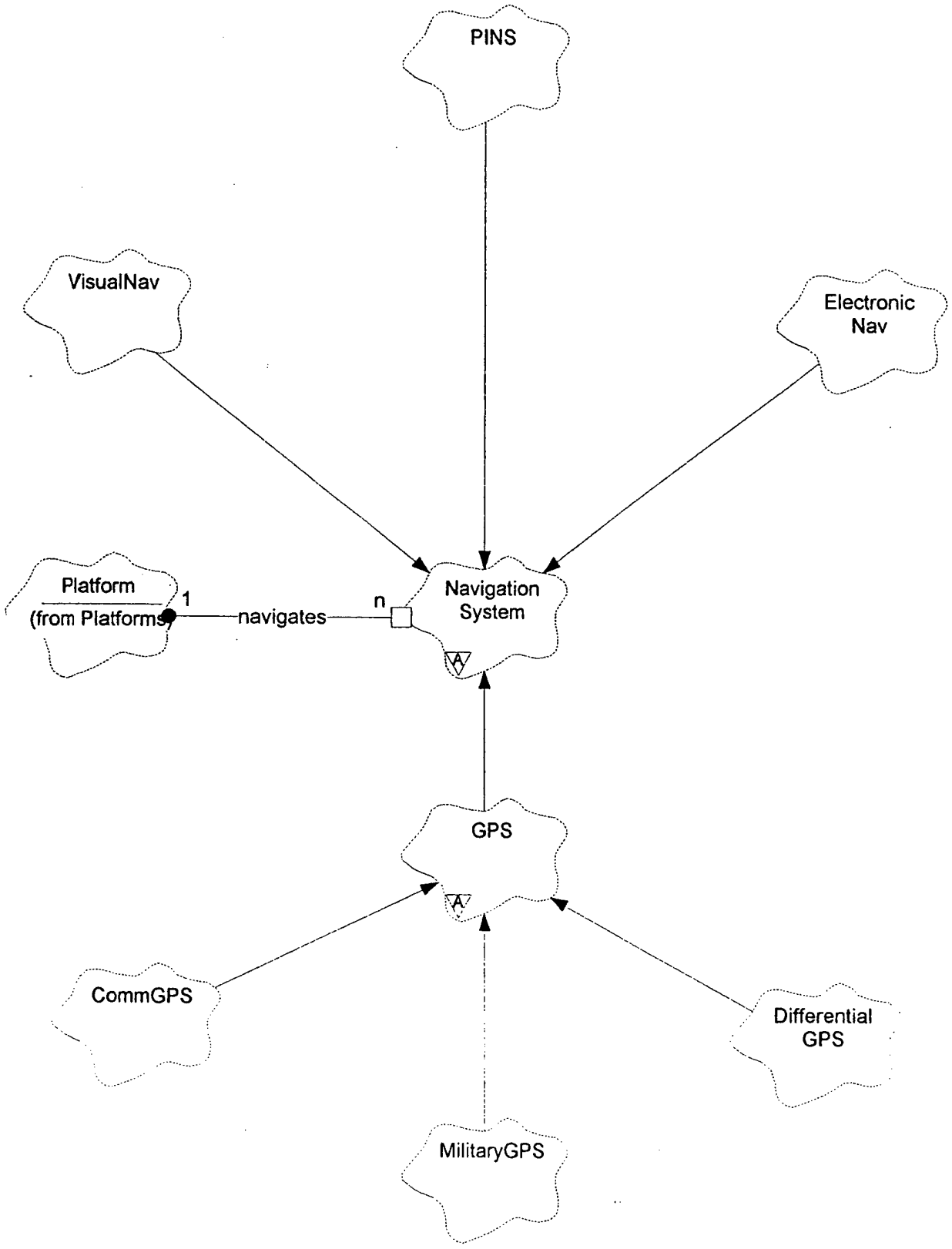
Class name:

Model

Category: MIW_Model
Documentation:
Represents computational models for predicting events or MOE values.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Uses:
Event creates

State machine: No
Concurrency: Sequential
Persistence: Transient


PINS
VisualNav
Electronic Nav
Platform
(from Platforms)
1
navigates
n
Navigation System
GPS
CommGPS
Differential GPS
MilitaryGPS

Class name:

NavigationSystem

Category: Navigation
Documentation:
Represents systems used by ships and aircraft for safe navigation.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:

float Accuracy
Represents the accuracy of the navigation system.

boolean mechstatus
Represents the mechanical status of the navigation system. ".T." means the navigation system is operational, ".F." means it is not.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

PINS

Category: Navigation
Documentation:
Represents the Precise Integrated Navigation System.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: NavigationSystem
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

VisualNav

Category: Navigation
Documentation:
Represents visual navigation.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: NavigationSystem
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

ElectronicNav

Category: Navigation
Documentation:
Represents electronic navigation systems (e.g. LORAN)

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: NavigationSystem
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

GPS

Category: Navigation
Documentation:
Represents various types of Global Positioning Systems.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: NavigationSystem
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

CommGPS

Category: Navigation
Documentation:
Represents commercial GPS systems.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: GPS
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

MilitaryGPS

Category: Navigation
Documentation:
Represents military GPS systems.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: GPS
Public Interface:
Has-A Relationships:

boolean antispoof
The status of antispoofing techniques. " .T." means either antispoofing is on and required crypto is inserted or antispoofing is not on. ".F." means antispoofing is on and required crypto is not inserted.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

DifferentialGPS

Category: Navigation
Documentation:
Represents differential GPS systems.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: GPS
Public Interface:
Has-A Relationships:

boolean diffstastatus
Represents the status of receiving differential stations. ".T." means differential stations are available. ".F." means differential stations are not available.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Platform

Category: Platforms
Documentation:

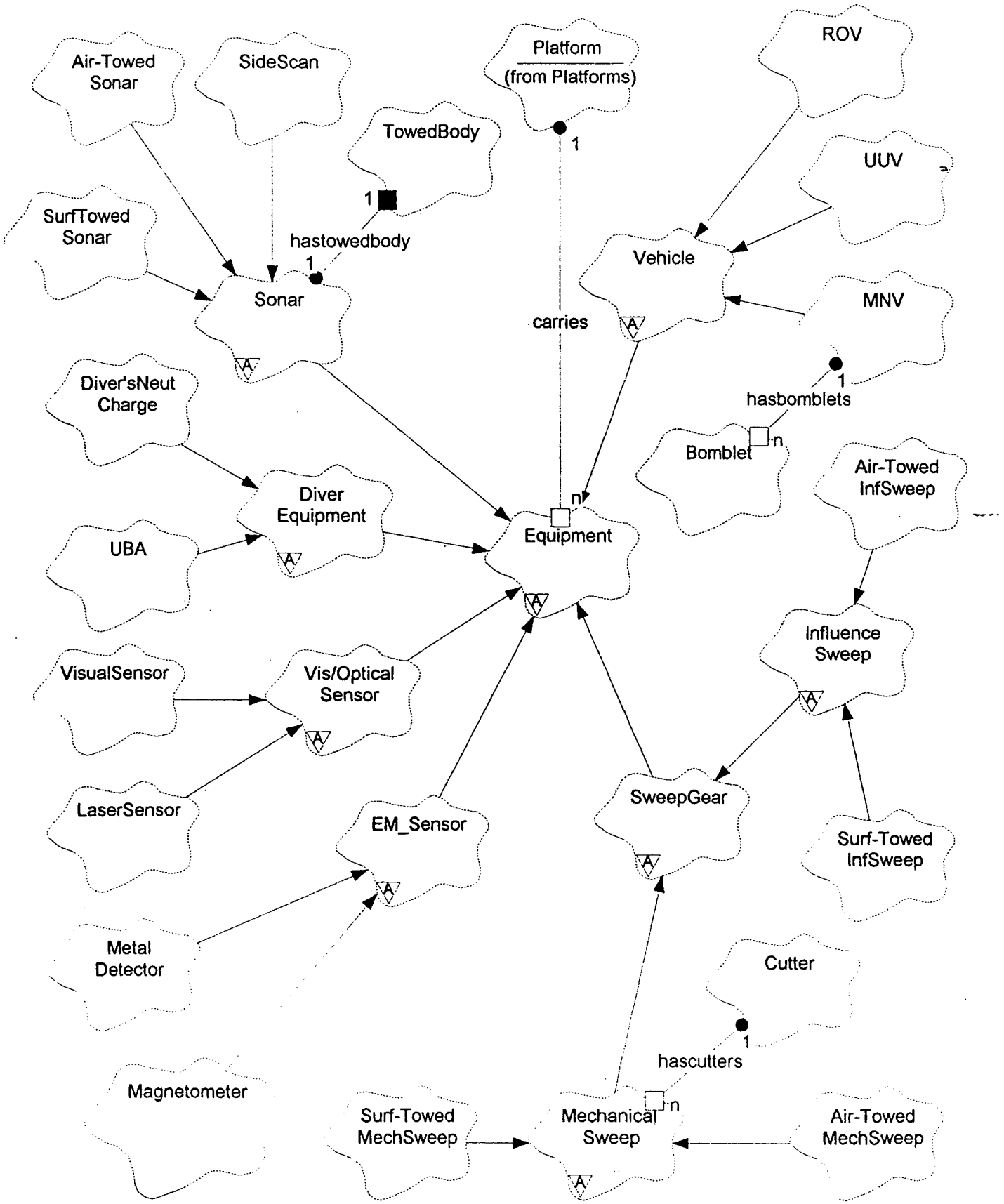
Represents a moving thing such as a ship or aircraft.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:

Equipment carries
Represents the association between a platform and the equipment it carries.

NavigationSystem navigates

State machine: No
Concurrency: Sequential
Persistence: Transient


Air-Towed Sonar
SideScan
Platform
(from Platforms)
ROV
TowedBody
UUV
SurfTowed Sonar
1
hastowedbody
1
Sonar
Vehicle
MNV
carries
Diver'sNeut Charge
1
hasbomblets
Bomblet
n
Air-Towed InfSweep
Diver Equipment
n
Equipment
UBA
Influence Sweep
VisualSensor
Vis/Optical Sensor
LaserSensor
EM_Sensor
SweepGear
Surf-Towed InfSweep
Metal Detector
Cutter
1
hascutters
Magnetometer
Surf-Towed MechSweep
Mechanical Sweep
n
Air-Towed MechSweep

Class name:

Equipment

Category: MCM_Equip
Documentation:
Represents the various types of MCM equipments.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:

nomenclature . nomenclature
Name for the equipment (e.g. AN/SQQ-32)

boolean operstatus
Represents the operational status of the equipment.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Sonar

Category: MCM_Equip
Documentation:
Represents the minehunting sonars.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Equipment
Public Interface:
Has-A Relationships:

TowedBody hastowedbody
Relationship between the sonar and the towed body that carries the hydrophones for the sonar.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Vehicle

Category: MCM_Equip
Documentation:
Represents underwater vehicles used in MCM.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Equipment
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:
SweepGear

Category: MCM_Equip
Documentation:
Represents minesweeping systems.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Equipment
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:
EM_Sensor

Category: MCM_Equip
Documentation:
Represents electromagnetic minehunting sensors.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Equipment
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Vis/OpticalSensor

Category: MCM_Equip
Documentation:
Represents visual or optical minehunting systems.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Equipment
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:
InfluenceSweep

Category: MCM_Equip
Documentation:
Represents influence sweep gear.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: SweepGear
Public Interface:
Has-A Relationships:
influencetype influencetype
The type of influence sweep: acoustic or magnetic.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:
Air-TowedInfSweep

Category: MCM_Equip
Documentation:
Represents influence minesweeping systems that are towed from AMCM helicopters.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: InfluenceSweep
State machine: No

Concurrency: Sequential
Persistence: Transient

Class name:

Surf-TowedInfSweep

Category: MCM_Equip
Documentation:
Represents influence minesweeping systems towed by surface ships.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: InfluenceSweep
Public Interface:
Has-A Relationships:
int dangerfront
Represents the dangerous front for the towing vessel.

Operations:
computedangerfront (int, waveform)

State machine: No
Concurrency: Sequential
Persistence: Transient

Operation name:

computedangerfront

Public member of: Surf-TowedInfSweep
Return Class: dangerfront
Arguments:
int cablescope
waveform infwaveform
Documentation:
Algorithm for computing the danger front for the towing platform.

Concurrency: Sequential

Class name:

Mechanical Sweep

Category: MCM_Equip
Documentation:
Represents family of mechanical minesweeping systems.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: SweepGear
Public Interface:
Has-A Relationships:

int cuttercount
The number of cutters carried by the sweep gear.

Operations:

addcutter ( )
subtractcutter ( )

State machine: No
Concurrency: Sequential
Persistence: Transient

Operation name:

addcutter

Public member of: Mechanical Sweep
*Return Class:*cuttercount
Documentation:
Increment the cutter count by 1.

Semantics:
cuttercount = cuttercount + 1

Object diagram: (Unspecified)

Concurrency: Sequential

Operation name:

subtractcutter

Public member of: Mechanical Sweep
*Return Class:*cuttercount
Documentation:
Subtracts one from the cutter count.

Semantics:
cuttercount = cuttercount - 1

Object diagram: (Unspecified)

Concurrency: Sequential

Class name:

Air-TowedMechSweep

Category: MCM_Equip
Documentation:
Represents mechanical minesweeping systems towed by aircraft.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Mechanical Sweep
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Surf-TowedMechSweep

Category: MCM_Equip
Documentation:
Represents mechanical minesweeping systems towed by surface ships.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Mechanical Sweep
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

MetalDetector

Category: MCM_Equip
Documentation:
Represents metal detecting minehunting systems.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: EM_Sensor
State machine: No
Concurrency: Sequential

Persistence: Transient

Class name:

Magnetometer

Category: MCM_Equip
Documentation:
Represents magnetometer minehunting systems.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: EM_Sensor
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

ROV

Category: MCM_Equip
Documentation:
Represents remotely operated MCM vehicles.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Vehicle
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

MNV

Category: MCM_Equip
Documentation:
Represents mine neutralization vehicles.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Vehicle
Public Interface:
Has-A Relationships:

Bomblet hasbomblets
Relates the MNV to the bomblets carried by it.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:
UUV

Category: MCM_Equip
Documentation:
Represents underwater unmanned vehicles.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Vehicle
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:
VisualSensor

Category: MCM_Equip
Documentation:
Represents visual minehunting systems.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Vis/OpticalSensor
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:
LaserSensor

Category: MCM_Equip
Documentation:
Represents laser minehunting systems.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Vis/OpticalSensor
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

SideScan

Category: MCM_Equip
Documentation:
Represents family of side scan sonars.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Sonar
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Air-TowedSonar

Category: MCM_Equip
Documentation:
Represents family of air-towed sonars.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Sonar
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

SurfTowedSonar

Category: MCM_Equip
Documentation:
Represents family of surface-towed sonars.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Sonar
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

DiverEquipment

Category: MCM_Equip
Documentation:
Represents neutralization equipments such as the diver's charges.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Equipment
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Diver'sNeutCharge

Category: MCM_Equip
Documentation:
Represents neutralization charge carried by diver.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: DiverEquipment
Public Interface:
Has-A Relationships:

int explosiveamt
Amount of explosive charge.

explosivetype explosivetype
Represents the type of explosive in the charge.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Bomblet

Category: MCM_Equip
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
State machine: No
Concurrency: Sequential
Persistence: Transient
Relates the MNV to the bomblets carried by it.

Class name:

Cutter

Category: MCM_Equip
Documentation:
Represents the cable cutters carried on the mechanical sweep gear.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:
float [0,1] cuttingprobability
Represents the probability the cutter will successfully cut the cable for the tethered mine.

Mechanical Sweep hascutters

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

TowedBody

Category: MCM_Equip
Documentation:
Represents the towed body that has the sonar hydrophones.

Export Control: Public
Cardinality: n
Hierarchy:

Superclasses: none
Public Interface:
Has-A Relationships:
int depth
The depth of the towed body.

int trailback
Distance astern the towed body is.

Operations:
computetrailback (int)

State machine: No
Concurrency: Sequential
Persistence: Transient

Operation name:
computetrailback

Public member of: TowedBody
*Return Class:*trailback
Arguments:
int cablescope
Documentation:
Algorithm to compute the trailback distance for the sonar.

Concurrency: Sequential
Relationship between the sonar and the towed body that carries the hydrophones for the sonar.

Class name:
Platform

Category: Platforms
Documentation:
Represents a moving thing such as a ship or aircraft.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:
Equipment carries
Represents the association between a platform and the equipment it carries.

NavigationSystem navigates

State machine: No
Concurrency: Sequential
Persistence: Transient
Represents the association between a platform and the equipment it carries.

Class name:

UBA

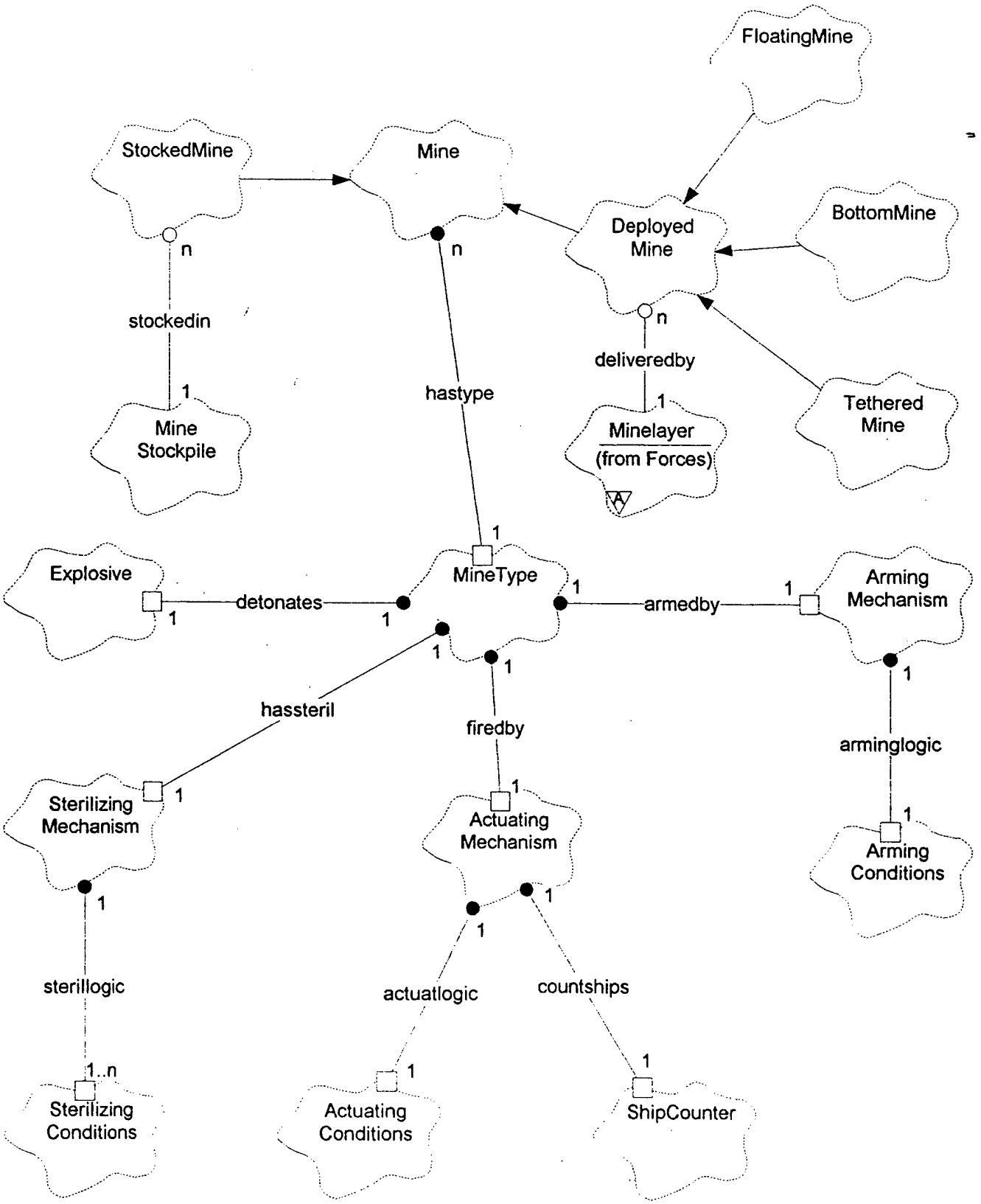
Category: MCM_Equip
Documentation:
Represents diver's underwater breathing apparatus.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: DiverEquipment
Public Interface:
Has-A Relationships:

gastypes breathinggas
The breathing gas carried in the UBA.

int capacity
Capacity of the UBA.

State machine: No
Concurrency: Sequential
Persistence: Transient


FloatingMine
StockedMine
Mine
BottomMine
Deployed Mine
n
n
n
stockedin
1
Mine Stockpile
hastype
deliveredby
1
Minelayer
(from Forces)
Tethered Mine
1
Explosive
1
detonates
1
MineType
1
armedby
1
Arming Mechanism
1
1
1
1
hassteril
firedby
arminglogic
1
Sterilizing Mechanism
1
1
Actuating Mechanism
1
1
Arming Conditions
1
1
1
sterillogic
actuatlogic
countships
1..n
1
1
Sterilizing Conditions
Actuating Conditions
ShipCounter

Class name:

Mine

Category: Threat
Documentation:
Represents a naval mine.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:

MineType hastype
Relates a mine to its type.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

MineType

Category: Threat
Documentation:
Template for a given type of mine

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:

ArmingMechanism armedby
Relates the mine to its arming mechanism.

Explosive detonates
Relates mine type to its explosive.

ActuatingMechanism firedby
Relates the mine type to the actuating mechanism.

SterilizingMechanism hassteril
Relates mine type to the mechanism that sterilizes it.

char minetypename
The name of the mine type.

State machine: No
Concurrency: Sequential
Persistence: Transient
Relates a mine to its type.

Class name:

Explosive

Category: Threat
Documentation:
Represents the explosive carried by the mine.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:

int explosiveamt
Amount of explosive charge.

explosivetype explosivetype
The type of explosive.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

ArmingMechanism

Category: Threat
Documentation:
Represents the mechanism that arms the mine.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:

ArmingConditions arminglogic
Relates the arming mechanism to the conditions under which the mine is armed.

probability armingreliability
The probability the mechanism will arm the mine given that the arming conditions are met.

State machine: No
Concurrency: Sequential
Persistence: Transient
Relates mine type to its explosive.

Relates the mine to its arming mechanism.

Class name:
ArmingConditions

Category: Threat
Documentation:
Represents the logic that determines whether or not the mine arms. Typical conditions may include time delay or command signal.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
State machine: No
Concurrency: Sequential
Persistence: Transient
Relates the arming mechanism to the conditions under which the mine is armed.

Class name:
SterilizingMechanism

Category: Threat
Documentation:
The mechanism that sterilizes the mine.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:
probability Reliability
The reliability with which the sterilizing mechanism sterilizes when it should.

SterilizingConditions sterillogic

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

ActuatingMechanism

Category: Threat
Documentation:
Determines whether or not the mine fires or increments the ship counter.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:

ActuatingConditions actuatlogic
Relates the actuation logic to the actuation mechanism.

ShipCounter countships
Relates the actuating mechanism to the ship counter.

probability Reliability
The reliability of the actuating mechanism.

State machine: No
Concurrency: Sequential
Persistence: Transient
Relates mine type to the mechanism that sterilizes it.

Relates the mine type to the actuating mechanism.

Class name:

SterilizingConditions

Category: Threat
Documentation:
The conditions under which the mine sterilizes. These may include time delay, end of battery life, or cutting of cable.

Export Control: Public
Cardinality: n

Hierarchy:
Superclasses: none
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:
ActuatingConditions

Category: Threat
Documentation:
The logic under which the mine actuates.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
State machine: No
Concurrency: Sequential
Persistence: Transient
Relates the actuation logic to the actuation mechanism.

Class name:
ShipCounter

Category: Threat
Documentation:
The device that counts the number of times the mine's actuating conditions are met.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:
int actualshipcount
The actual number of times the mine is actuated.

int shipcountsetting
The setting of the shipcounter

State machine: No
Concurrency: Sequential
Persistence: Transient
Relates the actuating mechanism to the ship counter.

Class name:

MineStockpile

Category: Threat
Documentation:
Represents the location where mines are stored.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:

int Capacity
The number of mines that can be stored at the mine stockpile

int array Mineinventory
Represents the number of mines by type in the stockpile

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Minelayer

Category: Forces
Documentation:
Represents the abstraction of a minelaying vessel.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Asset
Public Interface:
Has-A Relationships:

integer Minecapacity
The number of mines that the minelayer can carry

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

StockedMine

Category: Threat
Documentation:
Represents a mine that is stocked

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Mine
Public Uses:
MineStockpile stockedin

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:
DeployedMine

Category: Threat
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Mine
Public Uses:
Minelayer deliveredby

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:
TetheredMine

Category: Threat
Documentation:
A mine that is anchored or tethered to the bottom.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: DeployedMine
Public Interface:
Has-A Relationships:
int TetherLength
The length of the tether.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

BottomMine

Category: Threat
Documentation:
A mine that is on the bottom or buried in the bottom.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: DeployedMine
Public Interface:
Has-A Relationships:
probability PercentBurial
The amount the bottom mine is buried

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

FloatingMine

Category: Threat
Documentation:
A mine that is adrift.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: DeployedMine
State machine: No
Concurrency: Sequential
Persistence: Transient

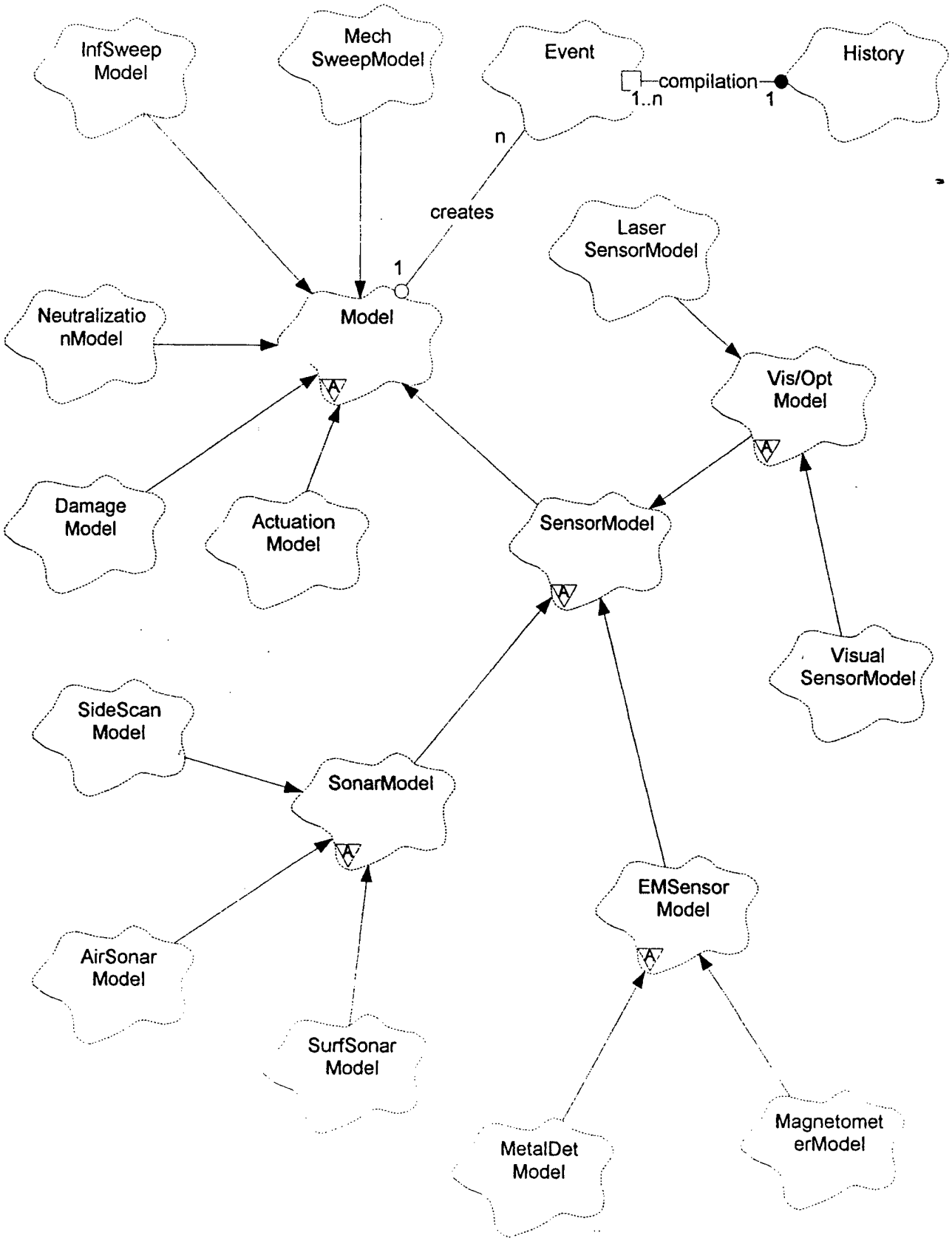

MCM_
Analyst

Class name:

MCM_Analyst

Category: MCM_Analysis
Documentation:
Surrogate for analyst concerned with MCM system performance and effectiveness.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
State machine: No
Concurrency: Sequential
Persistence: Transient


InfSweep Model
Mech SweepModel
Event
compilation
1..n
1
History
n
creates
1
Laser SensorModel
Neutralizatio nModel
Model
Vis/Opt Model
Damage Model
Actuation Model
SensorModel
Visual SensorModel
SideScan Model
SonarModel
EMSensor Model
AirSonar Model
SurfSonar Model
MetalDet Model
Magnetomet erModel

Class name:

Model

Category: MIW_Model
Documentation:
Represents computational models for predicting events or MOE values.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Uses:
Event creates

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Event

Category: MIW_Model
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:
time eventtime
The time the event occurs.

eventtype eventtype
The type of event.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

History

Category: MIW_Model
Documentation:
The list of events.

Export Control: Public

Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:

Event compilation
The history is the compilation of the events.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

InfSweepModel

Category: MIW_Model
Documentation:
Models the performance of influence minesweeping systems.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Model
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

MechSweepModel

Category: MIW_Model
Documentation:
Models the performance of mechanical minesweeping systems.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Model
State machine: No
Concurrency: Sequential
Persistence: Transient
The history is the compilation of the events.

Class name:

NeutralizationModel

Category: MIW_Model
Documentation:
Models the performance of mine neutralization systems.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Model
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

SensorModel

Category: MIW_Model
Documentation:
Models the effectiveness of sensors

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Model
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

SonarModel

Category: MIW_Model
Documentation:
Models the performance of sonar systems

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: SensorModel
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

DamageModel

Category: MIW_Model
Documentation:
Models the damage to ships caused by mines.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Model
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

SideScanModel

Category: MIW_Model
Documentation:
Models the performance of side scan sonar systems.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: SonarModel
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

AirSonarModel

Category: MIW_Model
Documentation:
Models the performance of air sonar systems.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: SonarModel
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

SurfSonarModel

Category: MIW_Model
Documentation:
Models the performance of surface sonar systems.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: SonarModel
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

EMSensorModel

Category: MIW_Model
Documentation:
Models the performance of electromagnetic sensors.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: SensorModel
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

MetalDetModel

Category: MIW_Model
Documentation:
Models the performance of metal detector sensors.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: EMSensorModel
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

MagnetometerModel

Category: MIW_Model
Documentation:
Models the performance of magnetometer sensors.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: EMSensorModel
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Vis/OptModel

Category: MIW_Model
Documentation:
Models the performance of visual or optical sensors.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: SensorModel
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

VisualSensorModel

Category: MIW_Model
Documentation:
Models the effectiveness of visual sensors.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Vis/OptModel
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

LaserSensorModel

Category: MIW_Model
Documentation:
Models the effectiveness of laser minehunting sensors.

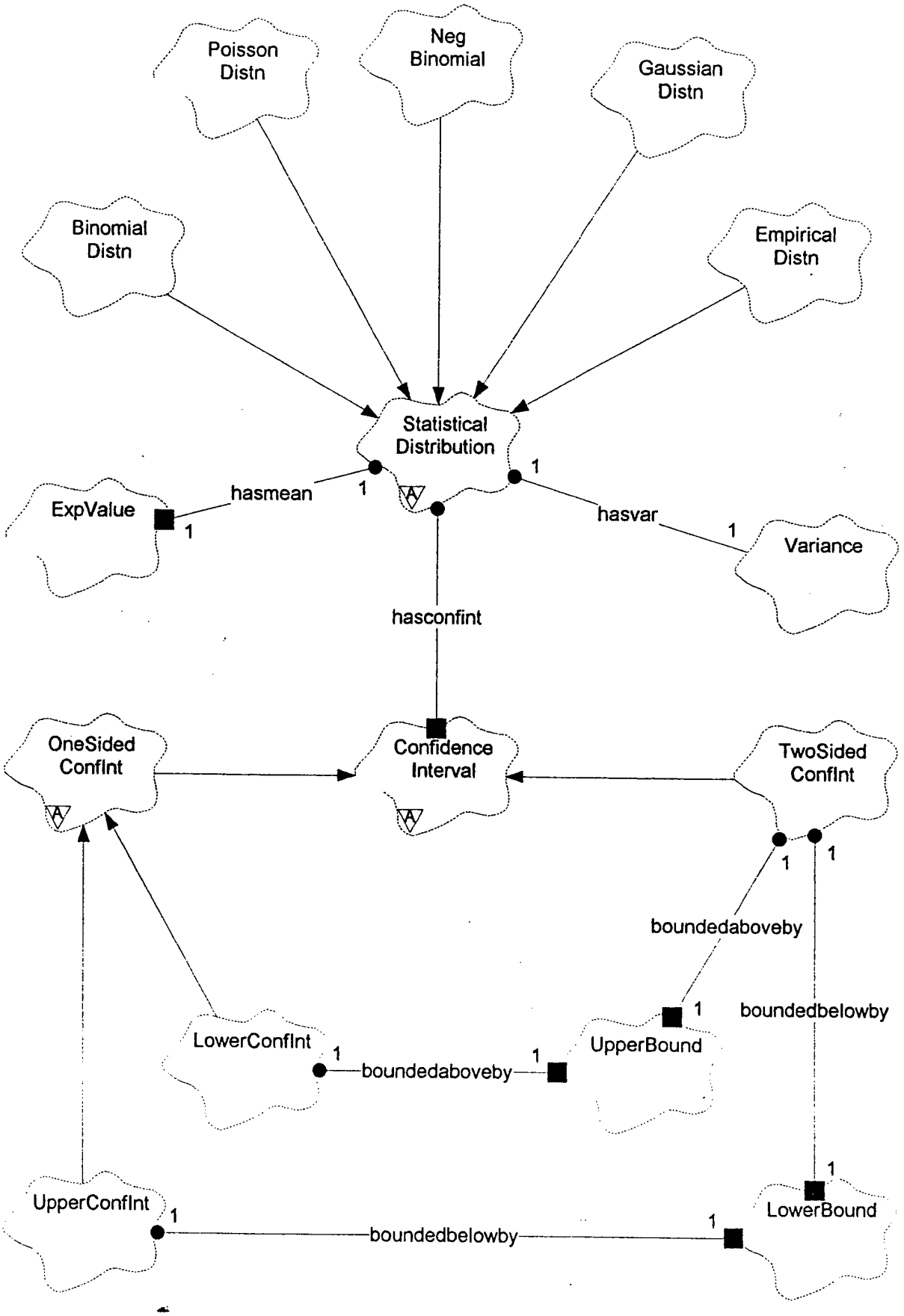
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Vis/OptModel
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

ActuationModel

Category: MIW_Model
Documentation:
Models the actuation of mines.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Model
State machine: No
Concurrency: Sequential
Persistence: Transient



File: c:\rose\semmiw\semmiw.mdl Sun Nov 26 10:12:12 1995 Class Diagram: MOE_Stats / Main

Class name:

StatisticalDistribution

Category: MOE_Stats
Documentation:
Represents the statistical distribution function.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:

ConfidenceInterval hasconfint
Relates the statistical distribution to its confidence intervals.

ExpValue hasmean
Relates the statistical distribution to its mean value.

Variance hasvar
The relationship between the statistical distribution and its variance.

Operations:

computemean ( )
computevar ( )

State machine: No
Concurrency: Sequential
Persistence: Transient

Operation name:

computemean

Public member of: StatisticalDistribution
*Return Class:*ExpValue
Documentation:
The operation to compute the mean of the statistical distribution.

Concurrency: Sequential

Operation name:

computevar

Public member of: StatisticalDistribution
*Return Class:*Variance
Documentation:
Computes the variance of the statistical distribution.

Concurrency: Sequential

Class name:

ExpValue

Category: MOE_Stats
Documentation:
Represents the expected value, or mean of the statistical distribution.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:
float value
The actual value of the mean.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Variance

Category: MOE_Stats
Documentation:
Represents the variance of the statistical distribution.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:
float value
The actual value of the variance.

State machine: No
Concurrency: Sequential
Persistence: Transient
Relates the statistical distribution to its mean value.

The relationship between the statistical distribution and its variance.

Class name:

ConfidenceInterval

Category: MOE_Stats
Documentation:
Represents a one- or two-sided confidence interval

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:
probability Confidencelevel
The value associated with the confidence interval.

State machine: No
Concurrency: Sequential
Persistence: Transient
Relates the statistical distribution to its confidence intervals.

Class name:

OneSidedConfInt

Category: MOE_Stats
Documentation:
Represents a one-sided confidence interval.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: ConfidenceInterval
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

TwoSidedConfInt

Category: MOE_Stats

Documentation:
Represents two-sided confidence intervals.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: ConfidenceInterval
Public Interface:
Has-A Relationships:
UpperBound boundedaboveby
Relates confidence inteval to its upper bound.

LowerBound boundedbelowby
Relates confidence interval to its lower bound.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

UpperBound

Category: MOE_Stats
Documentation:
The upper bound for the confidence interval.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:
float boundvalue
The value of the confidence bound

State machine: No
Concurrency: Sequential
Persistence: Transient
Relates confidence inteval to its upper bound.

Class name:

LowerBound

Category: MOE_Stats

Documentation:
The lower bound for the confidence interval.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:
float boundvalue
The value of the bound.

State machine: No
Concurrency: Sequential
Persistence: Transient
Relates confidence interval to its lower bound.

Class name:
LowerConfInt

Category: MOE_Stats
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: OneSidedConfInt
Public Interface:
Has-A Relationships:
UpperBound boundedaboveby
Relates the lower confidence interval to its upper bound.

State machine: No
Concurrency: Sequential
Persistence: Transient
Relates the lower confidence interval to its upper bound.

Class name:
UpperConfInt

Category: MOE_Stats
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: OneSidedConfInt

Public Interface:
Has-A Relationships:
LowerBound boundedbelowby
Relates the upper confidence interval to its lower bound.

State machine: No
Concurrency: Sequential
Persistence: Transient
Relates the upper confidence interval to its lower bound.

Class name:
BinomialDistn

Category: MOE_Stats
Documentation:
Represents the family of binomial distributions.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: StatisticalDistribution
Public Interface:
Has-A Relationships:
int Numtrials
The number of trials of the statistical distribution.

probability Successprob
The probability of success of an individual trial.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:
PoissonDistn

Category: MOE_Stats
Documentation:
Represents the family of Poisson Distributions.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: StatisticalDistribution

Public Interface:
Has-A Relationships:

Float Delta
Delta is the parameter of the Poisson Distribution.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

NegBinomialDistn

Category: MOE_Stats
Documentation:
Represents the negative binomial distn

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: StatisticalDistribution
Public Interface:
Has-A Relationships:

int Numtrials
The number of trials associated with the negative binomial distribution.

probability Successprob
The probability of success of an individual trial.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

GaussianDistn

Category: MOE_Stats
Documentation:
Represents the Gaussian family of distributions.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: StatisticalDistribution
Public Interface:
Has-A Relationships:

float Mu
The mean of the Gaussian distribution.

float SigmaSquared
The variance of the Gaussian distribution.

State machine: No
Concurrency: Sequential
Persistence: Transient

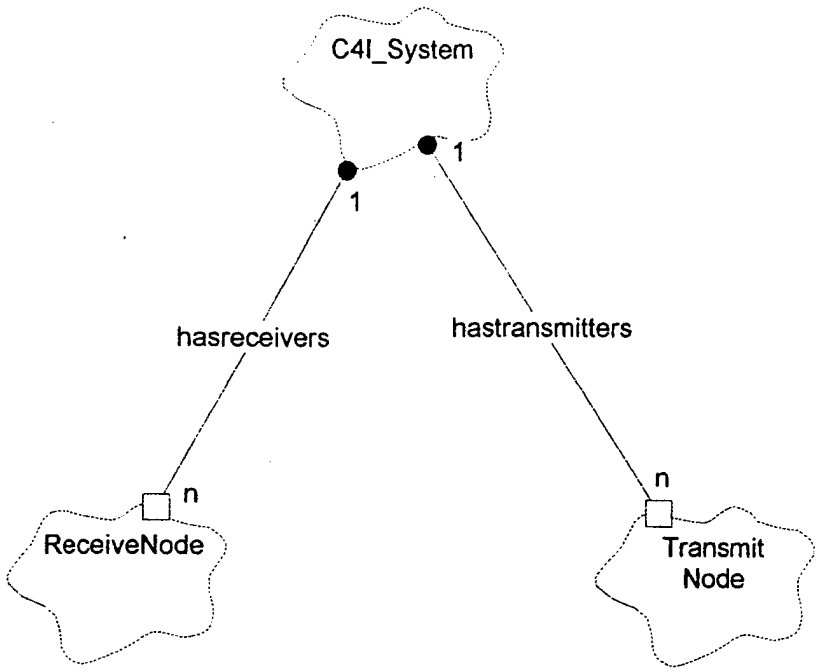
Class name:

EmpiricalDistn

Category: MOE_Stats
Documentation:
Represents an empirical distribution function for observed events.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: StatisticalDistribution
Public Interface:
Has-A Relationships:
dataarray XY_array
The number of points for each X value and the X values.

State machine: No
Concurrency: Sequential
Persistence: Transient


C4I_System
1
1
hasreceivers
hastransmitters
n
n
ReceiveNode
Transmit
Node

Class name:

C4I_System

Category: C4I
Documentation:
Represents systems used to exchange tactical information.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:

boolean emconstatus
Represents the status of emissions controls in use.

ReceiveNode hasreceivers
A C4I System contains receiving nodes.

TransmitNode hastransmitters
A C4I System has transmitting nodes.

boolean mechstatus
Represents the mechanical status of the C4I system.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

ReceiveNode

Category: C4I
Documentation:
Represents a receiving node of a C4I system.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:

boolean mechstatus
Represents the mechanical status of the receiving node.

State machine: No
Concurrency: Sequential

Persistence: Transient

Class name:

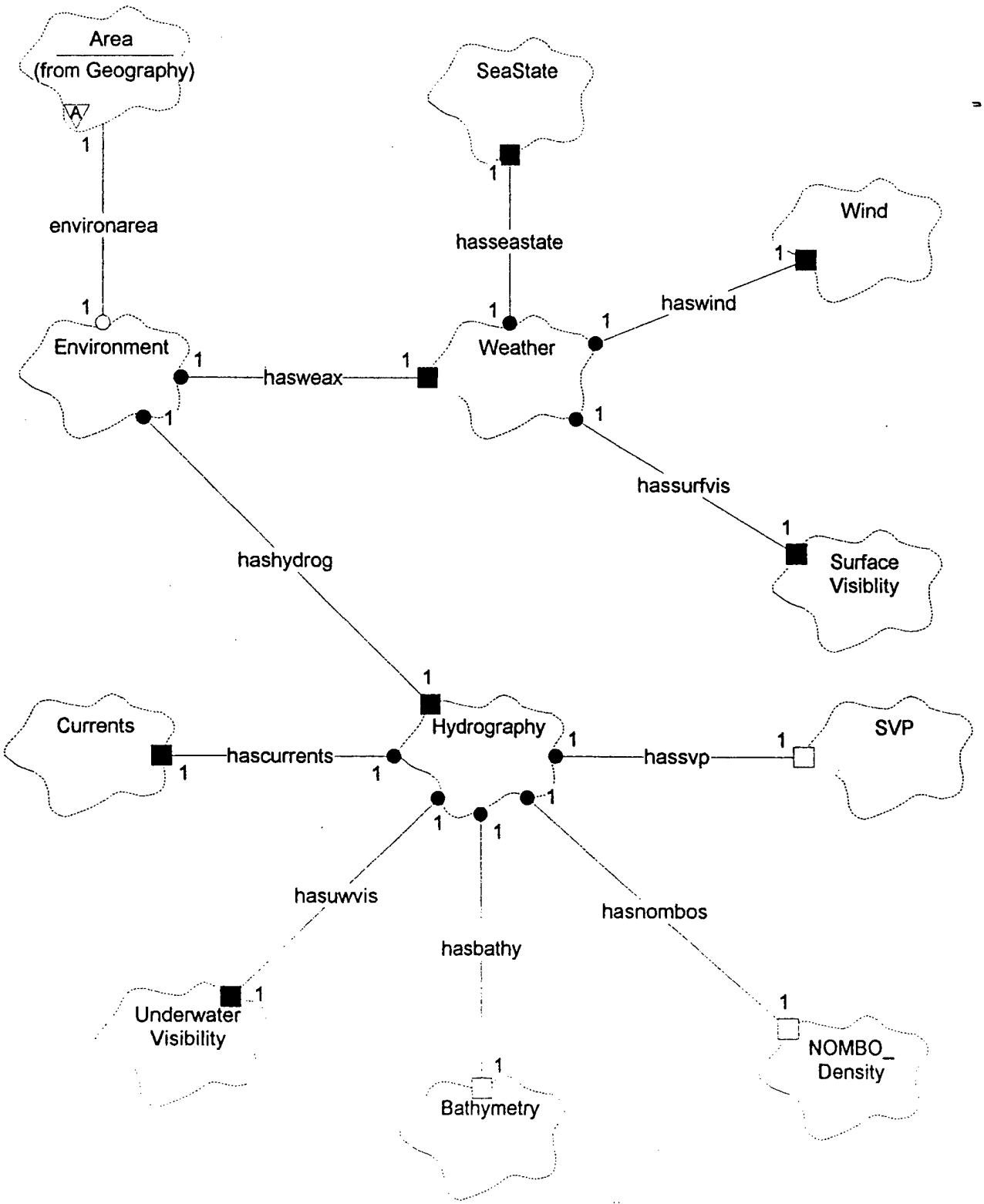
TransmitNode

Category: C4I
Documentation:
Represents a transmitting node of a C4I system

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:
boolean mechstatus
Represents the mechanical status of the transmitting node.

State machine: No
Concurrency: Sequential
Persistence: Transient
A C4I System contains receiving nodes.

A C4I System has transmitting nodes.


Area
(from Geography)
SeaState
Wind
environarea
hasseastate
haswind
Environment
hasweax
Weather
hassurfvis
Surface
Visiblity
hashydrog
Currents
hascurrents
Hydrography
hassvp
SVP
hasuwvis
hasbathy
hasnombos
Underwater
Visibility
Bathymetry
NOMBO_
Density

Class name:

Environment

Category: Oceanography
Documentation:
Represents the set of environmental data for an area.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Uses:
Area environarea

Public Interface:
Has-A Relationships:
Hydrography hashydrog
Relates the environment to the conditions below the surface of the water.

Weather hasweax
Relates the environment to the weather conditions.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Weather

Category: Oceanography
Documentation:
Represents the meteorlogical conditions (at or above the surface of the water)

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:
SeaState hasseastate
Relates the weather to the sea state.

SurfaceVisiblity hassurfvis
Relates the weather to the visibility at or above the surface.

Wind haswind
Relates the weather to the wind data.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

SeaState

Category: Oceanography
Documentation:
Represents the sea state

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:

int seastatevalue
The value of the sea state.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Wind

Category: Oceanography
Documentation:
Represents the wind conditions.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:

int maxgustspeed
The maximum speed of wind gusts.

int sustainedwindspeed
The sustained wind speed.

State machine: No
Concurrency: Sequential

Persistence: Transient

Class name:

SurfaceVisiblity

Category: Oceanography
Documentation:
The visibility conditions at or above the surface of the water.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:

int ceilingheight
The height of the cloud cover.

int visibilityrange
The range of visibility.

State machine: No
Concurrency: Sequential
Persistence: Transient
Relates the environment to the weather conditions.

Relates the weather to the sea state.

Relates the weather to the wind data.

Relates the weather to the visibility at or above the surface.

Class name:

Hydrography

Category: Oceanography
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:

Bathymetry hasbathy
Relates hydrography to bathymetry.

Currents hascurrents
Relates the hydrography to the currents.

NOMBO_Density hasnombos
Relates hydrography to NOMBOs

SVP hassvp
Relates hydrography to SVP.

UnderwaterVisibility hasuwvis
Relates hydrography to underwater visibility

State machine: No
Concurrency: Sequential
Persistence: Transient
Relates the environment to the conditions below the surface of the water.

Class name:

Currents

Category: Oceanography
Documentation:
Represents the currents at the surface and at the bottom of the water.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:

currentspeed bottcurrentspeed
The speed of the current below the surface.

currentspeed surfcurrentspeed
The speed of the surface currents.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

UnderwaterVisibility

Category: Oceanography

Documentation:
The underwater visibility for divers or vehicles.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:
int uwvisrange
Visibility range underwater.

State machine: No
Concurrency: Sequential
Persistence: Transient
Relates the hydrography to the currents.

Relates hydrography to underwater visibility

Class name:

Bathymetry

Category: Oceanography
Documentation:
Depth for a given location.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:
bathyarray Bathydata
Array of water depths for each given location.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

SVP

Category: Oceanography
Documentation:
The speed of sound as a function of depth.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:

svparray SVParray
A list of depths with speeds of sound given for each depth.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

NOMBO_Density

Category: Oceanography
Documentation:
Density of mine-like contacts that are not mines in an area.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:

float nombodensvalue
The value of the NOMBO density.

State machine: No
Concurrency: Sequential
Persistence: Transient
Relates hydrography to bathymetry.

Relates hydrography to SVP.

Relates hydrography to NOMBOs

Class name:

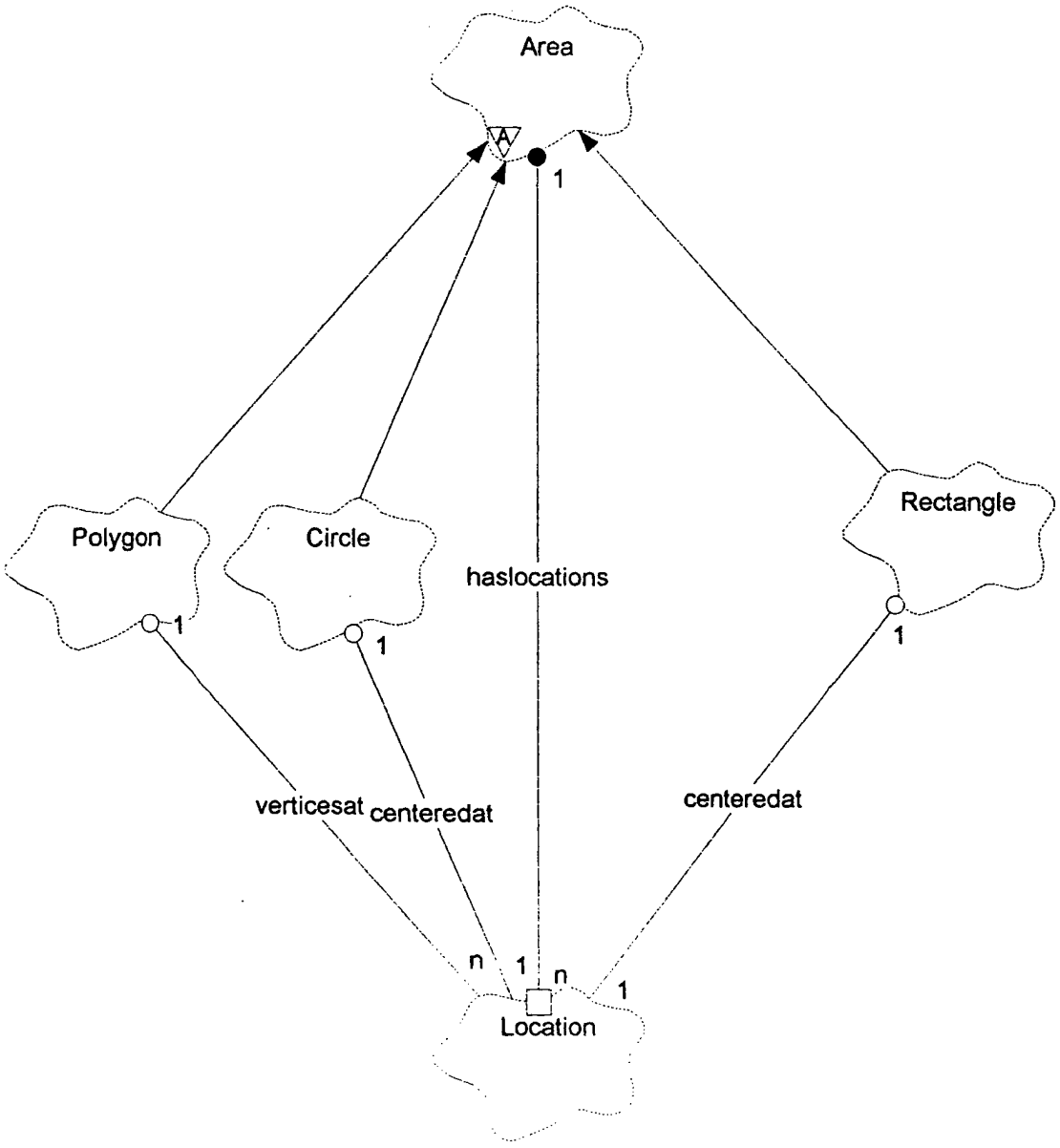
Area

Category: Geography

Documentation:
Represents a geographic area.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:
Location haslocations
Relates an area to locations contained in it.

State machine: No
Concurrency: Sequential
Persistence: Transient


Area
1
Polygon
Circle
Rectangle
haslocations
1
1
1
verticesat
centeredat
centeredat
n
1
n
1
Location

Class name:

Location

Category: Geography
Documentation:
Represents a physical location on the globe.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:

int depth
Depth of the location, referenced to sea level.

lattype latitude
The latitude of the location

longtype longitude
The longitude of the location.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Area

Category: Geography
Documentation:
Represents a geographic area.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Interface:
Has-A Relationships:

Location haslocations
Relates an area to locations contained in it.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Circle

Category: Geography
Documentation:
A circular area.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Area
Public Uses:

Location centeredat

Public Interface:
Has-A Relationships:

int Radius
The radius of the circle.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Rectangle

Category: Geography
Documentation:
Represents a rectangular area.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Area
Public Uses:

Location centeredat

Public Interface:
Has-A Relationships:

int Length
Length of the rectangle.

bearing orientation
Orientation of the rectangle axis.

int Width
The width of the rectangle.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

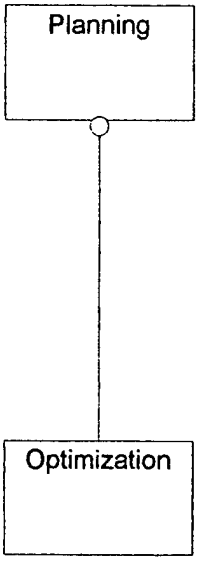
Polygon

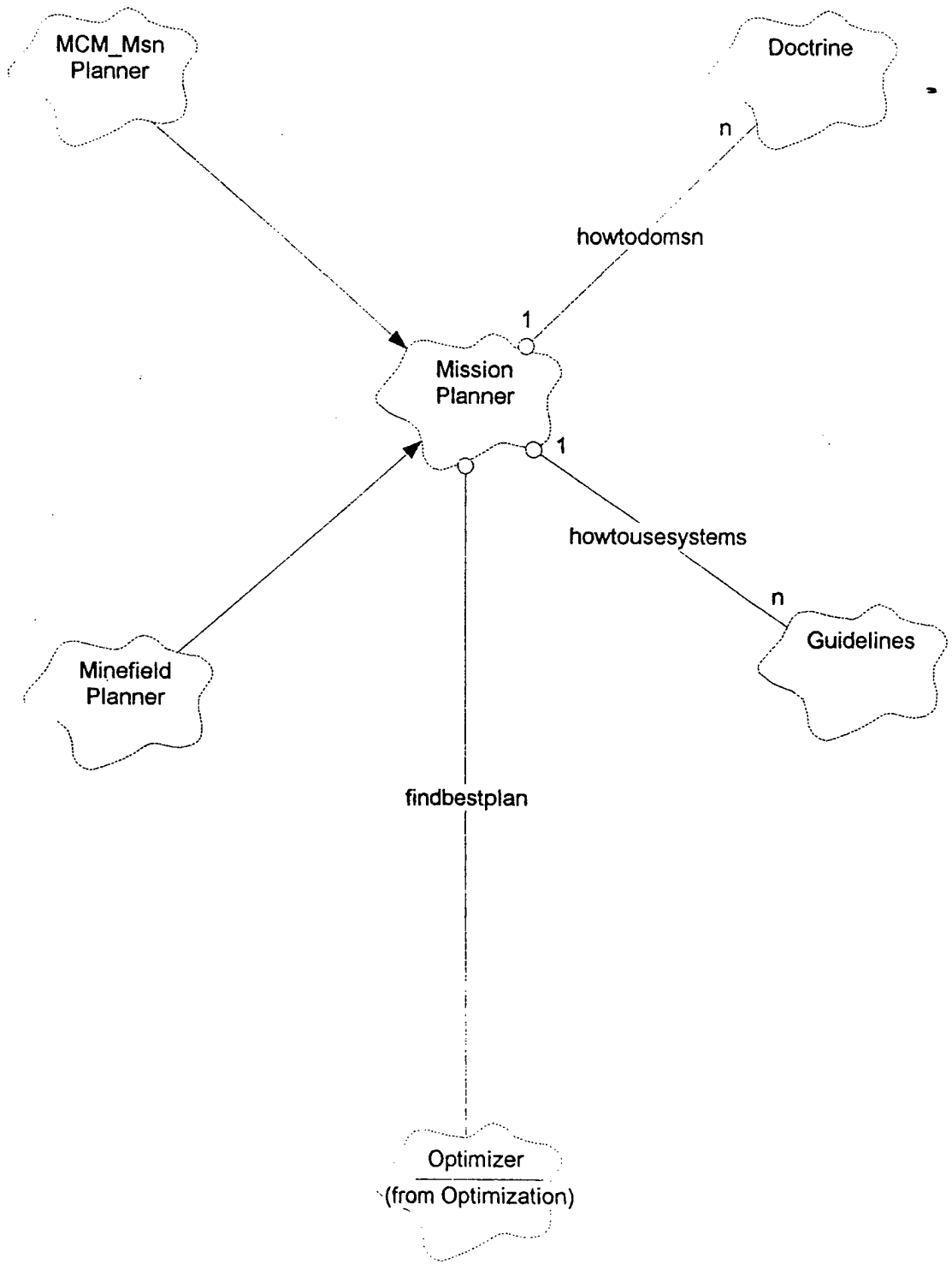
Category: Geography
Documentation:
A polygon of arbitrary number of vertices.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Area
Public Uses:
Location verticesat

Public Interface:
Has-A Relationships:
vertexlist vertexlist
The list of vertices of the polygon.

State machine: No
Concurrency: Sequential
Persistence: Transient
Relates an area to locations contained in it.


Planning
Optimization


MCM_Msn Planner
Doctrine
n
howtodomsn
1
Mission Planner
1
howtousesystems
n
Guidelines
Minefield Planner
findbestplan
Optimizer
(from Optimization)

Class name:

MissionPlanner

Category: Planning
Documentation:
Represents the plan for a mine warfare mission

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Uses:

Optimizer	findbestplan
Doctrine	howtodomsn
Guidelines	howtousesystems

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

MCM_MsnPlanner

Category: Planning
Documentation:
Planning procedure for a minecountermeasures mission.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: MissionPlanner
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

MinefieldPlanner

Category: Planning
Documentation:
Planning procedure for minefield placement.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: MissionPlanner
State machine: No

Concurrency: Sequential
Persistence: Transient

Class name:
Optimizer

Category: Optimization
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:
Doctrine

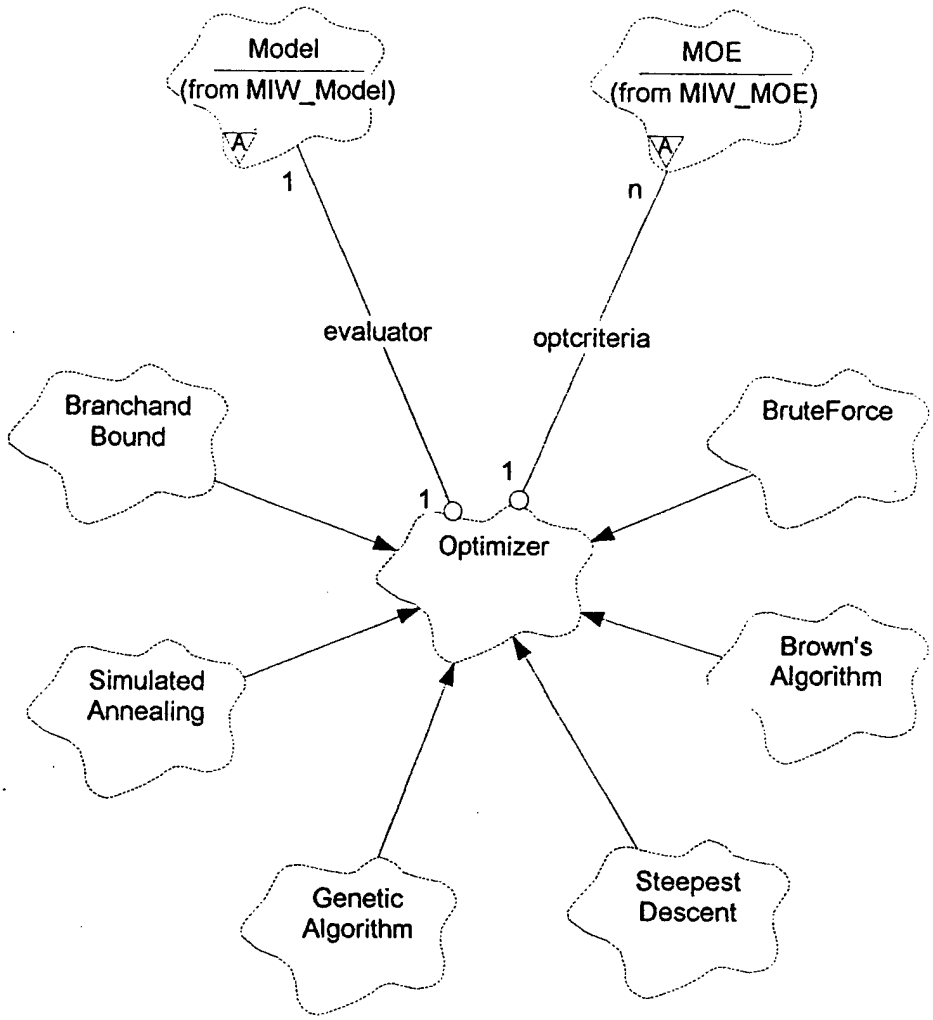
Category: Planning
Documentation:
Represents body of corporate knowledge concerning best use of assets in a particular mission.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:
Guidelines

Category: Planning
Documentation:
Represents guidance on best use of an MCM system.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
State machine: No
Concurrency: Sequential
Persistence: Transient


Model
(from MIW_Model)
MOE
(from MIW_MOE)
1
n
evaluator
optcriteria
Branchand
Bound
BruteForce
1
1
Optimizer
Simulated
Annealing
Brown's
Algorithm
Genetic
Algorithm
Steepest
Descent

Class name:

Model

Category: MIW_Model
Documentation:
Represents computational models for predicting events or MOE values.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Uses:

Event	creates

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Optimizer

Category: Optimization
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Uses:

MOE	optcriteria
Model	evaluator

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

MOE

Category: MIW_MOE
Documentation:
Represents a measure of effectiveness.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Public Uses:

Model	computedby

Public Interface:

Has-A Relationships:

float MOE_value
Represents the value of the MOE.

State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

GeneticAlgorithm

Category: Optimization
Documentation:
An optimization technique based on the genetic algorithm.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Optimizer
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

Brown'sAlgorithm

Category: Optimization
Documentation:
An optimization method based on Brown's Algorithm.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Optimizer
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

BruteForce

Category: Optimization
Documentation:
An optimization method based on exhaustively evaluating all possibilities.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Optimizer
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

BranchandBound

Category: Optimization
Documentation:
An optimization algorithm of the branch and bound type.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Optimizer
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

SimulatedAnnealing

Category: Optimization
Documentation:
An optimization method that uses simulated annealing.

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: Optimizer
State machine: No
Concurrency: Sequential
Persistence: Transient

Class name:

SteepestDescent

Category: Optimization
Documentation:
An optimization method based on the method of steepest descent.

Export Control:	Public
Cardinality:	n
Hierarchy:	
Superclasses:	Optimizer
State machine:	No
Concurrency:	Sequential
Persistence:	Transient

Chapter 3: Object Scenario Diagrams and Descriptions

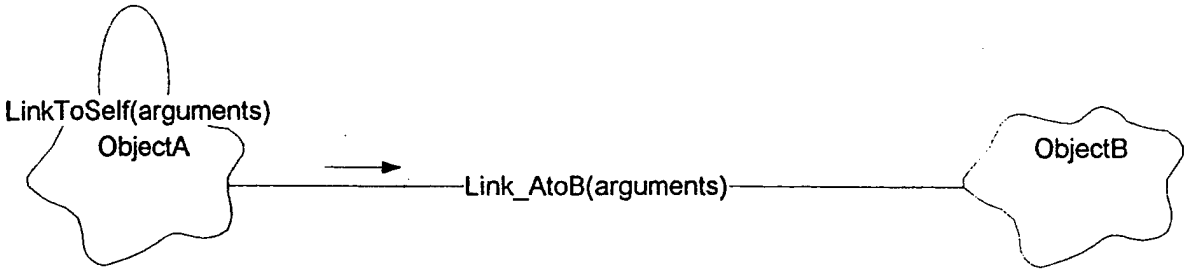
Definitions

A **scenario** is defined as a sequence of events that occur during system execution. In general, a scenario corresponds to an individual function of the system.

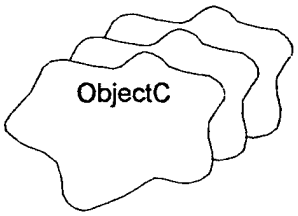
A **use case** as defined by Jacobson [3] is "a particular form or pattern or exemplar of usage". As used in this document, a use case corresponds to a basic system function and consists of a set of related scenarios.

Legend for Scenario Diagrams in the Booch Notation:

A. Links Between Objects



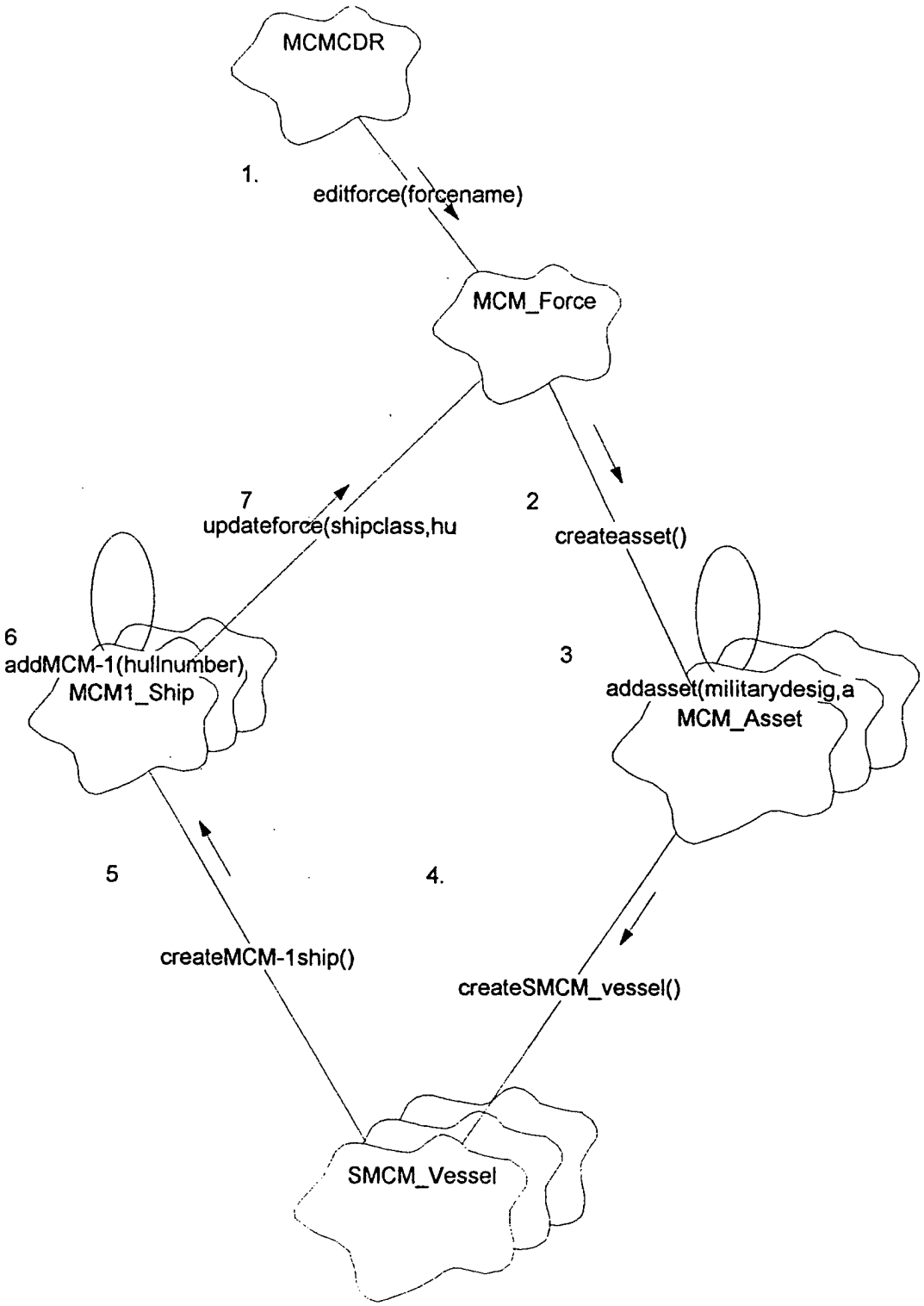
B. Multiple Instances of An Object



Use Case: Specify the Force

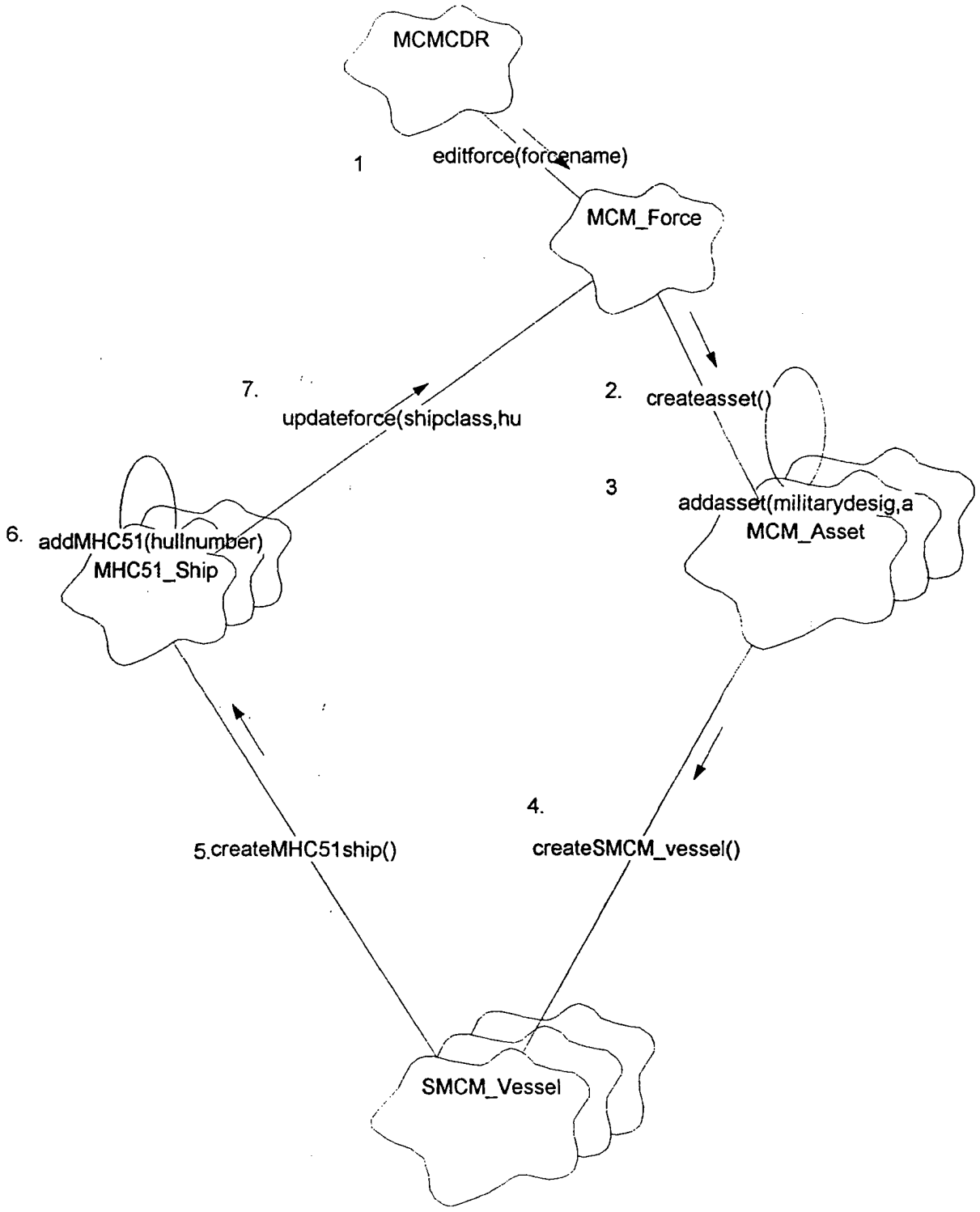
Scenario: Add New MCM-1 Ship

1. The user selects "Edit Force" on the Commander's Display.
2. The user selects "Add Asset". The "Add Asset screen appears".
3. The user enters the military designation of the new asset and selects "SMCM Vessel" as the asset type.
4. The user selects "MCM-1" as the SMCM vessel type.
5. An object of class MCM-1 is created.
6. The user selects the hull number of the MCM-1 class ship.
7. The MCM Force is updated with the new MCM-1 ship.


MCMCDR
1.
editforce(forcename)
MCM_Force
7
updateforce(shipclass,hu
2
createasset()
6
addMCM-1(hullnumber)
MCM1_Ship
3
addasset(militarydesig,a
MCM_Asset
5
4.
createMCM-1ship()
createSMCM_vessel()
SMCM_Vessel

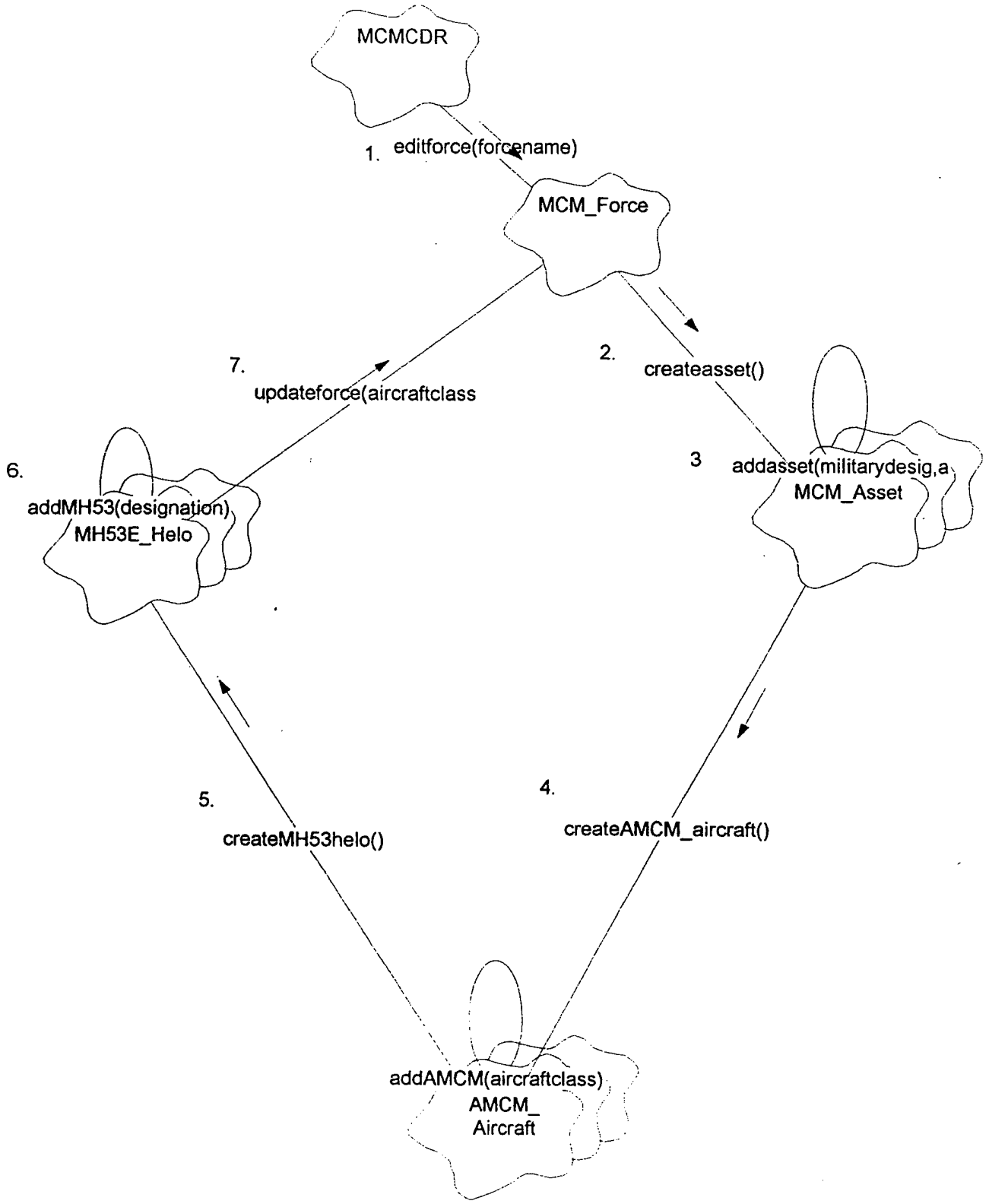
Scenario: Add New MHC-51 Ship

1. The user selects "Edit Force" on the Commander's Display.
2. The user selects "Add Asset". The "Add Asset screen appears".
3. The user enters the military designation of the new asset and selects "SMCM Vessel" as the asset type.
4. The user selects "MHC-51" as the SMCM vessel type.
5. An object of class MHC-51 is created.
6. The user selects the hull number of the MHC-51 class ship.
7. The MCM Force is updated with the new MHC-51 ship.


MCMCDR
1
editforce(forcename)
MCM_Force
7.
updateforce(shipclass,hu
2.
createasset()
3
addasset(militarydesig,a
MCM_Asset
6.
addMHC51(hullnumber)
MHC51_Ship
4.
5.createMHC51ship()
createSMCM_vessel()
SMCM_Vessel

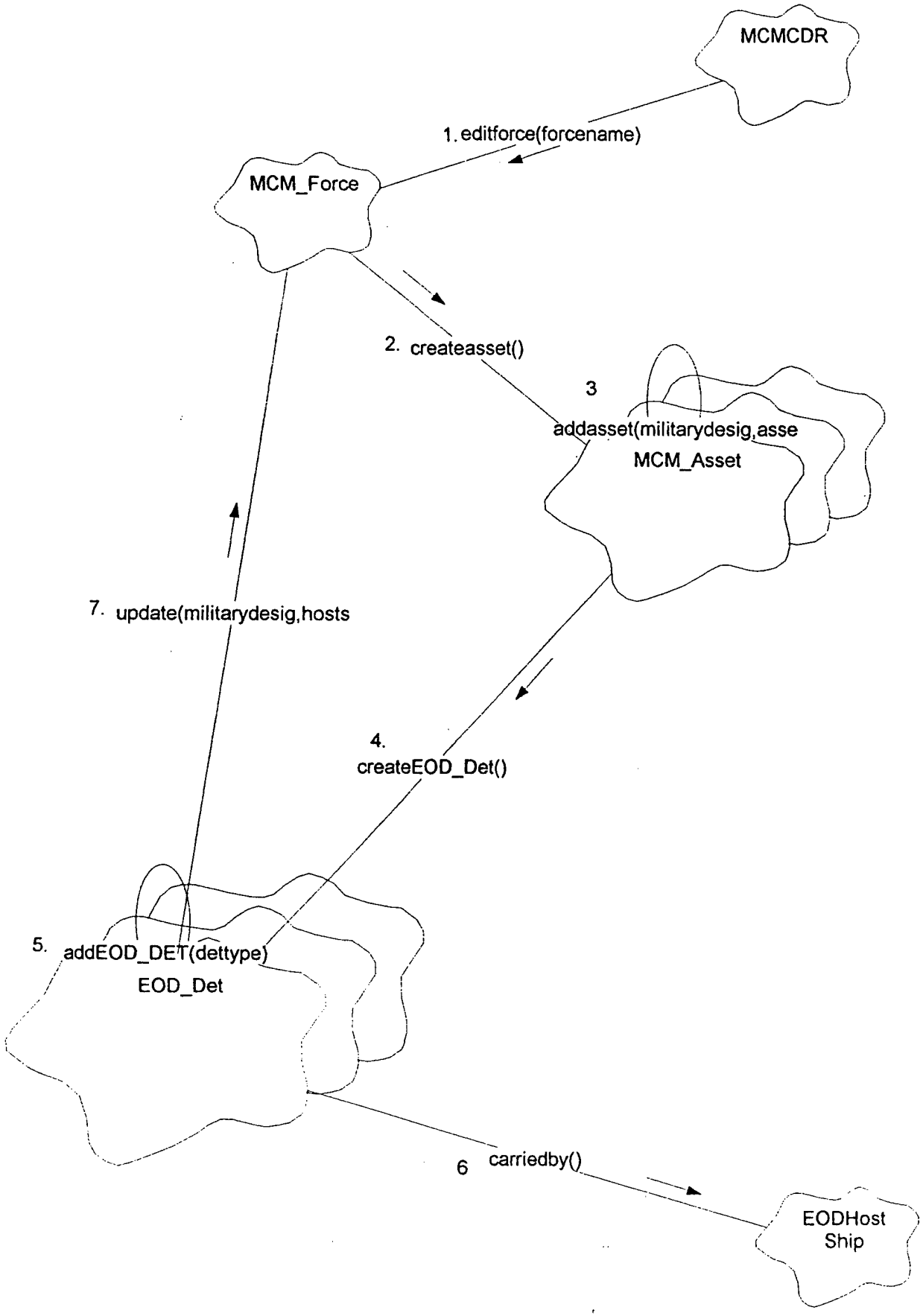
Scenario: Add New MH-53 Helicopter

1. The user selects "Edit Force" on the Commander's Display.
2. The user selects "Add Asset". The "Add Asset screen appears".
3. The user enters the military designation of the new asset and selects "AMCM Helicopter" as the asset type.
4. The user selects "MH-53E" as the AMCM helicopter type.
5. An object of class MH-53E is created.
6. The user selects the identifier for the MH-53E class helicopter.
7. The MCM Force is updated with the new MH53-E helicopter.


MCMCDR
1. editforce(forcename)
MCM_Force
2.
createasset()
3
addasset(militarydesig,a
MCM_Asset
4.
createAMCM_aircraft()
addAMCM(aircraftclass)
AMCM_
Aircraft
5.
createMH53helo()
6.
addMH53(designation)
MH53E_Helo
7.
updateforce(aircraftclass

Scenario: Add New EOD Detachment

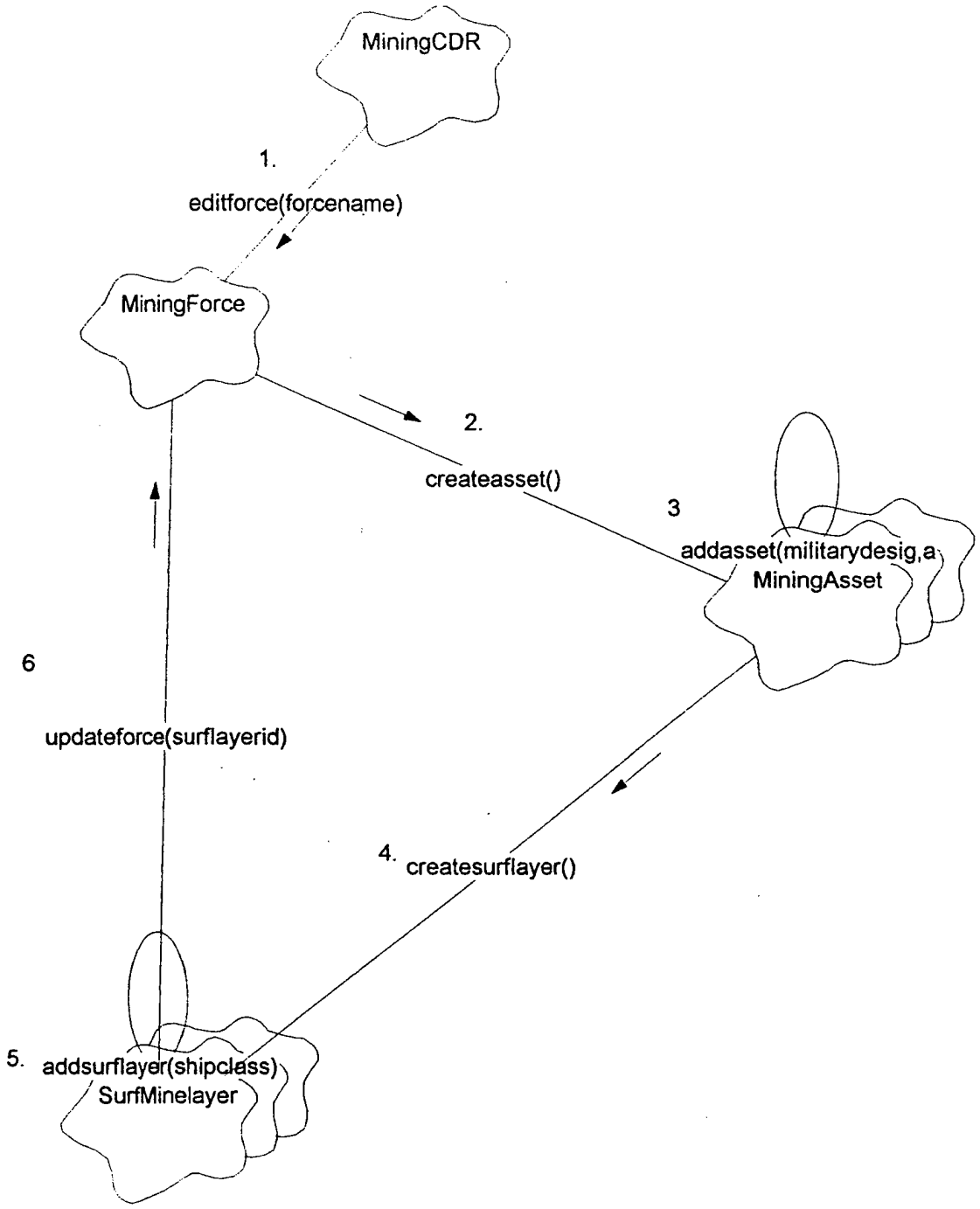
1. The user selects "Edit Force" on the Commander's Display.
2. The user selects "Add Asset". The "Add Asset screen appears".
3. The user enters the military designation of the new asset and selects "EOD Detachment" as the asset type.
4. An object of class EOD Detachment is created.
5. The user selects the type of EOD Detachment.
6. The user selects the host platform for the EOD Detachment.
7. The MCM Force is updated with the new EOD Detachment.



File: c:\rose\semmiw.mdl Tue Nov 14 13:45:03 1995 Object Diagram: <Top Level> / Add EOD Det to Force

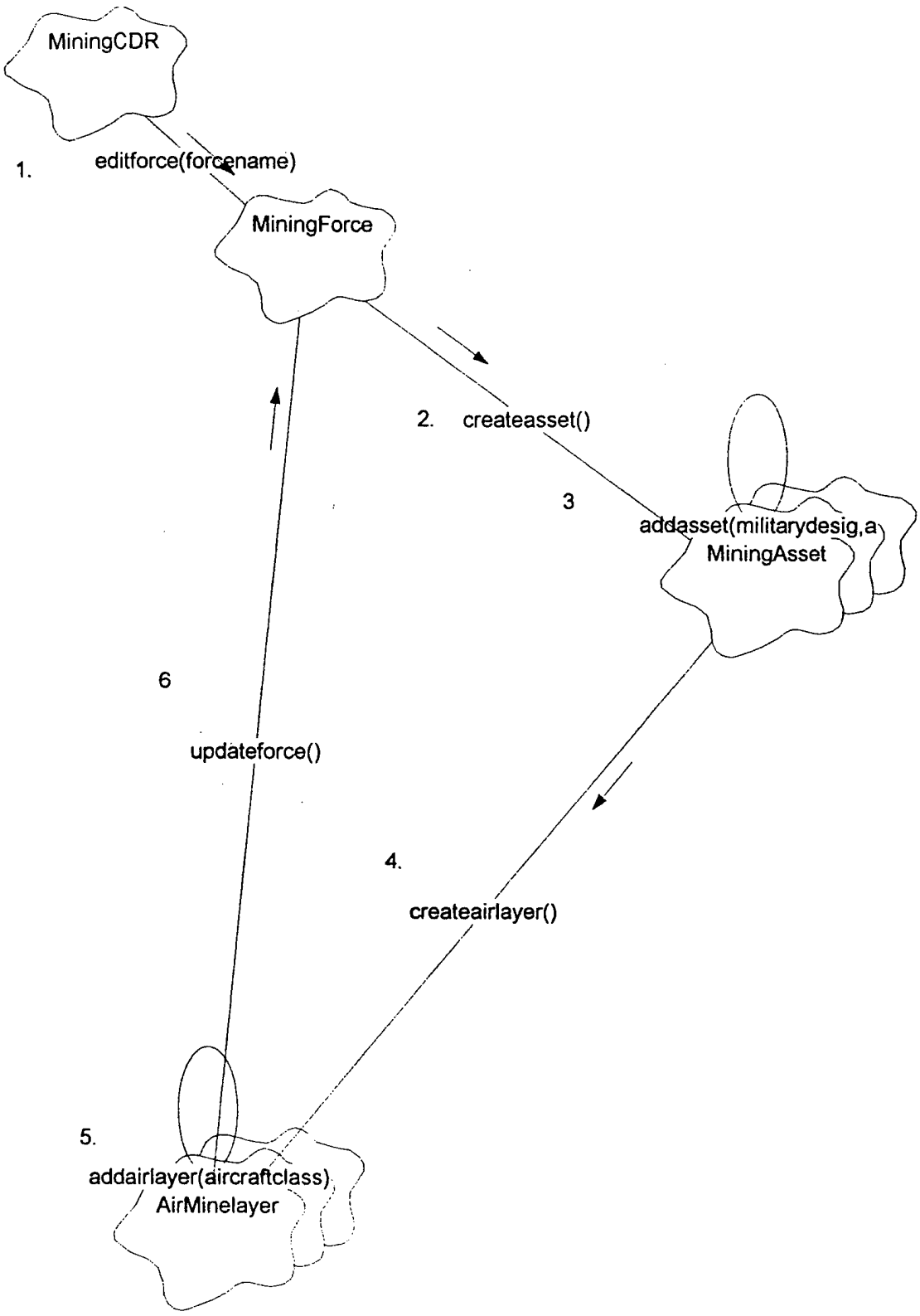
Scenario: Add New Surface Minelayer

1. The user selects "Edit Force" on the Commander's Display.
2. The user selects "Add Asset". The "Add Asset screen appears".
3. The user enters the military designation of the new asset and selects "Surface Minelayer" as the asset type.
4. A surface minelayer object is created.
5. The user selects the identifier for the surface minelayer.
6. The Mining Force is updated with the new surface minelayer.


MiningCDR
1.
editforce(forcename)
MiningForce
2.
createasset()
3
addasset(militarydesig,a
MiningAsset
6
updateforce(surflayerid)
4.
createsurflayer()
5.
addsurflayer(shipclass)
SurfMinelayer

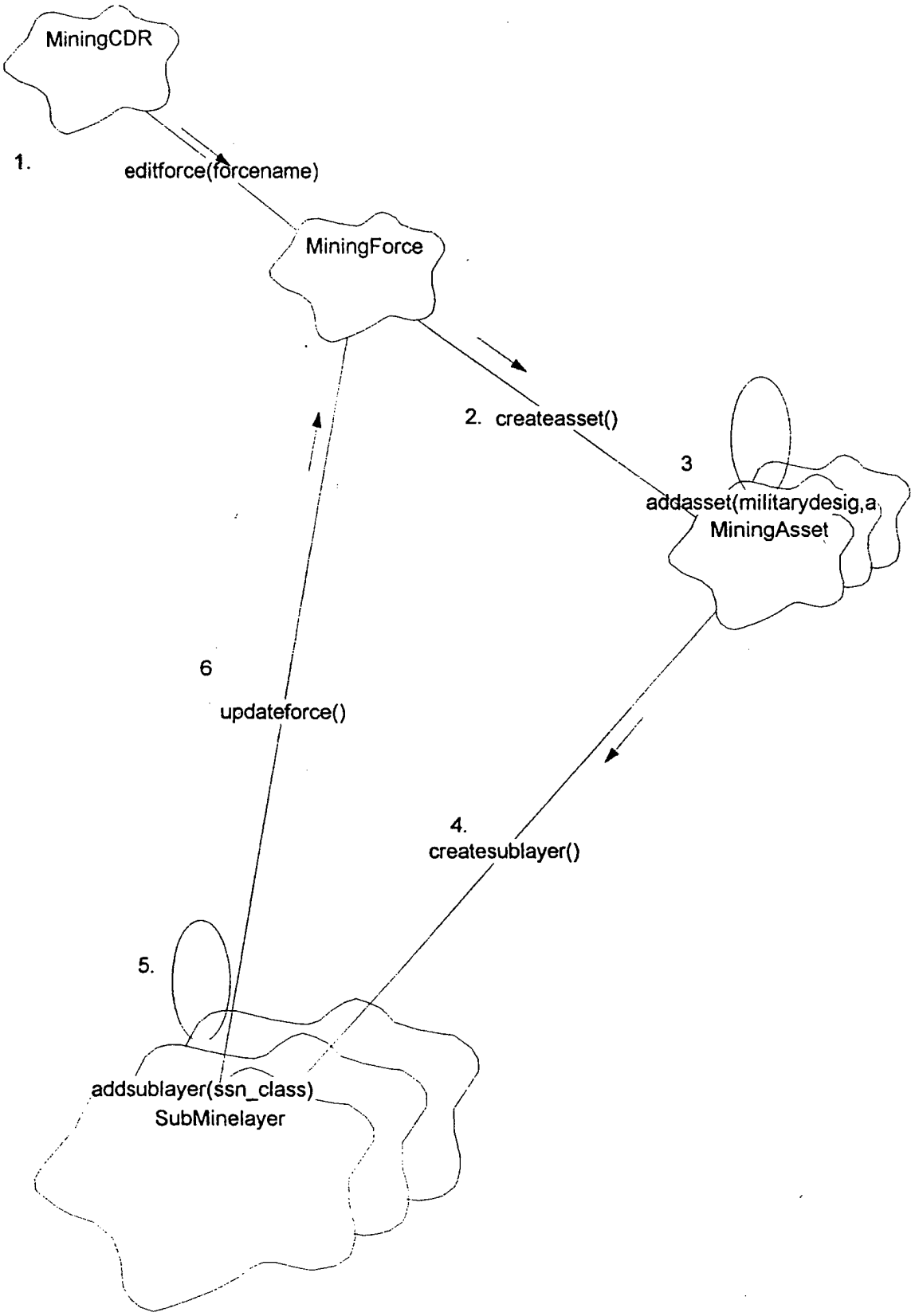
Scenario: Add New Air Minelayer

1. The user selects "Edit Force" on the Commander's Display.
2. The user selects "Add Asset". The "Add Asset screen appears".
3. The user enters the military designation of the new asset and selects "Air Minelayer" as the asset type.
4. A air minelayer object is created.
5. The user selects the identifier for the air minelayer.
6. The Mining Force is updated with the new air minelayer.


MiningCDR
1.
editforce(forcename)
MiningForce
2.
createasset()
3
addasset(militarydesig,a
MiningAsset
6
updateforce()
4.
createairlayer()
5.
addairlayer(aircraftclass)
AirMinelayer

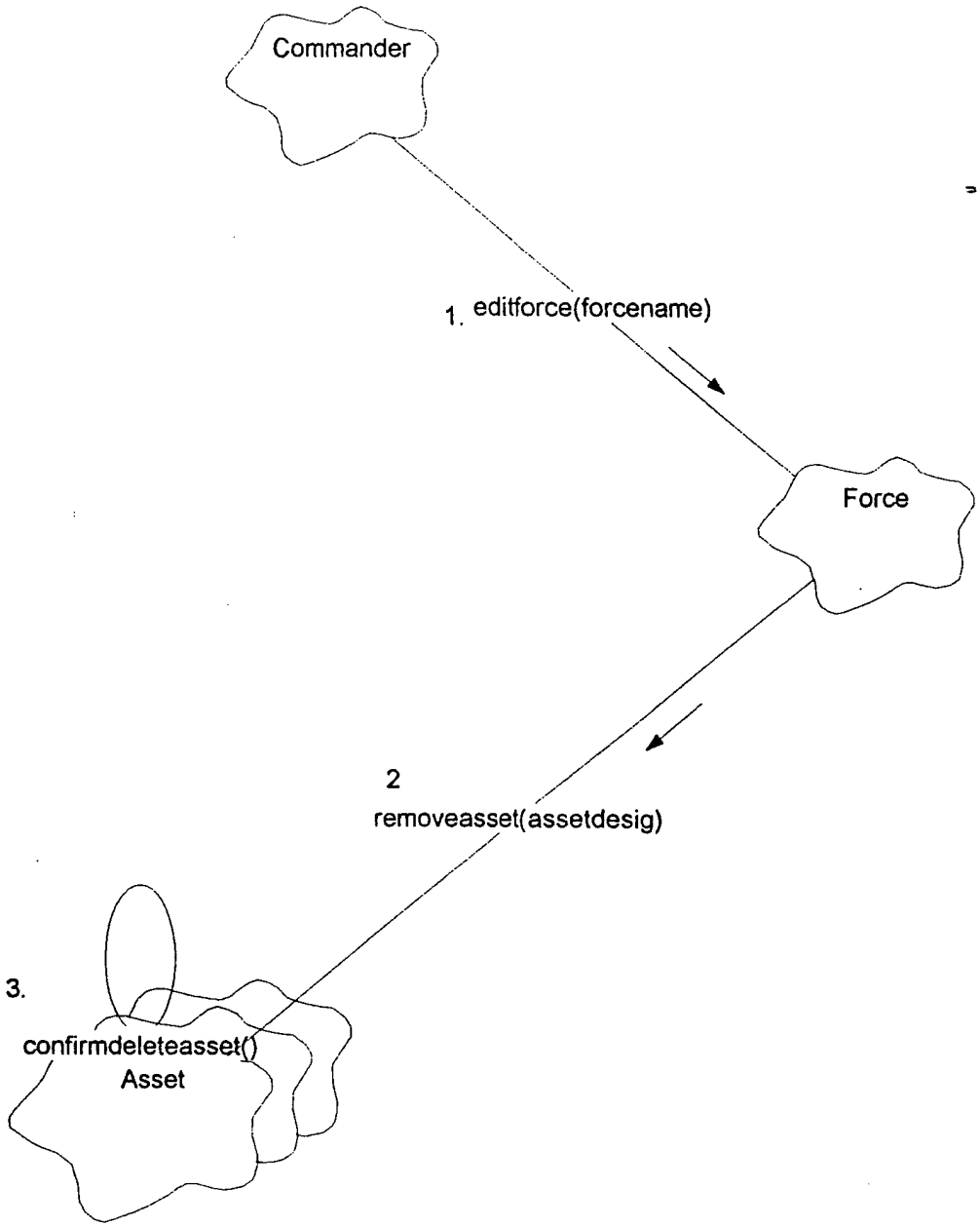
Scenario: Add New Submarine Minelayer

1. The user selects "Edit Force" on the Commander's Display.
2. The user selects "Add Asset". The "Add Asset screen appears".
3. The user enters the military designation of the new asset and selects "Submarine Minelayer" as the asset type.
4. A submarine minelayer object is created.
5. The user selects the hull number for the submarine minelayer.
6. The Mining Force is updated with the new submarine minelayer.


MiningCDR
1.
editforce(forcename)
MiningForce
2. createasset()
3
addasset(militarydesig,a
MiningAsset
6
updateforce()
4.
createsublayer()
5.
addsublayer(ssn_class)
SubMinelayer

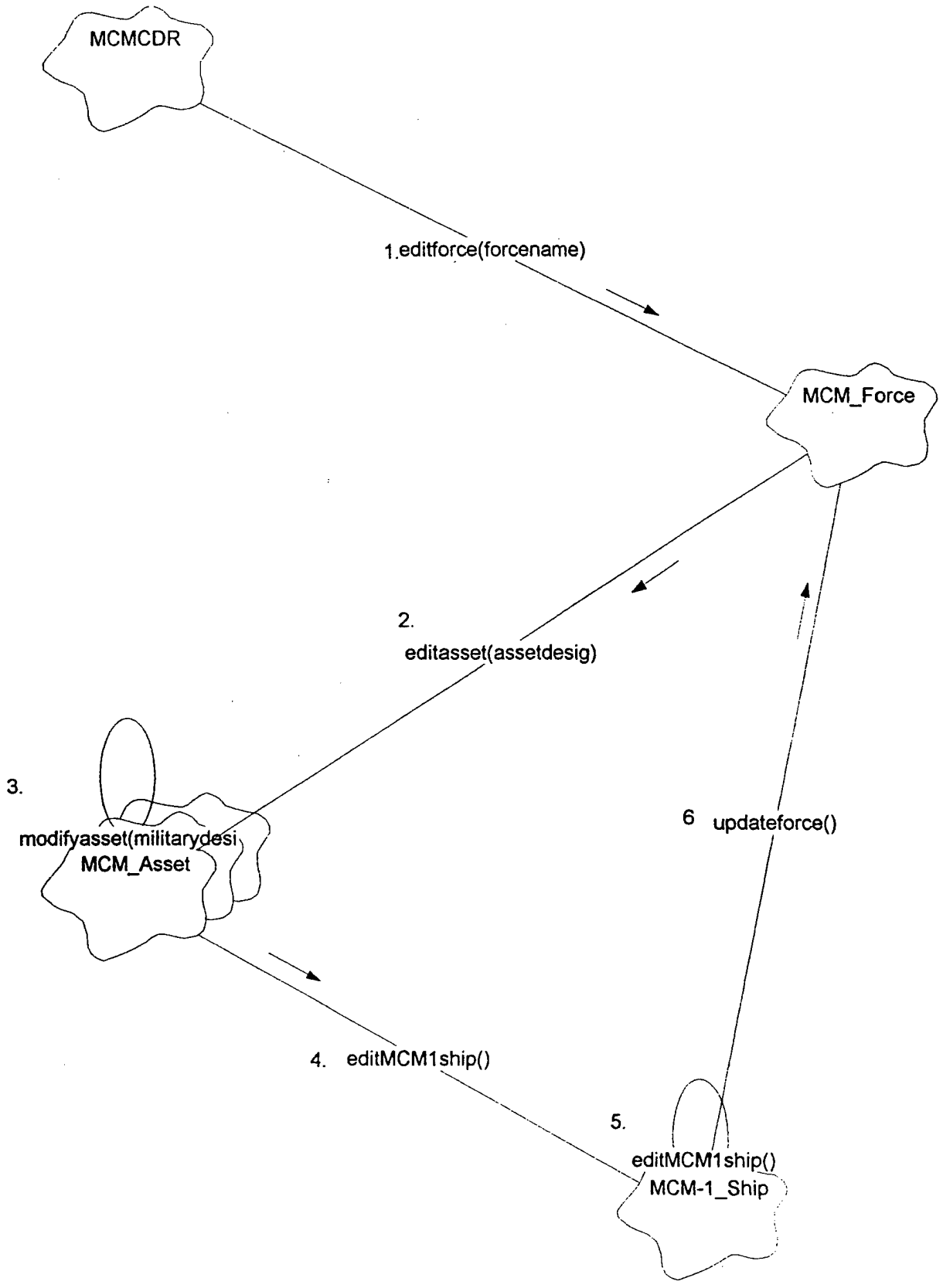
Scenario: Delete Asset From Force

1. The user selects "Edit Force" on the Commander's Display.
2. The user selects an asset and "Delete Asset".
3. The user confirms the deletion of the asset.


Commander
1. editforce(forcename)
Force
2
removeasset(assetdesig)
3.
confirmdeleteasset()
Asset

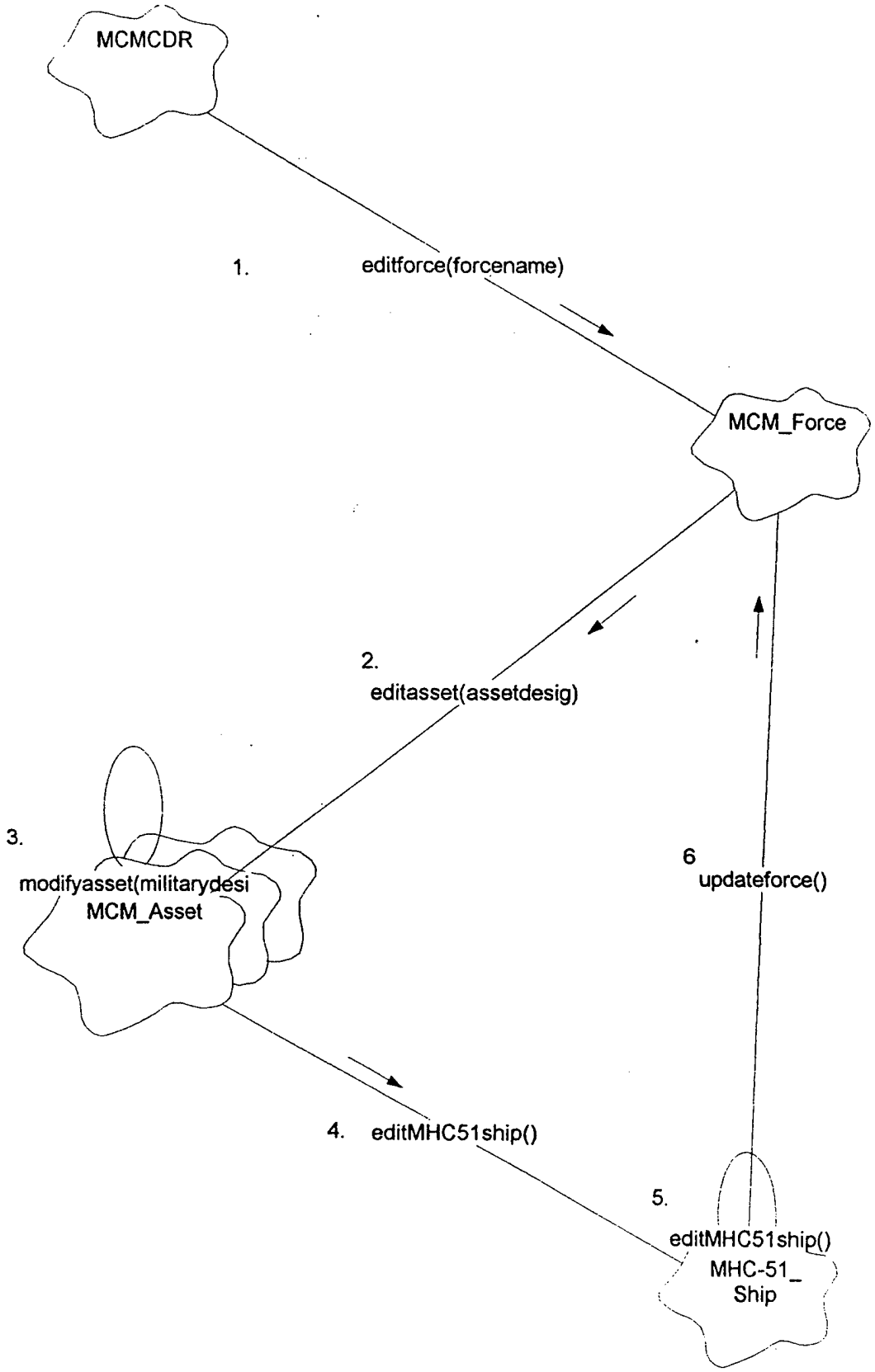
Scenario: Edit MCM-1 Member of the Force

1. The user selects "Edit Force" on the Commander's Display.
2. The user selects an asset (in this case a MCM-1 ship) and "Edit Asset".
3. The user edits the common MCM asset data.
4. The user edits the common SMCM ship data.
5. The user edits the specific data for the MCM-1 ship.
6. The MCM Force is updated with the new data for the MCM-1 ship.


MCMCDR
1.editforce(forcename)
MCM_Force
2.
editasset(assetdesig)
3.
modifyasset(militarydesi
MCM_Asset
6 updateforce()
4. editMCM1ship()
5.
editMCM1ship()
MCM-1_Ship

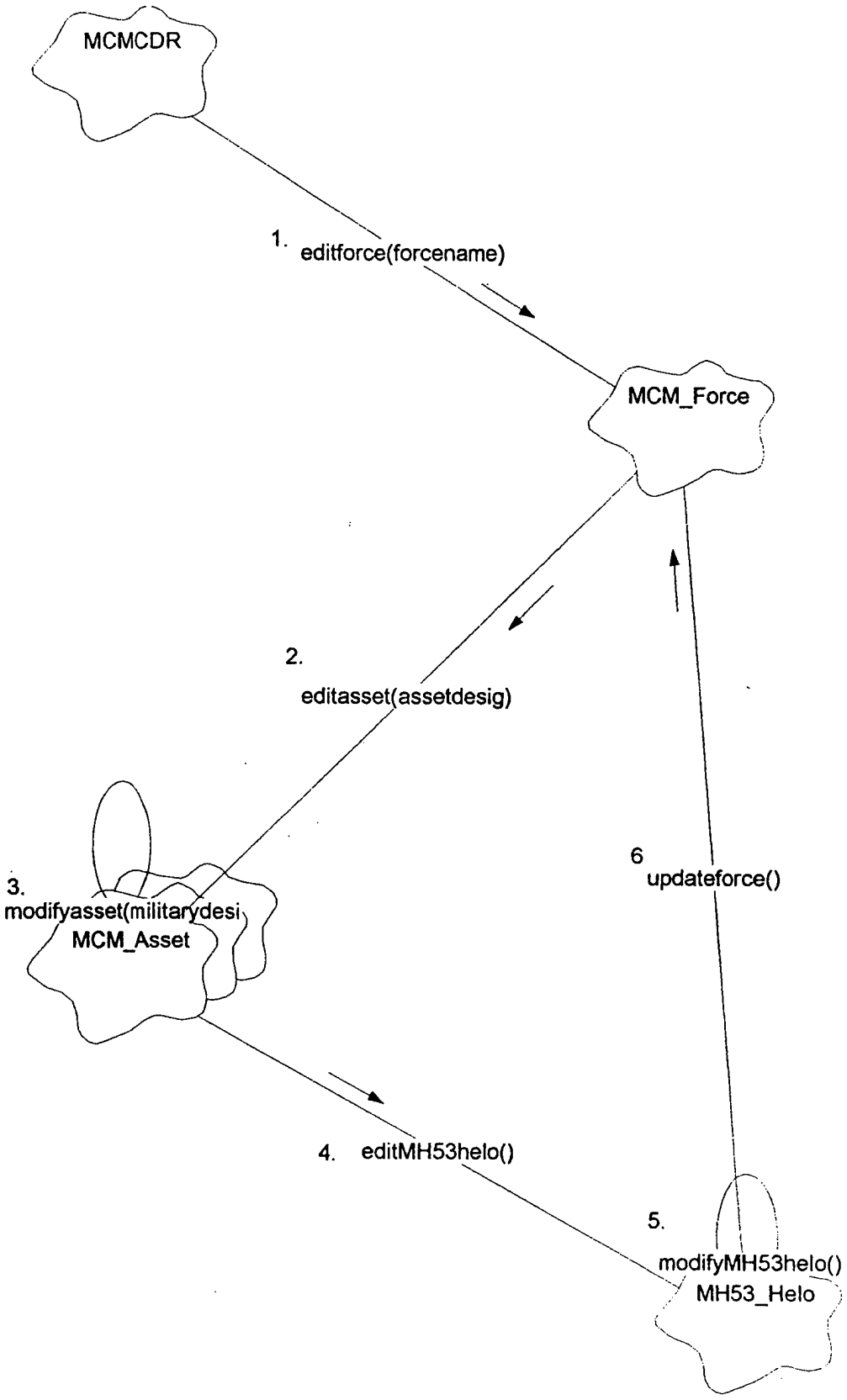
Scenario: Edit MHC-51 Member of the Force

1. The user selects "Edit Force" on the Commander's Display.
2. The user selects an asset (in this case a MHC-51 ship) and "Edit Asset".
3. The user edits the common MCM asset data.
4. The user edits the common SMCM ship data.
5. The user edits the specific data for the MHC-51 ship.
6. The MCM Force is updated with the new data for the MHC-51 ship.


MCMCDR
1.
editforce(forcename)
MCM_Force
2.
editasset(assetdesig)
3.
modifyasset(militarydesi
MCM_Asset
6
updateforce()
4.
editMHC51ship()
5.
editMHC51ship()
MHC-51_
Ship

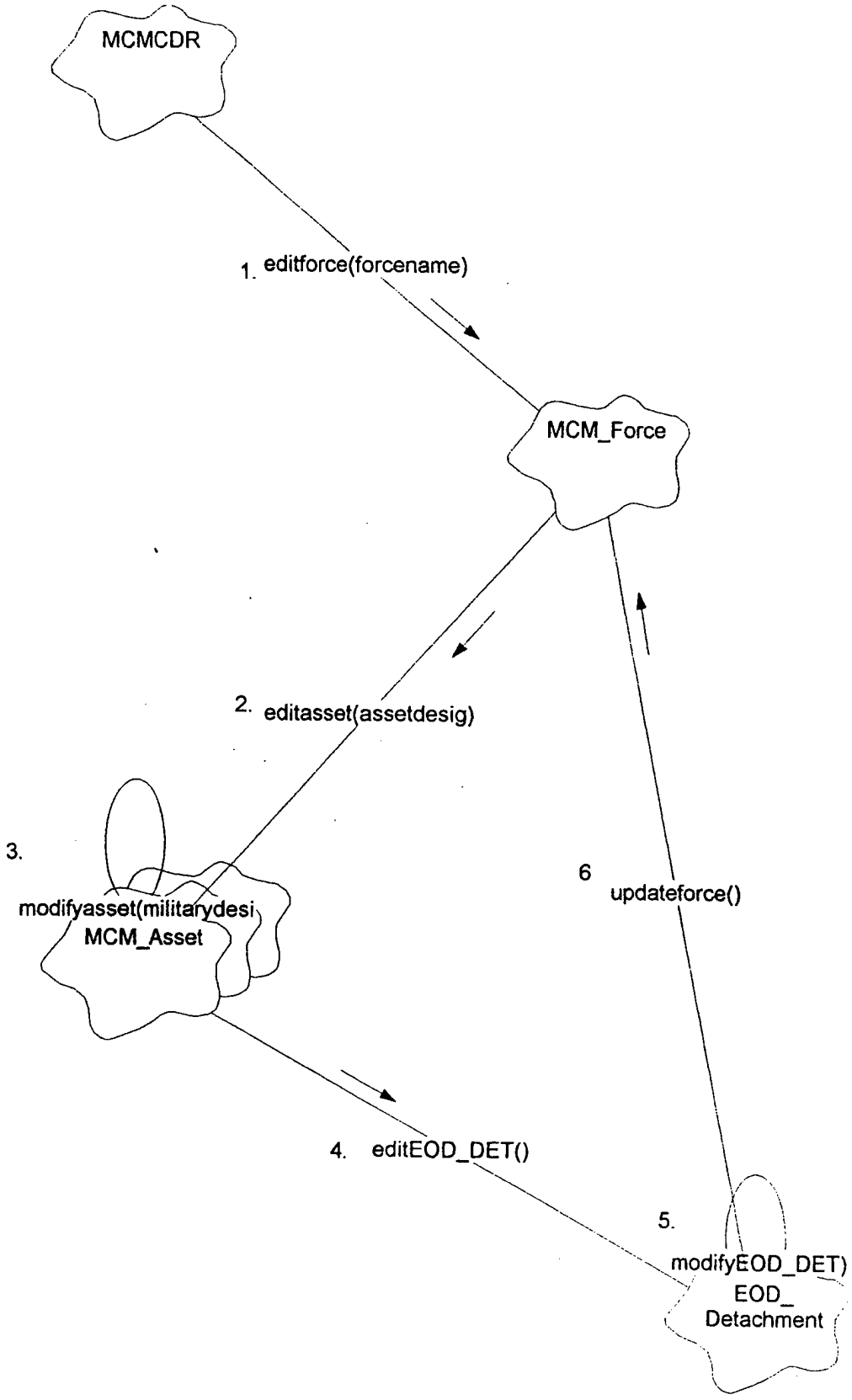
Scenario: Edit MH-53E Member of the Force

1. The user selects "Edit Force" on the Commander's Display.
2. The user selects an asset (in this case a MH-53E helicopter) and "Edit Asset".
3. The user edits the common MCM asset data.
4. The user edits the common AMCM helicopter data.
5. The user edits the specific data for the MH-53E helicopter.
6. The MCM Force is updated with the new data for the MH-53E helicopter.


MCMCDR
1. editforce(forcename)
MCM_Force
2.
editasset(assetdesig)
3.
modifyasset(militarydesi
MCM_Asset
6
updateforce()
4. editMH53helo()
5.
modifyMH53helo()
MH53_Helo

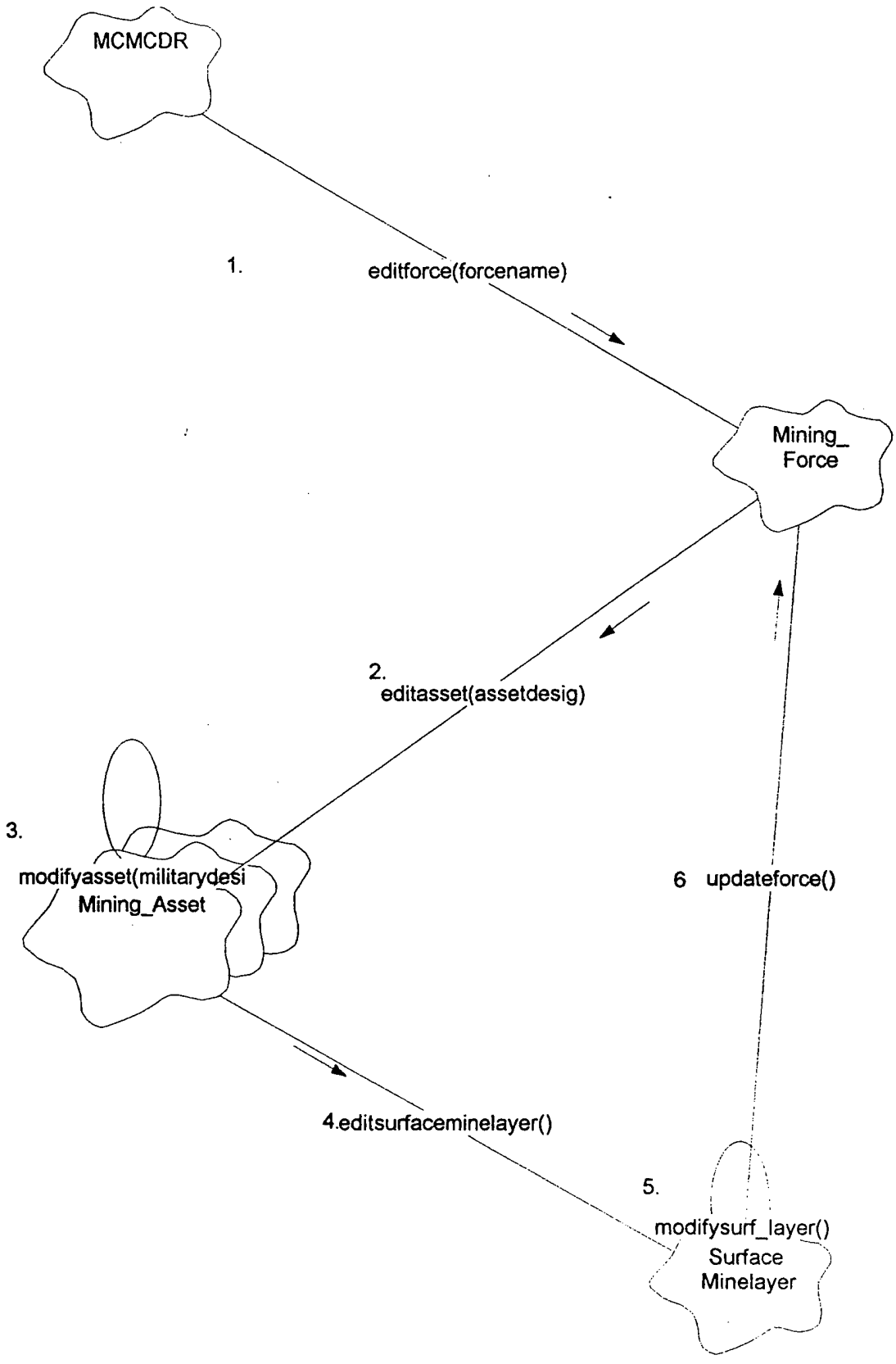
Scenario: Edit EOD Detachment Member of the Force

1. The user selects "Edit Force" on the Commander's Display.
2. The user selects an asset (in this case a EOD Detachment) and "Edit Asset".
3. The user edits the common MCM asset data.
4. The user edits the specific data for the EOD Detachment.
5. The MCM Force is updated with the new data for the EOD Detachment.


MCMCDR
1. editforce(forcename)
MCM_Force
2. editasset(assetdesig)
3.
modifyasset(militarydesi
MCM_Asset
6
updateforce()
4. editEOD_DET()
5.
modifyEOD_DET)
EOD_
Detachment

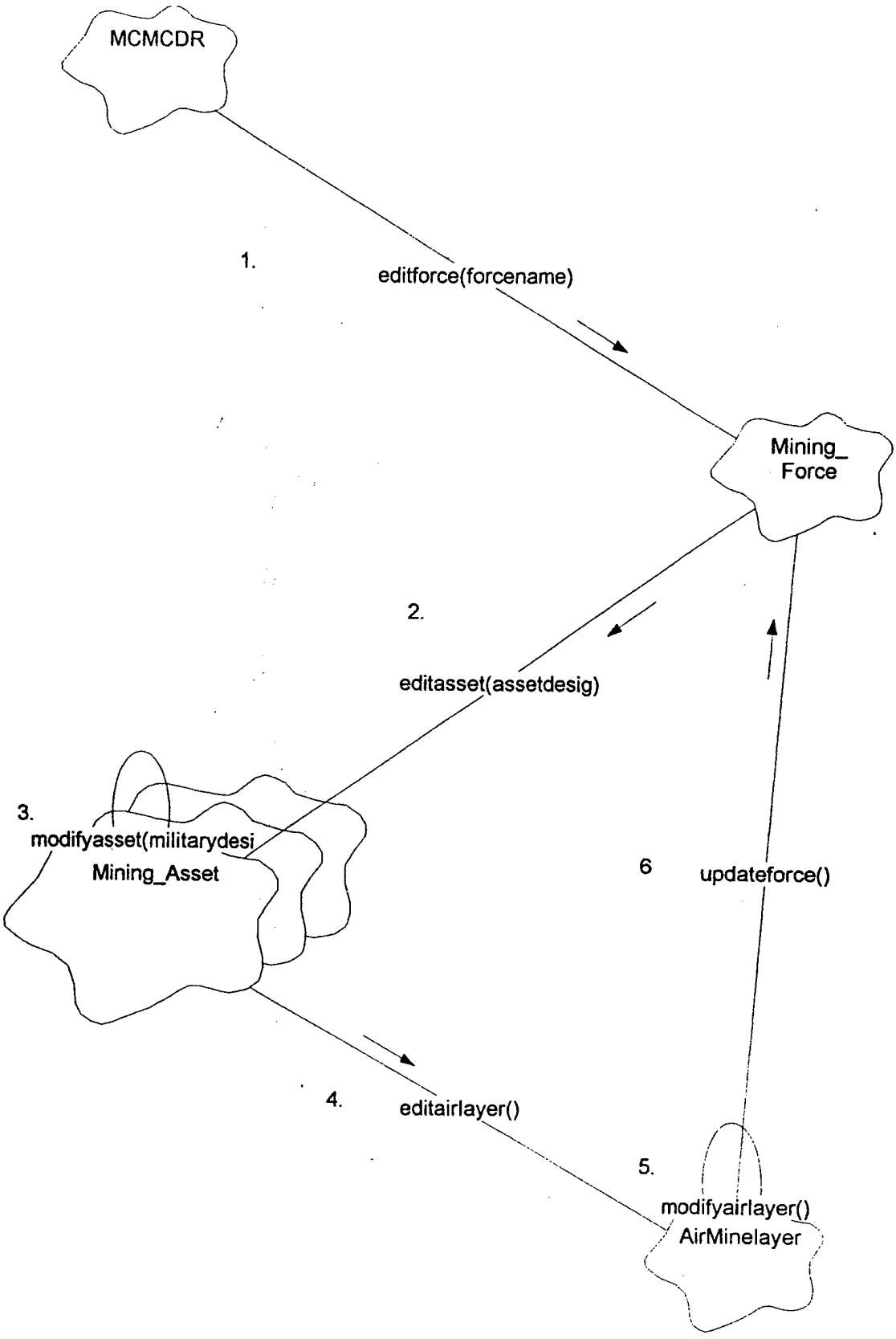
Scenario: Edit Surface Minelayer Member of the Force

1. The user selects "Edit Force" on the Commander's Display.
2. The user selects an asset (in this case a surface minelayer) and "Edit Asset".
3. The user edits the common mining asset data.
4. The user edits the specific data for the surface minelayer.
5. The MCM Force is updated with the new data for the surface minelayer.


MCMCDR
1.
editforce(forcename)
Mining_
Force
2.
editasset(assetdesig)
3.
modifyasset(militarydesi
Mining_Asset
6 updateforce()
4.editsurfaceminelayer()
5.
modifysurf_layer()
Surface
Minelayer

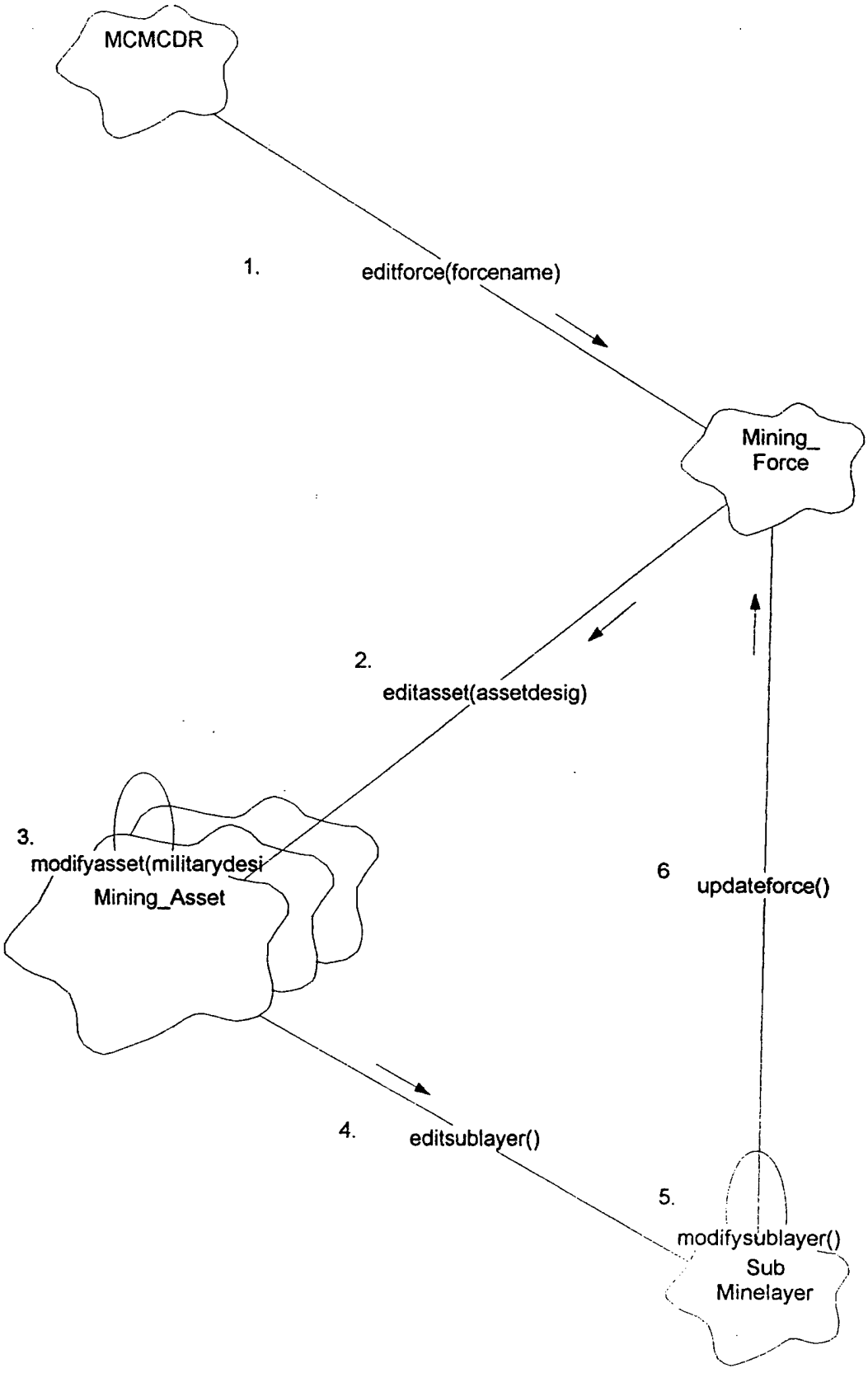
Scenario: Edit Air Minelayer of the Force

1. The user selects "Edit Force" on the Commander's Display.
2. The user selects an asset (in this case an air minelayer) and "Edit Asset".
3. The user edits the common MCM asset data.
4. The user edits the specific data for the air minelayer.
5. The MCM Force is updated with the new data for the air minelayer.


MCMCDR
1.
editforce(forcename)
Mining_
Force
2.
editasset(assetdesig)
3.
modifyasset(militarydesi
Mining_Asset
6
updateforce()
4.
editairlayer()
5.
modifyairlayer()
AirMinelayer

Scenario: Edit Submarine Minelayer Member of the Force

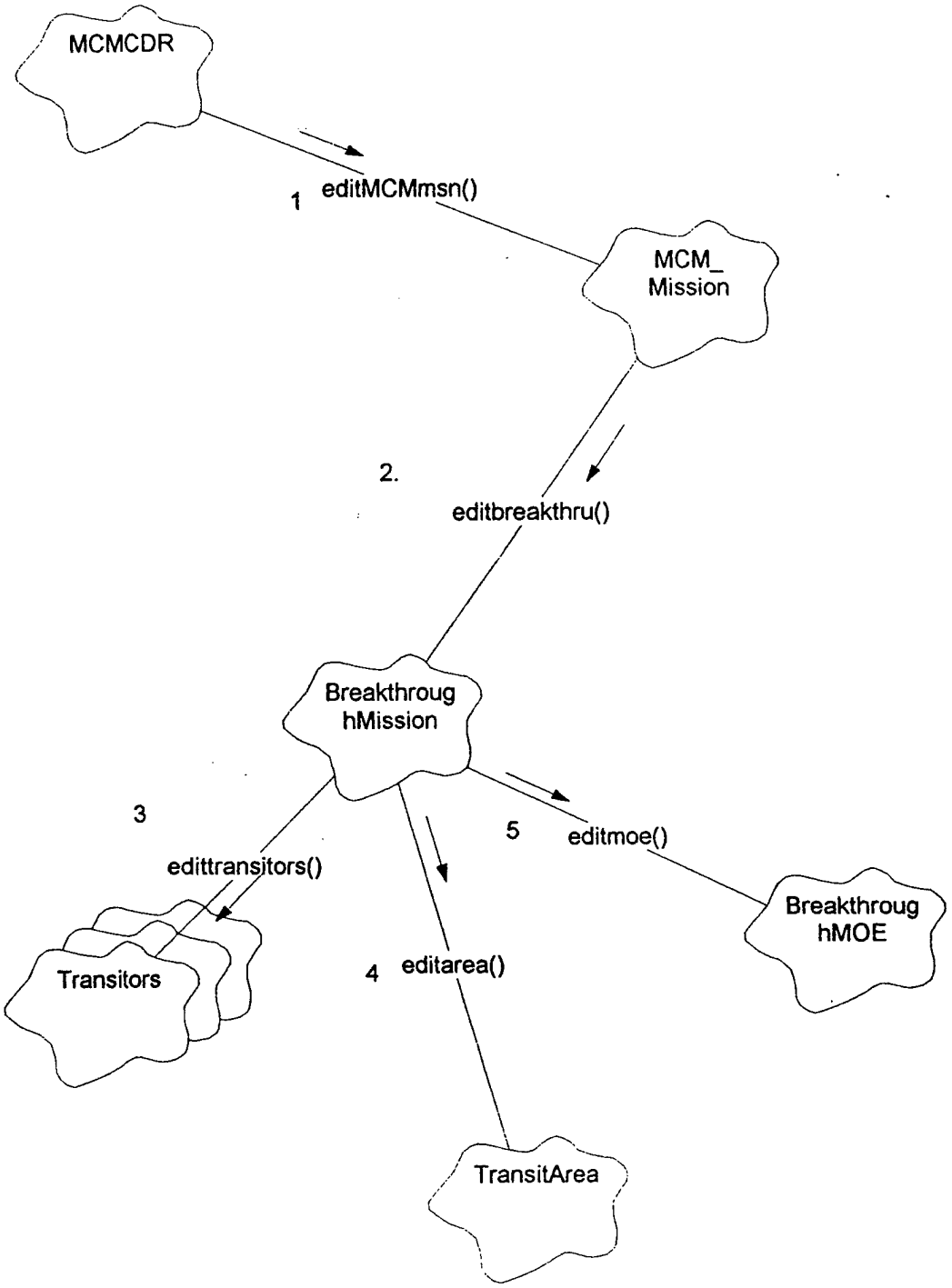
1. The user selects "Edit Force" on the Commander's Display.
2. The user selects an asset (in this case a submarine minelayer) and "Edit Asset".
3. The user edits the common MCM asset data.
4. The user edits the specific data for the submarine minelayer.
5. The MCM Force is updated with the new data for the submarine minelayer.


MCMCDR
1.
editforce(forcename)
Mining_
Force
2.
editasset(assetdesig)
3.
modifyasset(militarydesi
Mining_Asset
6
updateforce()
4.
editsublayer()
5.
modifysublayer()
Sub
Minelayer

Use Case: Specify the Mission and MOE

Scenario: Specify Breakthrough Mission and MOE

1. The user selects "Edit Mission" from the Commander's Main Menu. This brings up the Select Mission display.
2. The user selects "Breakthrough Mission". The Edit Breakthrough Mission display appears.
3. The user selects "Transitors" and selects or adds tranisting ships.
4. The user selects "Transit Area" and selects an area.
5. The user selects "Breakthrough MOE" and selects the primary MOE and secondary MOEs.


MCMCDR
1
editMCMmsn()
MCM_
Mission
2.
editbreakthru()
Breakthroug
hMission
3
edittransitors()
Transitors
4
editarea()
TransitArea
5
editmoe()
Breakthroug
hMOE

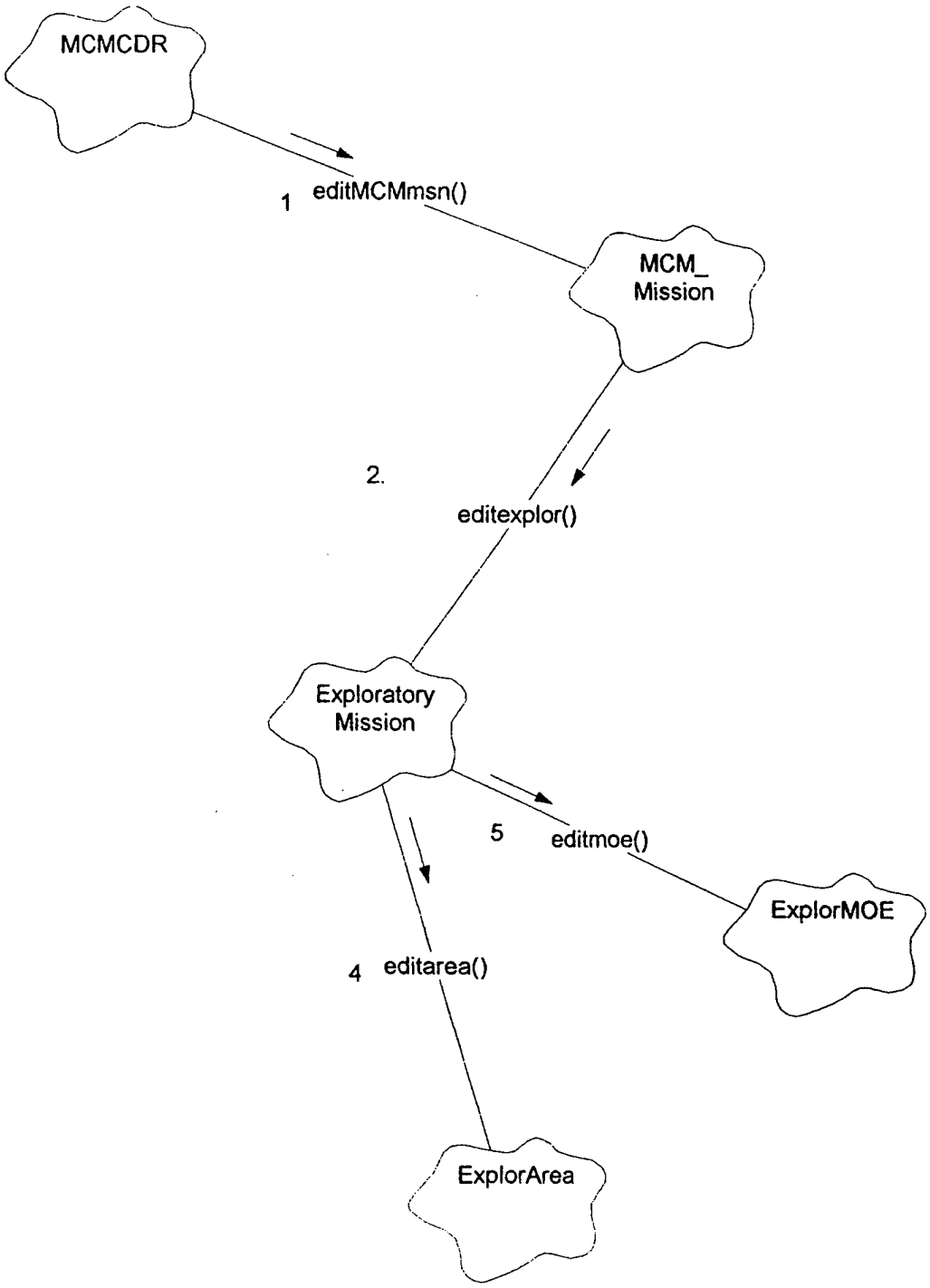
Scenario: Specify Exploratory MCM Mission and MOE

1. The user selects "Edit Mission" from the Commander's Main Menu. This brings up the Select Mission display.

2. The user selects "Exploratory MCM Mission". The Edit Exploratory MCM Mission display appears.

3. The user selects "Exploratory MCM Area" and selects an area.

4. The user selects "Exploratory MCM MOE" and selects the primary MOE and secondary MOEs.


MCMCDR
1 editMCMmsn()
MCM_
Mission
2.
editexplor()
Exploratory
Mission
5 editmoe()
ExplorMOE
4 editarea()
ExplorArea

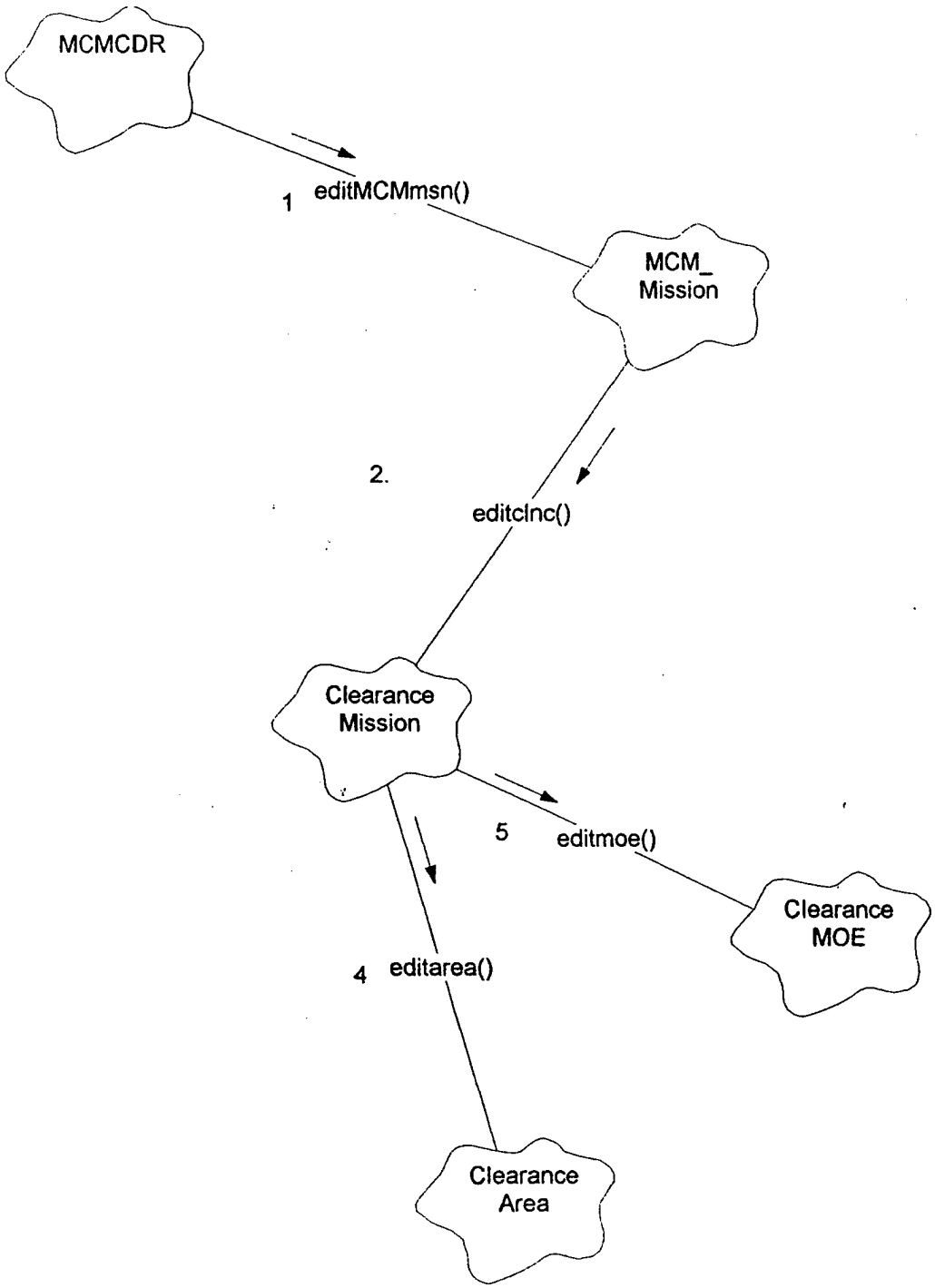
Scenario: Specify Clearance Mission and MOE

1. The user selects "Edit Mission" from the Commander's Main Menu. This brings up the Select Mission display.

2. The user selects "Clearance Mission". The Edit Clearance MCM Mission display appears.

3. The user selects "Clearance Area" and selects an area.

4. The user selects "Clearance MOE" and selects the primary MOE and secondary MOEs.


MCMCDR
1 editMCMmsn()
MCM_
Mission
2.
editclnc()
Clearance
Mission
5 editmoe()
Clearance
MOE
4 editarea()
Clearance
Area

Scenario: Specify Damage High Value Unit Mission and MOE

1. The user selects "Edit Mission" from the Commander's Main Menu. This brings up the Select Mission display.

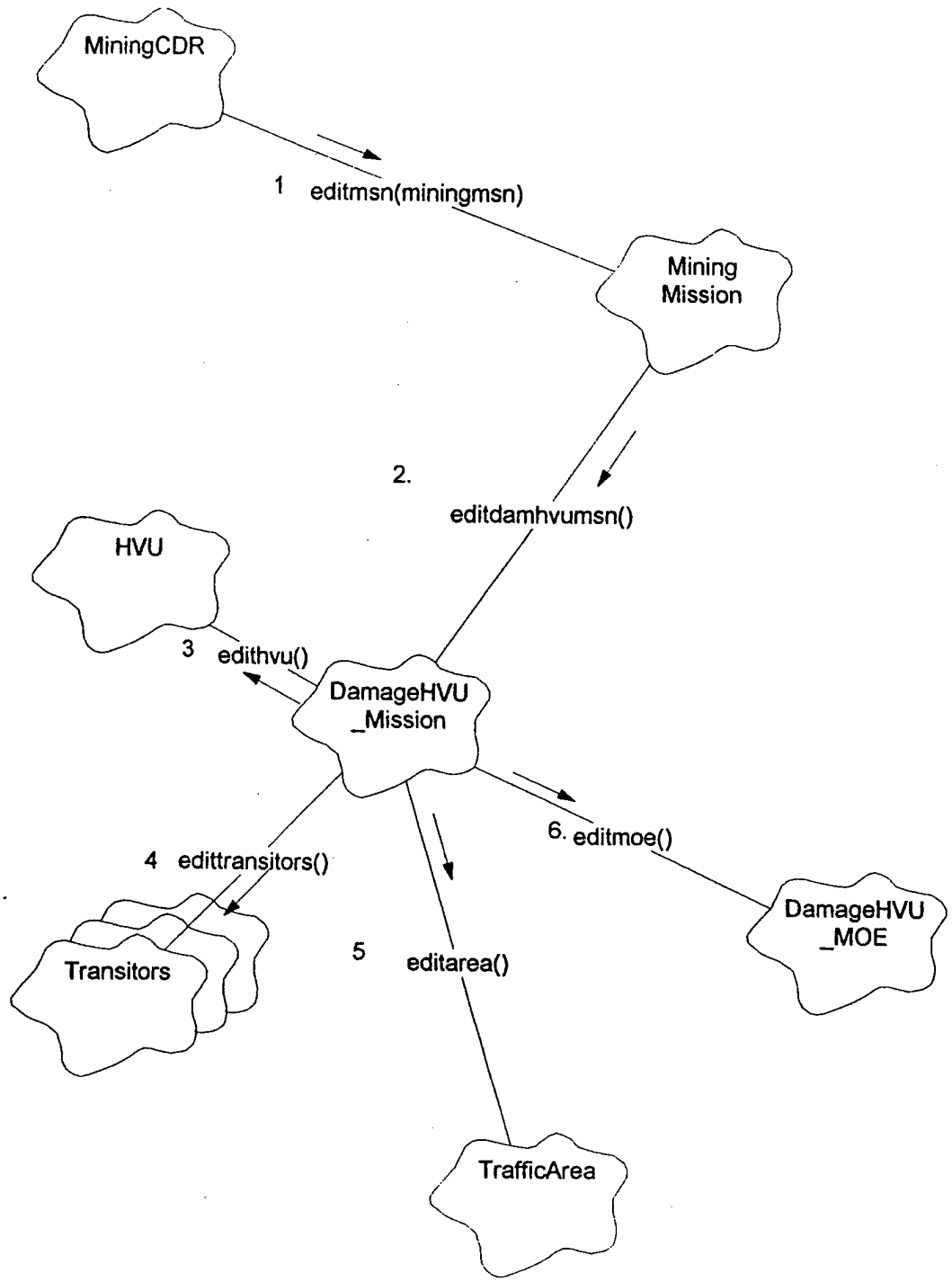
2. The user selects "Damage High Value Unit". The Edit Damage HVU Mission display appears.

3. The user selects or adds the HVU.

4. The user selects "Transitors" and selects or adds tranisting ships.

5 The user selects "Traffic Area" and selects an area.

6. The user selects "Damage HVU MOE" and selects the primary MOE and secondary MOEs.


MiningCDR
1 editmsn(miningmsn)
Mining Mission
2.
editdamhvumsn()
HVU
3 edithvu()
DamageHVU _Mission
4 edittransitors()
Transitors
5
editarea()
TrafficArea
6. editmoe()
DamageHVU _MOE

Scenario: Specify Antitonnage Mission and MOE

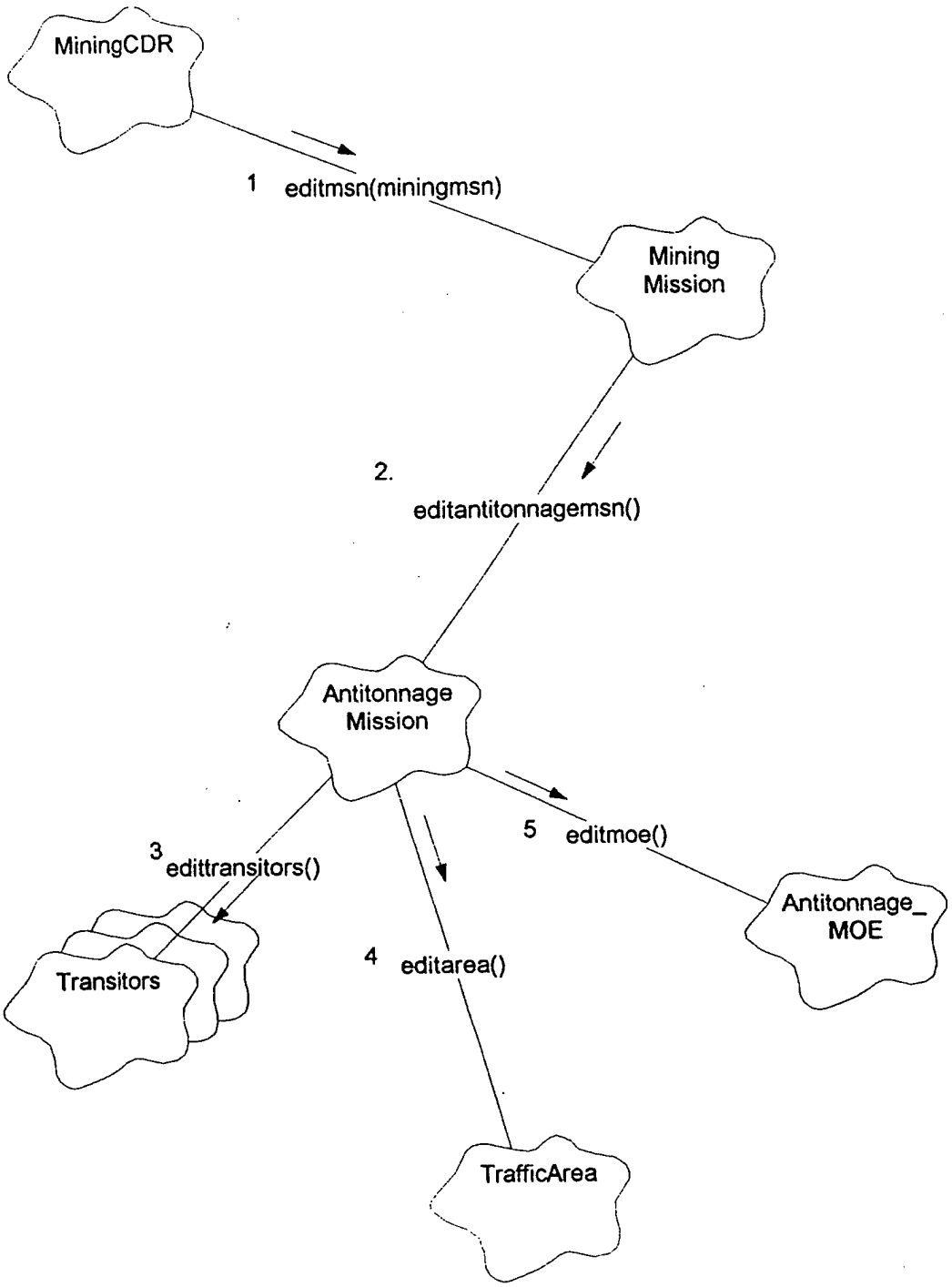
1. The user selects "Edit Mission" from the Commander's Main Menu. This brings up the Select Mission display.

2. The user selects "Anti-Tonnage Mission". The Edit Anti-Tonnage Mission display appears.

3. The user selects "Transitors" and selects or adds tranisting ships.

4. The user selects "Traffic Area" and selects an area.

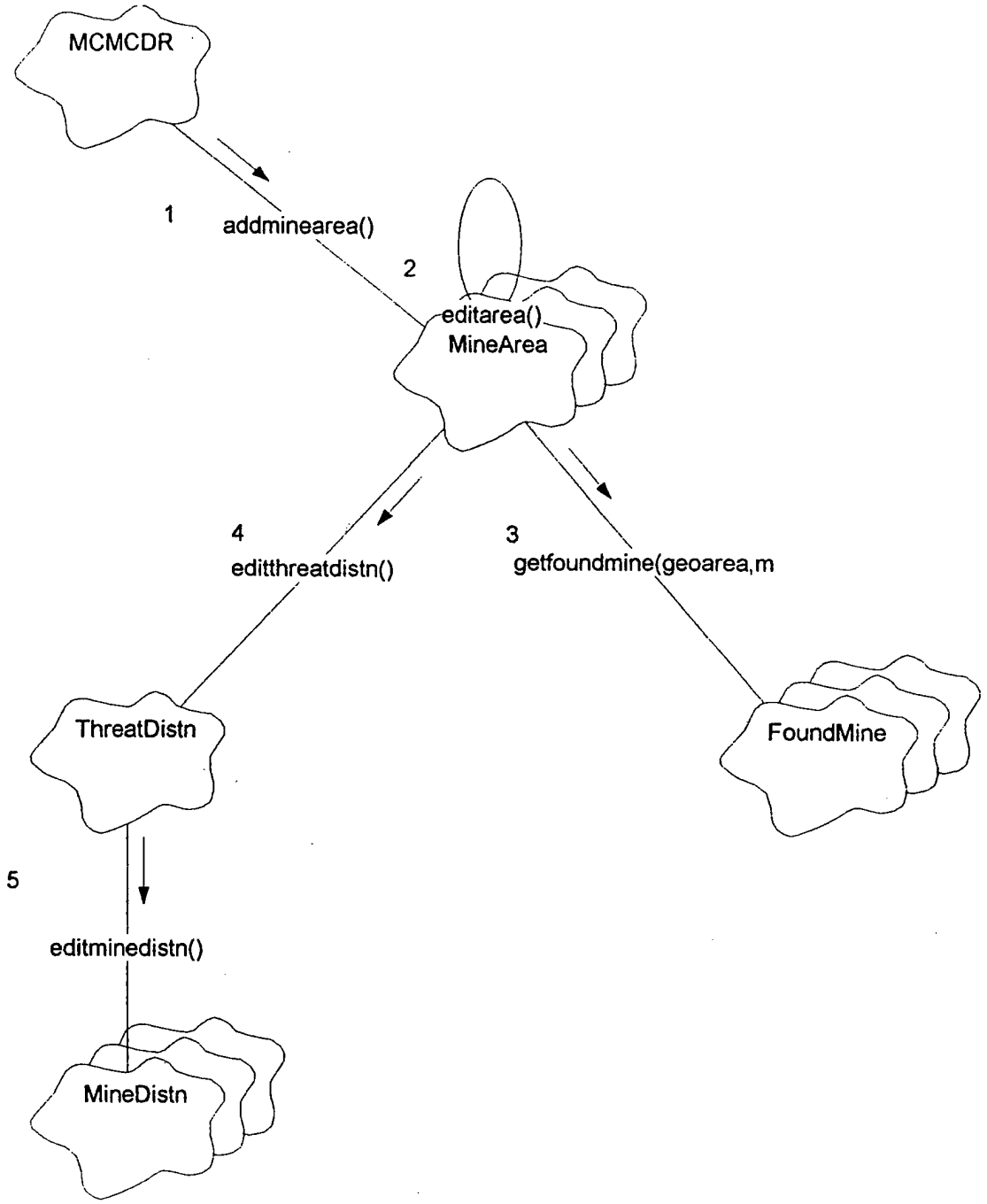
5. The user selects "Antitonnage MOE" and selects the primary MOE and secondary MOEs.


MiningCDR
1 editmsn(miningmsn)
Mining Mission
2.
editantitonnagemsn()
Antitonnage Mission
3 edittransitors()
Transitors
4 editarea()
TrafficArea
5 editmoe()
Antitonnage_ MOE

Use Case: Specify Mine Area

Scenario: Specify Mine Area

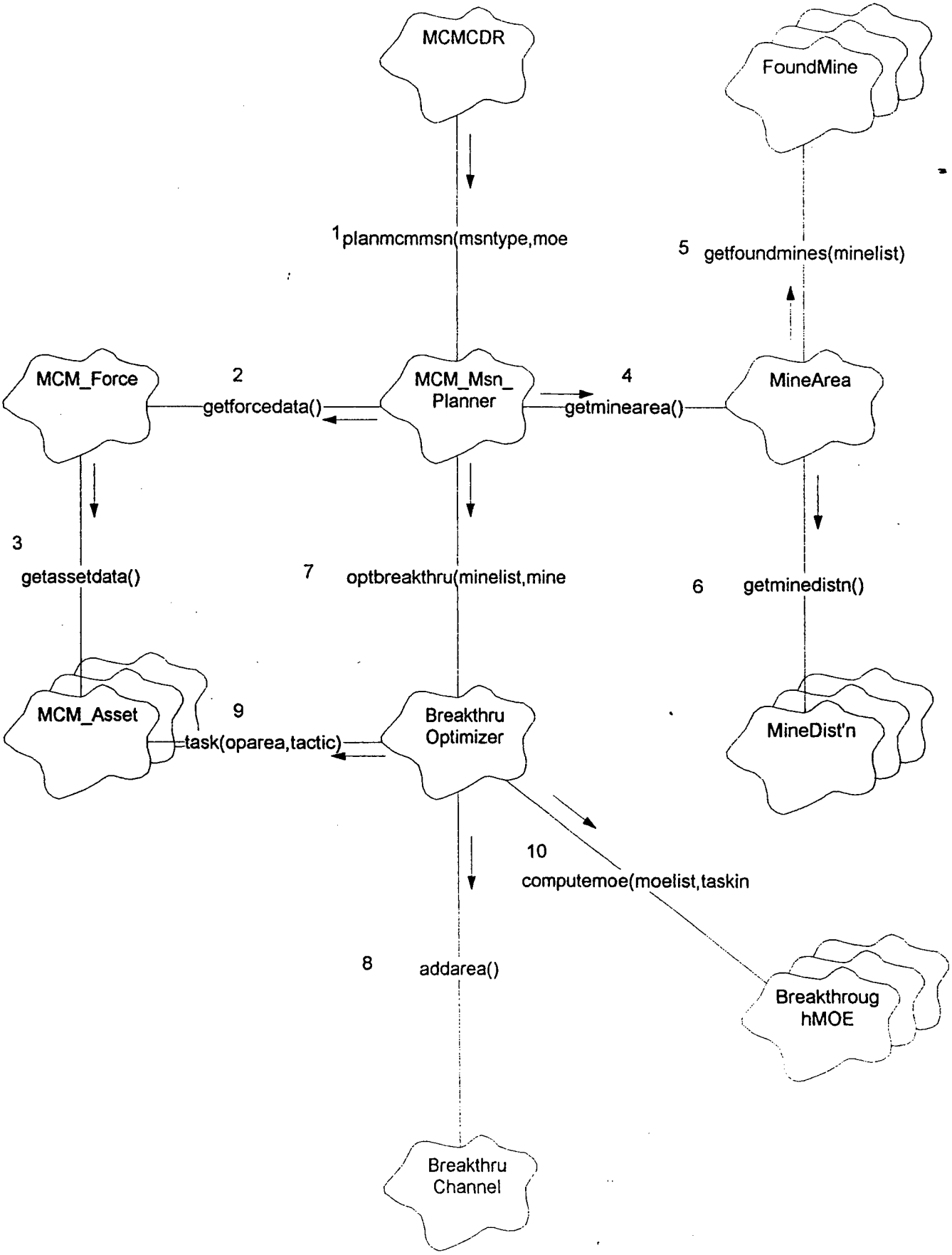
1. The user selects "Mine Area" from the Commander's Main Menu. This brings up the Select Mine Area display.
2. The user edits the area.
3. The Mine Area gets the identification and location of found mines in the mine area and displays them.
4. The Edit Mine Distribution display appears for the user to edit the mine distribution parameters associated with the mine area.


MCMCDR
1
addminearea()
2
editarea()
MineArea
4
editthreatdistn()
3
getfoundmine(geoarea,m
ThreatDistn
FoundMine
5
editminedistn()
MineDistn

Use Case: Plan MCM Mission

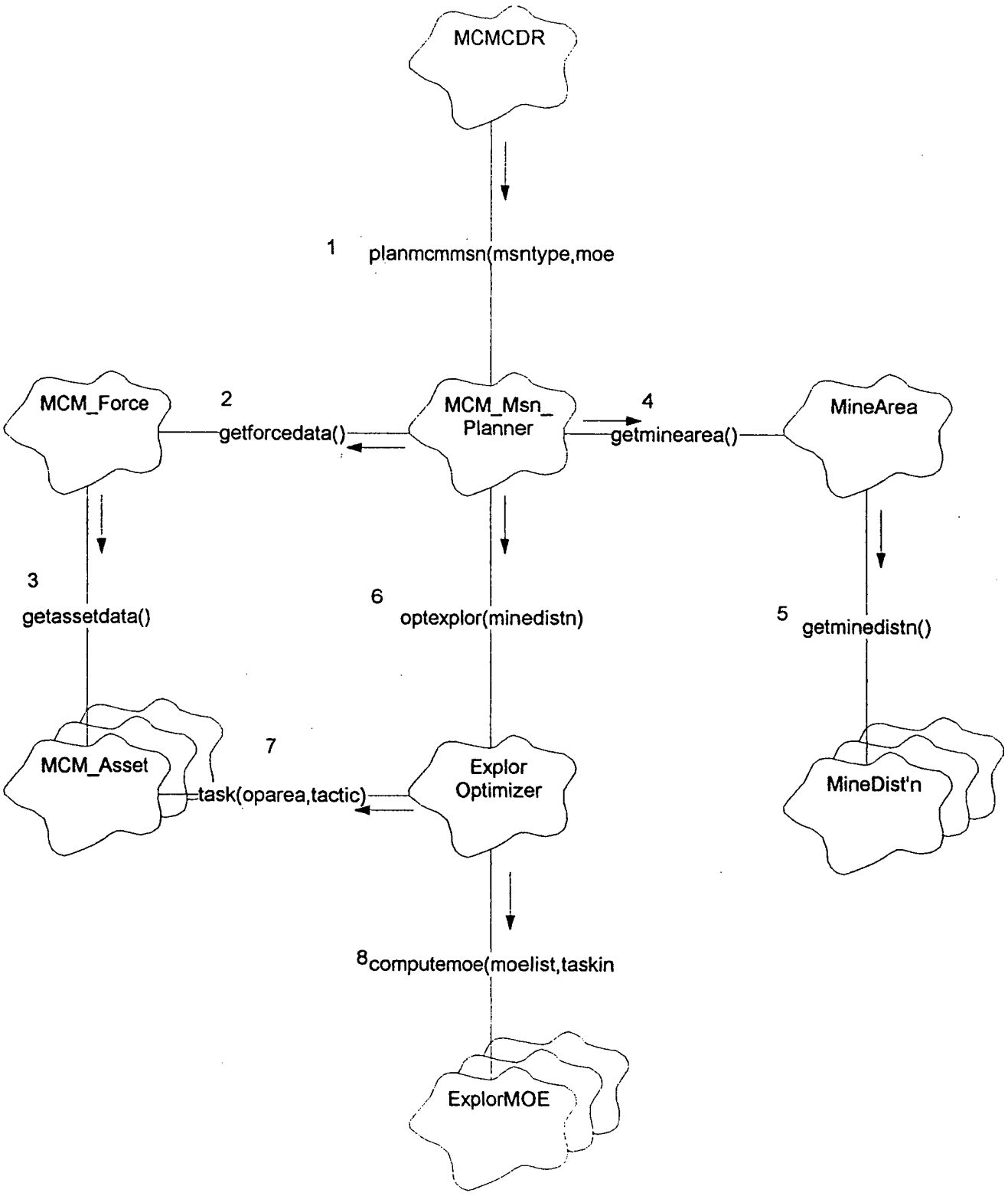
Scenario: Plan Breakthrough Mission

1. The MCM Commander selects "Plan Mission". A breakthrough mission and MOE set have already been selected.
2. The MCM Mission Planner calls the MCM Force for force data.
3. The MCM Force calls the assets for asset information (asset types and capabilities)
4. The MCM Mission Planner calls the Mine Area for information on found mines and threat mine distribution
5. The mine area calls the found mine database for found mines in the mine area.
6. The mine area calls the threat mine distribution.
7. The MCM Mission Planner invokes the Breakthrough Optimizer with the list of found mines, the threat mine distributions, the list of available assets and capabilities, and the Breakthrough MOE.
8. The Breakthrough Optimizer determines the channel for breakthrough.
9. The Breakthrough Optimizer assigns operating areas and tactics to MCM Assets.
10. The Breakthrough Optimizer calls the MOEs for values.


MCMCDR
FoundMine
1 planmcmmsn(msntype,moe
5 getfoundmines(minelist)
MCM_Force
2
getforcedata()
MCM_Msn_
Planner
4
getminearea()
MineArea
3
getassetdata()
7
optbreakthru(minelist,mine
6
getminedistn()
MCM_Asset
9
task(oparea,tactic)
Breakthru
Optimizer
MineDist'n
10
computemoe(moelist,taskin
8
addarea()
Breakthroug
hMOE
Breakthru
Channel

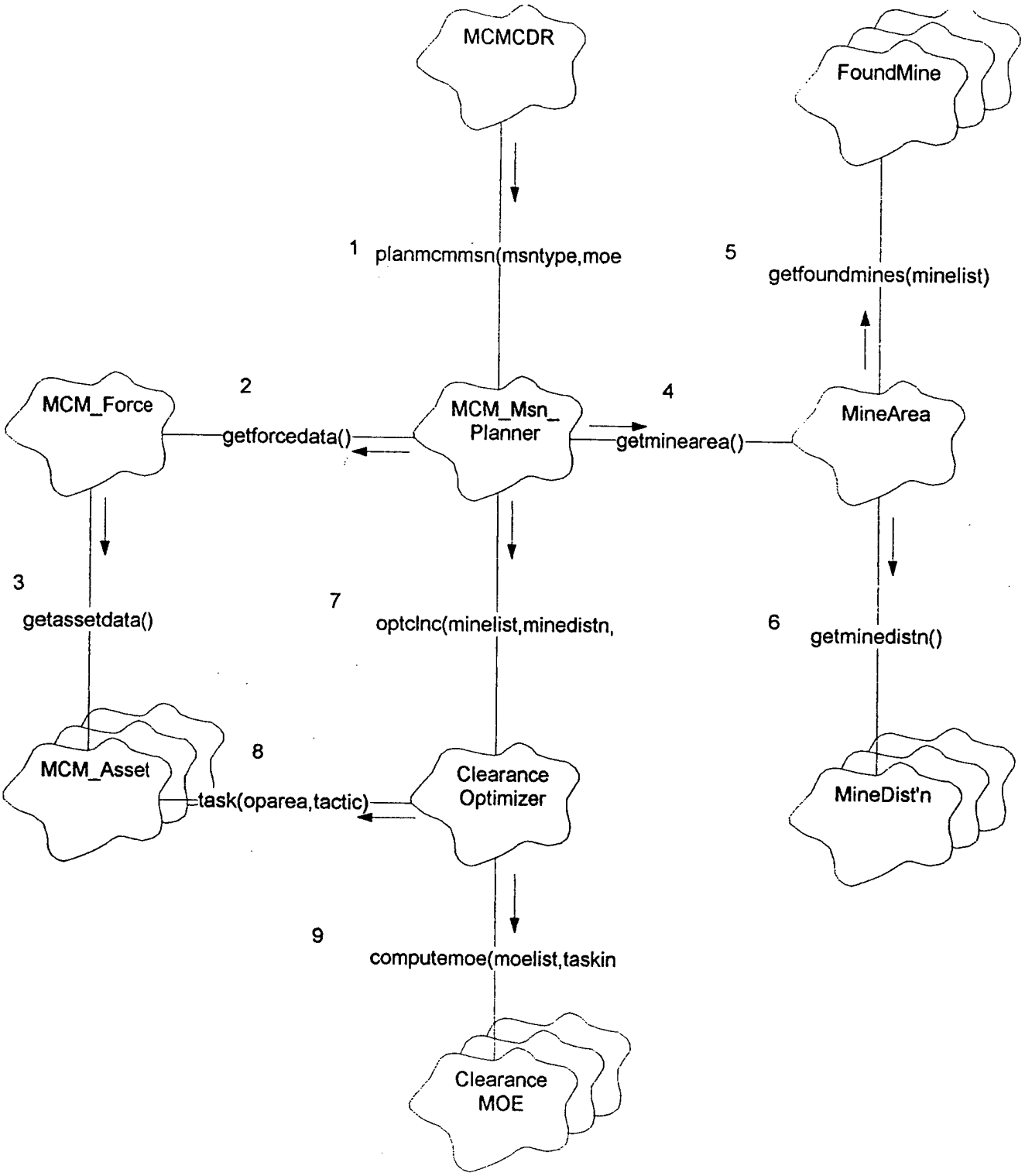
Scenario: Plan Exploratory MCM Mission

1. The MCM Commander selects "Plan Mission". An exploratory mission and MOE set have already been selected.
2. The MCM Mission Planner calls the MCM Force for force data.
3. The MCM Force calls the assets for asset information (asset types and capabilities)
4. The MCM Mission Planner calls the Mine Area for information on mine distributions.
5. The mine area calls the threat mine distribution.
6. The MCM Mission Planner invokes the Exploratory MCM Optimizer with the threat mine distributions, the list of available assets and capabilities, and the Exploratory MCM MOE.
7. The Exploratory MCM Optimizer assigns operating areas and tactics to MCM Assets.
8. The Exploratory Optimizer calls the MOEs for values.


MCMCDR
1
planmcmmsn(msntype,moe
MCM_Force
2
getforcedata()
MCM_Msn_
Planner
4
getminearea()
MineArea
3
getassetdata()
6
optexplor(minedistn)
5
getminedistn()
MCM_Asset
7
task(oparea,tactic)
Explor
Optimizer
MineDist'n
8computemoe(moelist,taskin
ExplorMOE

Scenario: Plan Clearance Mission

1. The MCM Commander selects "Plan Mission". A clearance mission and MOE set have already been selected.
2. The MCM Mission Planner calls the MCM Force for force data.
3. The MCM Force calls the assets for asset information (asset types and capabilities)
4. The MCM Mission Planner calls the Mine Area for information on found mines and threat mine distribution
5. The mine area calls the found mine database for found mines in the mine area.
6. The mine area calls the threat mine distribution.
7. The MCM Mission Planner invokes the Clearance Optimizer with the list of found mines, the threat mine distributions, the list of available assets and capabilities, and the Clearance MOE.
8. The Clearance Optimizer assigns operating areas and tactics to MCM Assets.
9. The Clearance Optimizer calls the MOEs for values.


MCMCDR
FoundMine
1 planmcmmsn(msntype,moe
5 getfoundmines(minelist)
MCM_Force
2
getforcedata()
MCM_Msn_
Planner
4
getminearea()
MineArea
3
getassetdata()
7
optclnc(minelist,minedistn,
6
getminedistn()
MCM_Asset
8
task(oparea,tactic)
Clearance
Optimizer
MineDist'n
9
computemoe(moelist,taskin
Clearance
MOE

Scenario: Specify Antitonnage Mission and MOE

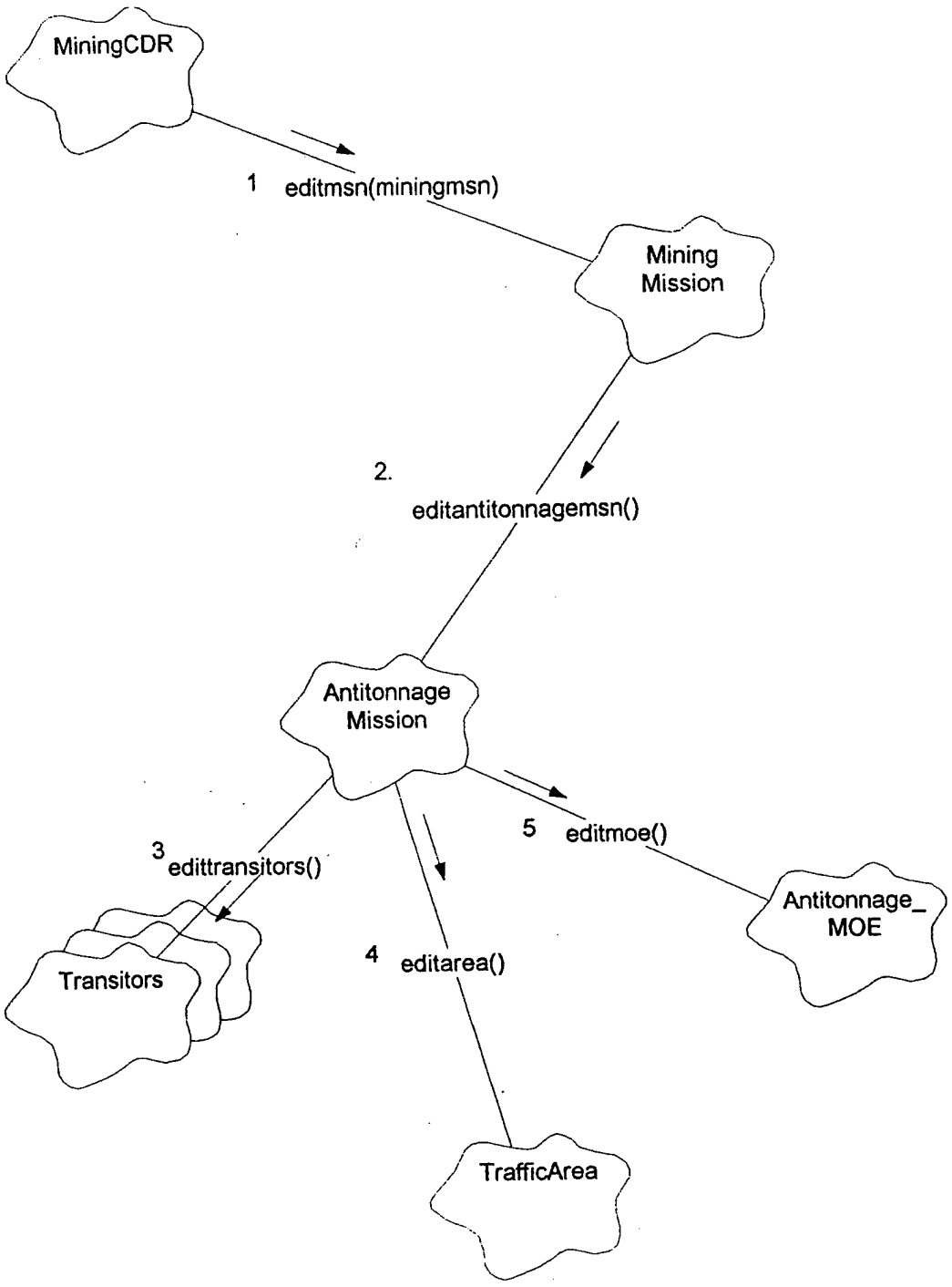
1. The user selects "Edit Mission" from the Commander's Main Menu. This brings up the Select Mission display.

2. The user selects "Anti-Tonnage Mission". The Edit Anti-Tonnage Mission display appears.

3. The user selects "Transitors" and selects or adds tranisting ships.

4. The user selects "Traffic Area" and selects an area.

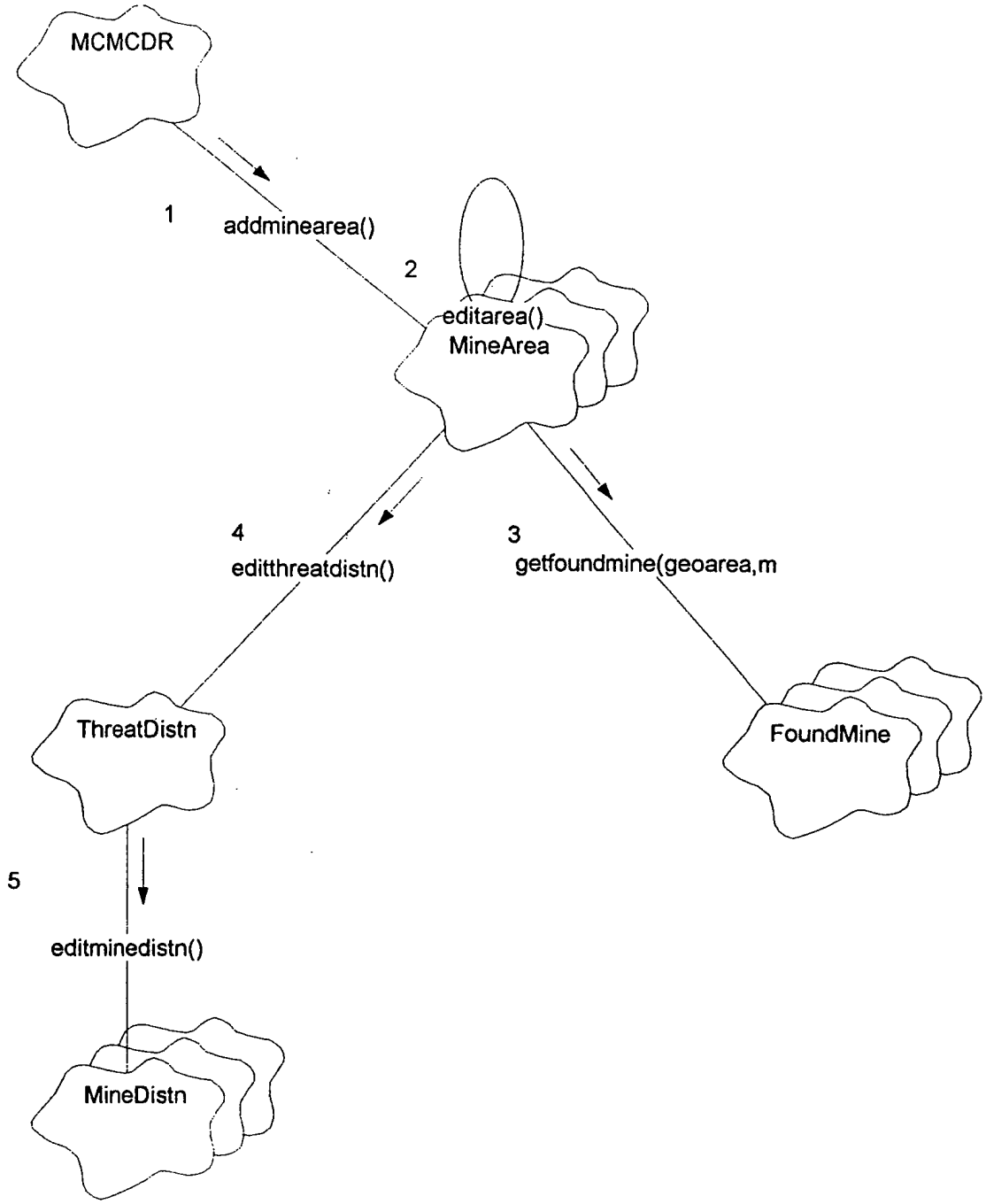
5. The user selects "Antitonnage MOE" and selects the primary MOE and secondary MOEs.


MiningCDR
1 editmsn(miningmsn)
Mining Mission
2. editantitonnagemsn()
Antitonnage Mission
3 edittransitors()
Transitors
4 editarea()
TrafficArea
5 editmoe()
Antitonnage_ MOE

Use Case: Specify Mine Area

Scenario: Specify Mine Area

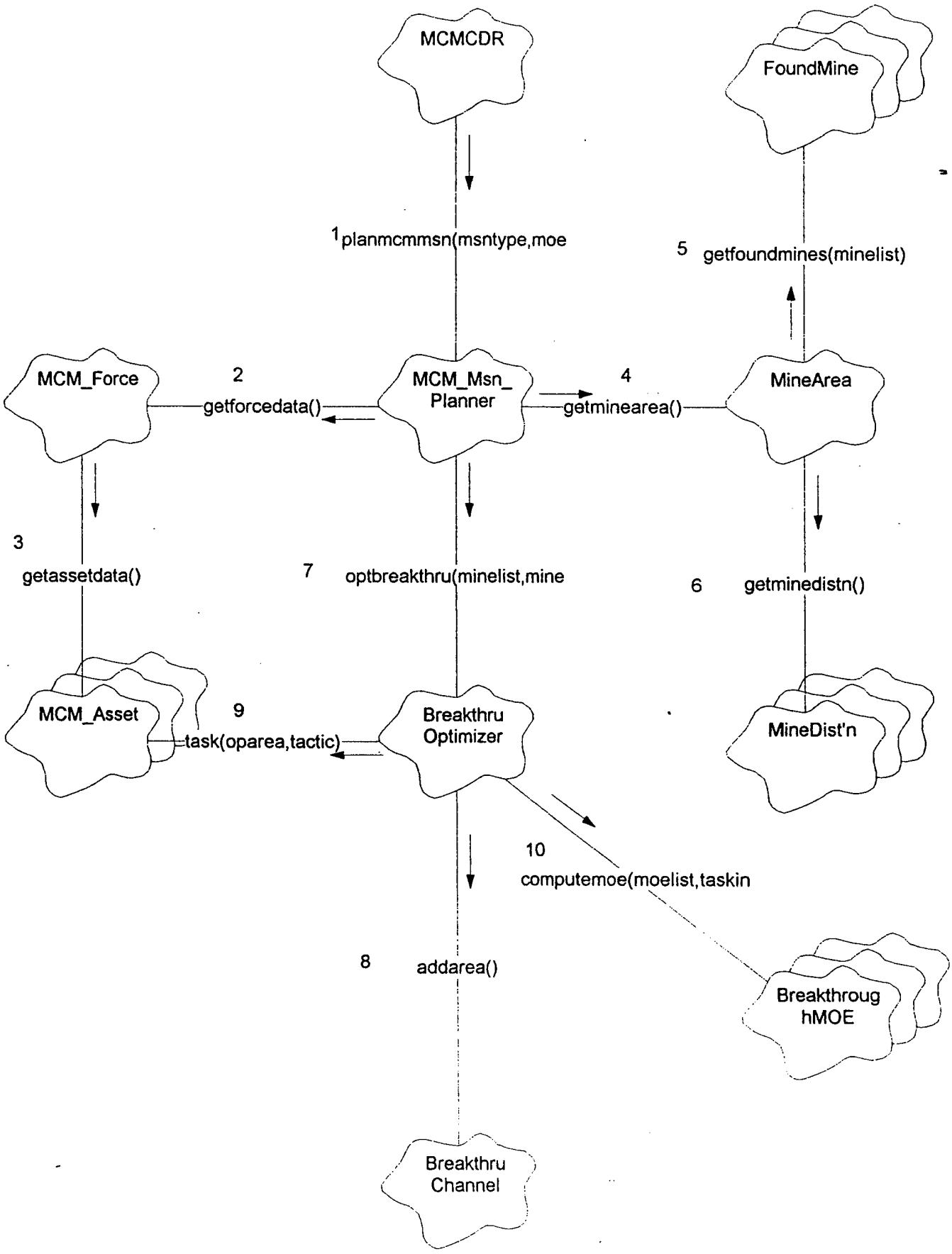
1. The user selects "Mine Area" from the Commander's Main Menu. This brings up the Select Mine Area display.
2. The user edits the area.
3. The Mine Area gets the identification and location of found mines in the mine area and displays them.
4. The Edit Mine Distribution display appears for the user to edit the mine distribution parameters associated with the mine area.


MCMCDR
1
addminearea()
2
editarea()
MineArea
4
editthreatdistn()
3
getfoundmine(geoarea,m
ThreatDistn
FoundMine
5
editminedistn()
MineDistn

Use Case: Plan MCM Mission

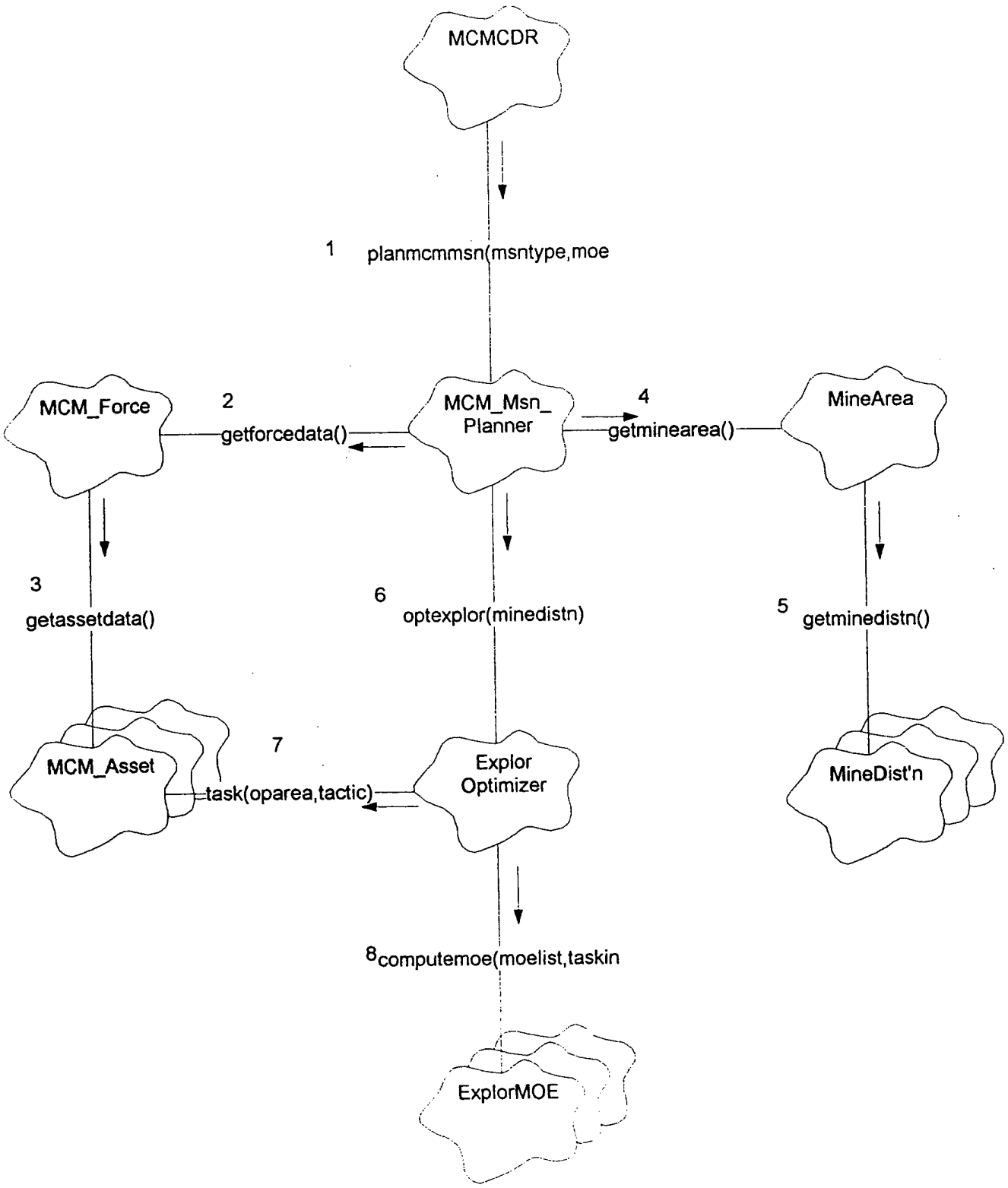
Scenario: Plan Breakthrough Mission

1. The MCM Commander selects "Plan Mission". A breakthrough mission and MOE set have already been selected.
2. The MCM Mission Planner calls the MCM Force for force data.
3. The MCM Force calls the assets for asset information (asset types and capabilities)
4. The MCM Mission Planner calls the Mine Area for information on found mines and threat mine distribution
5. The mine area calls the found mine database for found mines in the mine area.
6. The mine area calls the threat mine distribution.
7. The MCM Mission Planner invokes the Breakthrough Optimizer with the list of found mines, the threat mine distributions, the list of available assets and capabilities, and the Breakthrough MOE.
8. The Breakthrough Optimizer determines the channel for breakthrough.
9. The Breakthrough Optimizer assigns operating areas and tactics to MCM Assets.
10. The Breakthrough Optimizer calls the MOEs for values.


MCMCDR
FoundMine
1 planmcmmsn(msntype,moe
5 getfoundmines(minelist)
MCM_Force
2
getforcedata()
MCM_Msn_
Planner
4
getminearea()
MineArea
3
getassetdata()
7
optbreakthru(minelist,mine
6
getminedistn()
MCM_Asset
9
task(oparea,tactic)
Breakthru
Optimizer
MineDist'n
10
computemoe(moelist,taskin
8
addarea()
Breakthroug
hMOE
Breakthru
Channel

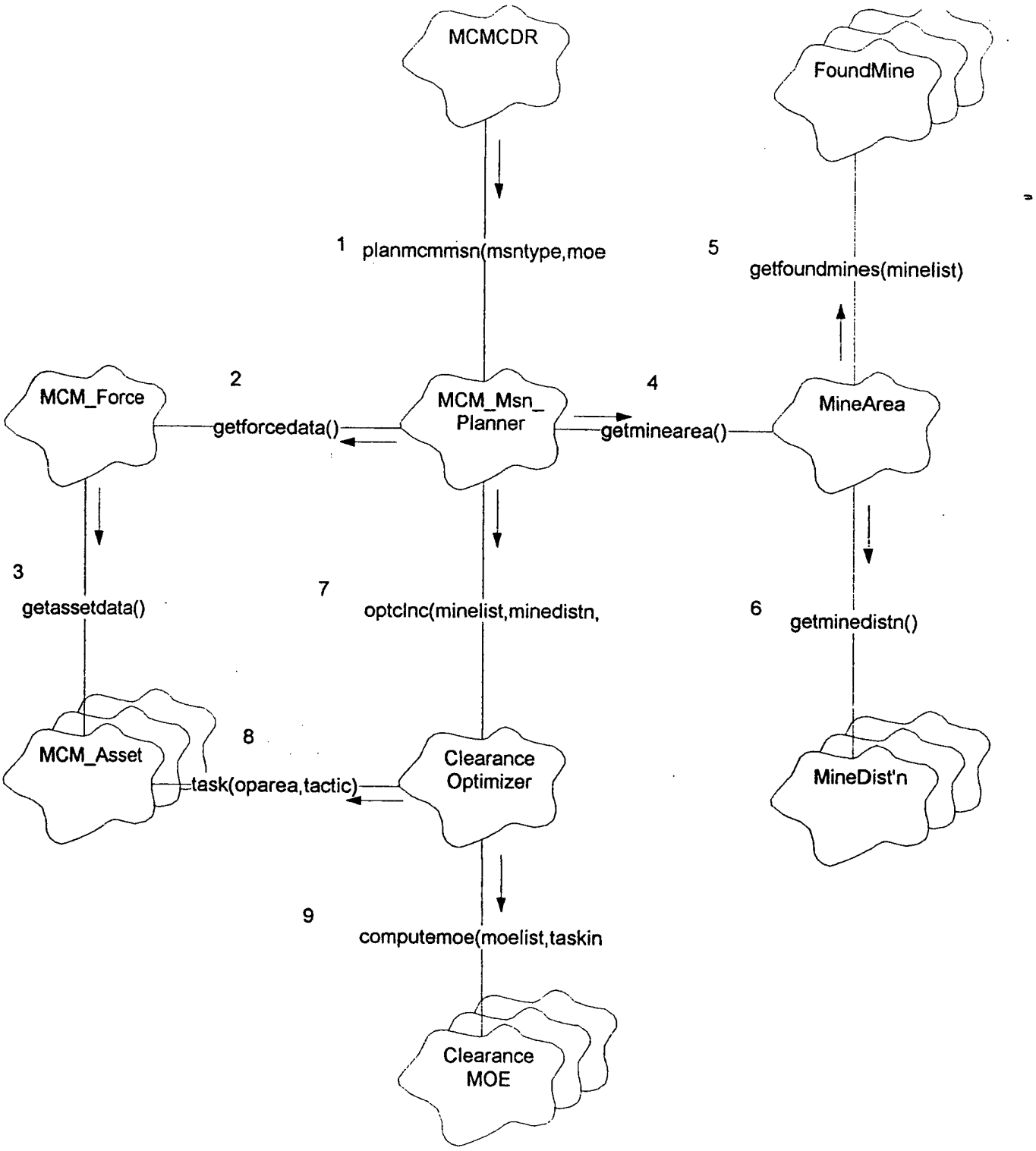
Scenario: Plan Exploratory MCM Mission

1. The MCM Commander selects "Plan Mission". An exploratory mission and MOE set have already been selected.
2. The MCM Mission Planner calls the MCM Force for force data.
3. The MCM Force calls the assets for asset information (asset types and capabilities)
4. The MCM Mission Planner calls the Mine Area for information on mine distributions.
5. The mine area calls the threat mine distribution.
6. The MCM Mission Planner invokes the Exploratory MCM Optimizer with the threat mine distributions, the list of available assets and capabilities, and the Exploratory MCM MOE.
7. The Exploratory MCM Optimizer assigns operating areas and tactics to MCM Assets.
8. The Exploratory Optimizer calls the MOEs for values.


MCMCDR
1 planmcmmsn(msntype,moe
MCM_Force
2
getforcedata()
MCM_Msn_
Planner
4
getminearea()
MineArea
3
getassetdata()
6
optexplor(minedistn)
5
getminedistn()
MCM_Asset
7
task(oparea,tactic)
Explor
Optimizer
MineDist'n
8computemoe(moelist,taskin
ExplorMOE

Scenario: Plan Clearance Mission

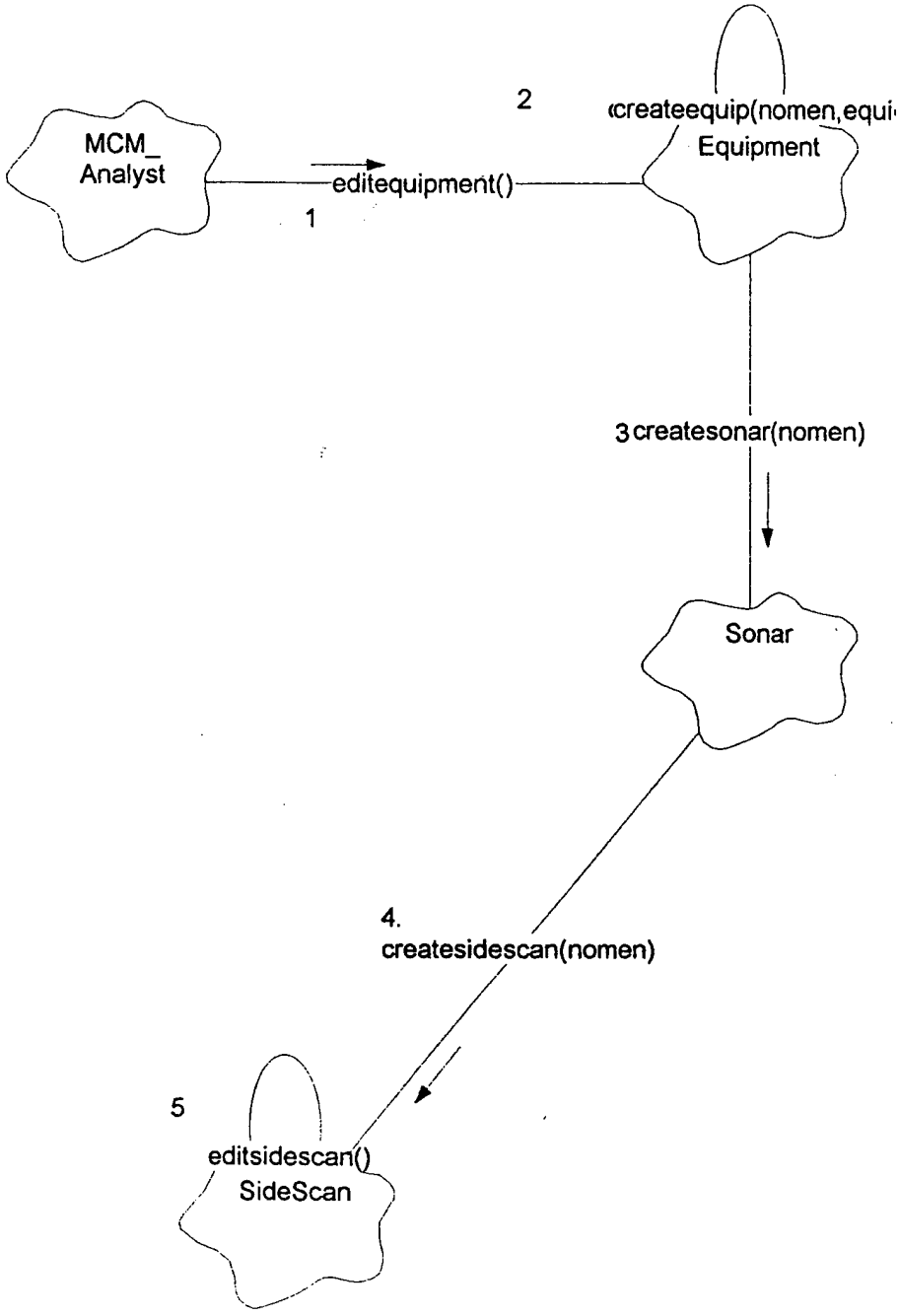
1. The MCM Commander selects "Plan Mission". A clearance mission and MOE set have already been selected.
2. The MCM Mission Planner calls the MCM Force for force data.
3. The MCM Force calls the assets for asset information (asset types and capabilities)
4. The MCM Mission Planner calls the Mine Area for information on found mines and threat mine distribution
5. The mine area calls the found mine database for found mines in the mine area.
6. The mine area calls the threat mine distribution.
7. The MCM Mission Planner invokes the Clearance Optimizer with the list of found mines, the threat mine distributions, the list of available assets and capabilities, and the Clearance MOE.
8. The Clearance Optimizer assigns operating areas and tactics to MCM Assets.
9. The Clearance Optimizer calls the MOEs for values.


MCMCDR
FoundMine
1 planmcmmsn(msntype,moe
5 getfoundmines(minelist)
MCM_Force
2
MCM_Msn_
Planner
4
MineArea
getforcedata()
getminearea()
3
getassetdata()
7
optclnc(minelist,minedistn,
6
getminedistn()
MCM_Asset
8
Clearance
Optimizer
MineDist'n
task(oparea,tactic)
9
computemoe(moelist,taskin
Clearance
MOE

Use Case: Specify Equipment Characteristics

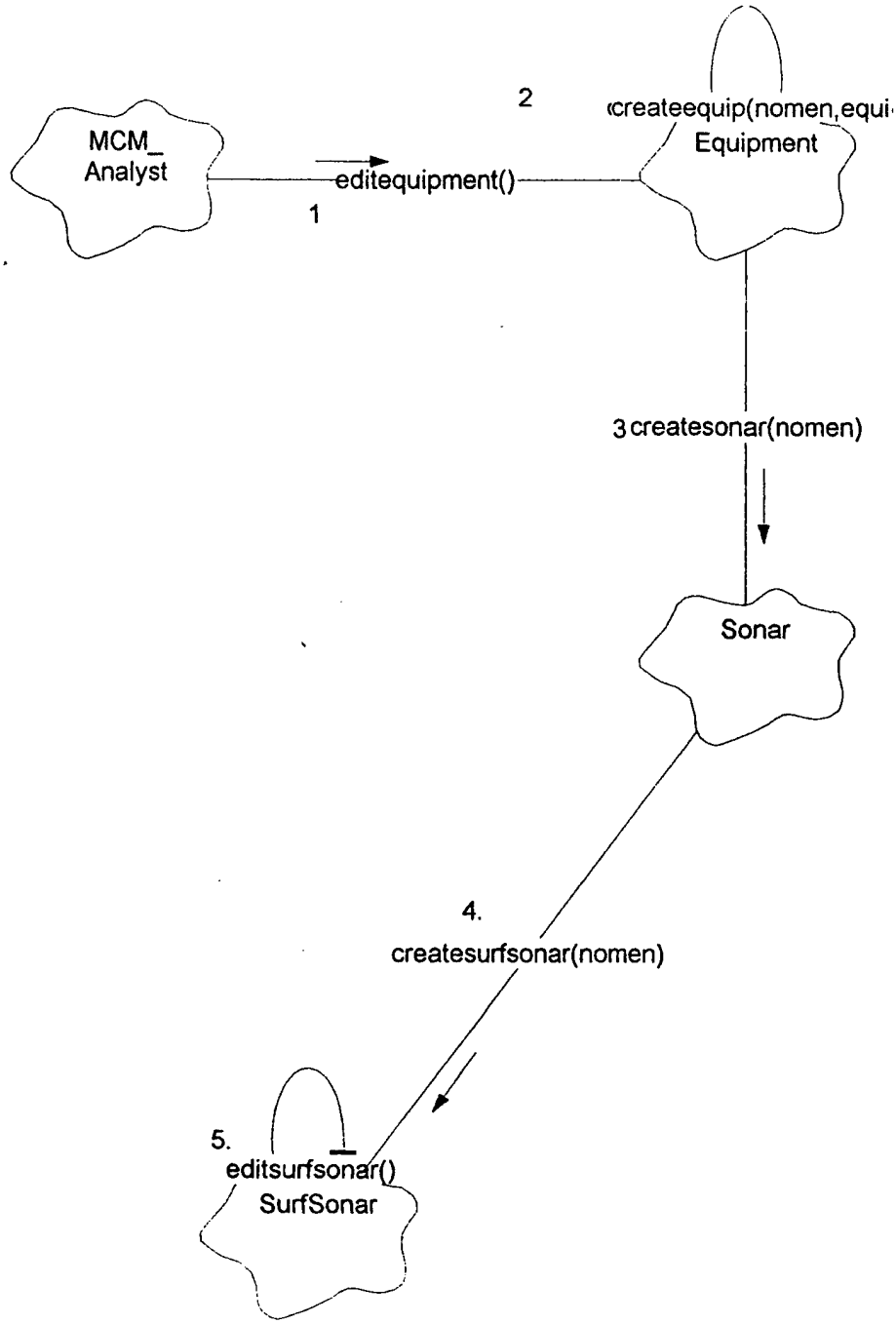
Add a new side scan sonar

1. The user selects "Edit MCM Equipment" on the MCM Analyst's display. The Edit Equipment display appears.
2. The user selects "Add Equipment".
3. The user selects "Side Scan Sonar" as the type of equipment and enters the nomenclature for the new side scan sonar. The Edit Sonar display appears.
4. The user enters the sonar characteristics. The Edit Side Scan display appears.
5. The user edits the side scan sonar characteristics.


MCM_
Analyst
editequipment()
1
2
createequip(nomen,equi
Equipment
3 createsonar(nomen)
Sonar
4.
createsidescan(nomen)
5
editsidescan()
SideScan

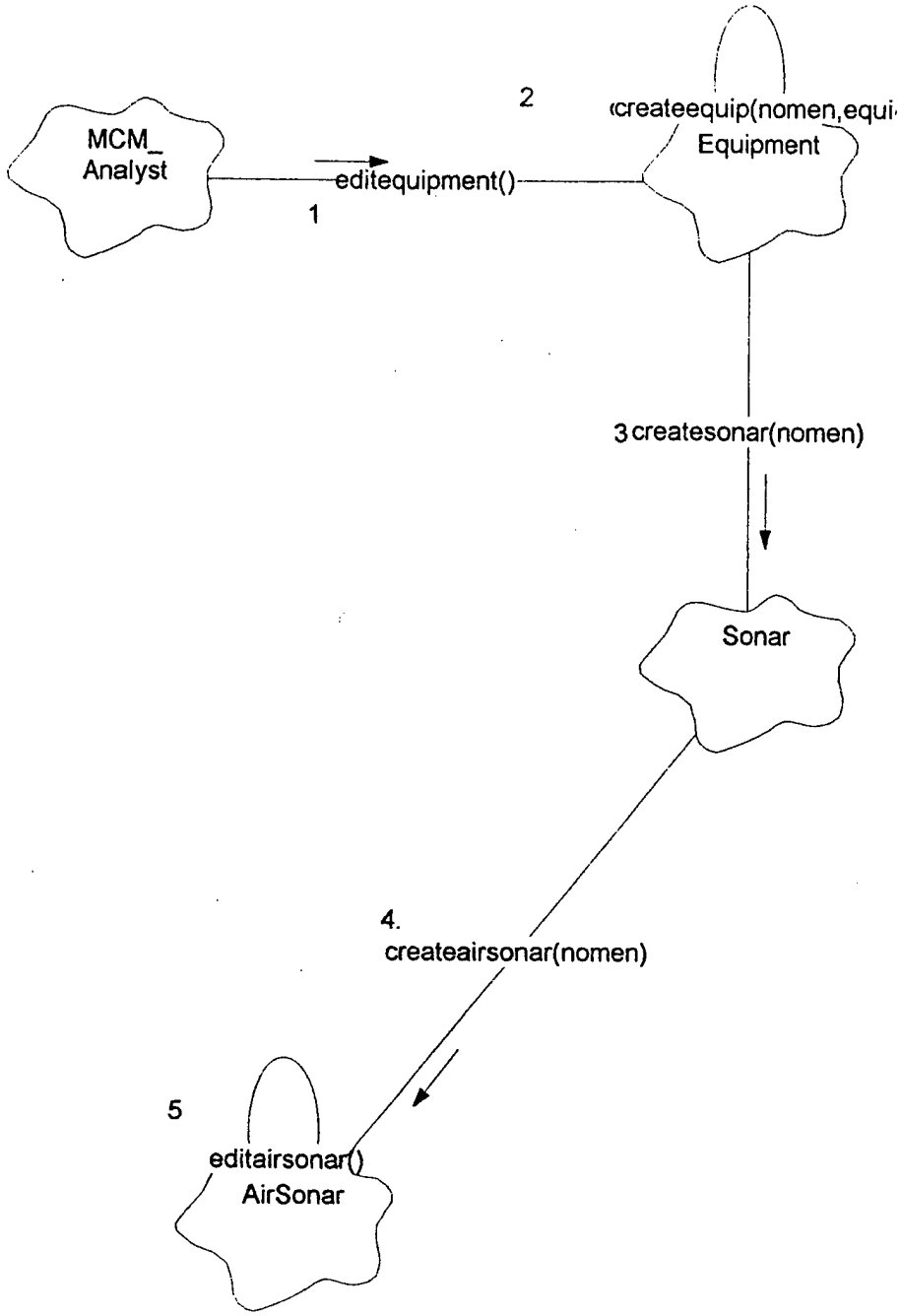
Add a New Surface Sonar

1. The user selects "Edit MCM Equipment" on the MCM Analyst's display. The Edit Equipment display appears.
2. The user selects "Add Equipment".
3. The user selects "Surface Sonar" as the type of equipment and enters the nomenclature for the new surface sonar. The Edit Sonar display appears.
4. The user enters the sonar characteristics. The Edit Surface Sonar display appears.
5. The user edits the surface sonar characteristics.


MCM_
Analyst
editequipment()
1
2
createequip(nomen,equi
Equipment
3 createsonar(nomen)
Sonar
4.
createsurfsonar(nomen)
5.
editsurfsonar()
SurfSonar

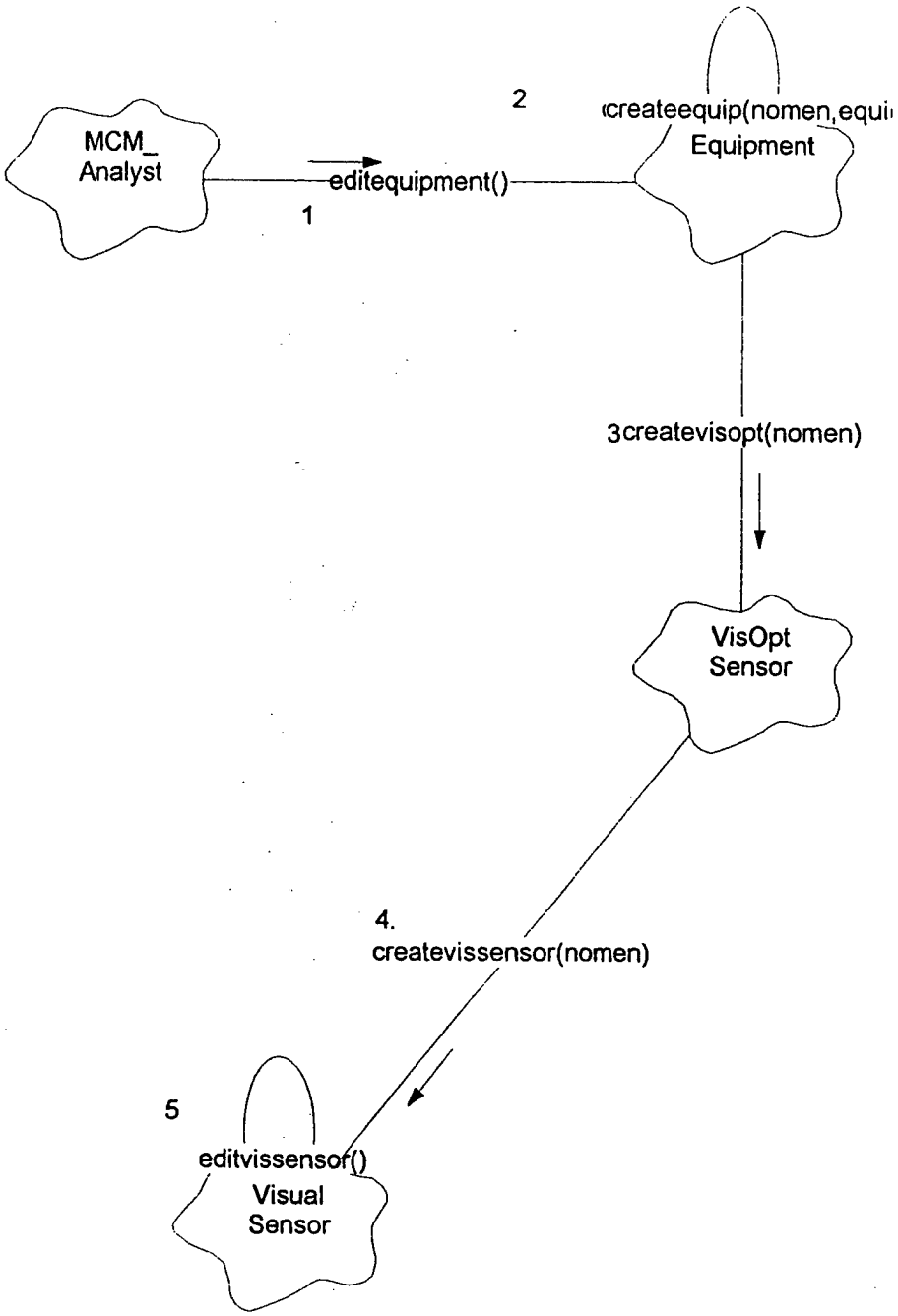
Add a New Air Sonar

1. The user selects "Edit MCM Equipment" on the MCM Analyst's display. The Edit Equipment display appears.
2. The user selects "Add Equipment".
3. The user selects "Air Sonar" as the type of equipment and enters the nomenclature for the new air sonar. The Edit Sonar display appears.
4. The user enters the sonar characteristics. The Edit Air Sonar display appears.
5. The user edits the air sonar characteristics.


MCM_
Analyst
editequipment()
1
2
createequip(nomen,equi
Equipment
3 createsonar(nomen)
Sonar
4.
createairsonar(nomen)
5
editairsonar()
AirSonar

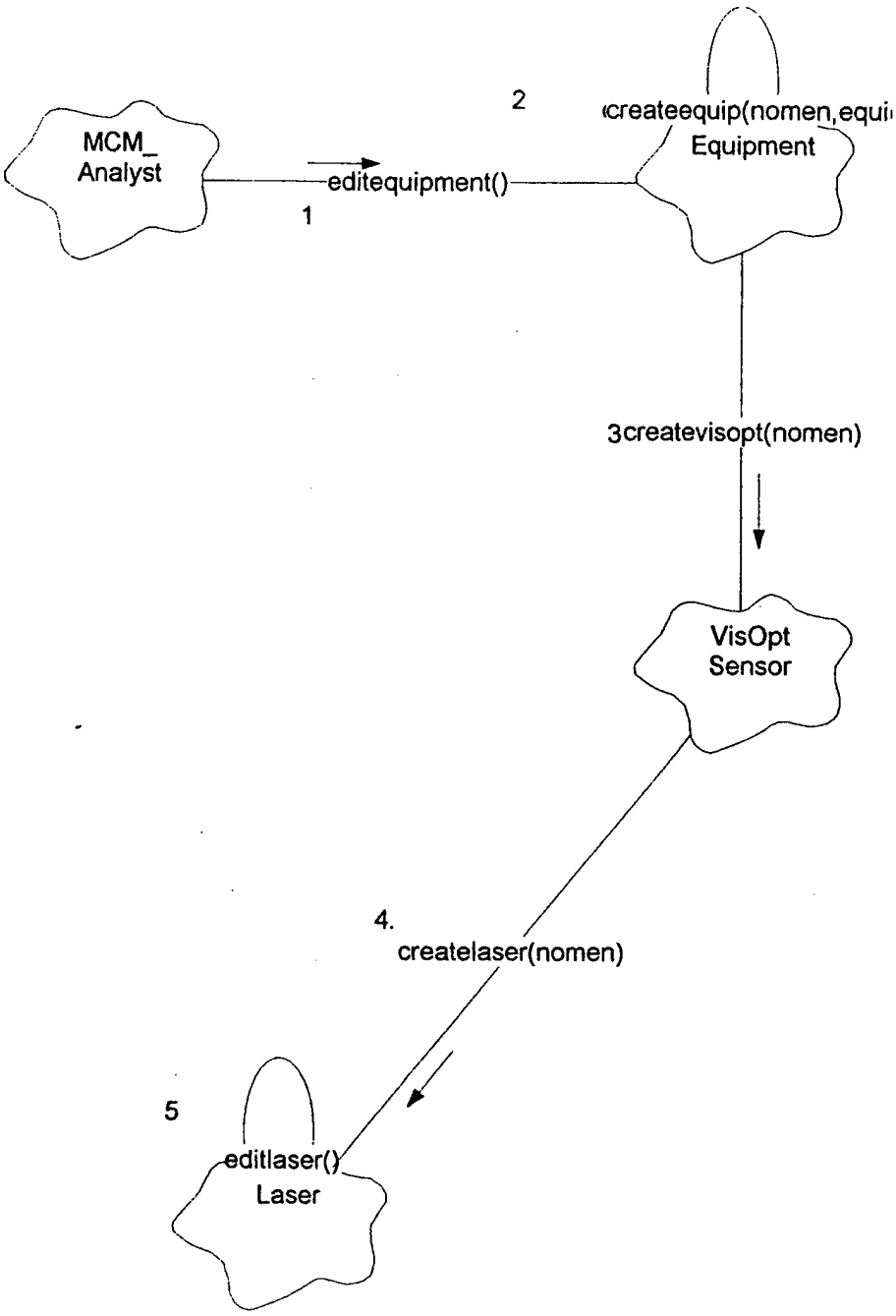
Add a New Visual Sensor

1. The user selects "Edit MCM Equipment" on the MCM Analyst's display. The Edit Equipment display appears.
2. The user selects "Add Equipment".
3. The user selects "Visual Sensor" as the type of equipment and enters the nomenclature for the new visual sensor. The Edit Visual/Optical Sensor display appears.
4. The user enters the visual/optical sensor characteristics. The Edit Visual Sensor display appears.
5. The user edits the visual sensor characteristics.


MCM_
Analyst
editequipment()
1
2
createequip(nomen,equi
Equipment
3createvisopt(nomen)
VisOpt
Sensor
4.
createvissensor(nomen)
5
editvissensor()
Visual
Sensor

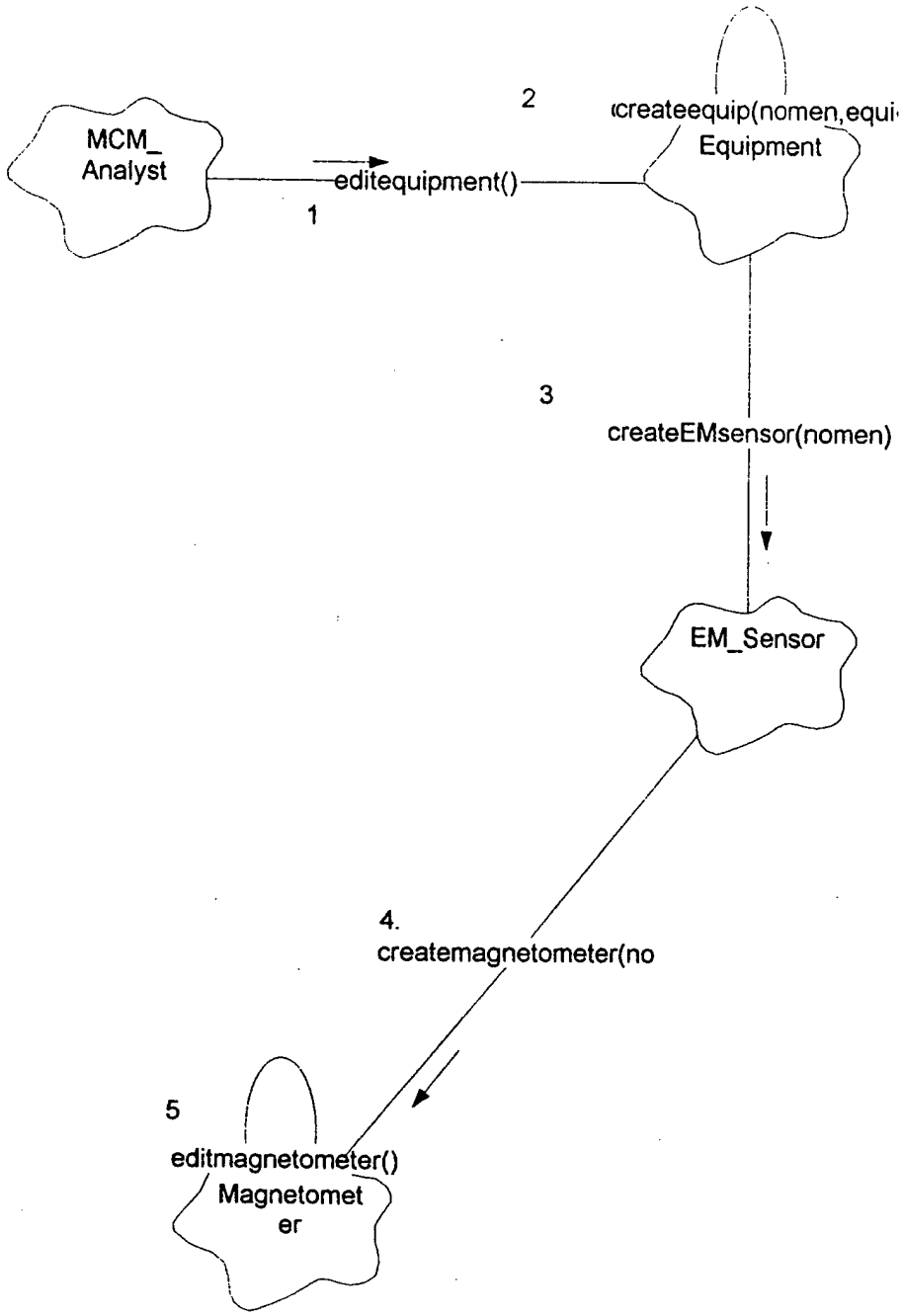
Add a New Laser Sensor

1. The user selects "Edit MCM Equipment" on the MCM Analyst's display. The Edit Equipment display appears.
2. The user selects "Add Equipment".
3. The user selects "Laser Sensor" as the type of equipment and enters the nomenclature for the new laser sensor. The Edit Visual/Optical Sensor display appears.
4. The user enters the visual/optical sensor characteristics. The Edit Laser Sensor display appears.
5. The user edits the laser sensor characteristics.


MCM_
Analyst
editequipment()
1
2
createequip(nomen,equi
Equipment
3createvisopt(nomen)
VisOpt
Sensor
4.
createlaser(nomen)
5
editlaser()
Laser

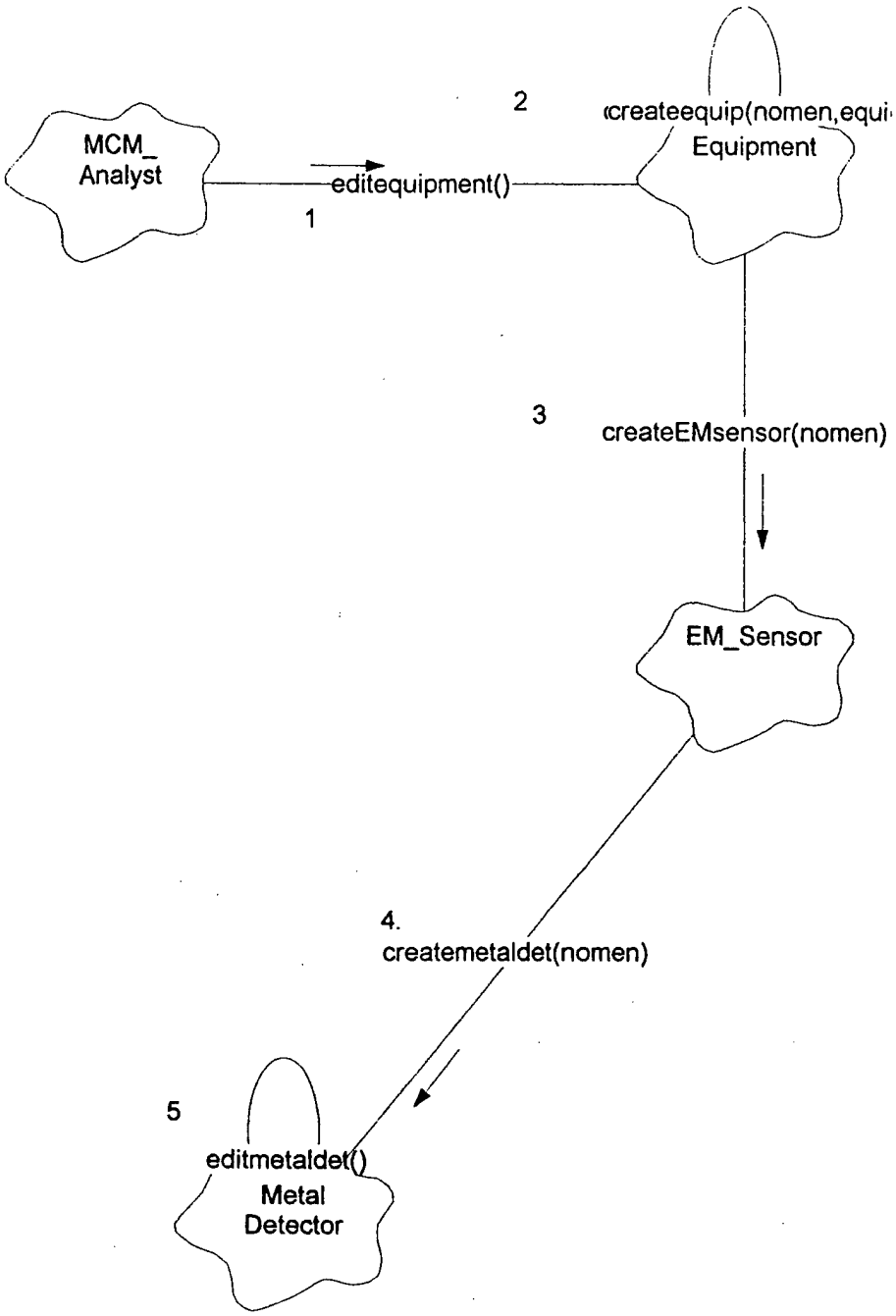
Add a New Magnetometer

1. The user selects "Edit MCM Equipment" on the MCM Analyst's display. The Edit Equipment display appears.
2. The user selects "Add Equipment".
3. The user selects "Magnetometer" as the type of equipment and enters the nomenclature for the new laser sensor. The Edit Electromagnetic Sensor display appears.
4. The user enters the electromagnetic sensor characteristics. The Edit Magnetometer display appears.
5. The user edits the magnetometer sensor characteristics.


MCM_
Analyst
editequipment()
1
2
createequip(nomen,equi
Equipment
3
createEMsensor(nomen)
EM_Sensor
4.
createmagnetometer(no
5
editmagnetometer()
Magnetomet
er

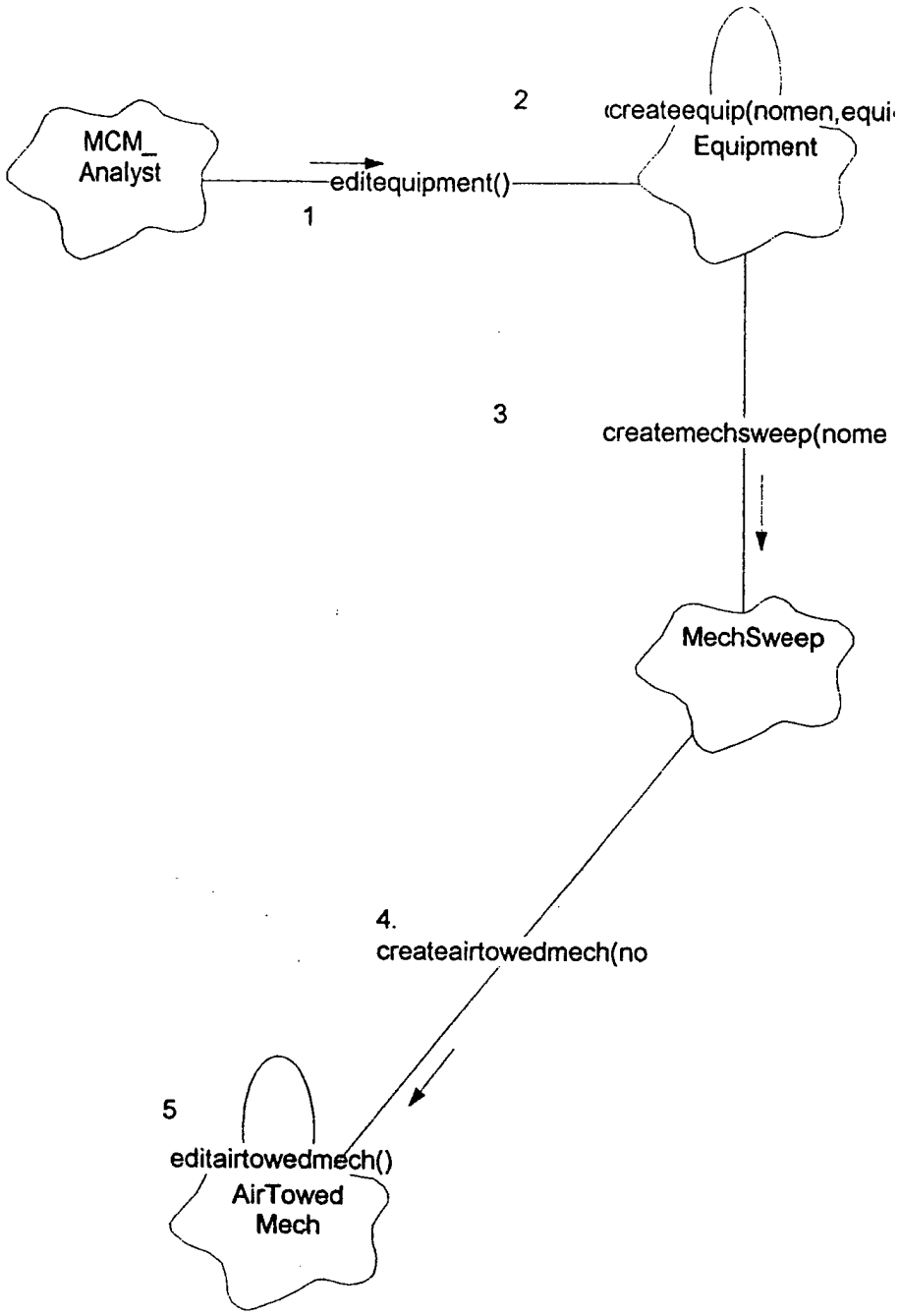
Add a New Metal Detector

1. The user selects "Edit MCM Equipment" on the MCM Analyst's display. The Edit Equipment display appears.
2. The user selects "Add Equipment".
3. The user selects "Metal Detector" as the type of equipment and enters the nomenclature for the new laser sensor. The Edit Electromagnetic Sensor display appears.
4. The user enters the electromagnetic sensor characteristics. The Edit Metal Detector display appears.
5. The user edits the metal detector sensor characteristics.


MCM_
Analyst
editequipment()
1
2
createequip(nomen,equi
Equipment
3
createEMsensor(nomen)
EM_Sensor
4.
createmetaldet(nomen)
5
editmetaldet()
Metal
Detector

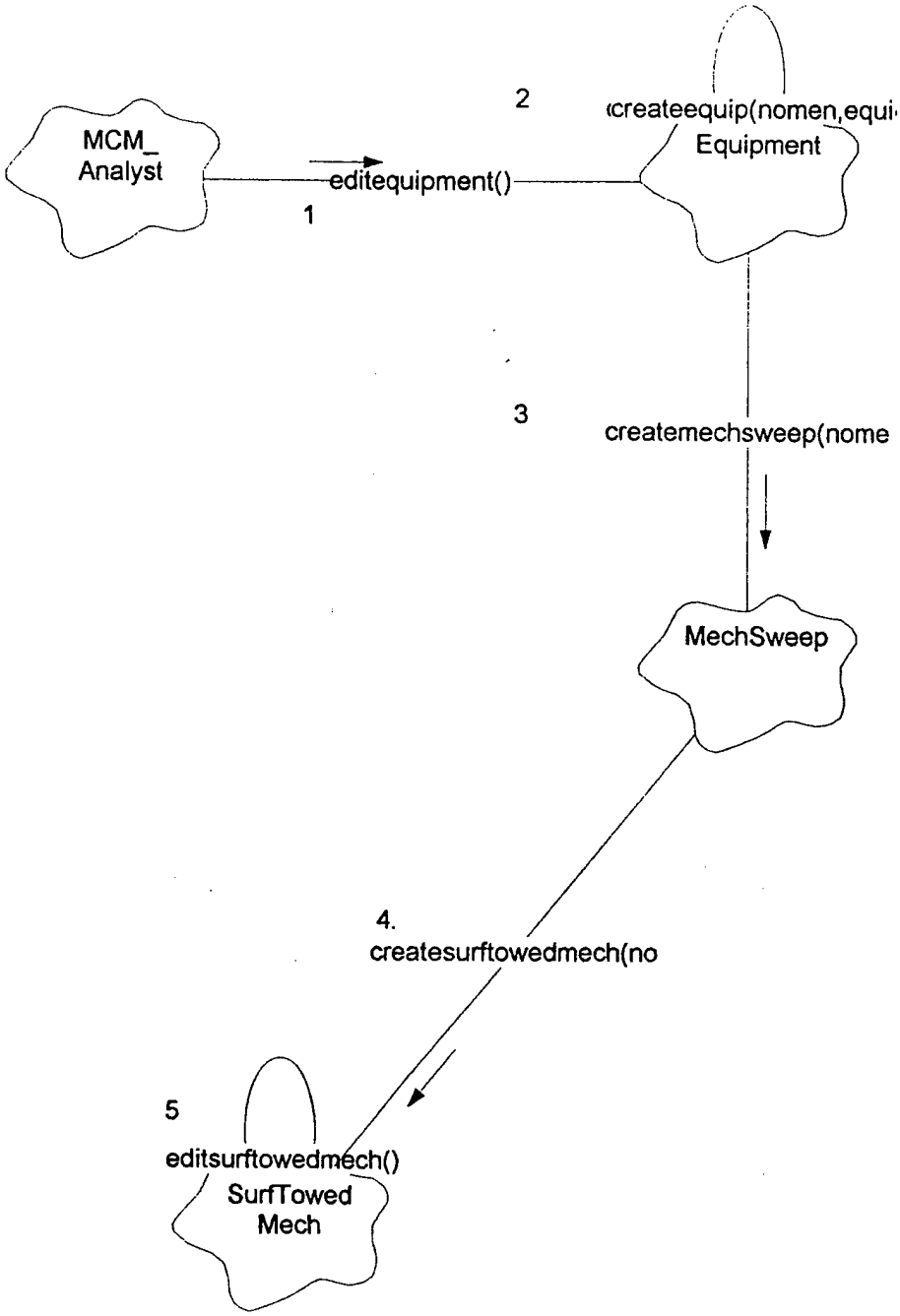
Add a New Air-Towed Mechanical Sweep Gear

1. The user selects "Edit MCM Equipment" on the MCM Analyst's display. The Edit Equipment display appears.
2. The user selects "Add Equipment".
3. The user selects "Air-Towed Mechanical Sweep" as the type of equipment and enters the nomenclature for the new mechanical sweep The Edit Mechanical Sweep Gear display appears.
4. The user enters the mechanical sweep gear characteristics. The Edit Air-Towed Mechanical Sweep display appears.
5. The user edits the Air-Towed Mechanical Sweep characteristics.


MCM_
Analyst
editequipment()
1
2
createequip(nomen,equi
Equipment
3
createmechsweep(nome
MechSweep
4.
createairtowedmech(no
5
editairtowedmech()
AirTowed
Mech

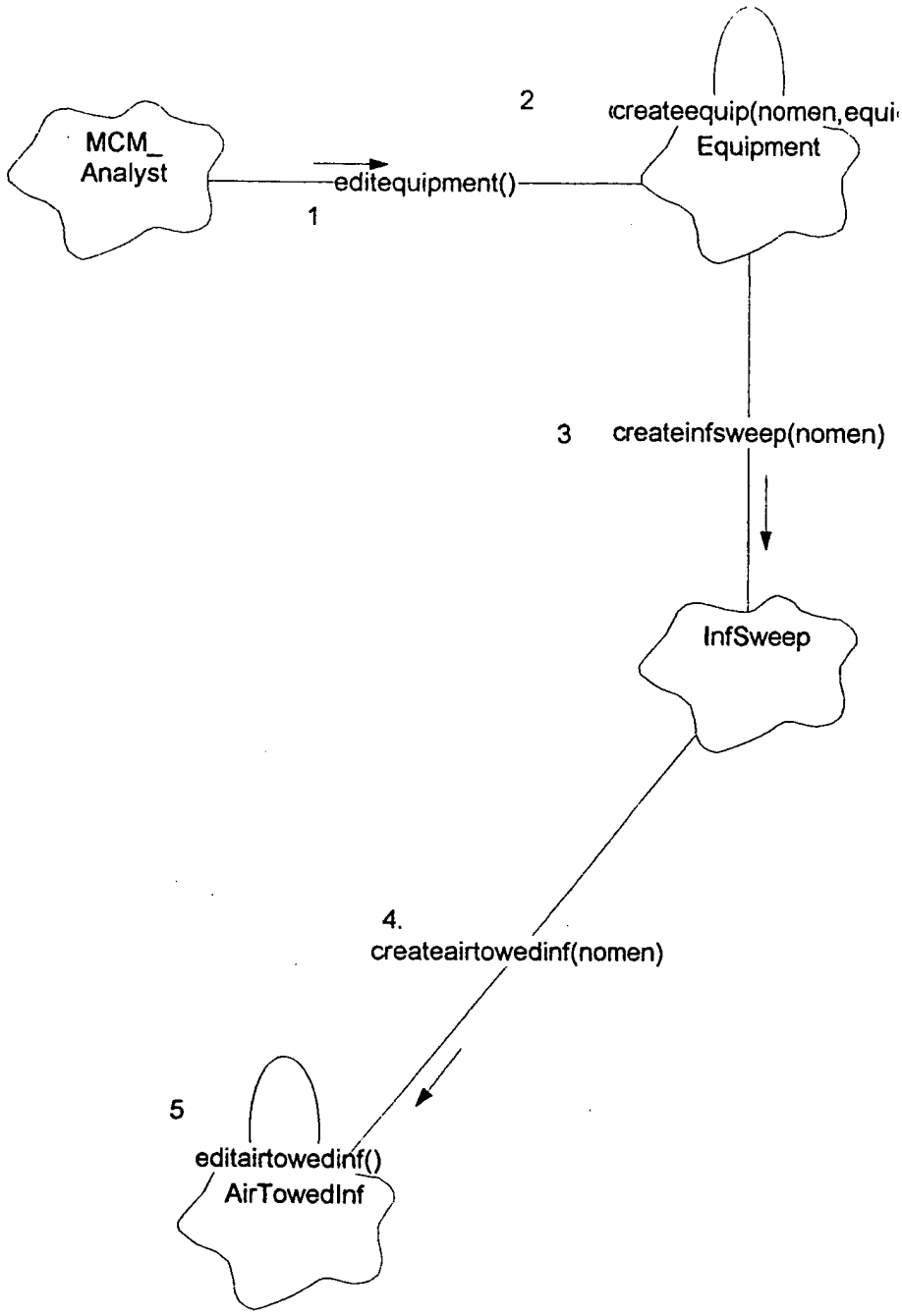
Add a New Surface-Towed Mechanical Sweep Gear

1. The user selects "Edit MCM Equipment" on the MCM Analyst's display. The Edit Equipment display appears.
2. The user selects "Add Equipment".
3. The user selects "Surface-Towed Mechanical Sweep" as the type of equipment and enters the nomenclature for the new mechanical sweep. The Edit Mechanical Sweep Gear display appears.
4. The user enters the mechanical sweep gear characteristics. The Edit Surface-Towed Mechanical Sweep display appears.
5. The user edits the Surface-Towed Mechanical Sweep characteristics.


MCM_
Analyst
editequipment()
1
2
createequip(nomen,equi
Equipment
3
createmechsweep(nome
MechSweep
4.
createsurftowedmech(no
5
editsurftowedmech()
SurfTowed
Mech

Add a New Air-Towed Influence Sweep Gear

1. The user selects "Edit MCM Equipment" on the MCM Analyst's display. The Edit Equipment display appears.
2. The user selects "Add Equipment".
3. The user selects "Air-Towed Influence Sweep" as the type of equipment and enters the nomenclature for the new mechanical sweep. The Edit Influence Sweep Gear display appears.
4. The user enters the influence sweep gear characteristics. The Edit Air-Towed Influence Sweep display appears.
5. The user edits the Air-Towed Influence Sweep characteristics.


MCM_
Analyst
editequipment()
1
2
createequip(nomen,equi
Equipment
3
createinfsweep(nomen)
InfSweep
4.
createairtowedinf(nomen)
5
editairtowedinf()
AirTowedInf

Add a New Surface-Towed Influence Sweep Gear

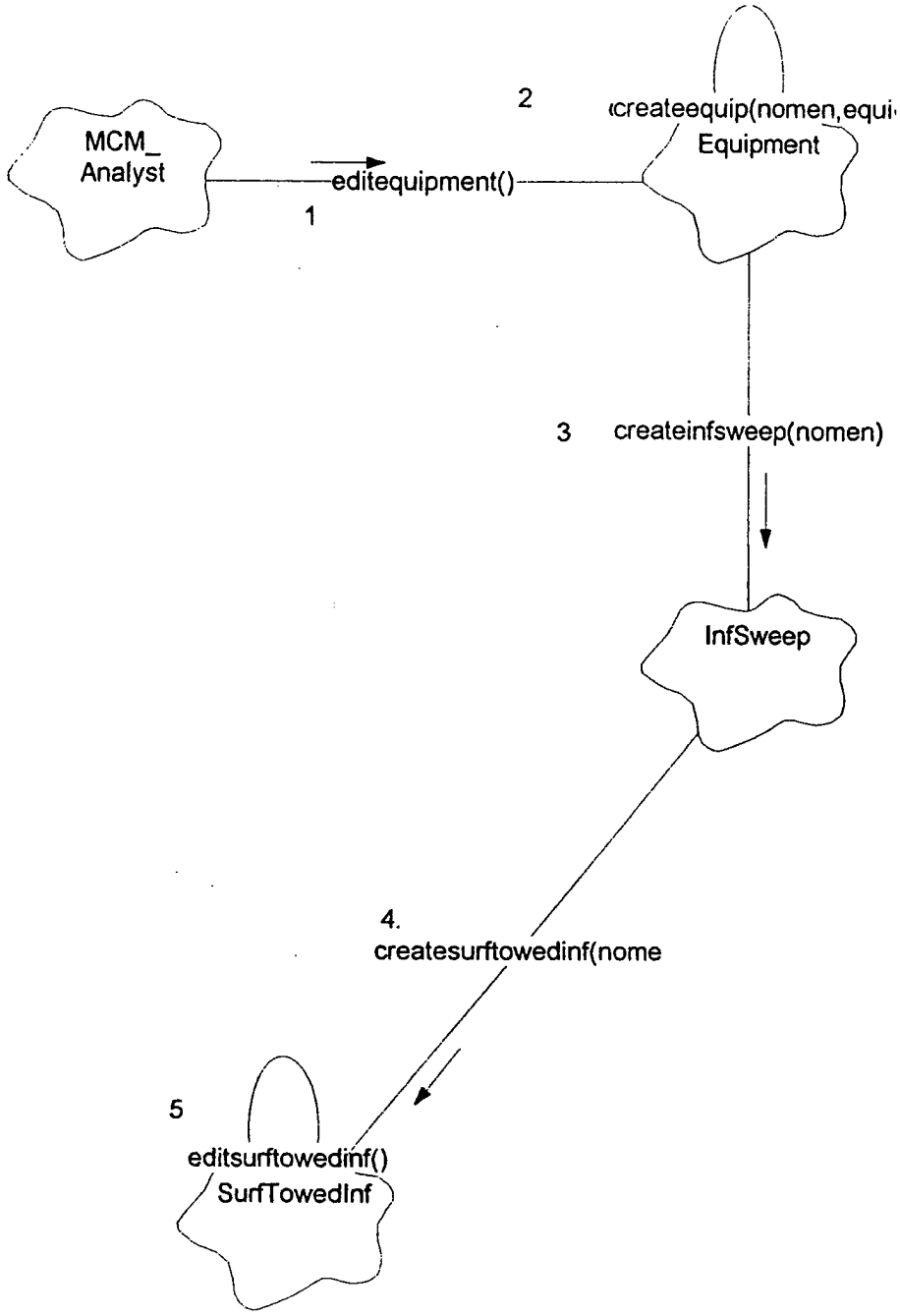
1. The user selects "Edit MCM Equipment" on the MCM Analyst's display. The Edit Equipment display appears.

2. The user selects "Add Equipment".

3. The user selects "Surface-Towed Influence Sweep" as the type of equipment and enters the nomenclature for the new mechanical sweep. The Edit Influence Sweep Gear display appears.

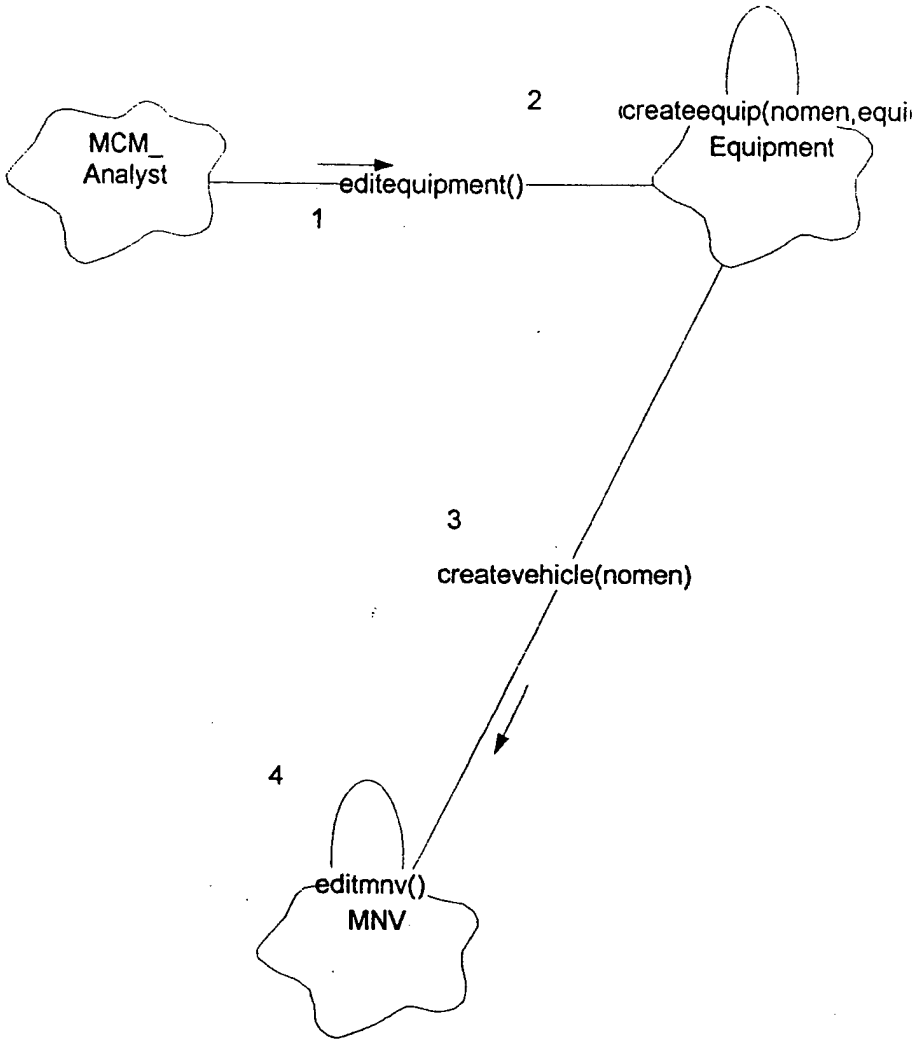
4. The user enters the influence sweep gear characteristics. The Edit Surface-Towed Influence Sweep display appears.

5. The user edits the Surface-Towed Influence Sweep characteristics.


MCM_
Analyst
editequipment()
1
2
createequip(nomen,equi
Equipment
3
createinfsweep(nomen)
InfSweep
4.
createsurftowedinf(nome
5
editsurftowedinf()
SurfTowedInf

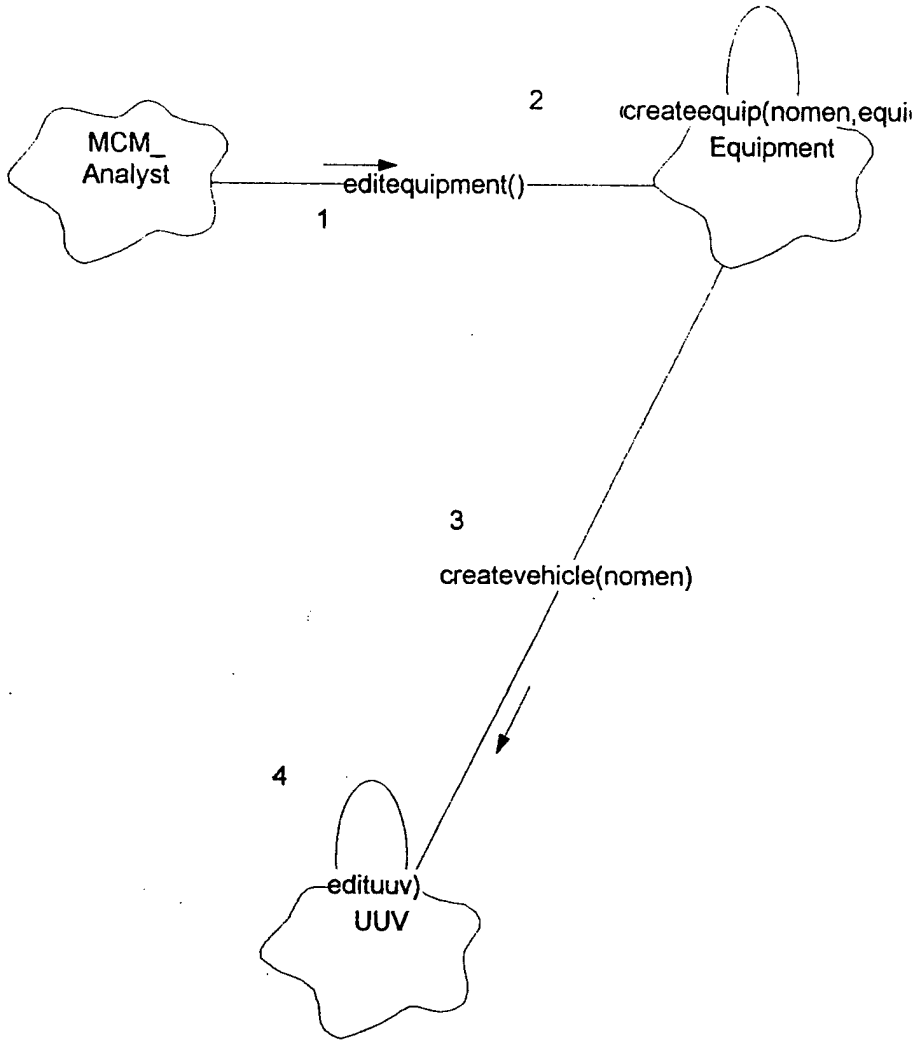
Add a New MNV

1. The user selects "Edit MCM Equipment" on the MCM Analyst's display. The Edit Equipment display appears.
2. The user selects "Add Equipment".
3. The user selects "Mine Neutralization Vehicle" as the type of equipment and enters the nomenclature for the new MNV. The Edit MNV display appears.
4. The user enters the MNV characteristics.


MCM_
Analyst
editequipment()
1
2
createequip(nomen,equi
Equipment
3
createvehicle(nomen)
4
editmnv()
MNV

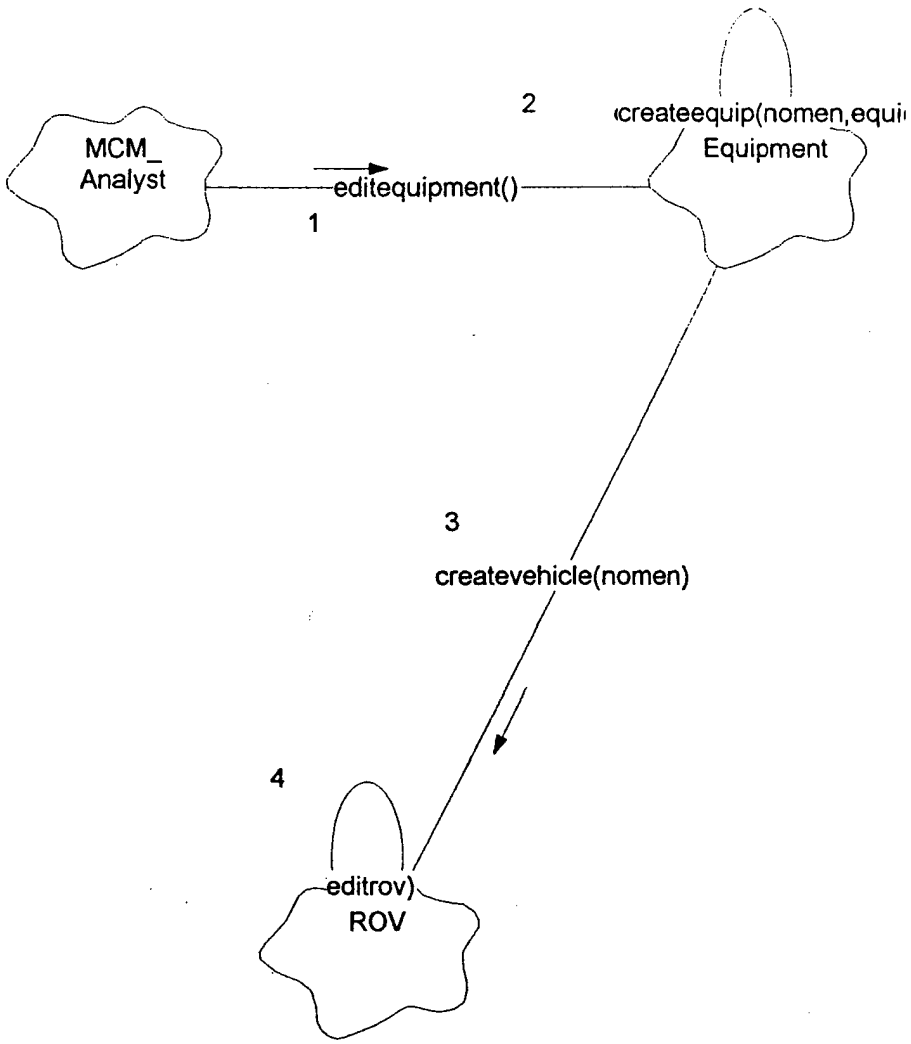
Add a New UUV

1. The user selects "Edit MCM Equipment" on the MCM Analyst's display. The Edit Equipment display appears.
2. The user selects "Add Equipment".
3. The user selects "Unmanned Underwater Vehicle" as the type of equipment and enters the nomenclature for the new UUV. The Edit UUV display appears.
4. The user enters the UUV characteristics.


MCM_
Analyst
editequipment()
1
2
createequip(nomen,equi
Equipment
3
createvehicle(nomen)
4
edituuv)
UUV

Add a New ROV

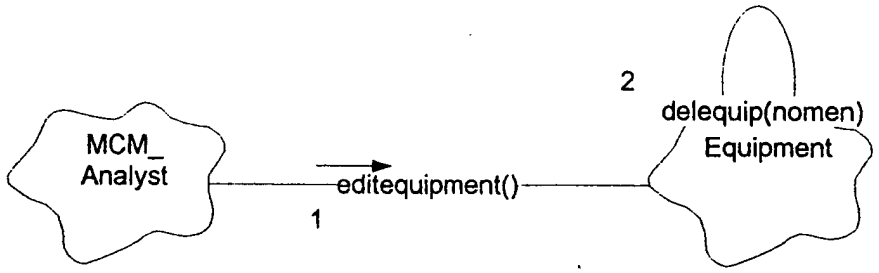
1. The user selects "Edit MCM Equipment" on the MCM Analyst's display. The Edit Equipment display appears.
2. The user selects "Add Equipment".
3. The user selects "Remotely Operated Vehicle" as the type of equipment and enters the nomenclature for the new ROV. The Edit ROV display appears.
4. The user enters the ROV characteristics.


MCM_
Analyst
editequipment()
1
2
createequip(nomen,equi
Equipment
3
createvehicle(nomen)
4
editrov)
ROV

Delete an MCM Equipment

1. The user selects "Edit Equipment". The Equipment object is called.

2. The user selects the equipment to be deleted and selects "Delete". The user confirms the deletion. The equipment of that nomenclature is removed.

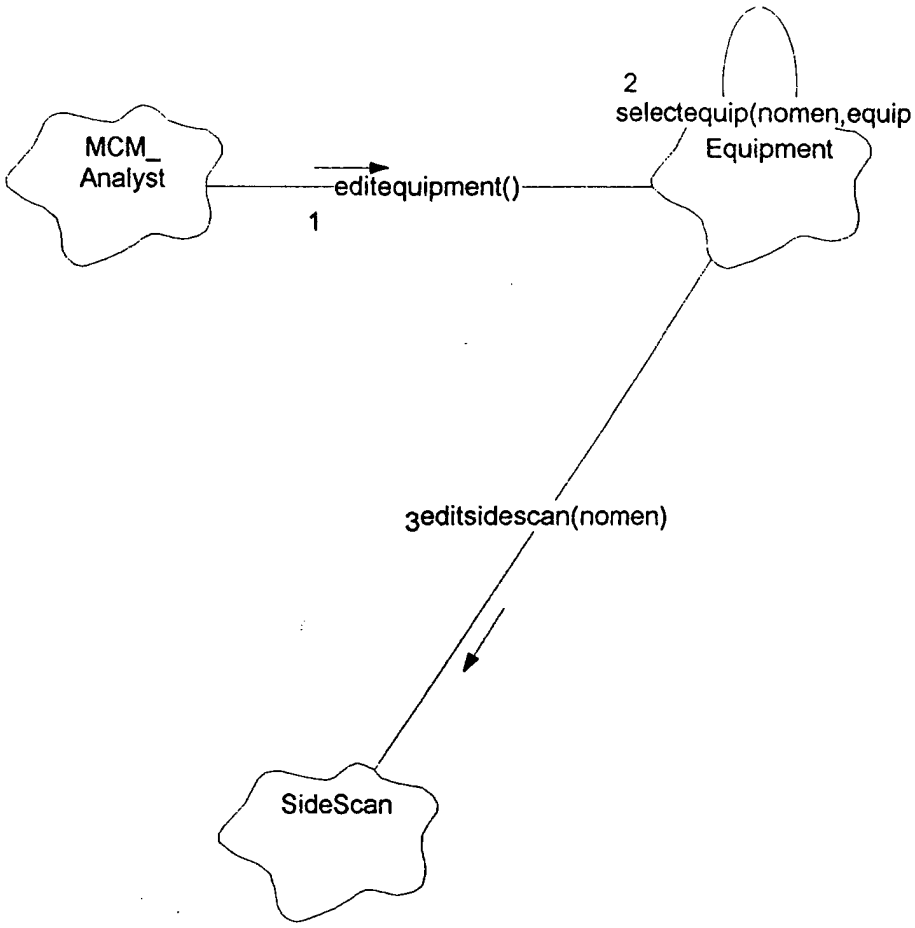

MCM_
Analyst
editequipment()
1
2
delequip(nomen)
Equipment

Edit a Side Scan Sonar

1. The user selects "Edit MCM Equipment" on the MCM Analyst's display. The Edit Equipment display appears.

2. The user selects the side scan sonar for editing. The Edit Side Scan Sonar display appears.

3 The user edits the side scan sonar characteristics.

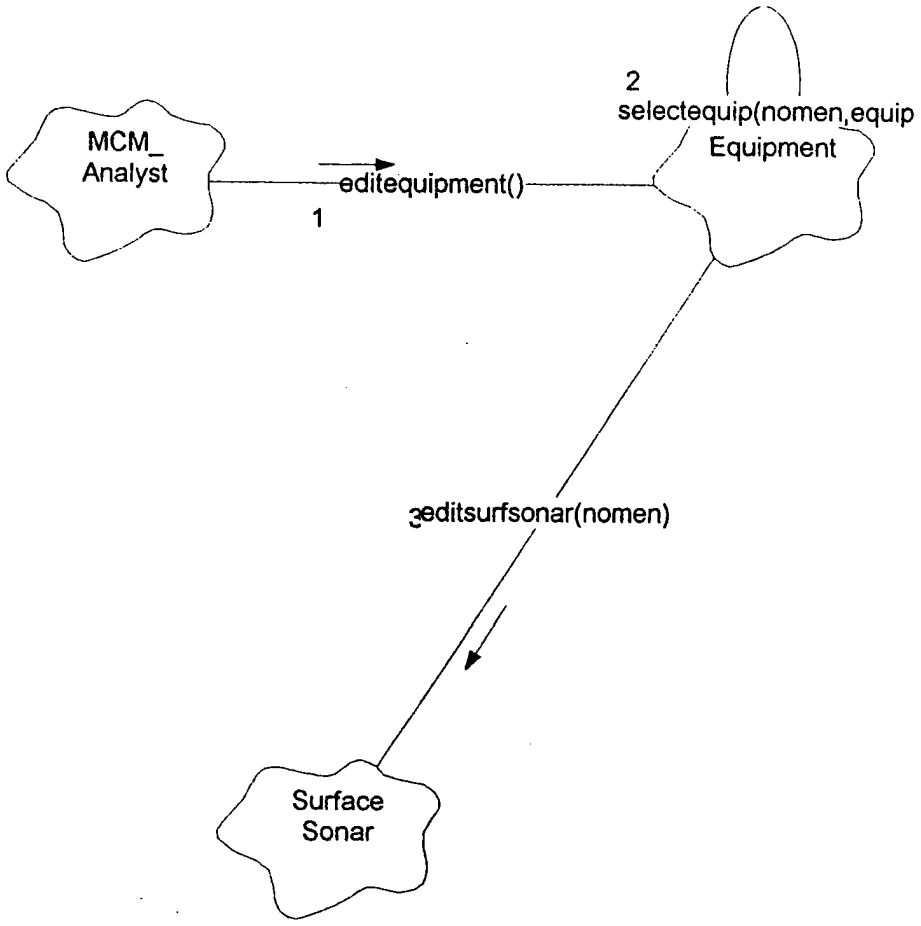

MCM_
Analyst
editequipment()
1
2
selectequip(nomen,equip
Equipment
3editsidescan(nomen)
SideScan

Edit a Surface Sonar

1. The user selects "Edit MCM Equipment" on the MCM Analyst's display. The Edit Equipment display appears.

2. The user selects the surface sonar for editing. The Edit Surface Sonar display appears.

3 The user edits the surface sonar characteristics.

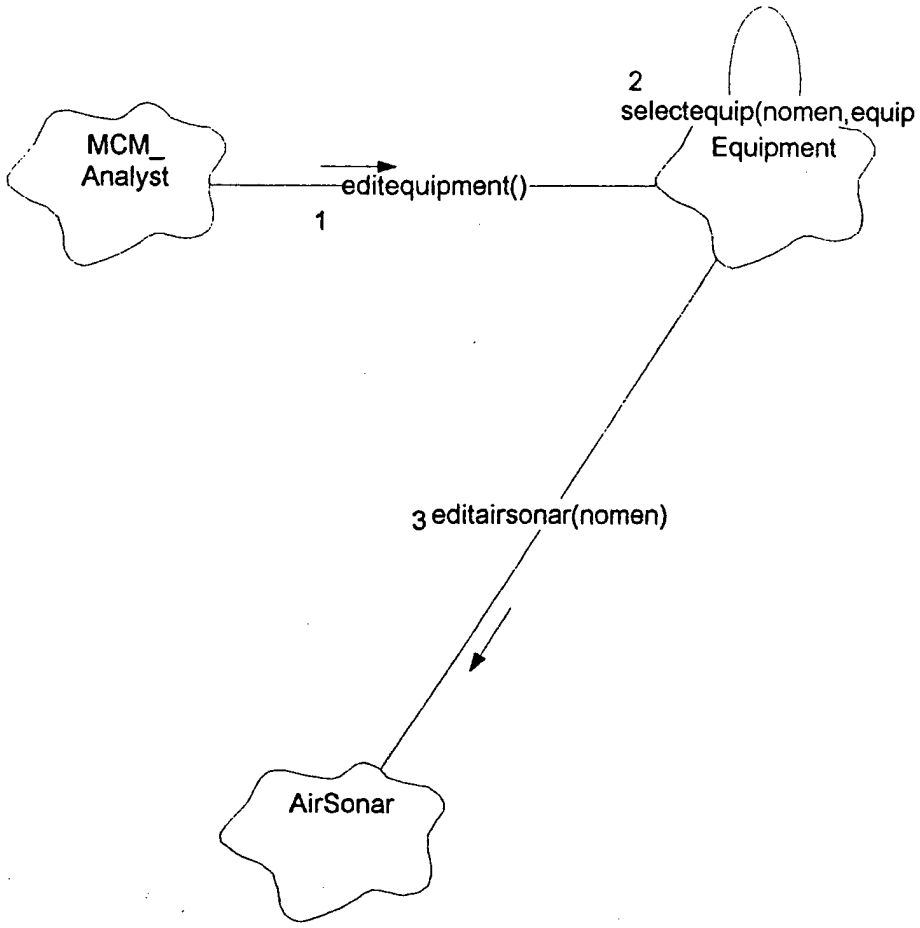

2
selectequip(nomen,equip
Equipment
MCM_
Analyst
editequipment()
1
3editsurfsonar(nomen)
Surface
Sonar

Edit a Air ~~Surface~~ Sonar

1. The user selects "Edit MCM Equipment" on the MCM Analyst's display. The Edit Equipment display appears.

2. The user selects the air sonar for editing. The Edit Air Sonar display appears.

3 The user edits the air sonar characteristics.

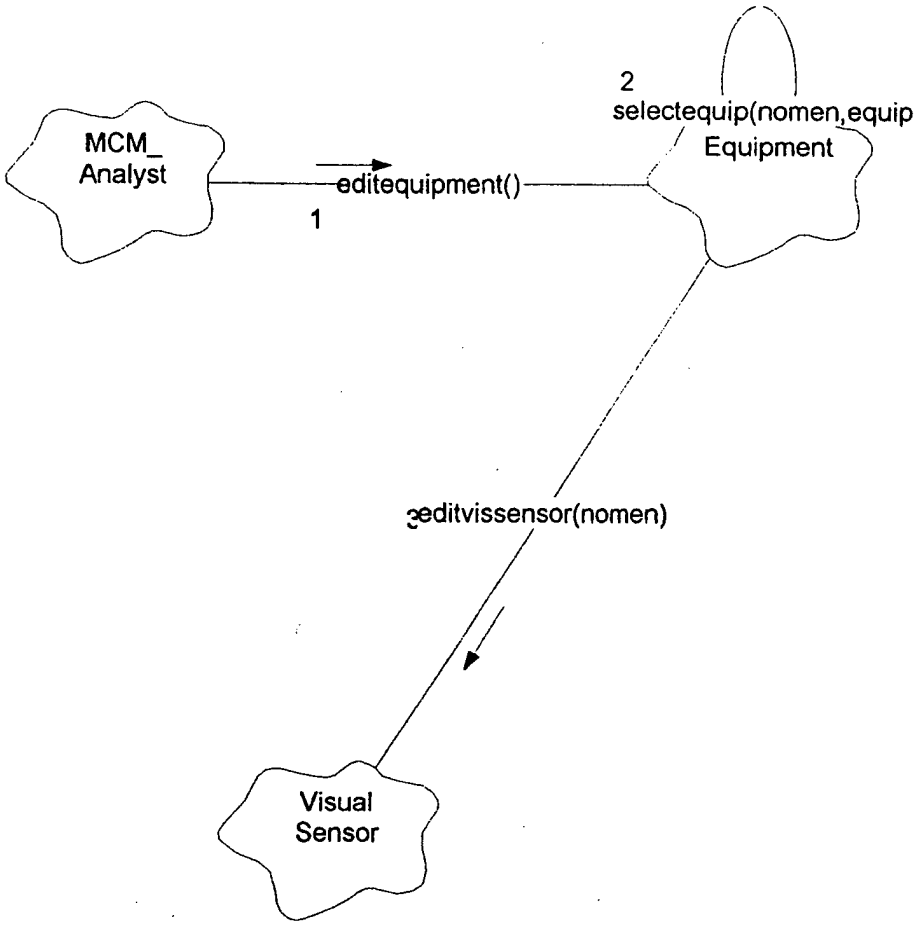

MCM_
Analyst
editequipment()
1
2
selectequip(nomen,equip
Equipment
3 editairsonar(nomen)
AirSonar

Edit a Visual Sensor

1. The user selects "Edit MCM Equipment" on the MCM Analyst's display. The Edit Equipment display appears.

2. The user selects the visual sensor for editing. The Edit Visual Sensor display appears.

3 The user edits the visual sensor characteristics.

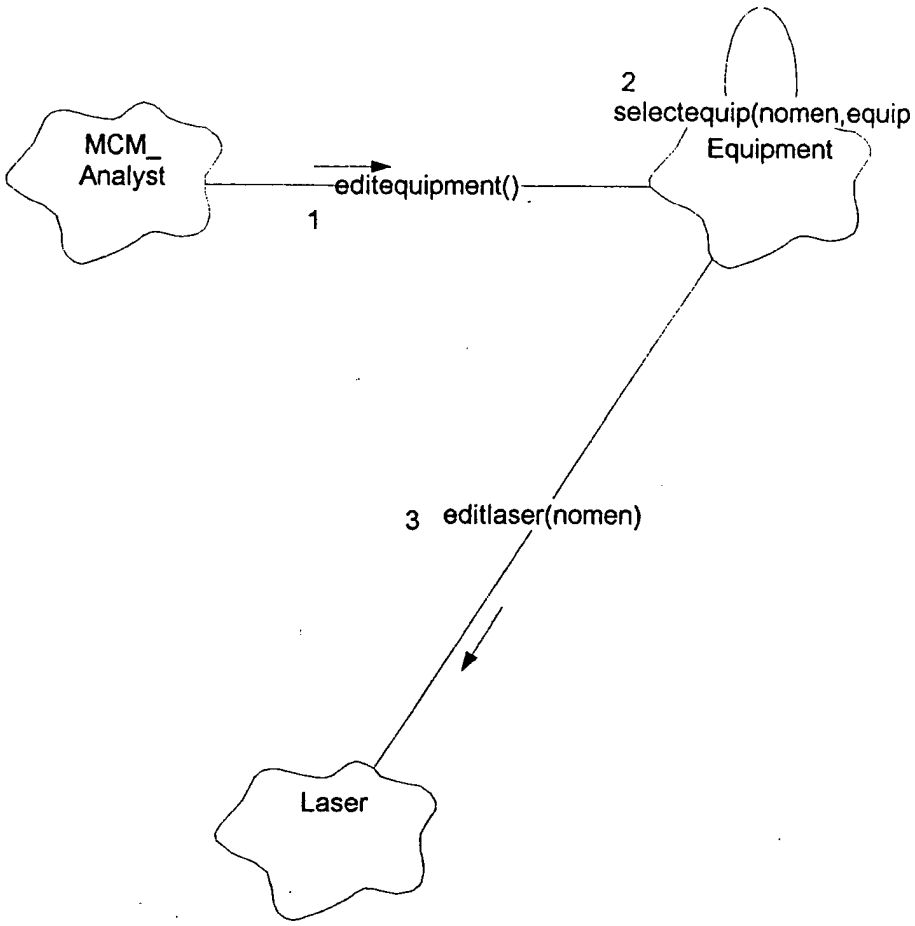

2
selectequip(nomen,equip
Equipment
MCM_
Analyst
editequipment()
1
editvissensor(nomen)
Visual
Sensor

Edit a Laser Sensor

1. The user selects "Edit MCM Equipment" on the MCM Analyst's display. The Edit Equipment display appears.

2. The user selects the laser sensor for editing. The Edit Laser Sensor display appears.

3 The user edits the laser sensor characteristics.

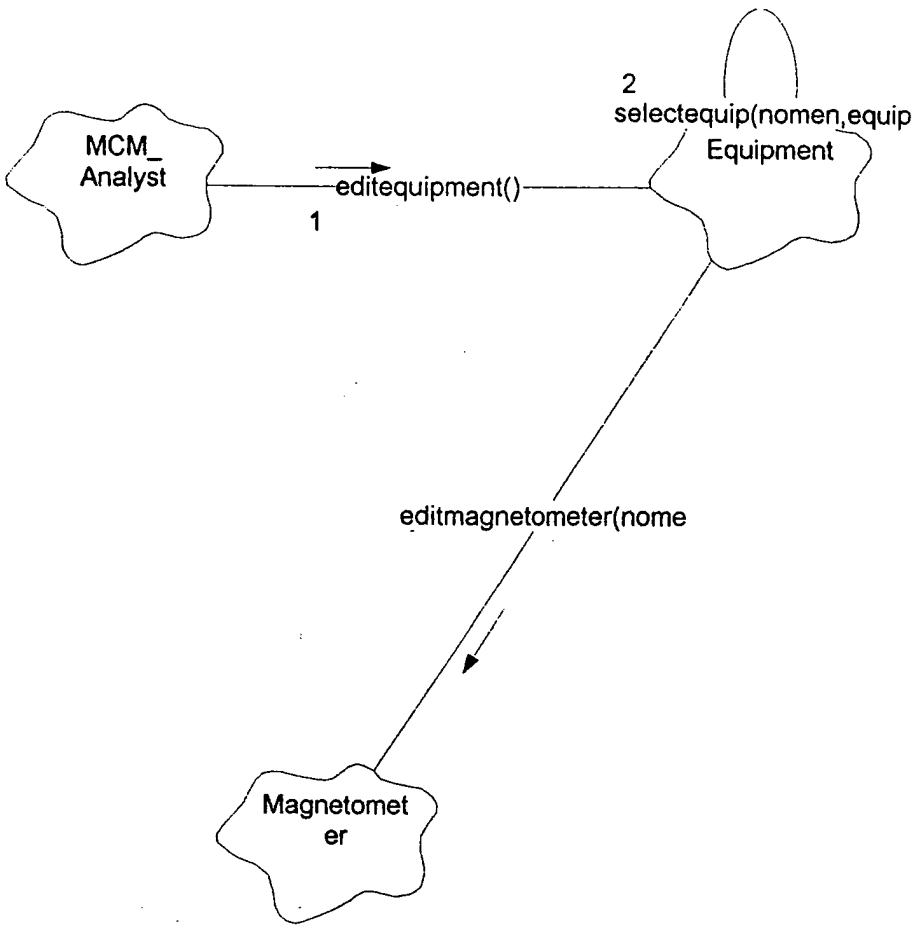

MCM_
Analyst
editequipment()
1
2
selectequip(nomen,equip
Equipment
3 editlaser(nomen)
Laser

Edit a Magnetometer

1. The user selects "Edit MCM Equipment" on the MCM Analyst's display. The Edit Equipment display appears.

2. The user selects the magnetometer for editing. The Edit Magnetometer display appears.

3 The user edits the magnetometer characteristics.

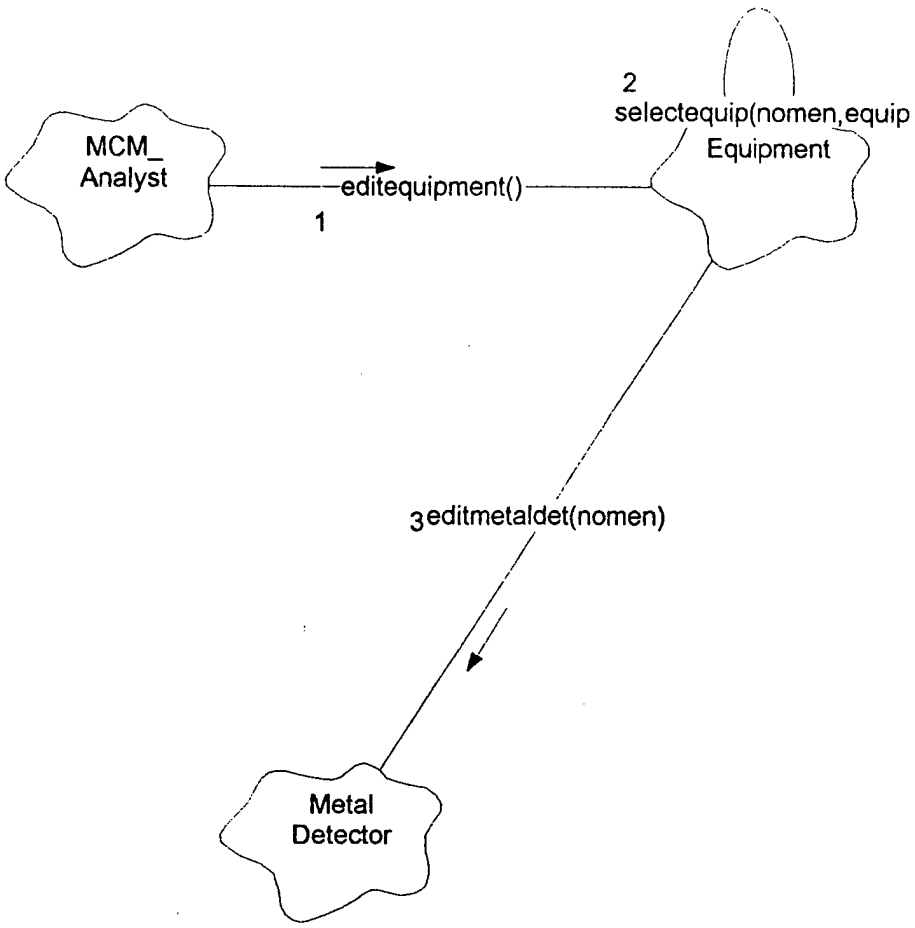

2
selectequip(nomen,equip
Equipment
MCM_
Analyst
editequipment()
1
editmagnetometer(nome
Magnetomet
er

Edit a Metal Detector

1. The user selects "Edit MCM Equipment" on the MCM Analyst's display. The Edit Equipment display appears.

2. The user selects the metal detector for editing. The Edit Metal Detector display appears.

3 The user edits the metal detector characteristics.

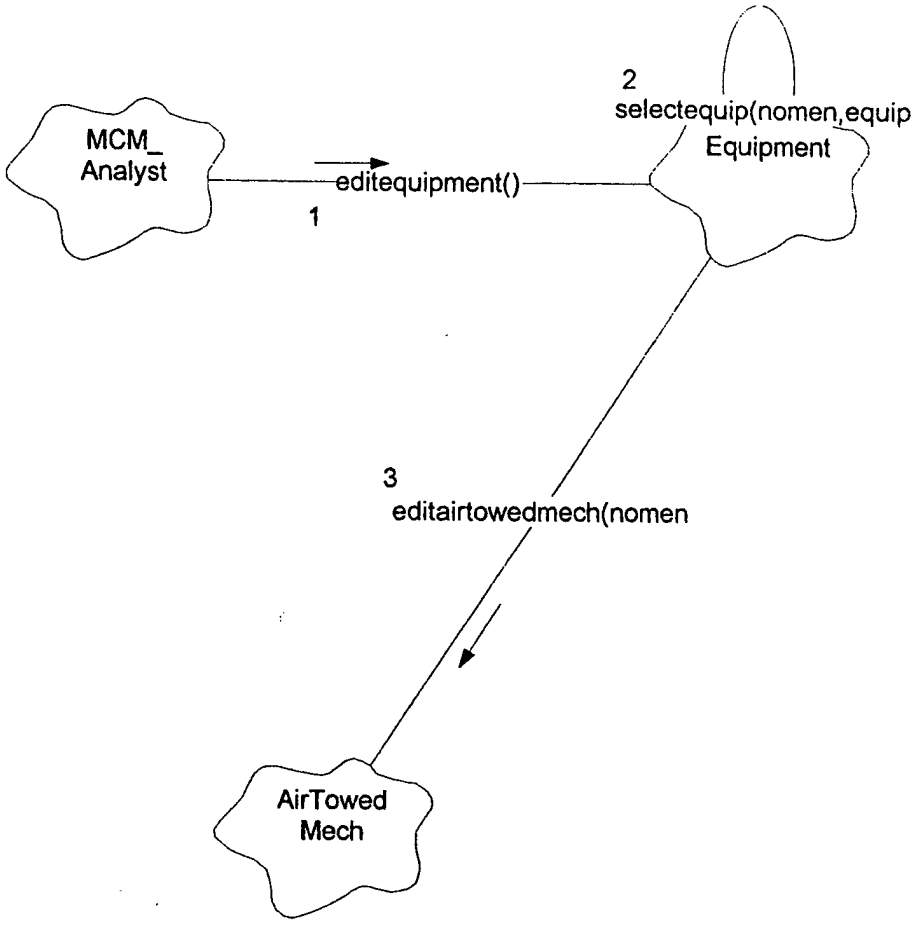

MCM_
Analyst
editequipment()
1
2
selectequip(nomen,equip
Equipment
3editmetaldet(nomen)
Metal
Detector

Edit an Air-Towed Mechanical Sweep Gear

1. The user selects "Edit MCM Equipment" on the MCM Analyst's display. The Edit Equipment display appears.

2. The user selects the air-towed mechanical sweep gear for editing. The Edit Air-Towed Mechanincal Sweep display appears.

3 The user edits the air-towed mechanical sweep characteristics.

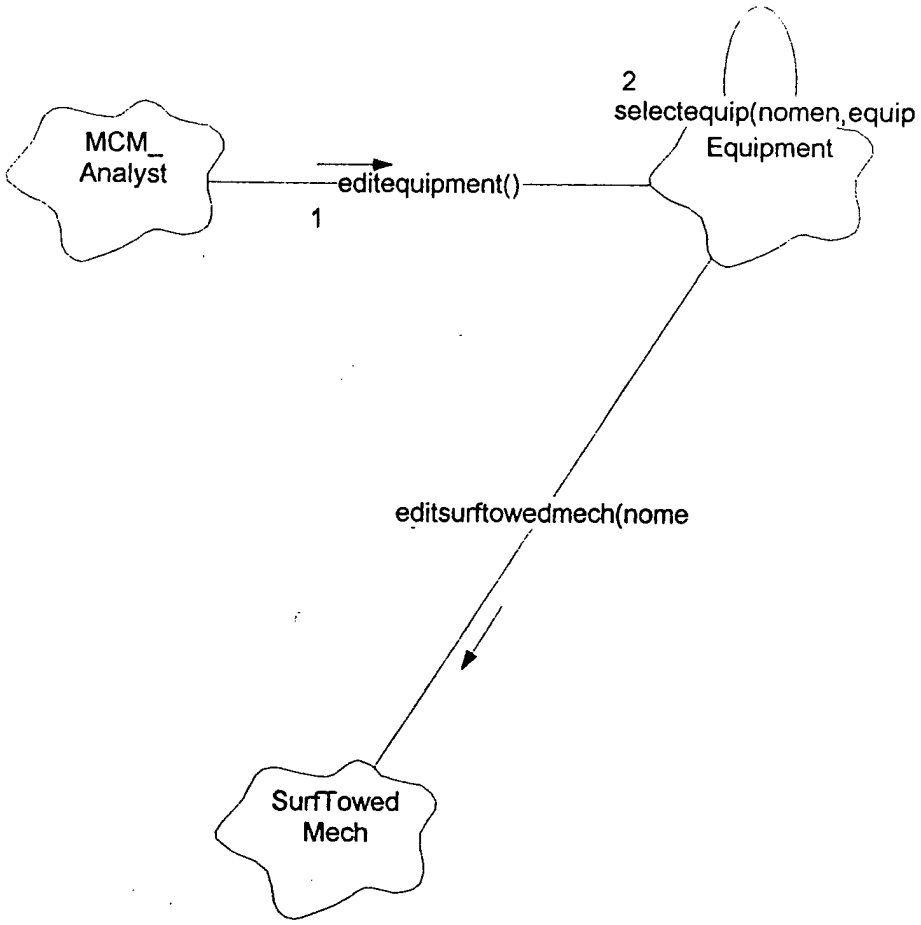

MCM_
Analyst
editequipment()
1
2
selectequip(nomen,equip
Equipment
3
editairtowedmech(nomen
AirTowed
Mech

Edit aSurface-Towed Mechanical Sweep Gear

1. The user selects "Edit MCM Equipment" on the MCM Analyst's display. The Edit Equipment display appears.

2. The user selects the surface-towed mechanical sweep gear for editing. The Edit Surface-Towed Mechanincal Sweep display appears.

3 The user edits the surface-towed mechanical sweep characteristics.

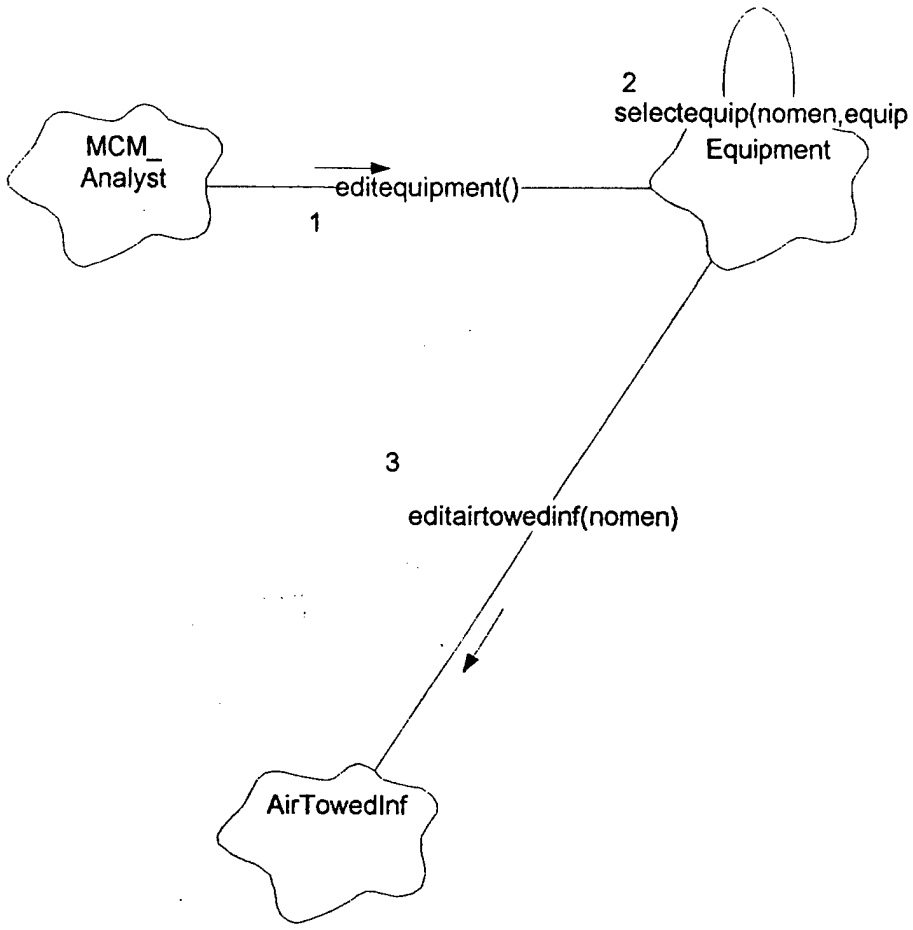

2
selectequip(nomen,equip
Equipment
MCM_
Analyst
editequipment()
1
editsurftowedmech(nome
SurfTowed
Mech

Edit an Air-Towed Influence Sweep Gear

1. The user selects "Edit MCM Equipment" on the MCM Analyst's display. The Edit Equipment display appears.

2. The user selects the air-towed influence sweep gear for editing. The Edit Air-Towed Influence Sweep display appears.

3 The user edits the air-towed influence sweep characteristics.

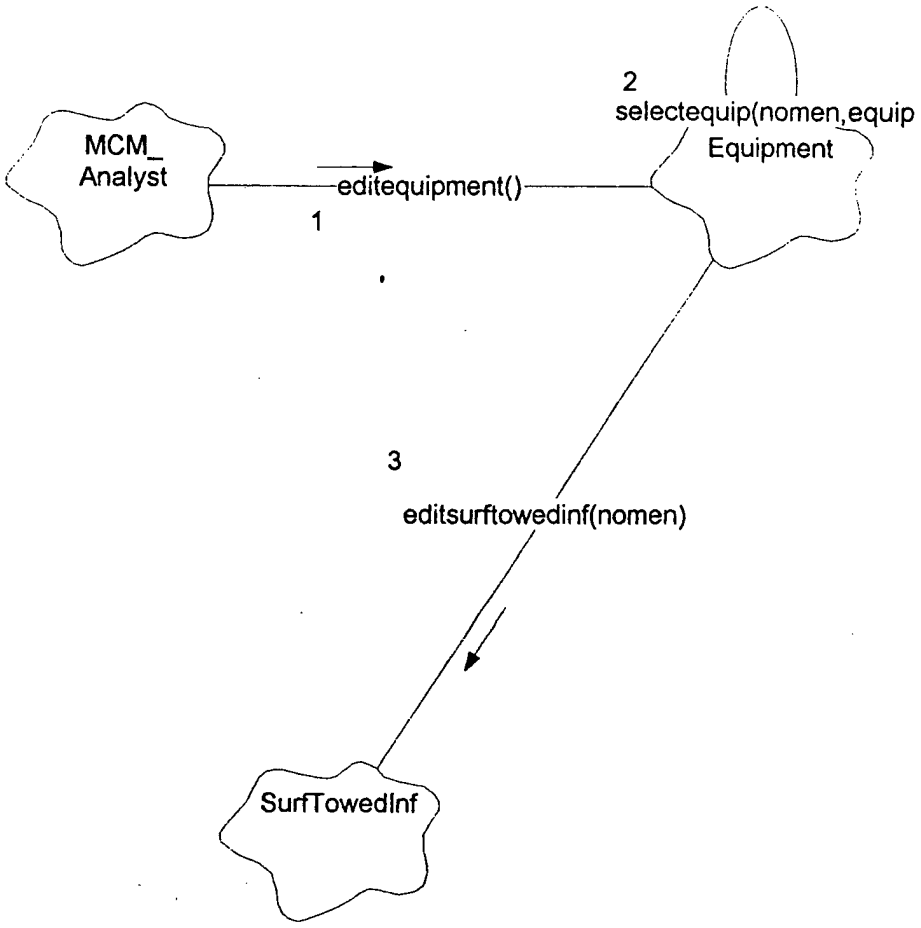

MCM_
Analyst
editequipment()
1
2
selectequip(nomen,equip
Equipment
3
editairtowedinf(nomen)
AirTowedInf

Edit a Surface-Towed Influence Sweep Gear

1. The user selects "Edit MCM Equipment" on the MCM Analyst's display. The Edit Equipment display appears.

2. The user selects the surface-towed influence sweep gear for editing. The Edit Surface-Towed Influence Sweep display appears.

3 The user edits the surface-towed influence sweep characteristics.

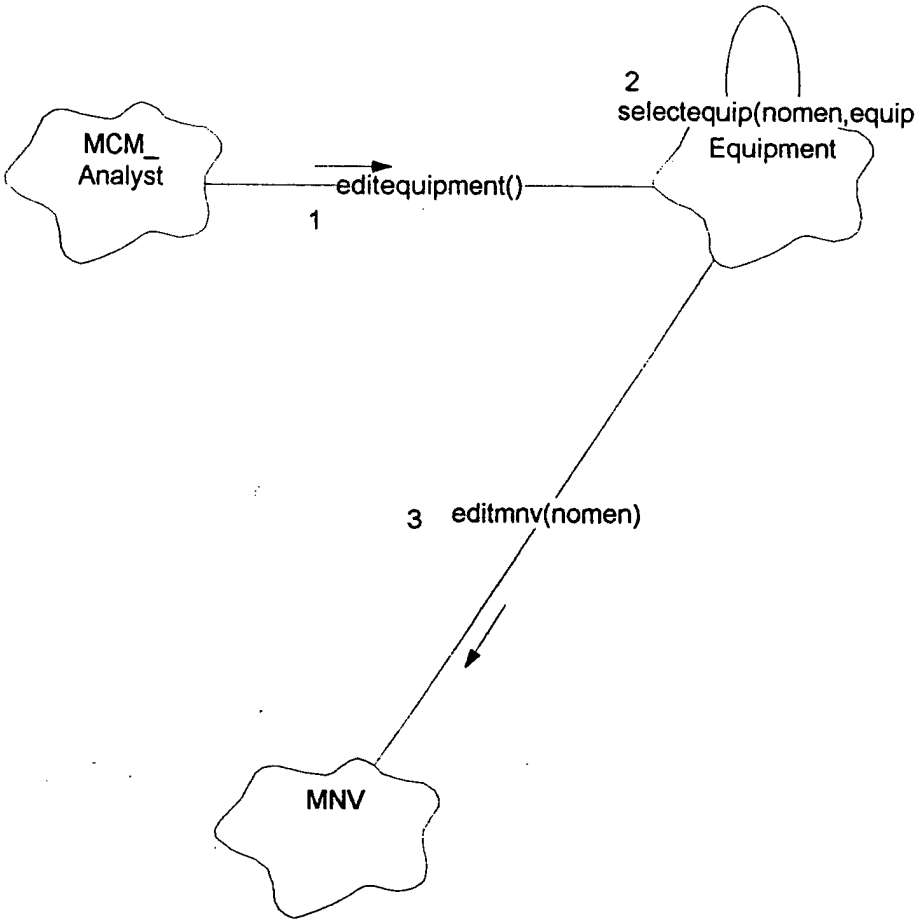

MCM_
Analyst
editequipment()
1
2
selectequip(nomen,equip
Equipment
3
editsurftowedinf(nomen)
SurfTowedInf

Edit a MNV

1. The user selects "Edit MCM Equipment" on the MCM Analyst's display. The Edit Equipment display appears.

2. The user selects the MNV for editing. The Edit MNV display appears.

3 The user edits the MNV characteristics.

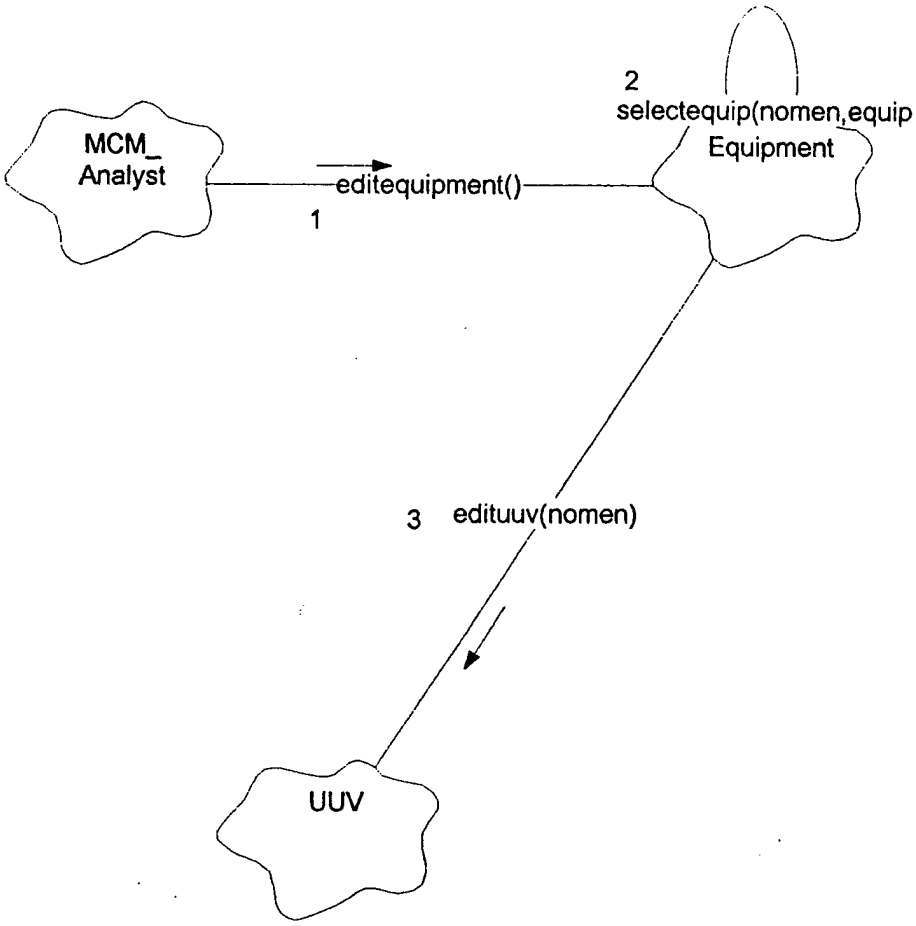

MCM_
Analyst
editequipment()
1
2
selectequip(nomen,equip
Equipment
3 editmnv(nomen)
MNV

Edit an UUV

1. The user selects "Edit MCM Equipment" on the MCM Analyst's display. The Edit Equipment display appears.

2. The user selects the UUV for editing. The Edit UUV display appears.

3 The user edits the UUV characteristics.

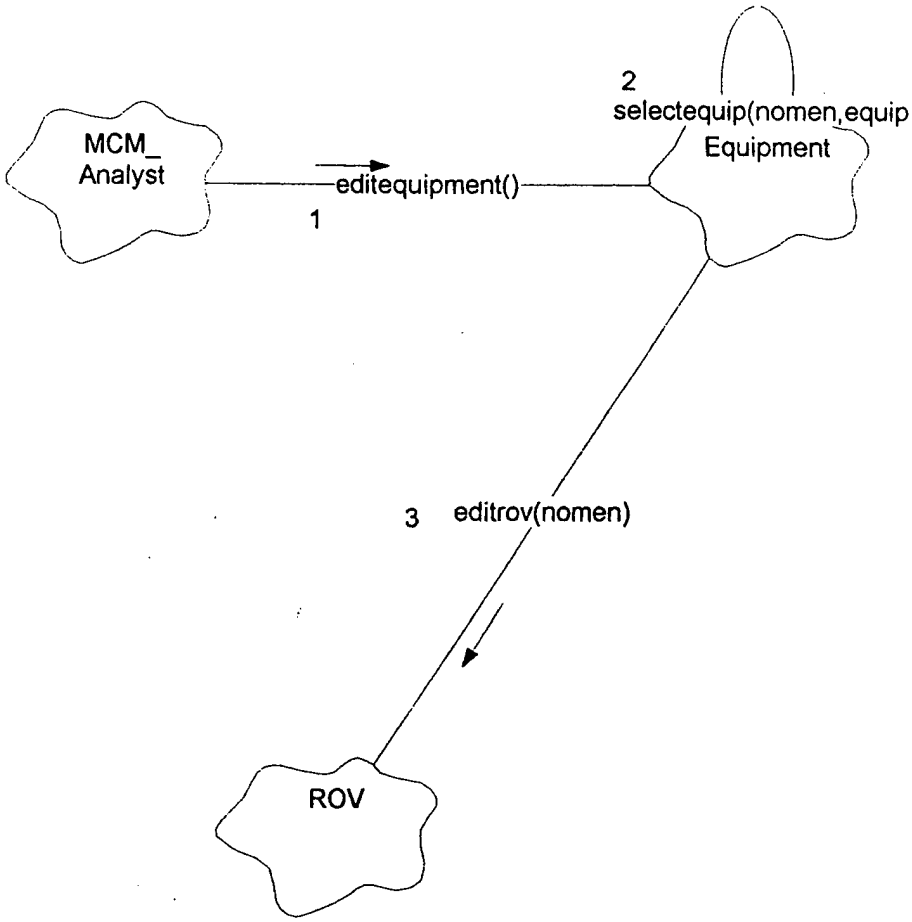

2
selectequip(nomen,equip
Equipment
MCM_
Analyst
editequipment()
1
3 edituuv(nomen)
UUV

Edit a ROV

1. The user selects "Edit MCM Equipment" on the MCM Analyst's display. The Edit Equipment display appears.

2. The user selects the ROV for editing. The Edit ROV display appears.

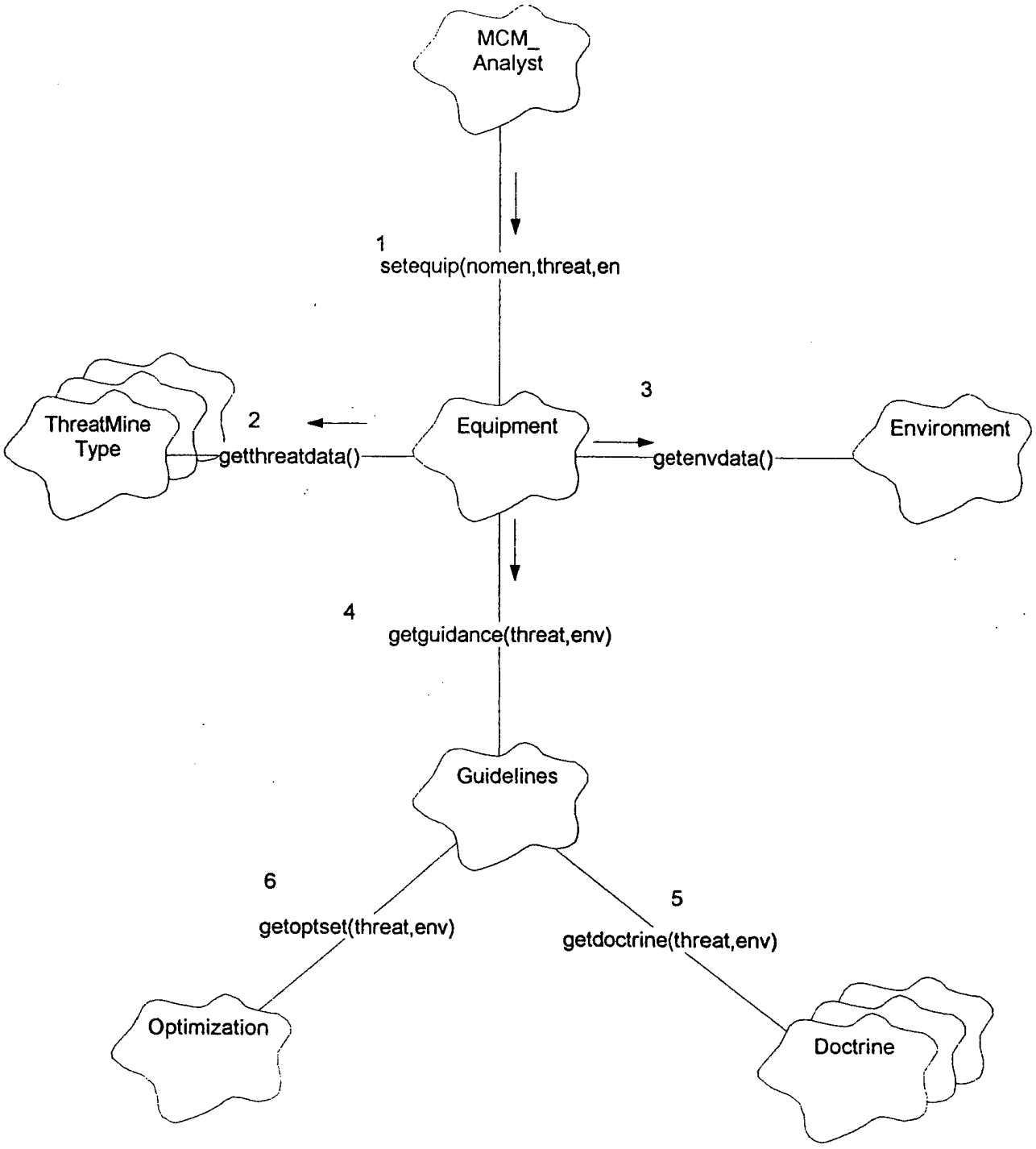
3 The user edits the ROV characteristics.


MCM_
Analyst
editequipment()
1
2
selectequip(nomen,equip
Equipment
3
editrov(nomen)
ROV

Use Case: Specify Equipment Settings

Scenario: Specify Equipment Settings

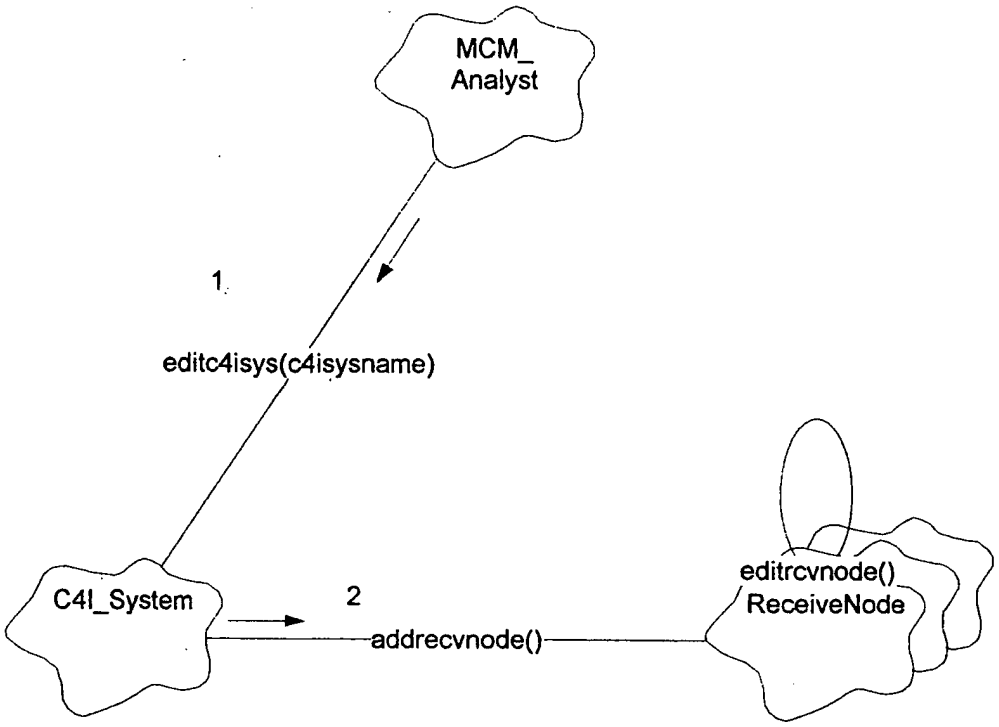
1. The user selects "Set Equipment".
2. The user selects the equipment for which the settings will be obtained.
3. The equipment object gets a list of the threat mines of interest.
4. The equipment gets environmental data.
5. The equipment calls the guidelines object with the threat and environmental data.
6. The guidelines object gets doctrinal guidance from doctrine objects.
7. The guidelines object calls an optimizer with the threat data, environmental data, and doctrinal guidance.


MCM_
Analyst
1
setequip(nomen,threat,en
ThreatMine
Type
2
getthreatdata()
Equipment
3
getenvdata()
Environment
4
getguidance(threat,env)
Guidelines
6
getoptset(threat,env)
5
getdoctrine(threat,env)
Optimization
Doctrine

Use Case: Define Characteristics for C4I Systems

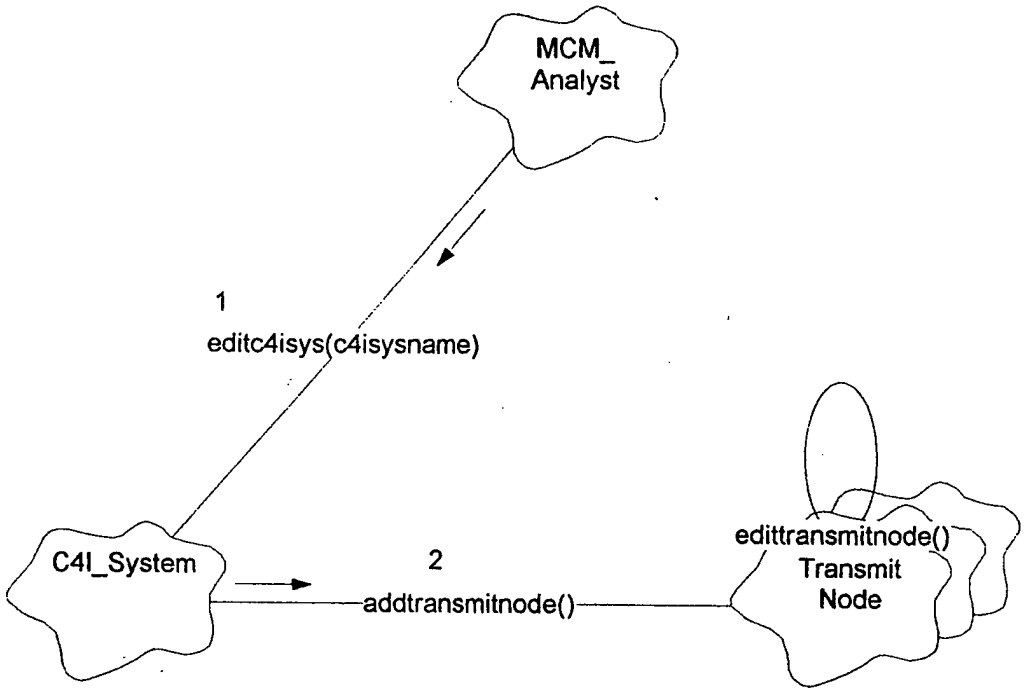
Scenario: Add a Receiving Node

1. The user selects "Edit C4I System" from the Analyst's Display.
2. The user selects "Add Receive Node" from the C4I System Display.
3. The user adds the identifier for the receiving node.


MCM_
Analyst
1
editc4isys(c4isysname)
C4I_System
2
addrecvnode()
editrcvnode()
ReceiveNode

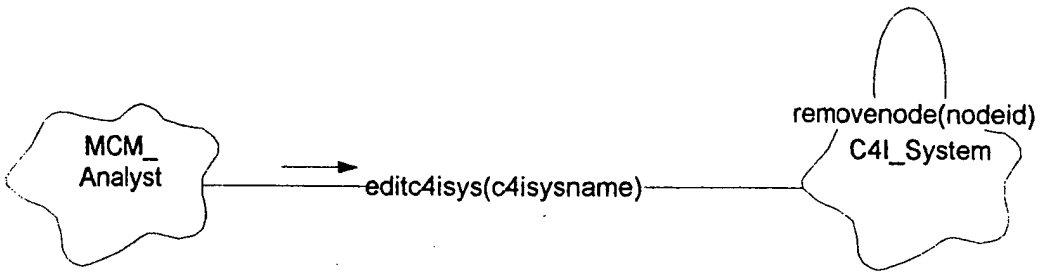
Scenario: Add a Transmit Node

1. The user selects "Edit C4I System" from the Analyst's Display.
2. The user selects "Add Transmit Node" from the C4I System Display.
3. The user adds the identifier for the transmitting node.


MCM_
Analyst
1
editc4isys(c4isysname)
C4I_System
2
addtransmitnode()
edittransmitnode()
Transmit
Node

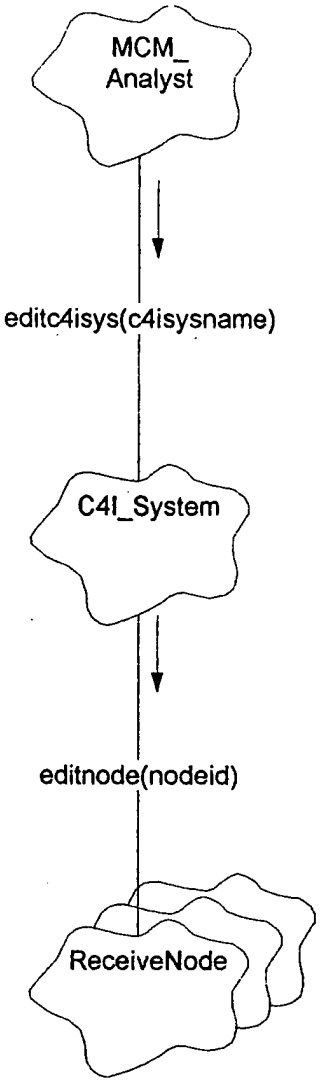
Scenario: Delete a Node

1. The user selects "Edit C4I System" from the Analyst's Display.
2. The user selects a node to be deleted and confirms the deletion.


MCM_
Analyst
editc4isys(c4isysname)
removenode(nodeid)
C4I_System

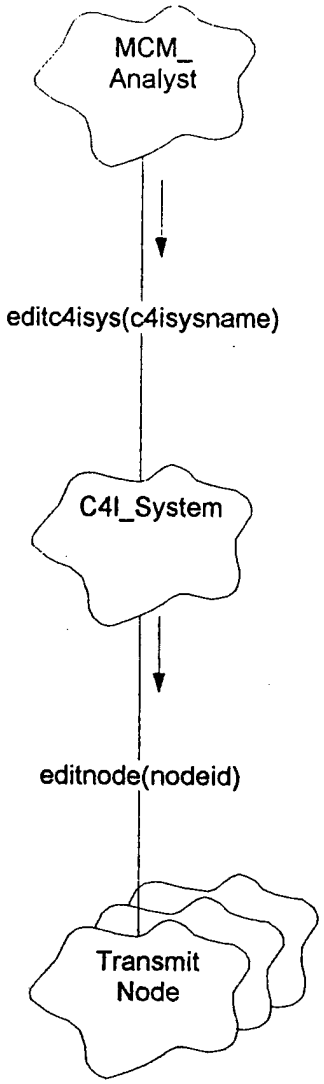
Scenario: Edit a Receiving Node

1. The user selects "Edit C4I System" from the Analyst's Display.
2. The user selects "Add Receive Node" from the C4I System Display.
3. The user selects and edits the receiving node.


MCM_
Analyst
editc4isys(c4isysname)
C4I_System
editnode(nodeid)
ReceiveNode

Scenario: Edit a Transmit Node

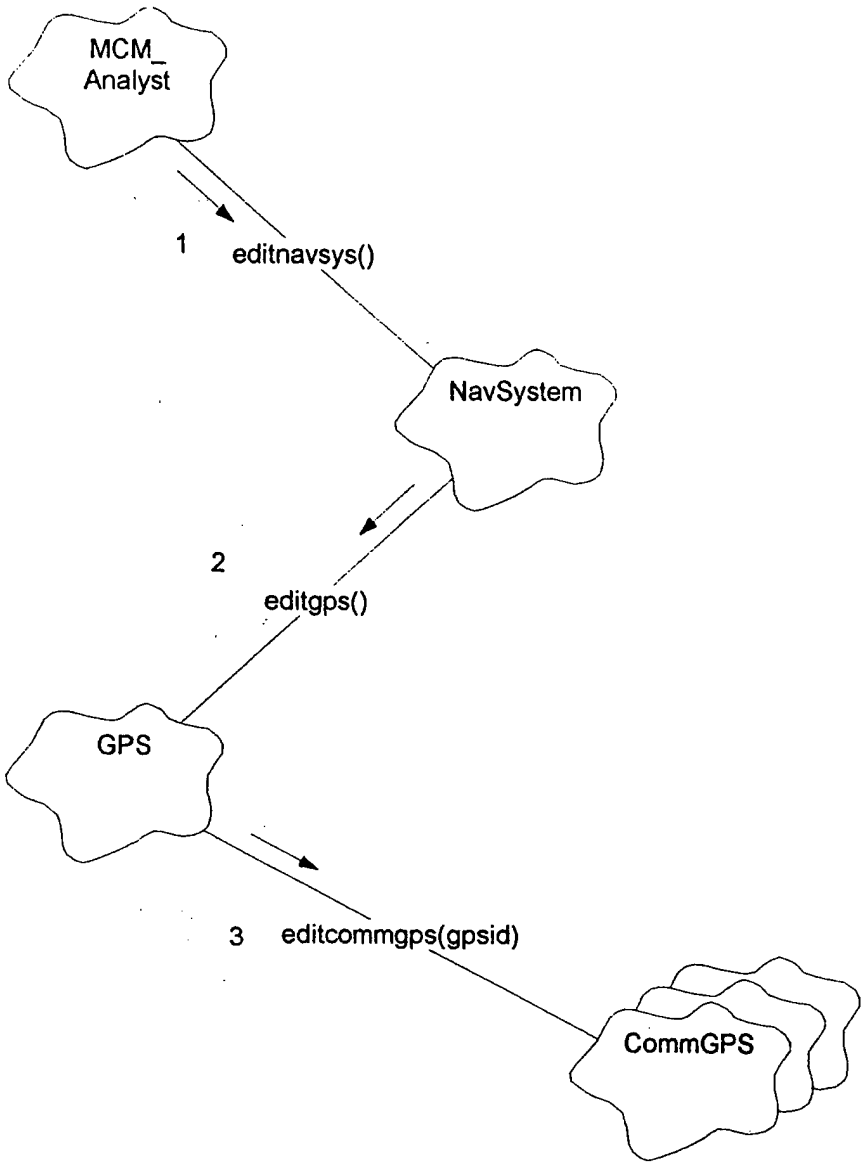
1. The user selects "Edit C4I System" from the Analyst's Display.
2. The user selects "Add Transmit Node" from the C4I System Display.
3. The user selects and edits the transmitting node.


MCM_
Analyst
editc4isys(c4isysname)
C4I_System
editnode(nodeid)
Transmit
Node

Use Case: Define Characteristics for Navigation Systems

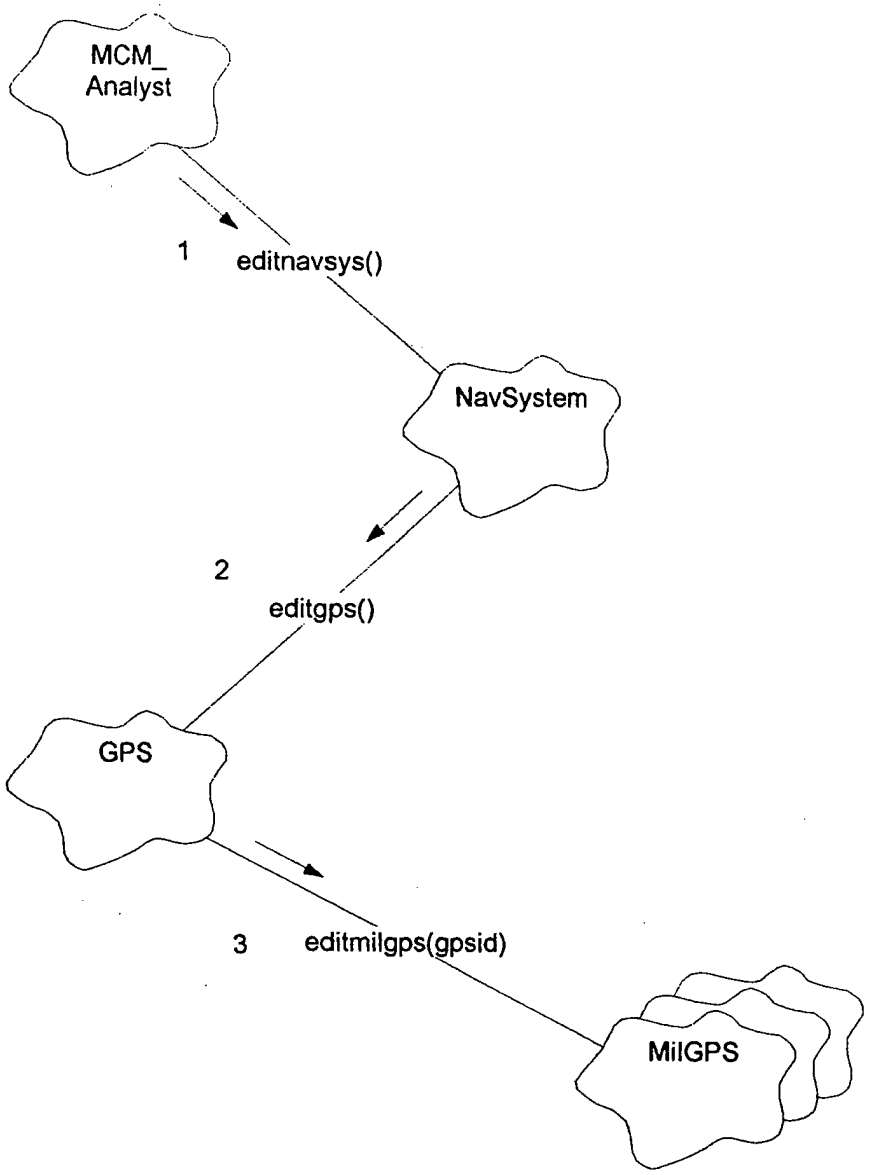
Scenario: Define Characteristics for Commercial GPS

1. The user selects "Edit Navigation System" in the MCM Analyst's Display.
2. The user selects "Edit GPS" in the Navigation Display.
3. The user selects a commercial GPS in the GPS Display.
4. The user edits the characteristics of the GPS selected.


MCM_
Analyst
1
editnavsys()
NavSystem
2
editgps()
GPS
3
editcommgps(gpsid)
CommGPS

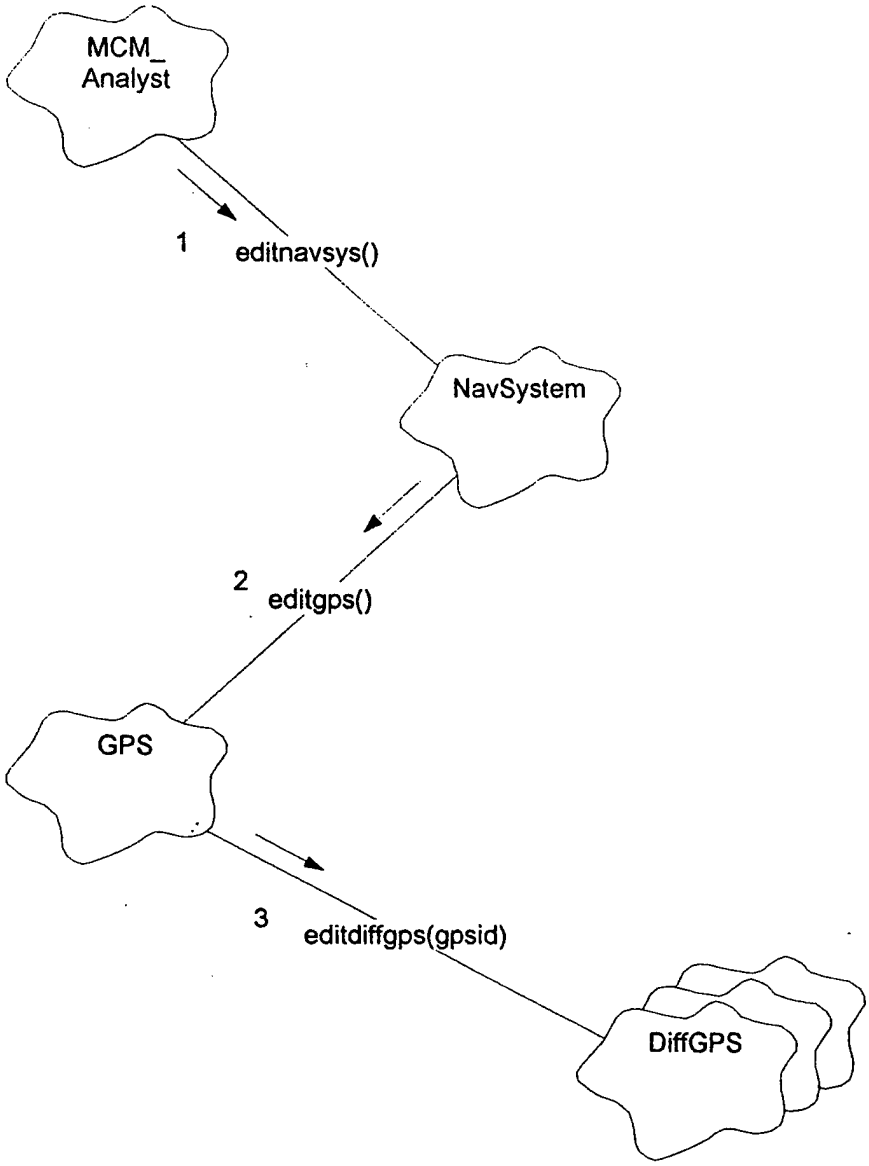
Scenario: Define Characteristics for Military GPS

1. The user selects "Edit Navigation System" in the MCM Analyst's Display.
2. The user selects "Edit GPS" in the Navigation Display.
3. The user selects a military GPS in the GPS Display.
4. The user edits the characteristics of the GPS selected.


MCM_
Analyst
1
editnavsys()
NavSystem
2
editgps()
GPS
3
editmilgps(gpsid)
MilGPS

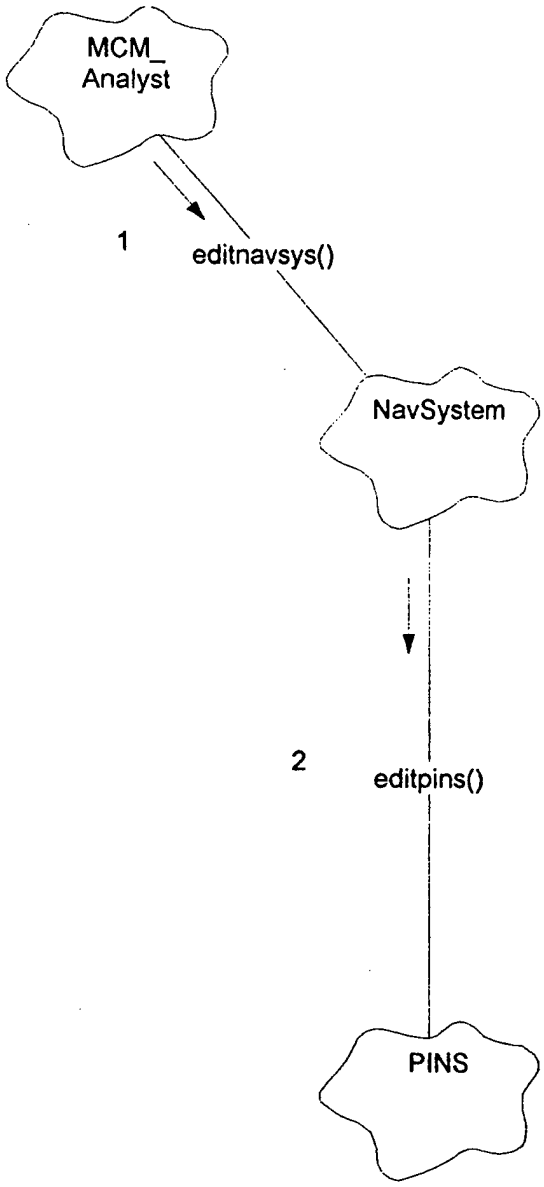
Scenario: Define Characteristics for Differential GPS

1. The user selects "Edit Navigation System" in the MCM Analyst's Display.
2. The user selects "Edit GPS" in the Navigation Display.
3. The user selects a differential GPS in the GPS Display.
4. The user edits the characteristics of the GPS selected.


MCM_
Analyst
1
editnavsys()
NavSystem
2
editgps()
GPS
3
editdiffgps(gpsid)
DiffGPS

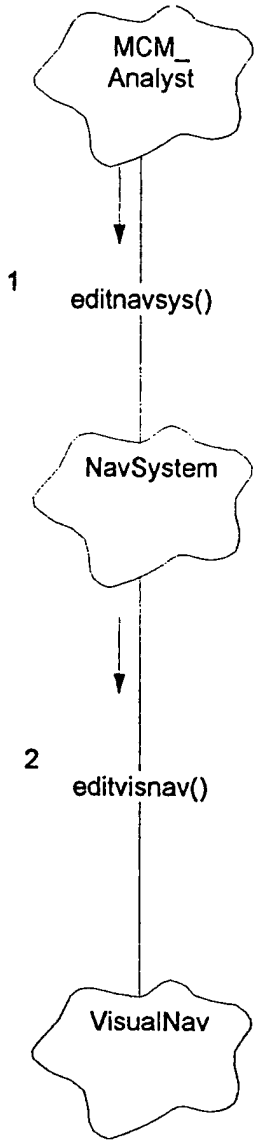
Scenario: Define Characteristics for PINS

1. The user selects "Edit Navigation System" in the MCM Analyst's Display.
2. The user selects "Edit PINS" in the Navigation Display.
3. The user edits the PINS characteristics.


MCM_
Analyst
1
editnavsys()
NavSystem
2
editpins()
PINS

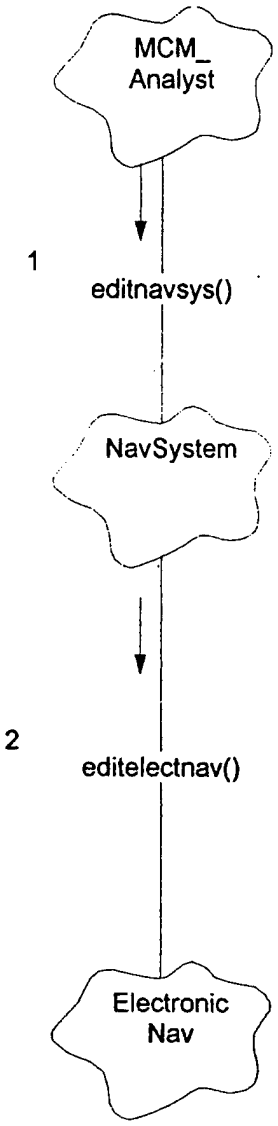
Scenario: Define Characteristics for Visual Navigation Systems

1. The user selects "Edit Navigation System" in the MCM Analyst's Display.
2. The user selects "Edit Visual Navigation" in the Navigation Display.
3. The user edits the visual navigation system characteristics.


MCM_
Analyst
1
editnavsys()
NavSystem
2
editvisnav()
VisualNav

Scenario: Define Characteristics for Electronic Navigation Systems

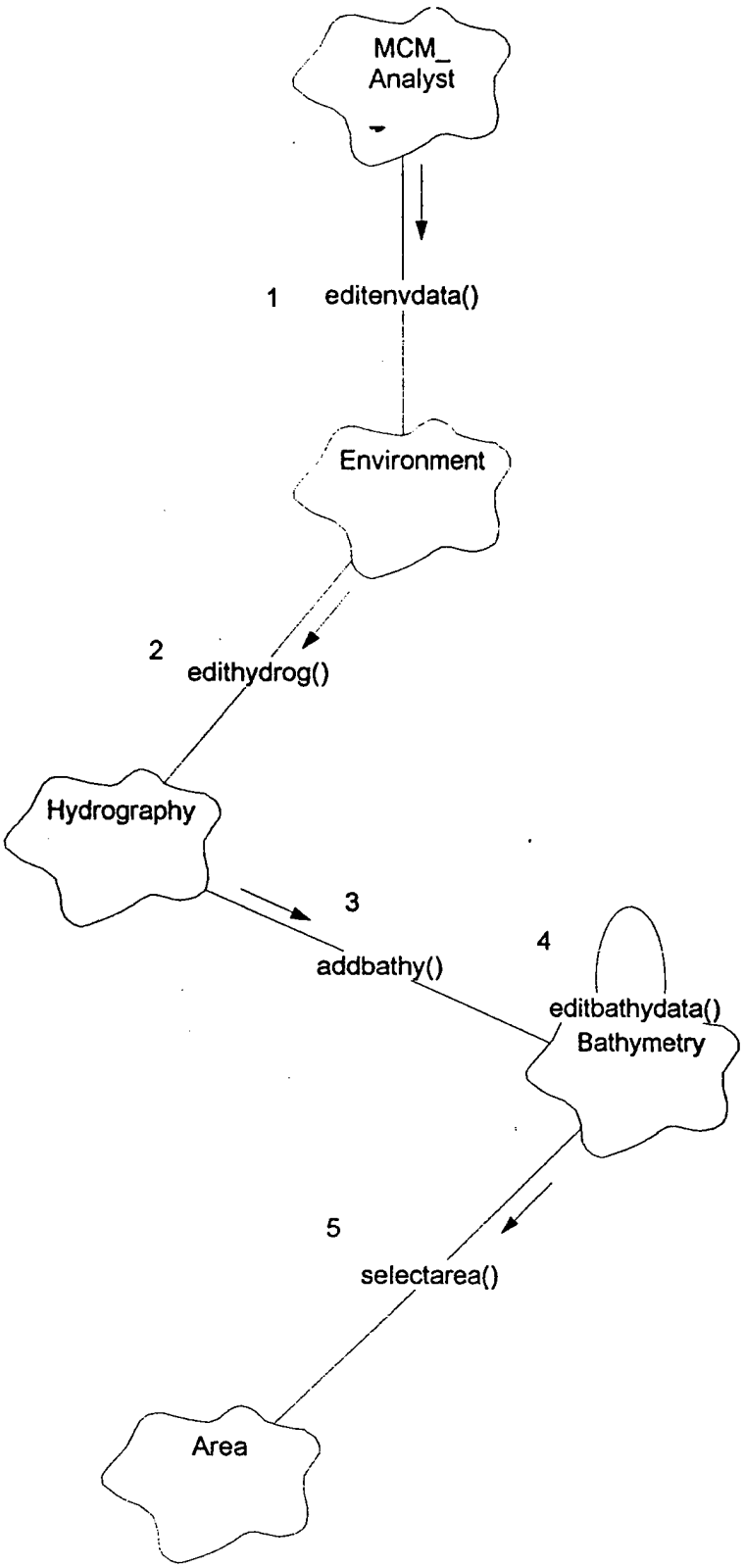
1. The user selects "Edit Navigation System" in the MCM Analyst's Display.
2. The user selects "Edit Electronic Navigation" in the Navigation Display.
3. The user edits the characteristics for the electronic navigation system.


MCM_
Analyst
1
editnavsys()
NavSystem
2
editelectnav()
Electronic
Nav

Use Case: Specify Environmental Data

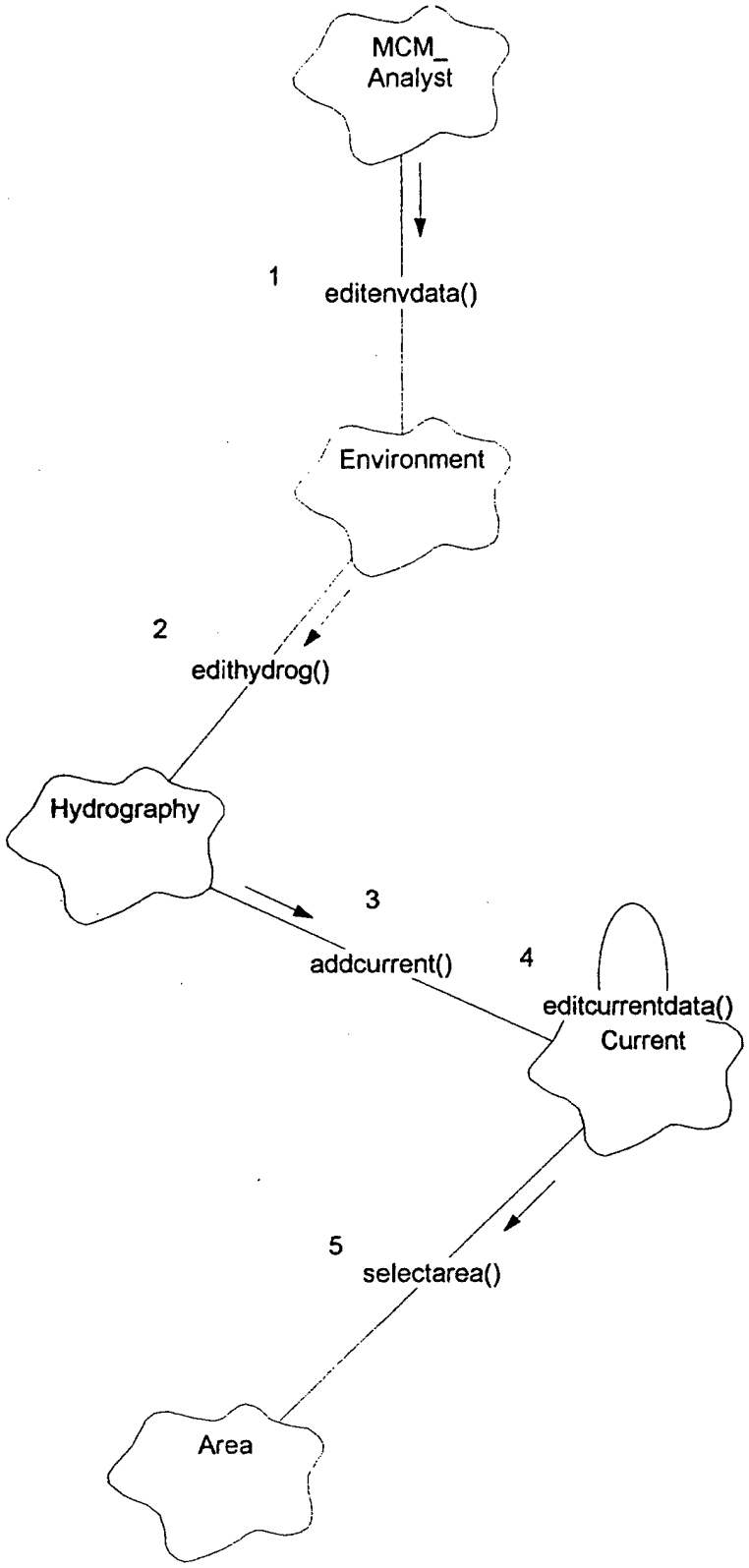
Scenario: Add Bathymetry Data for an Area

1. The user selects "Environment" from the MCM Analyst's Display.
2. The user selects "Hydrography" from the Environment Object.
3. The user selects "Add Bathymetry" from the Hydrography Display.
4. The user enters and edits the bathymetry data in the Bathymetry Display.
5. The user creates an area associated with the bathymetry data.


MCM_
Analyst
1
editenvdata()
Environment
2
edithydrog()
Hydrography
3
addbathy()
4
editbathydata()
Bathymetry
5
selectarea()
Area

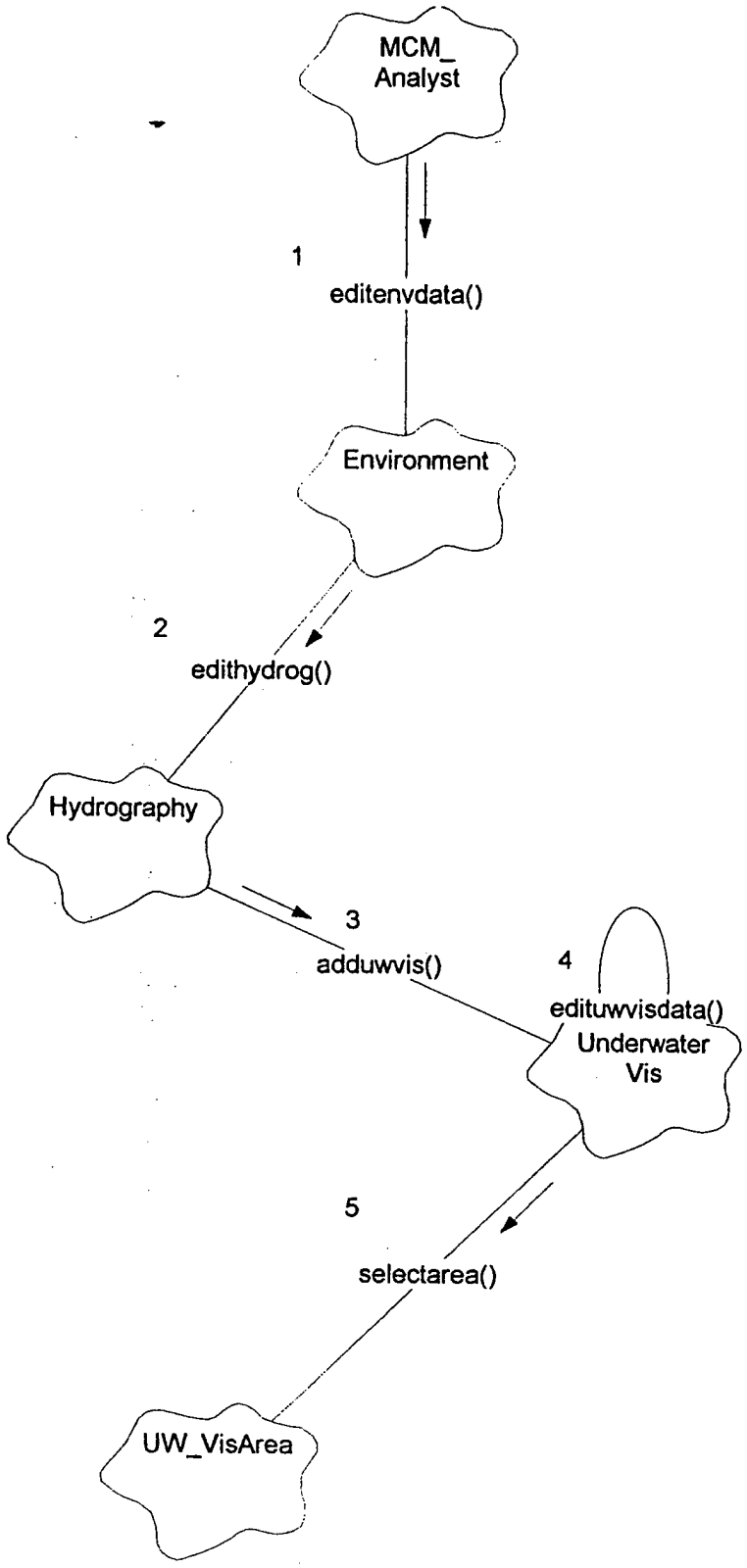
Scenario: Add Current Data

1. The user selects "Environment" from the MCM Analyst's Display.
2. The user selects "Hydrography" from the Environment Object.
3. The user selects "Add Current Data" from the Hydrography Display.
4. The user enters and edits the bathymetry data in the Currents Display.
5. The user creates an area associated with the current data.


MCM_
Analyst
1
editenvdata()
Environment
2
edithydrog()
Hydrography
3
addcurrent()
4
editcurrentdata()
Current
5
selectarea()
Area

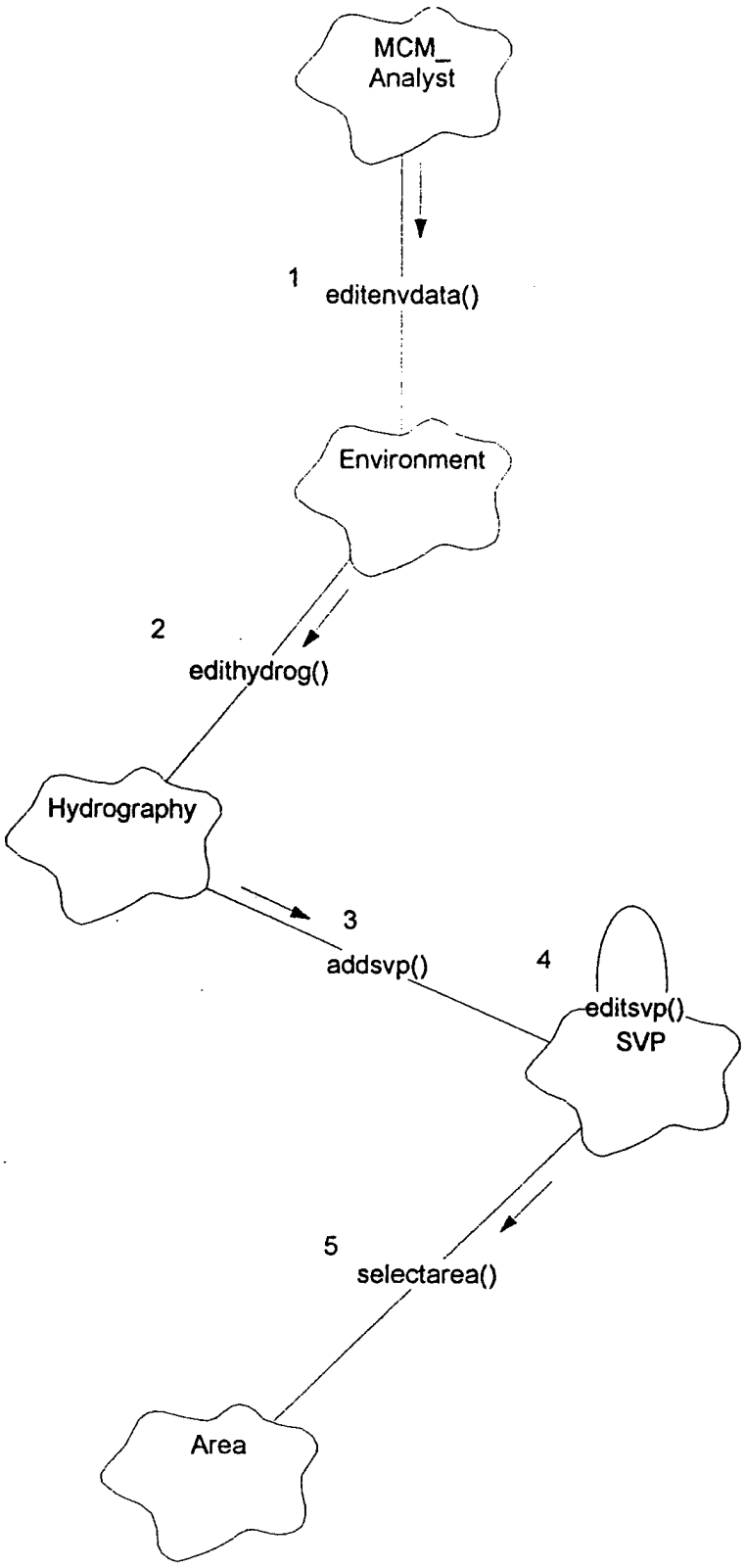
Scenario: Add Underwater Visibility Data

1. The user selects "Environment" from the MCM Analyst's Display.
2. The user selects "Hydrography" from the Environment Object.
3. The user selects "Add Underwater Visibility Data" from the Hydrography Display.
4. The user enters and edits the visibility data in the Underwater Visibility Display.
5. The user creates an area associated with the bathymetry data.


MCM_
Analyst
1
editenvdata()
Environment
2
edithydrog()
Hydrography
3
adduwvis()
4
edituwvisdata()
Underwater
Vis
5
selectarea()
UW_VisArea

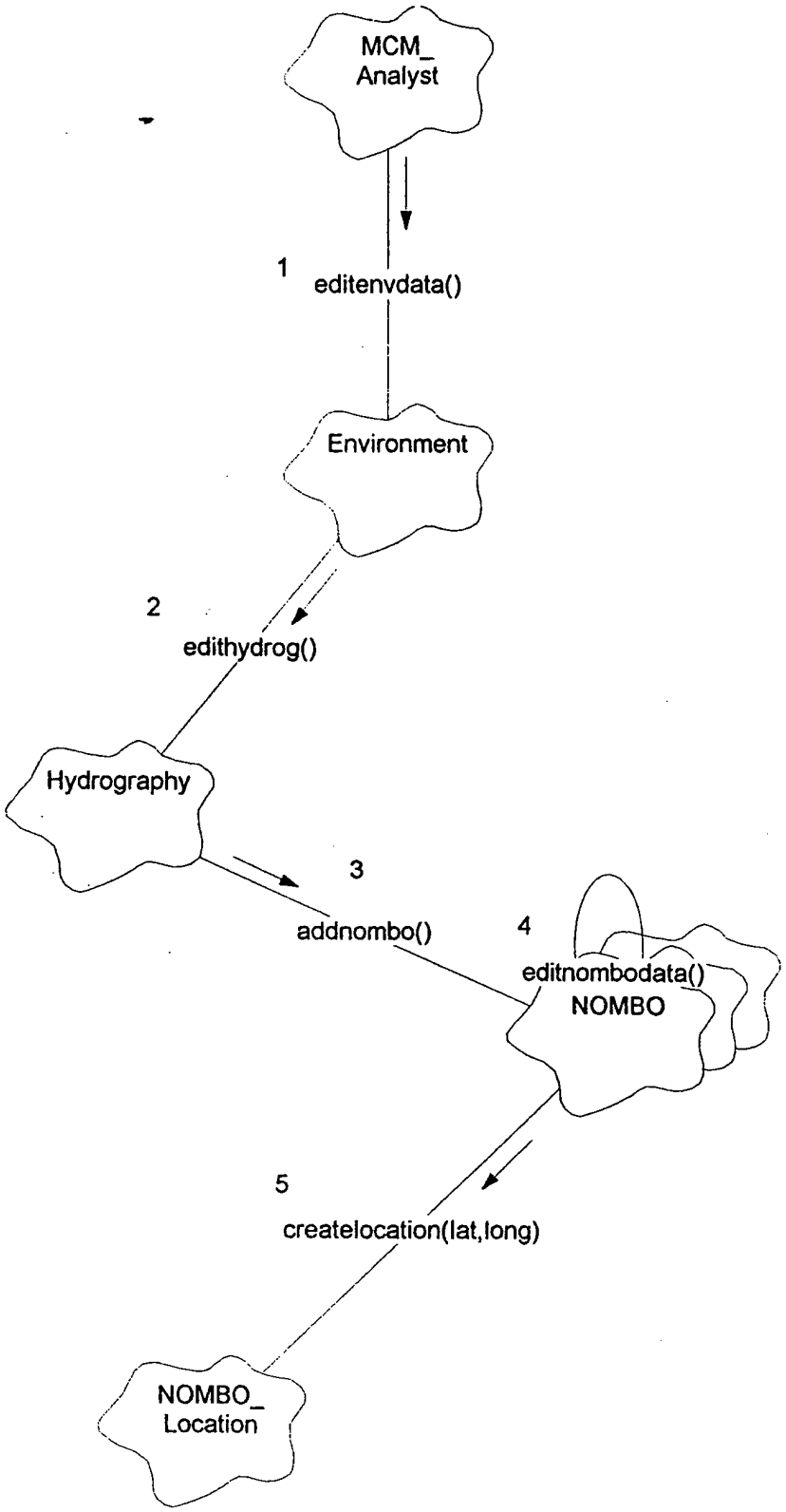
Scenario: Add SVP Data

1. The user selects "Environment" from the MCM Analyst's Display.
2. The user selects "Hydrography" from the Environment Object.
3. The user selects "Add SVP" from the Hydrography Display.
4. The user enters and edits the SVP data in the SVP Display.
5. The user creates an area associated with the SVP data.


MCM_
Analyst
1
editenvdata()
Environment
2
edithydrog()
Hydrography
3
addsvp()
4
editsvp()
SVP
5
selectarea()
Area

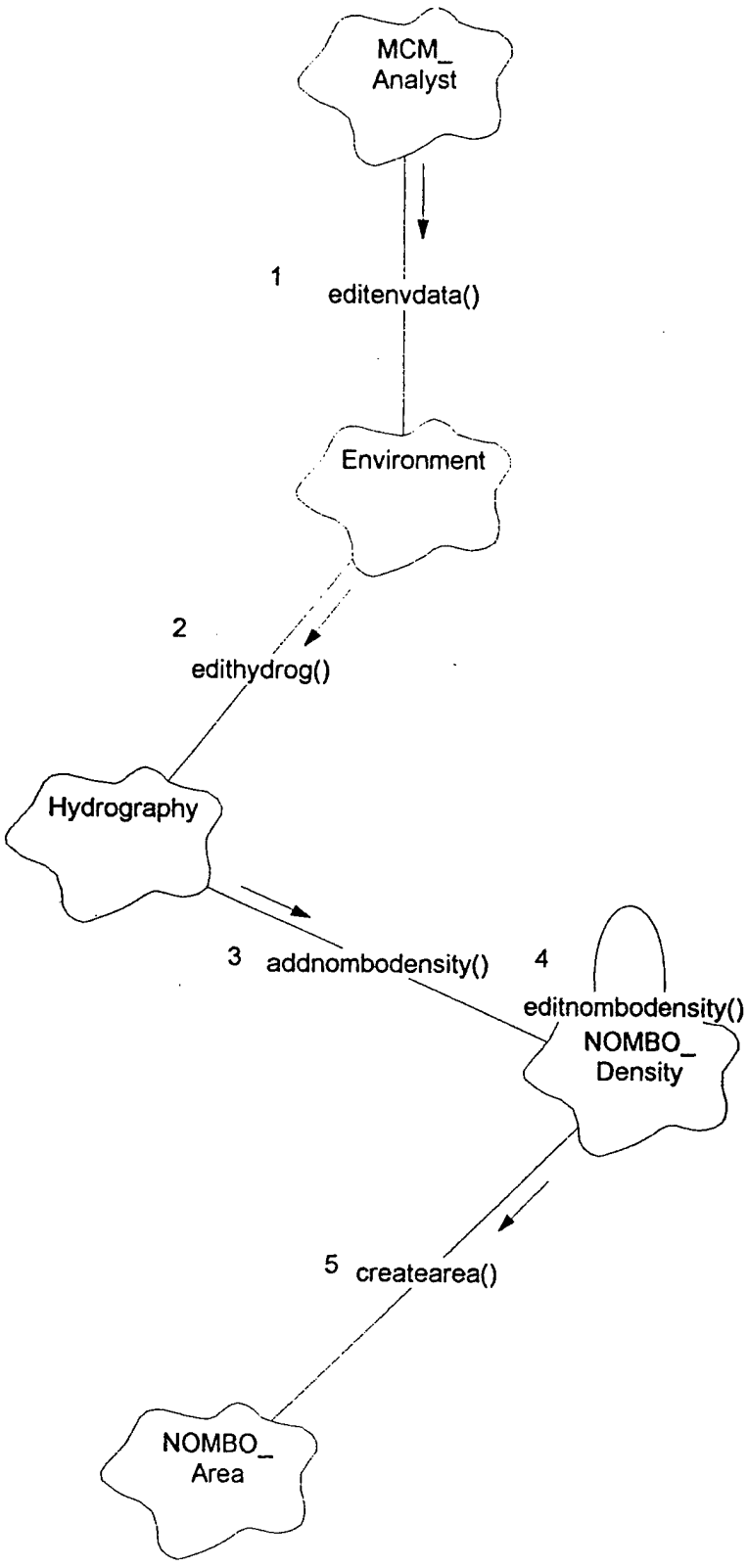
Scenario: Add NOMBO Location Data

1. The user selects "Environment" from the MCM Analyst's Display.
2. The user selects "Hydrography" from the Environment Object.
3. The user selects "Add NOMBO" from the Hydrography Display.
4. The user enters and edits the NOMBO data in the Edit NOMBO Display.
5. The user enters the location associated with the bathymetry data.


MCM_
Analyst
1
editenvdata()
Environment
2
edithydrog()
Hydrography
3
addnombo()
4
editnombodata()
NOMBO
5
createlocation(lat,long)
NOMBO_
Location

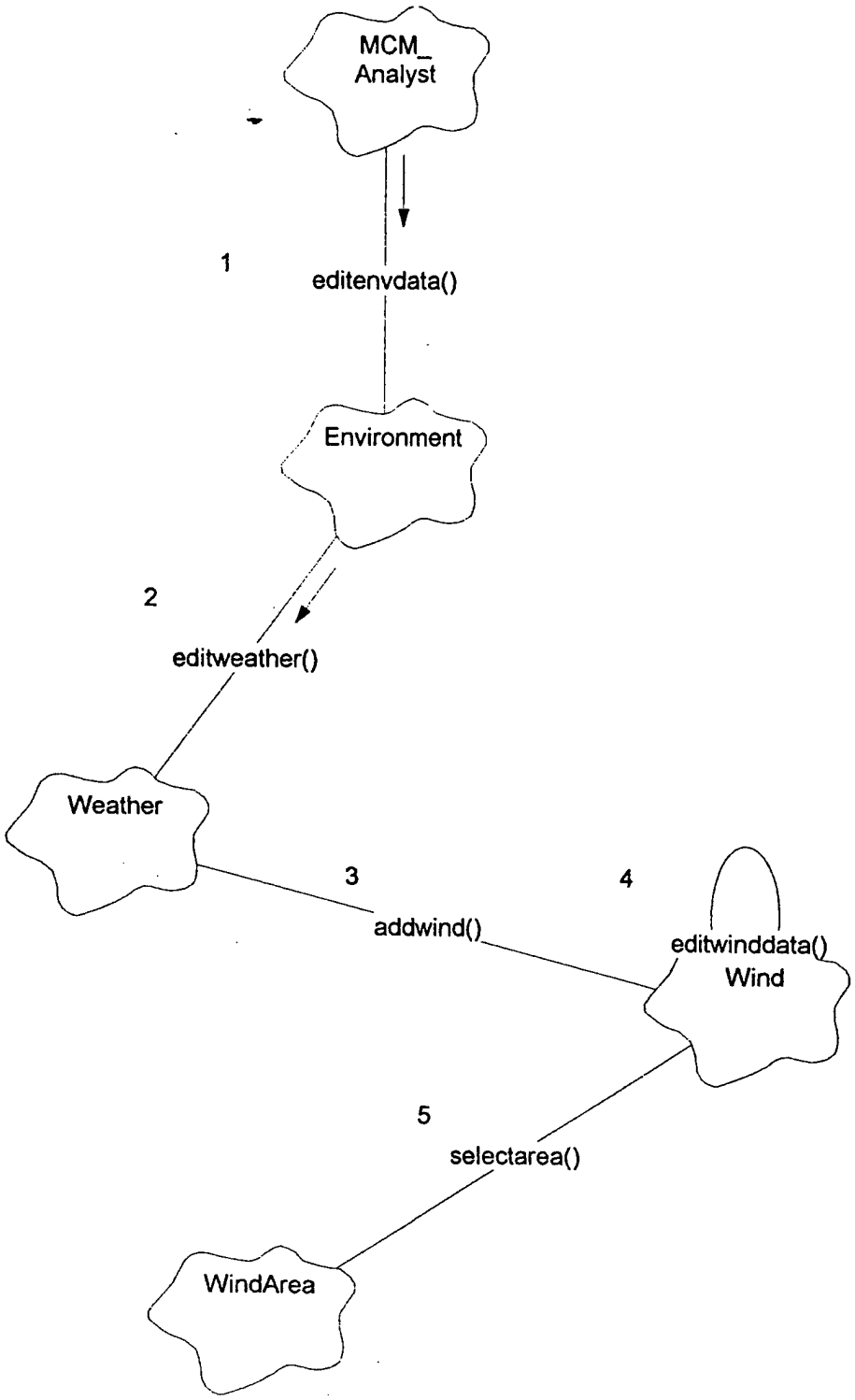
Scenario: Add NOMBO Density Data for an Area

1. The user selects "Environment" from the MCM Analyst's Display.
2. The user selects "Hydrography" from the Environment Object.
3. The user selects "Add NOMBO Density Data" from the Hydrography Display.
4. The user enters and edits the NOMBO density data in the NOMBO Density Display.
5. The user creates an area associated with the NOMBO density data.


MCM_
Analyst
1
editenvdata()
Environment
2
edithydrog()
Hydrography
3
addnombodensity()
4
editnombodensity()
NOMBO_
Density
5
createarea()
NOMBO_
Area

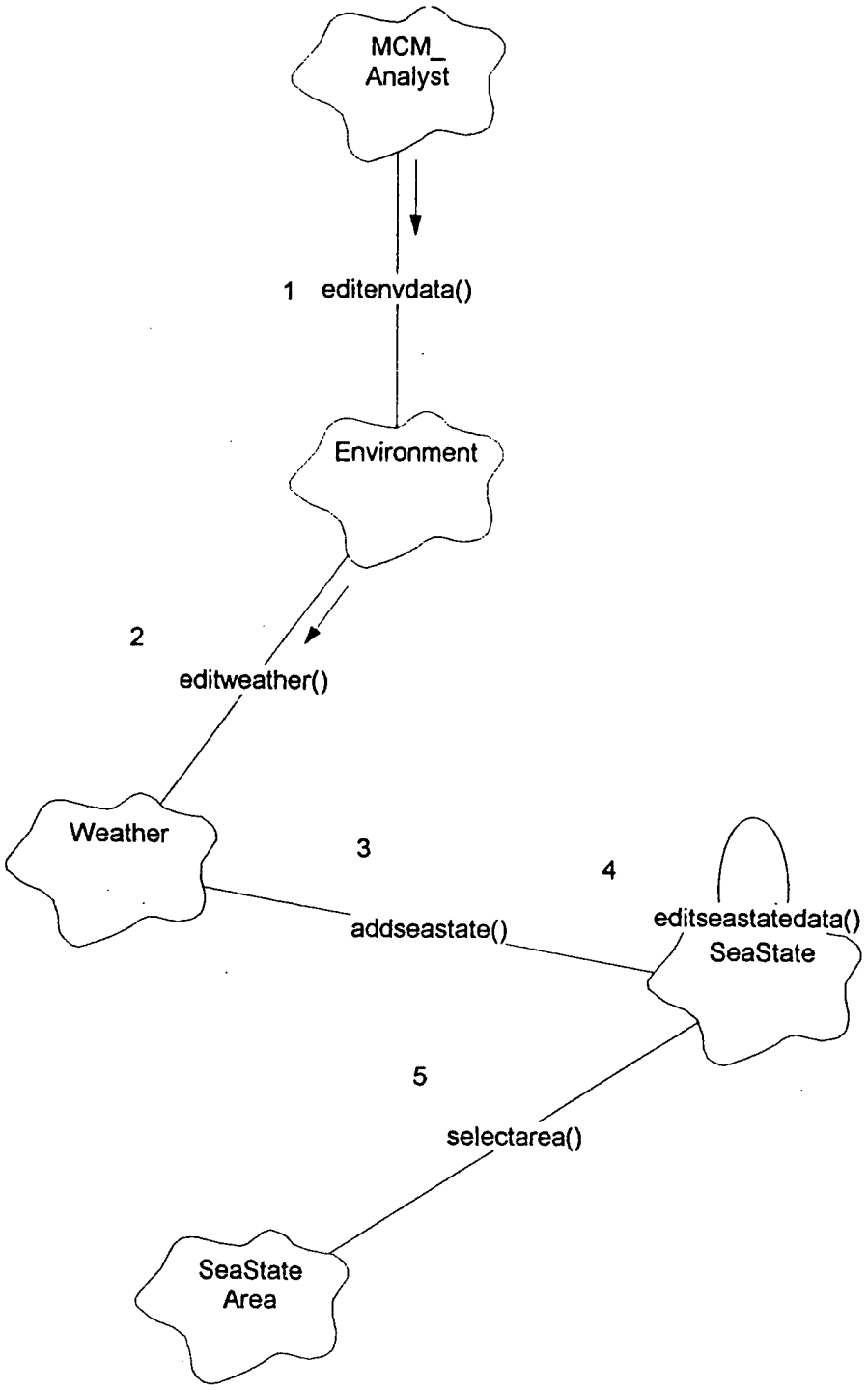
Scenario: Add Wind Data

1. The user selects "Environment" from the MCM Analyst's Display.
2. The user selects "Weather" from the Environment Object.
3. The user selects "Add Wind Data" from the Weather Display.
4. The user enters and edits the wind data in the Wind Display.
5. The user creates an area associated with the wind data.


MCM_
Analyst
1
editenvdata()
Environment
2
editweather()
Weather
3
addwind()
4
editwinddata()
Wind
5
selectarea()
WindArea

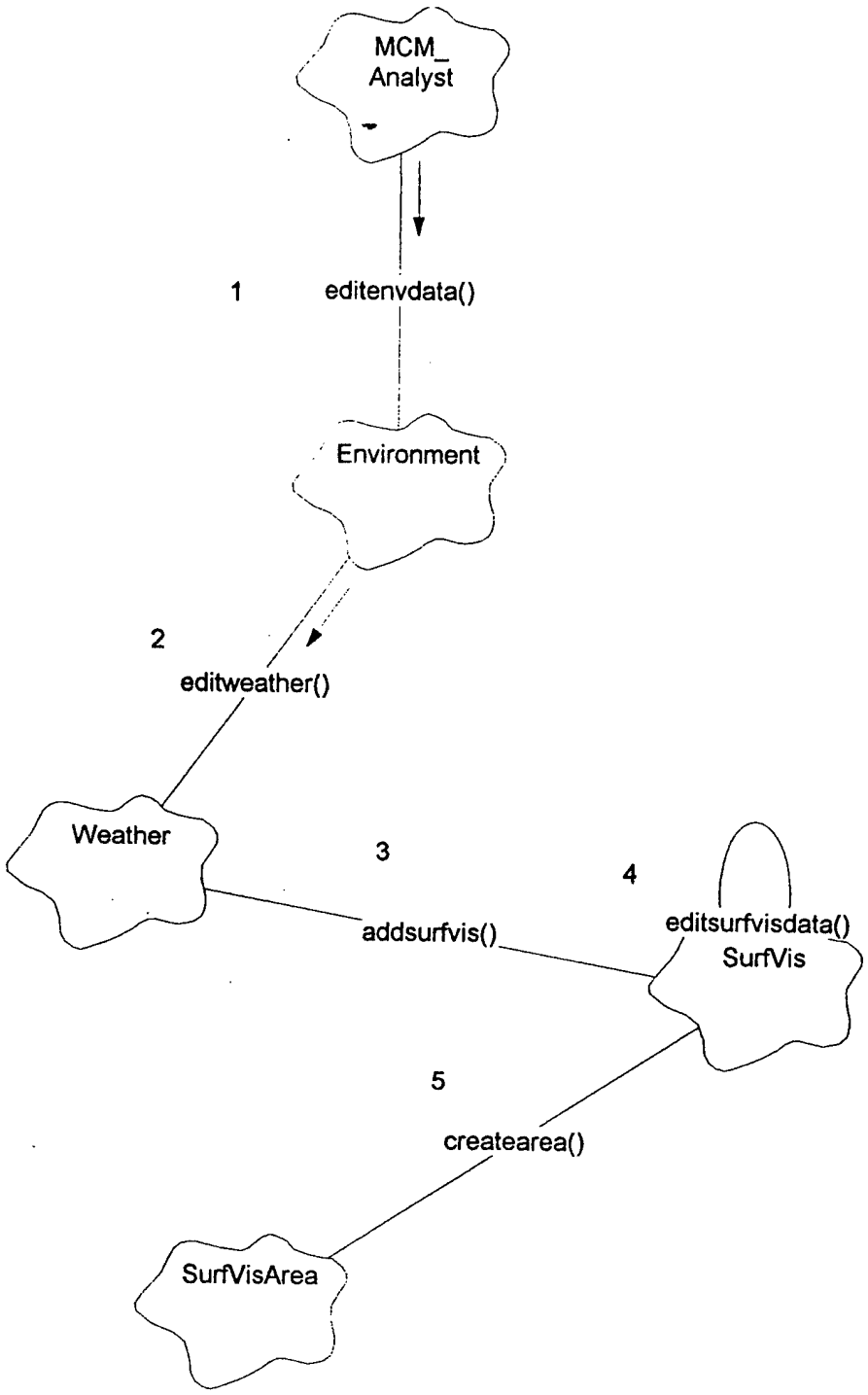
Scenario: Add Sea State Data

1. The user selects "Environment" from the MCM Analyst's Display.
2. The user selects "Weather" from the Environment Object.
3. The user selects "Add Sea State Data" from the Weather Display.
4. The user enters and edits the Sea State data in the Sea State Display.
5. The user creates an area associated with the Sea State data.


MCM_
Analyst
1 editenvdata()
Environment
2
editweather()
Weather
3
addseastate()
4
editseastatedata()
SeaState
5
selectarea()
SeaState
Area

Scenario: Add Surface Visibility Data

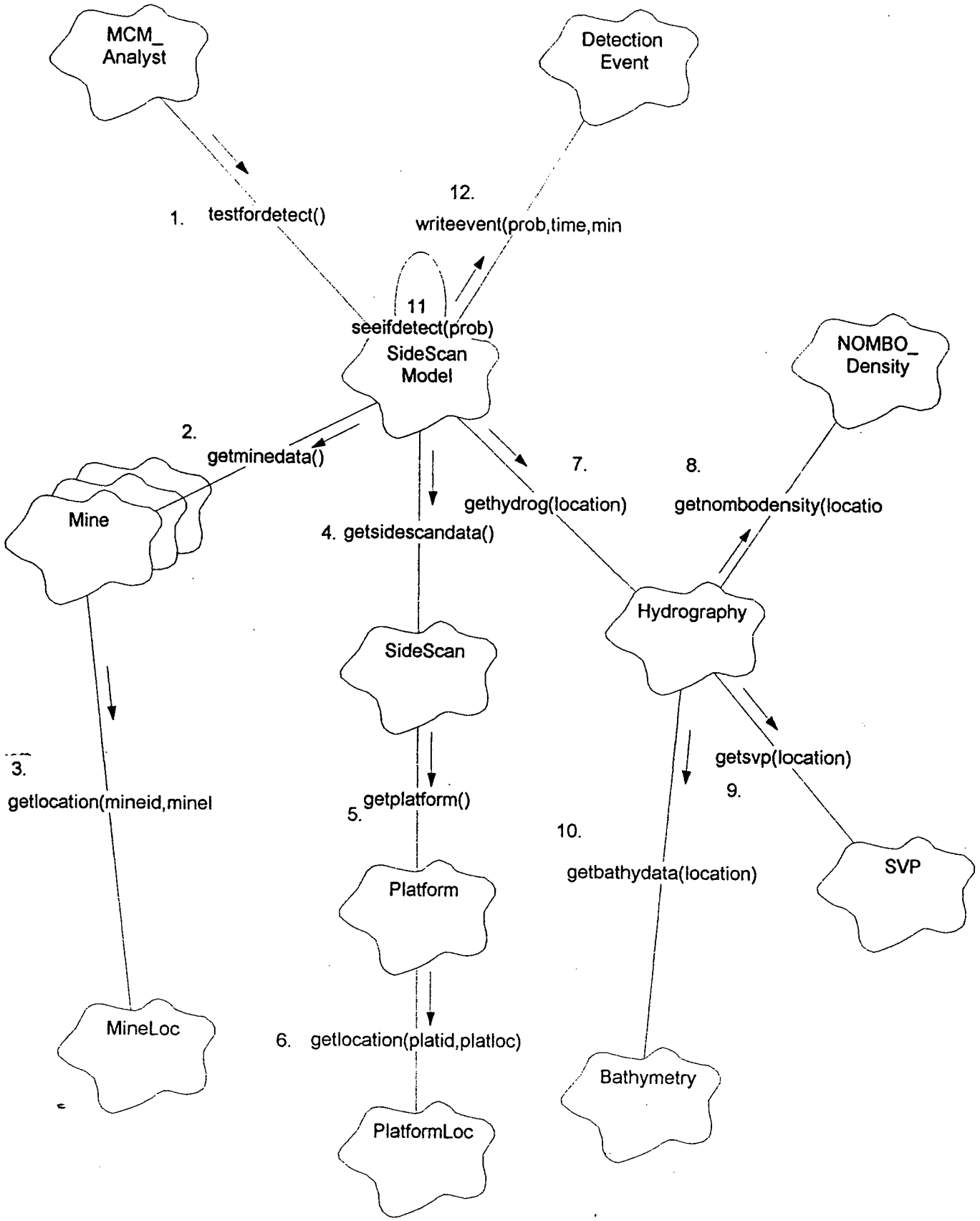
1. The user selects "Environment" from the MCM Analyst's Display.
2. The user selects "Weather" from the Environment Object.
3. The user selects "Add Surface Visibility Data" from the Weather Display.
4. The user enters and edits the surface visibility data in the Surface Visibility Display.
5. The user creates an area associated with the surface visibility data.


MCM_
Analyst
1
editenvdata()
Environment
2
editweather()
Weather
3
addsurfvis()
4
editsurfvisdata()
SurfVis
5
createarea()
SurfVisArea

Use Case: Determine MCM System Effectiveness

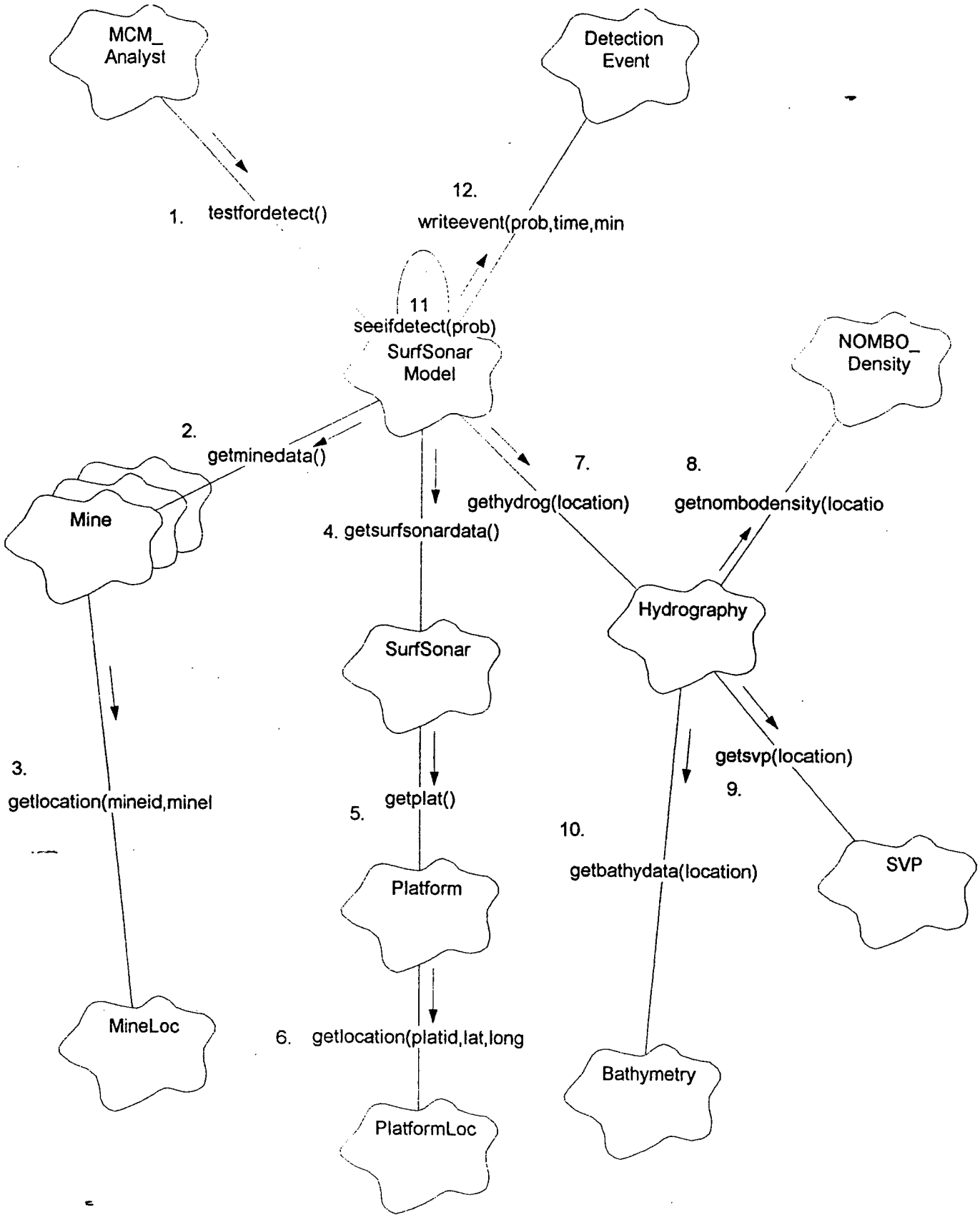
Scenario: Determine if Side Scan Sonar Detects Mine

1. The MCM Analyst calls the side scan detection model to test for a detection.
2. The side scan model calls the mines for mine data.
3. The mine object gets its location.
4. The side scan model calls the side scan sonar object for side scan data.
5. The side scan model calls its platform for platform data
6. The platform gets its location
7. The side scan model calls the hydrography object for hydrographic environmental data.
8. The hydrography object gets NOMBO density data.
9. The hydrography object gets SVP data.
10. The hydrography object gets bathymetric data.
11. The side scan models determines if detection occurs and the probability of detection.
12. The side scan model writes a detection event with the detection data.


MCM_
Analyst
Detection
Event
1. testfordetect()
12.
writeevent(prob,time,min
11
seeifdetect(prob)
SideScan
Model
NOMBO_
Density
2.
getminedata()
7.
gethydrog(location)
8.
getnombodensity(locatio
Mine
4. getsidescandata()
Hydrography
SideScan
3.
getlocation(mineid,minel
5. getplatform()
getsvp(location)
9.
10.
getbathydata(location)
SVP
Platform
MineLoc
6. getlocation(platid,platloc)
Bathymetry
PlatformLoc

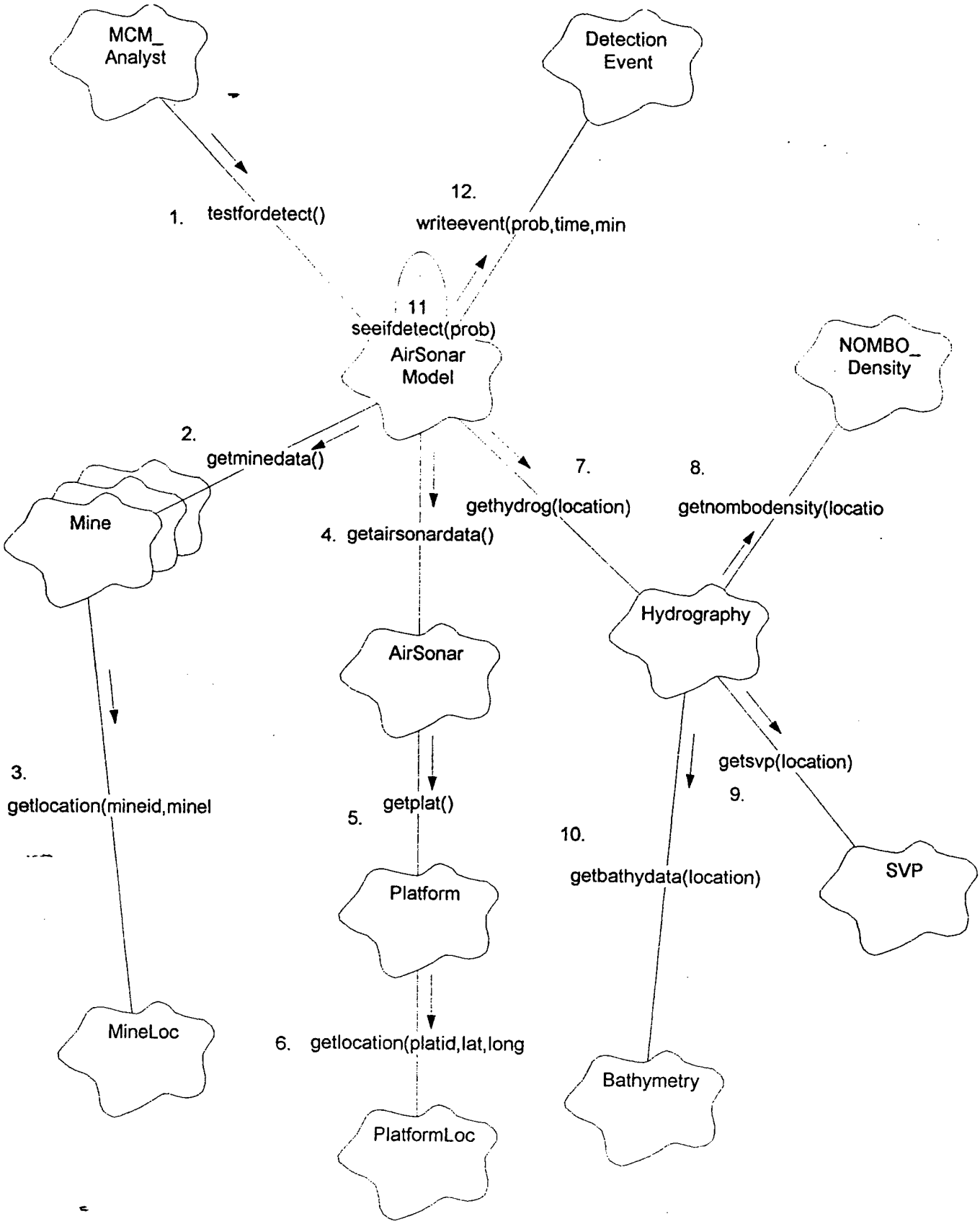
Scenario: Determine if Surface Sonar Detects Mine

1. The MCM Analyst calls the surface sonar detection model to test for a detection.
2. The surface sonar model calls the mines for mine data.
3. The mine object gets its location.
4. The surface sonar model calls the surface sonar object for surface sonar data.
5. The surface sonar model calls its platform for platform data
6. The platform gets its location
7. The surface sonar model calls the hydrography object for hydrographic environmental data.
8. The hydrography object gets NOMBO density data.
9. The hydrography object gets SVP data.
10. The hydrography object gets bathymetric data.
11. The surface sonar model determines if detection occurs and the probability of detection.
12. The surface sonar model writes a detection event with the detection data.


MCM_
Analyst
Detection
Event
1. testfordetect()
12.
writeevent(prob,time,min
11
seeifdetect(prob)
SurfSonar
Model
NOMBO_
Density
2.
getminedata()
7.
gethydrog(location)
8.
getnombodensity(locatio
Mine
4. getsurfsonardata()
Hydrography
SurfSonar
getsvp(location)
9.
3.
getlocation(mineid,minel
5.
getplat()
10.
getbathydata(location)
SVP
Platform
MineLoc
6. getlocation(platid,lat,long
Bathymetry
PlatformLoc

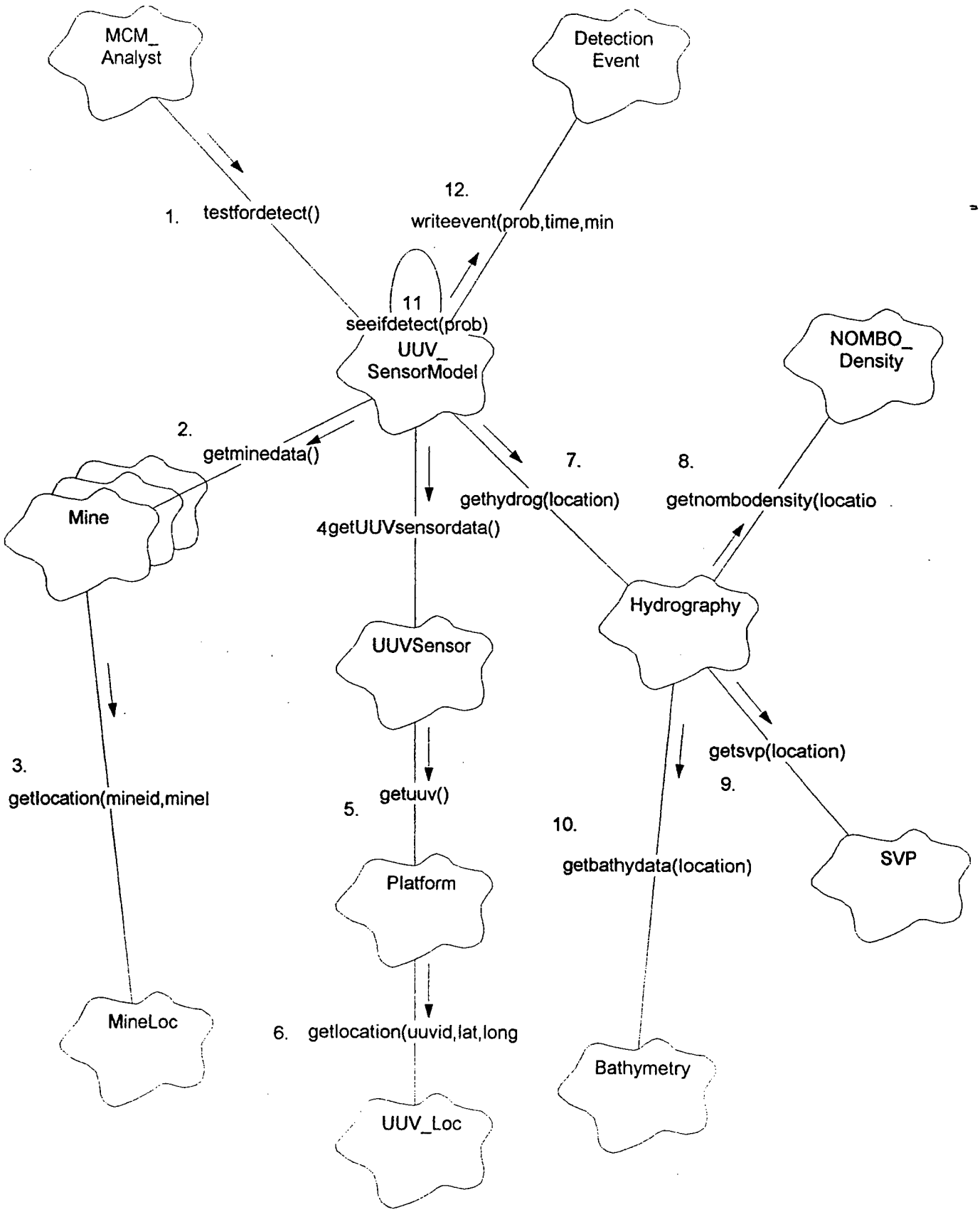
Scenario: Determine if Air Sonar Detects Mine

1. The MCM Analyst calls the air sonar detection model to test for a detection.
2. The air sonar model calls the mines for mine data.
3. The mine object gets its location.
4. The air sonar model calls the air sonar object for air sonar data.
5. The air sonar model calls its platform for platform data
6. The platform gets its location
7. The air sonar model calls the hydrography object for hydrographic environmental data.
8. The hydrography object gets NOMBO density data.
9. The hydrography object gets SVP data.
10. The hydrography object gets bathymetric data.
11. The air sonar model determines if detection occurs and the probability of detection.
12. The air sonar model writes a detection event with the detection data.


MCM_
Analyst
Detection
Event
1. testfordetect()
12.
writeevent(prob,time,min
11
seeifdetect(prob)
AirSonar
Model
NOMBO_
Density
2.
getminedata()
7.
gethydrog(location)
8.
getnombodensity(locatio
Mine
4. getairsonardata()
Hydrography
AirSonar
3.
getlocation(mineid,minel
5.
getplat()
getsvp(location)
9.
10.
getbathydata(location)
SVP
Platform
MineLoc
6. getlocation(platid,lat,long
Bathymetry
PlatformLoc

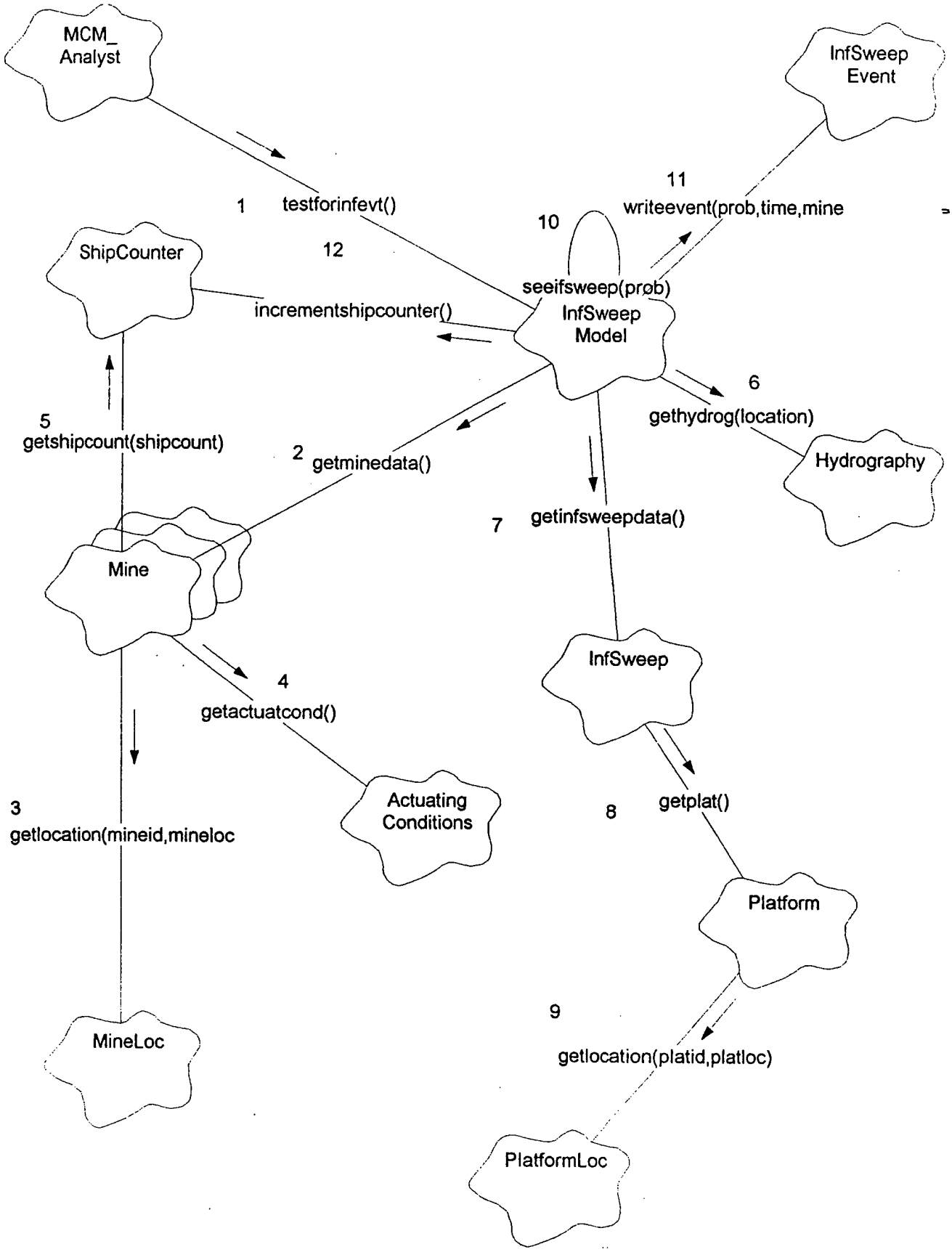
Scemaro: Determine if UUV Detects Mine

1. The MCM Analyst calls the UUV sensor detection model to test for a detection.
2. The UUV sensor model calls the mines for mine data.
3. The mine object gets its location.
4. The UUV sensor model calls the UUV sensor object for UUV sensor data.
5. The UUV sensor model calls its platform for platform data
6. The platform gets its location
7. The UUV sensor model calls the hydrography object for hydrographic environmental data.
8. The hydrography object gets NOMBO density data.
9. The hydrography object gets SVP data.
10. The hydrography object gets bathymetric data.
11. The UUV sensor model determines if detection occurs and the probability of detection.
12. The UUV sensor model writes a detection event with the detection data.


MCM_
Analyst
Detection
Event
1. testfordetect()
12.
writeevent(prob,time,min
11
seeifdetect(prob)
UUV_
SensorModel
NOMBO_
Density
2.
getminedata()
Mine
4getUUVsensordata()
7.
gethydrog(location)
8.
getnombodensity(locatio
Hydrography
UUVSensor
3.
getlocation(mineid,minel
5.
getuuv()
getsvp(location)
9.
10.
getbathydata(location)
SVP
Platform
MineLoc
6. getlocation(uuvid,lat,long
Bathymetry
UUV_Loc

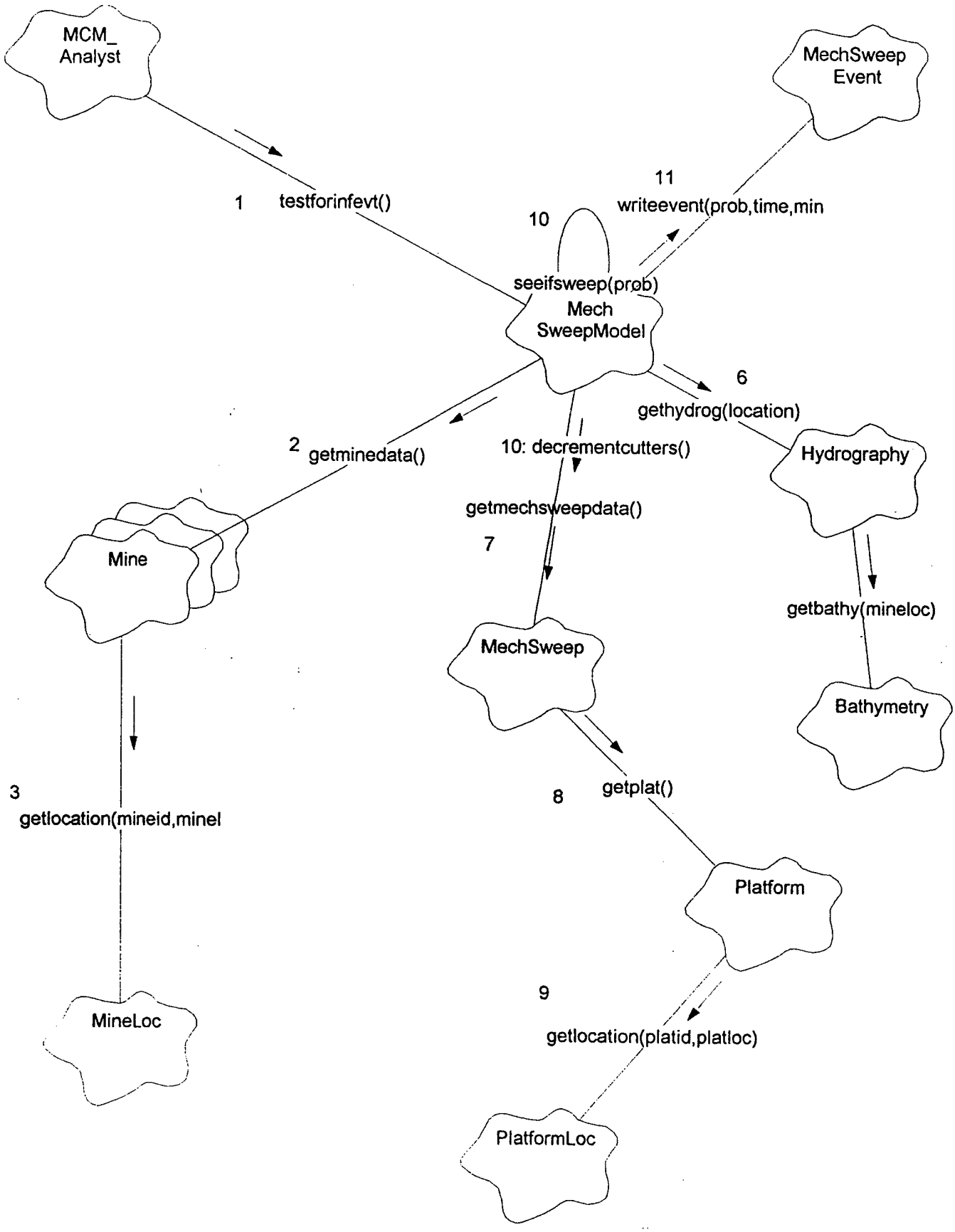
Scenario: Determine if Influence Sweep is Effective

1. The MCM Analyst calls the influence sweep model to test influence sweep effectiveness.
2. The influence sweep model calls the mines for mine data.
3. The mine object gets its location.
4. The mine object gets actuating conditions.
5. The mine object gets the ship count.
6. The influence sweep model calls the hydrography object for hydrographic environmental data.
7. The influence sweep model calls the influence sweep object for influence sweep data.
8. The influence sweep model calls its platform for platform data
9. The platform gets its location
10. The influence sweep models determines if actuation occurs and the probability of actuation.
11. The influence sweep model writes an influence sweep event with the event data.


MCM_
Analyst
InfSweep
Event
1
testforinfevt()
11
writeevent(prob,time,mine
10
12
ShipCounter
seeifsweep(prob)
incrementshipcounter()
InfSweep
Model
6
gethydrog(location)
5
getshipcount(shipcount)
2
getminedata()
Hydrography
7
getinfsweepdata()
Mine
InfSweep
4
getactuatcond()
Actuating
Conditions
3
getlocation(mineid,mineloc
8
getplat()
Platform
9
getlocation(platid,platloc)
MineLoc
PlatformLoc

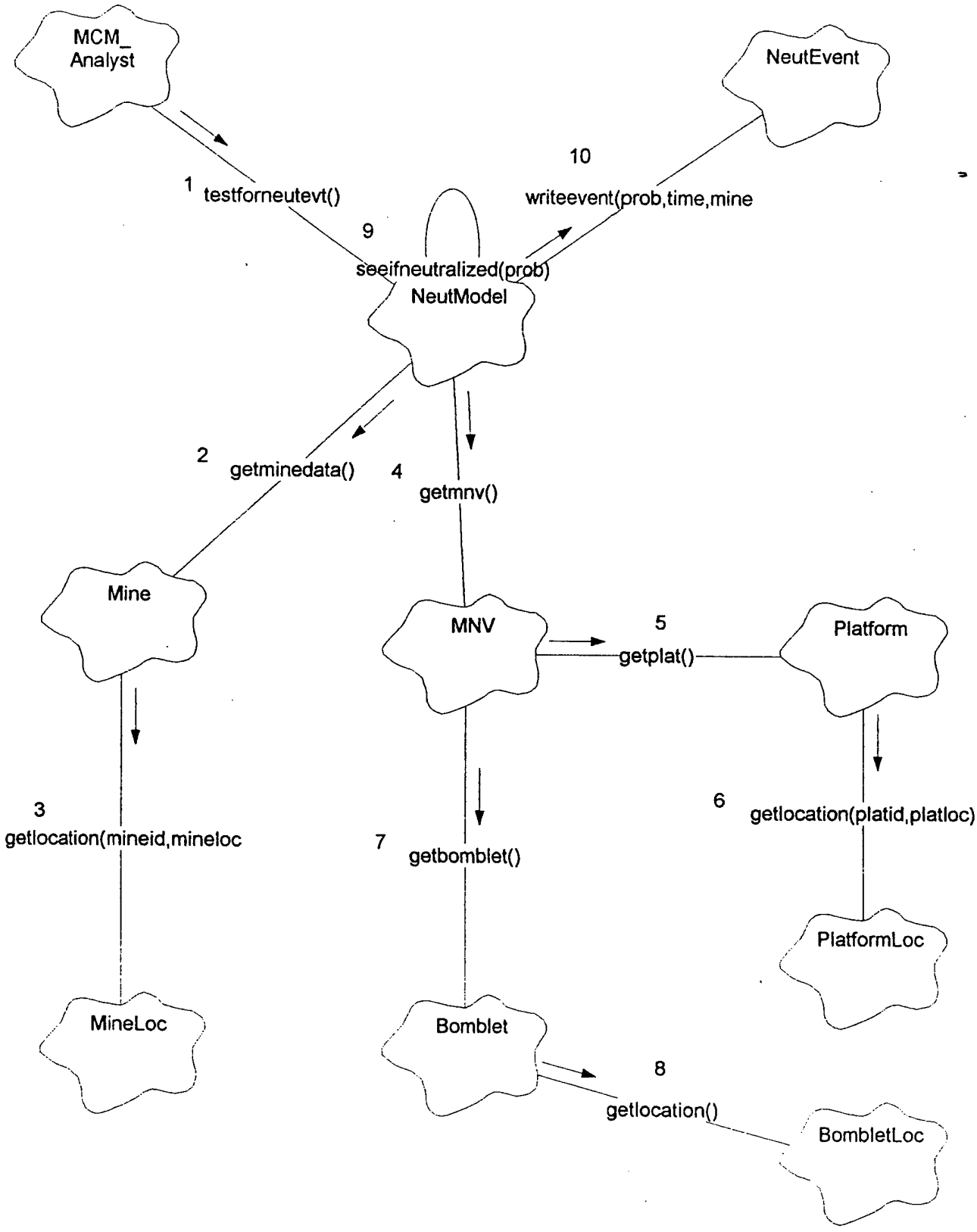
Scenario: Determine if Mechanical Sweep is Effective

1. The MCM Analyst calls the mechanical sweep model to test mechanical sweep effectiveness.
2. The mechanical sweep model calls the mines for mine data.
3. The mine object gets its location.
4. The mine object gets actuating conditions.
5. The mine object gets the ship count.
6. The mechanical sweep model calls the hydrography object for hydrographic environmental data.
7. The mechanical sweep model calls the mechanical sweep object for mechanical sweep data.
8. The mechanical sweep model calls its platform for platform data
9. The platform gets its location
10. The mechanical sweep models determines if sweeping occurs and the probability of sweeping.
11. The mechanical sweep model writes an mechanical sweep event with the event data.


MCM_
Analyst
MechSweep
Event
1
testforinfevt()
11
writeevent(prob,time,min
10
seeifsweep(prob)
Mech
SweepModel
6
gethydrog(location)
2
getminedata()
10: decrementcutters()
Hydrography
getmechsweepdata()
7
Mine
getbathy(mineloc)
MechSweep
Bathymetry
3
getlocation(mineid,minel
8
getplat()
Platform
9
MineLoc
getlocation(platid,platloc)
PlatformLoc

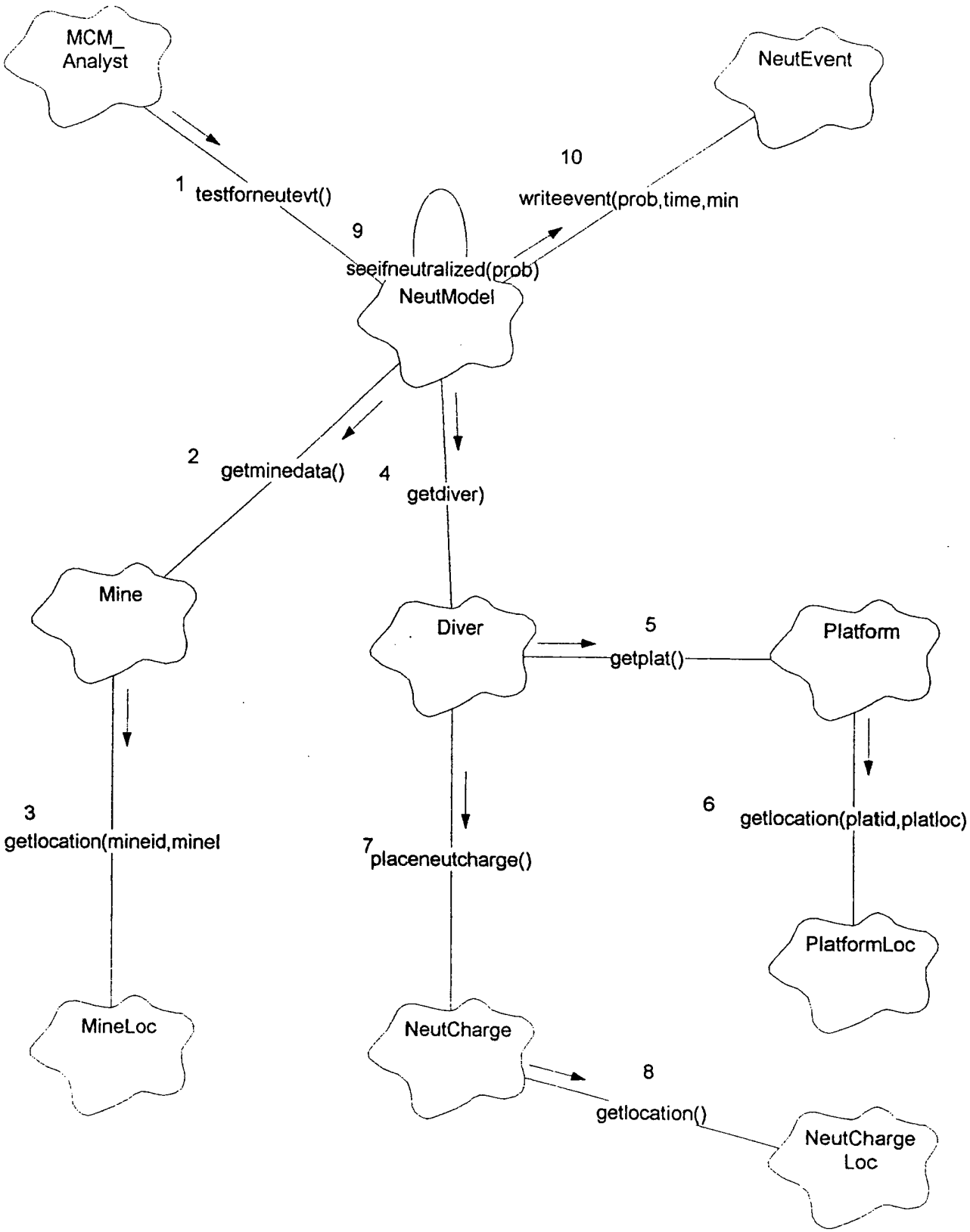
Scenario: Determine if MNV Neutralizes Mine

1. The MCM Analyst calls the MNV neutralization model to test for neutralization.
2. The MNV model calls the mines for mine data.
3. The mine object gets its location.
4. The MNV model calls the MNV object for MNV data.
5. The MNV object calls the platform for platform data.
6. The platform gets its location.
7. The MNV object calls the bomblet for bomblet data.
8. The bomblet gets its location
9. The MNV models determines if neutralization occurs and the probability of neutralization.
10. The MNV model writes a neutralization event with the neutralization data.


MCM_
Analyst
NeutEvent
1 testforneutevt()
10
writeevent(prob,time,mine
9
seeifneutralized(prob)
NeutModel
2
getminedata()
4
getmnv()
Mine
MNV
5
getplat()
Platform
3
getlocation(mineid,mineloc
7
getbomblet()
6
getlocation(platid,platloc)
PlatformLoc
MineLoc
Bomblet
8
getlocation()
BombletLoc

Scenario: Determine if Diver Neutralizes Mine

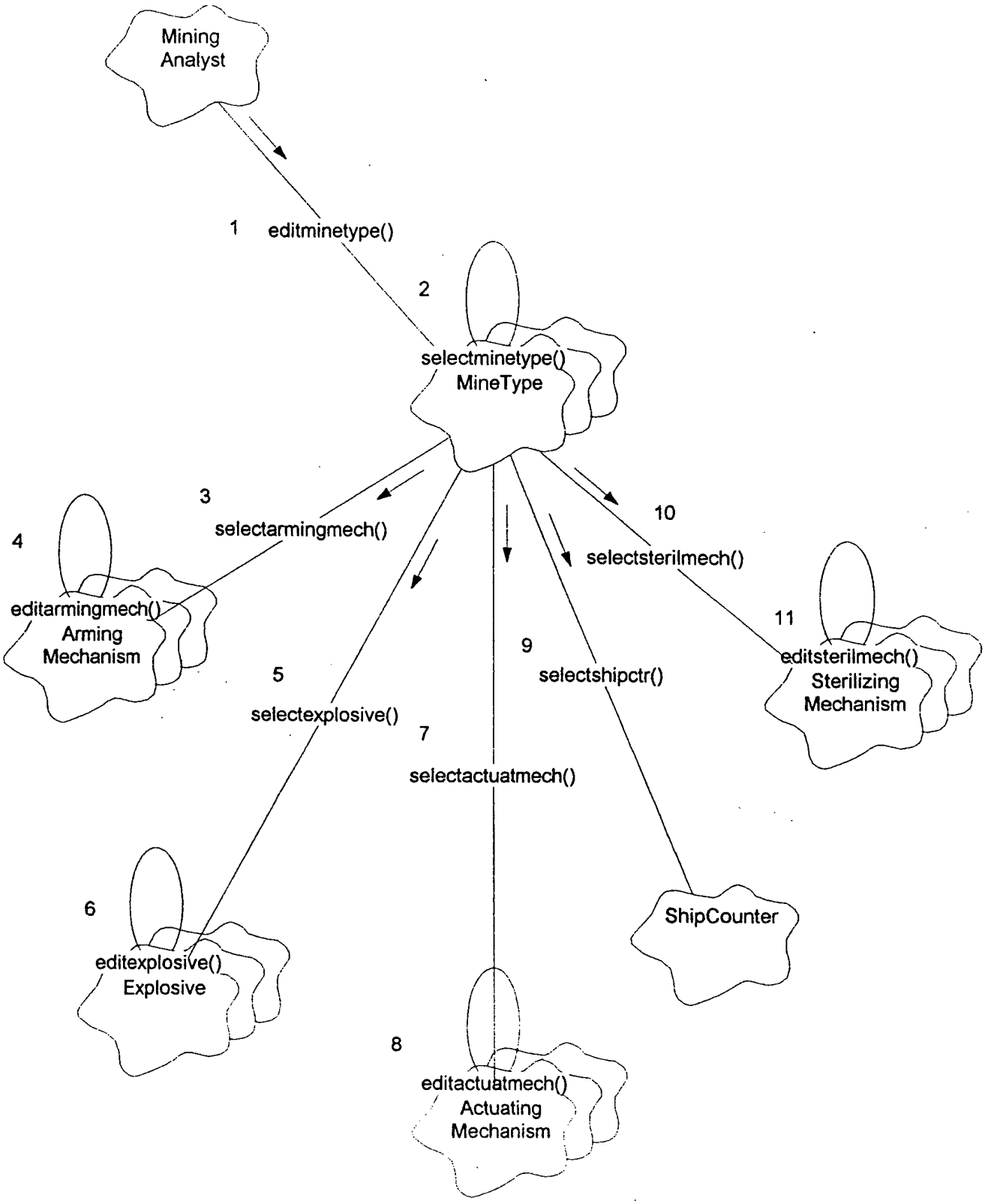
1. The MCM Analyst calls the neutralization model to test for neutralization.
2. The neutralization model calls the mines for mine data.
3. The mine object gets its location.
4. The neutralization model calls the diver object for diver data.
5. The diver object calls the platform for platform data.
6. The platform gets its location.
7. The diver object calls the neutralization charger for charge data.
8. The neutralization charge gets its location
9. The neutralization model determines if neutralization occurs and the probability of neutralization.
10. The neutralization model writes a neutralization event with the neutralization data.


MCM_
Analyst
NeutEvent
1 testforneutevt()
10
writeevent(prob,time,min
9
seeifneutralized(prob)
NeutModel
2
getminedata()
4
getdiver)
Mine
Diver
5
getplat()
Platform
3
getlocation(mineid,minel
7
placeneutcharge()
6
getlocation(platid,platloc)
PlatformLoc
MineLoc
NeutCharge
8
getlocation()
NeutCharge
Loc

Use Case: Specify Mine Characteristics

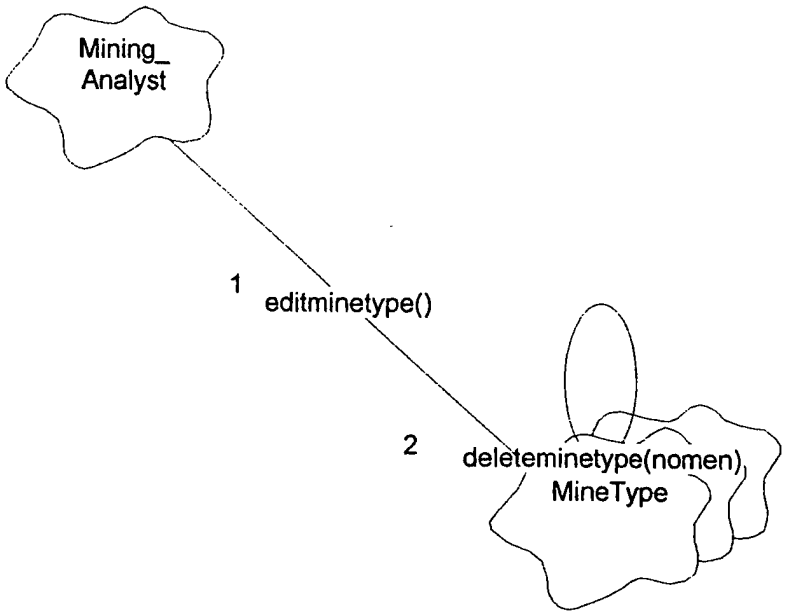
Scenario: Edit a Mine Type

1. The user selects "Edit Mine Type" in the Mining Analyst's Display.
2. The user selects the mine type for editing.
3. The user chooses "Select Arming Mechanism".
4. The user edits the arming mechanism data.
5. The user chooses "Select Explosive".
6. The user edits the explosive data.
7. The user chooses "Select Actuating Mechanism".
8. The user edits the actuating mechanism data.
9. The user chooses "Select Sterilizing Mechanism".
10. The user edits the sterilizing mechanism data.


Mining Analyst
1 editminetype()
2
selectminetype()
MineType
3
selectarmingmech()
4
editarmingmech()
Arming Mechanism
5
selectexplosive()
6
editexplosive()
Explosive
7
selectactuatmech()
8
editactuatmech()
Actuating Mechanism
9
selectshipctr()
ShipCounter
10
selectsterilmech()
11
editsterilmech()
Sterilizing Mechanism

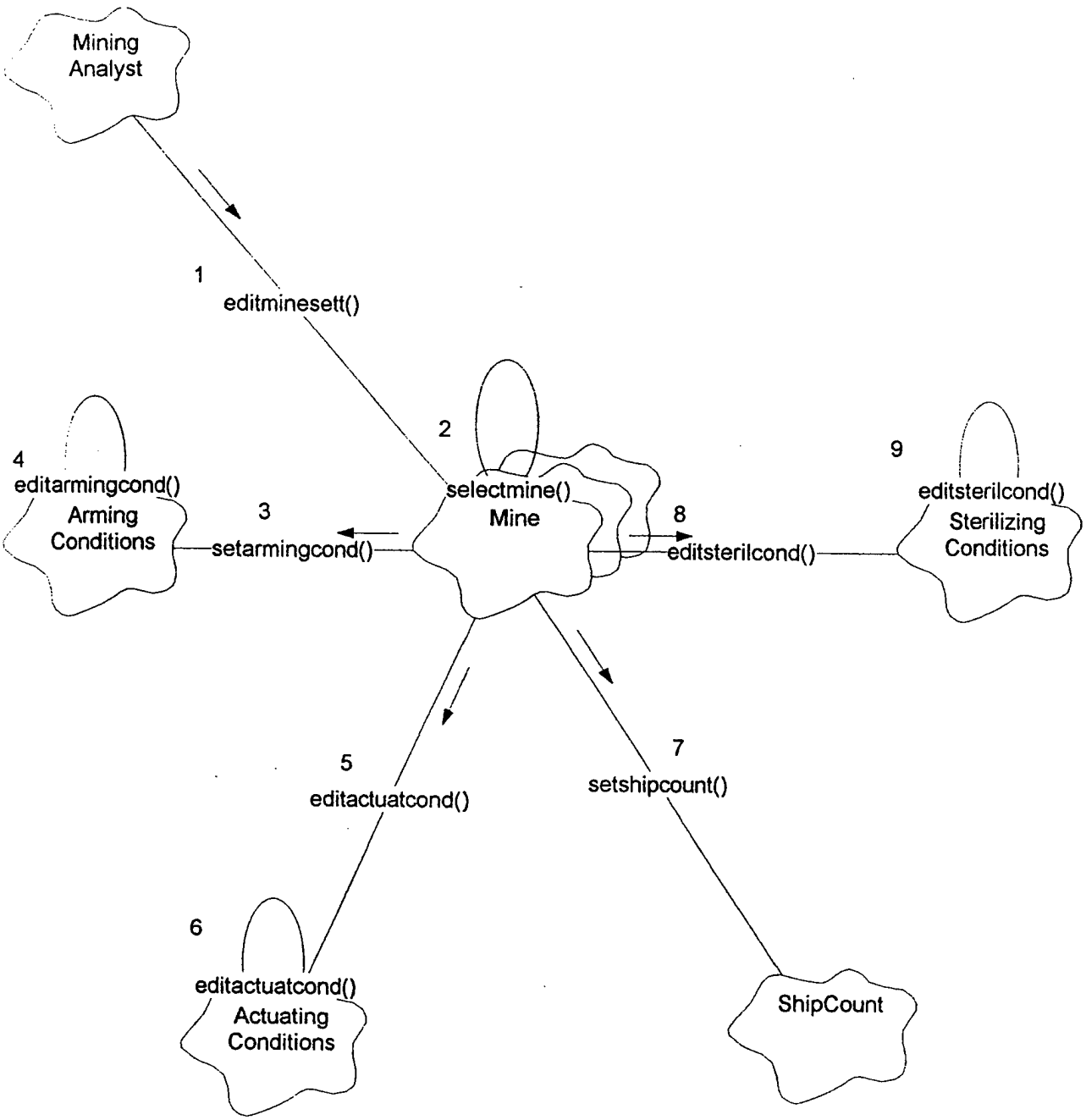
Scenario: Delete a Mine Type

1. The user selects "Delete Mine Type" in the Mining Analyst's Display.
2. The user selects the mine type for deletion and confirms the delection.


Mining_
Analyst
1
editminetype()
2
deleteminetype(nomen)
MineType

Scenario: Specify Mine Settings

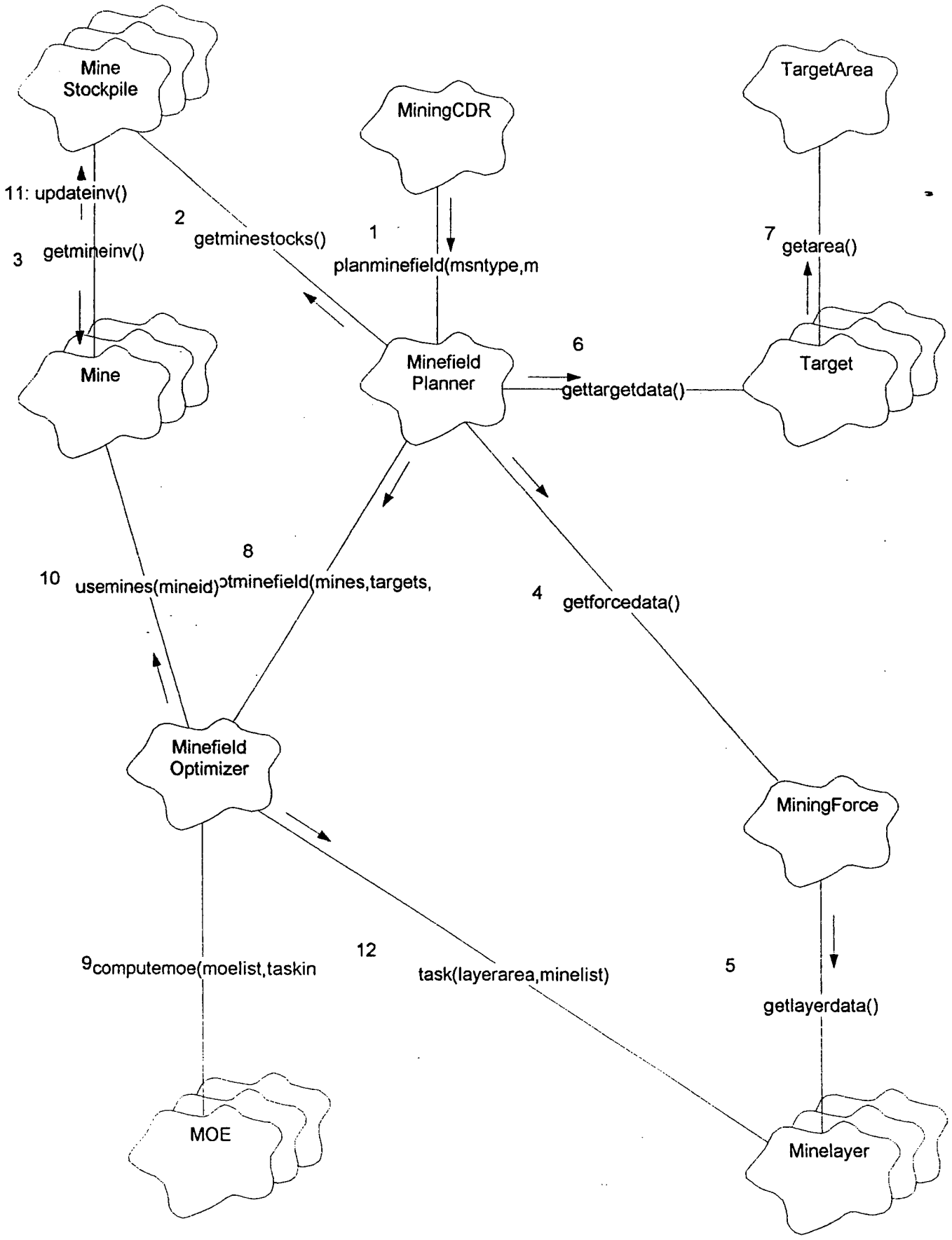
1. The user selects "Edit Mines" in the Mining Analyst's Display.
2. The user selects the mine for editing.
3. The user chooses "Set Arming Conditions".
4. The user edits the arming conditions.
5. The user chooses "Set Actuating Conditions".
6. The user edits the actuating conditions
7. The user sets the ship count.
8. The user chooses "Set Sterilizing Conditions".
9. The user edits the sterilizing conditions


Mining Analyst
1
editminesett()
2
selectmine()
Mine
4
editarmingcond()
Arming Conditions
3
setarmingcond()
8
editsterilcond()
9
editsterilcond()
Sterilizing Conditions
5
editactuatcond()
6
editactuatcond()
Actuating Conditions
7
setshipcount()
ShipCount

Use Case: Plan a Minefield

Scenario: Plan a Minefield

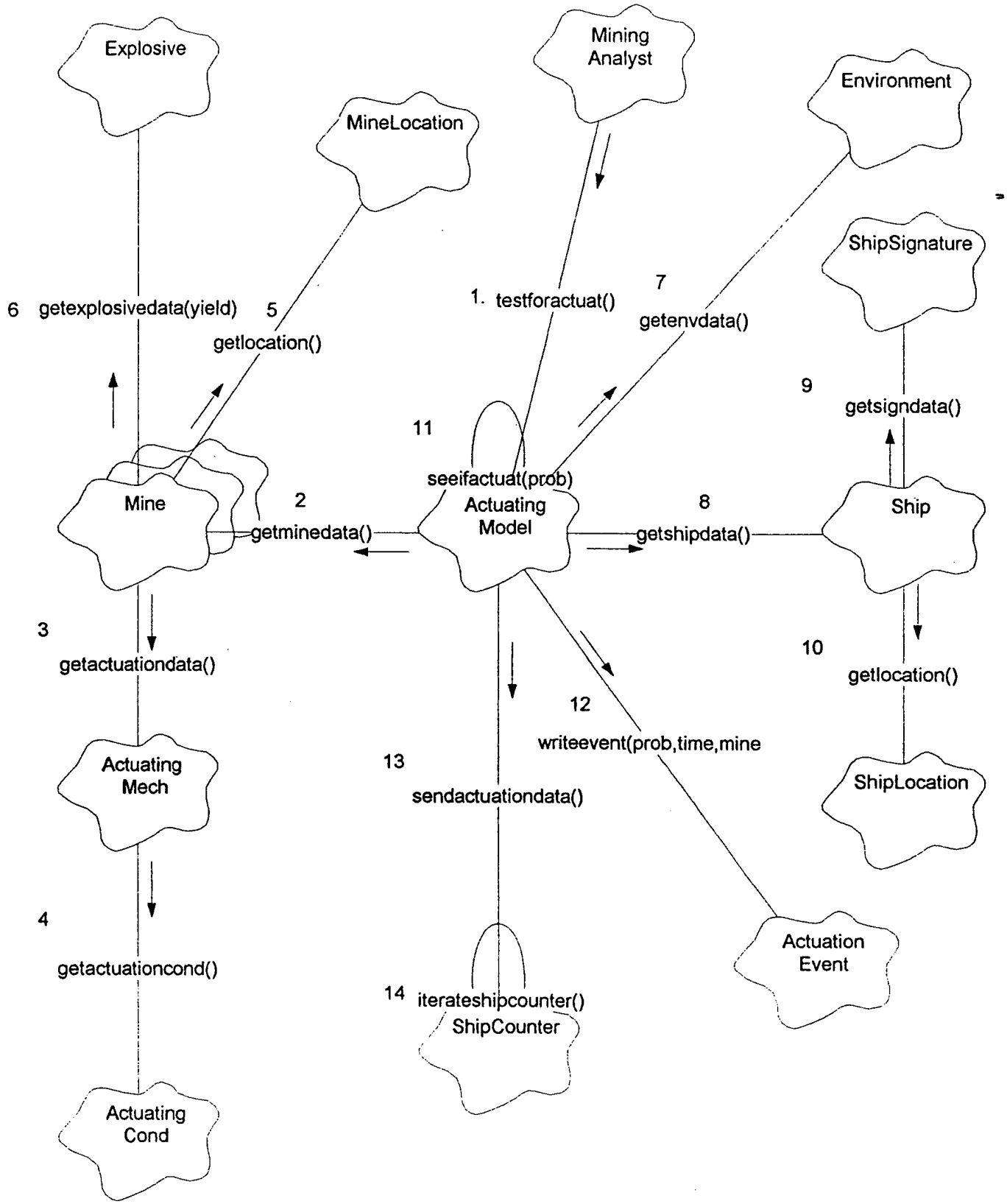
1. The Mining Commander invokes the Minefield Planner object with the mining mission and the MOEs.
2. The Minefield Planner calls the mine stockpiles for mine stock data.
3. Each mine stockpile gets data on mine availability from the mine objects.
4. The Minefield Planner calls the mining force for force data.
5. The mining force gets minelayer data.
6. The Minefield Planner calls for target data.
7. Each target gets data on its area.
8. The Minefield Planner calls the Minefield Optimizer with data on available mines, minelayers, and targets. The Minefield Optimizer creates an optimal plan.
9. The Minefield Optimizer calls the MOE objects for values.
10. The Minefield Optimizer marks the mines for use.
11. The mines update the stockpile inventory as they are marked for use.
12. The Minefield Optimizer sends the tasking (including area, pattern, mine types and numbers) to the minelayers.


Mine Stockpile
MiningCDR
TargetArea
11: updateinv()
2
getminestocks()
1
planminefield(msntype,m
7
getarea()
3
getmineinv()
Mine
Minefield Planner
6
gettargetdata()
Target
8
10
usemines(mineid)
otminefield(mines,targets,
4
getforcedata()
Minefield Optimizer
MiningForce
9computemoe(moelist,taskin
12
task(layerarea,minelist)
5
getlayerdata()
MOE
Minelayer

Use Case: Determine Mine Effectiveness

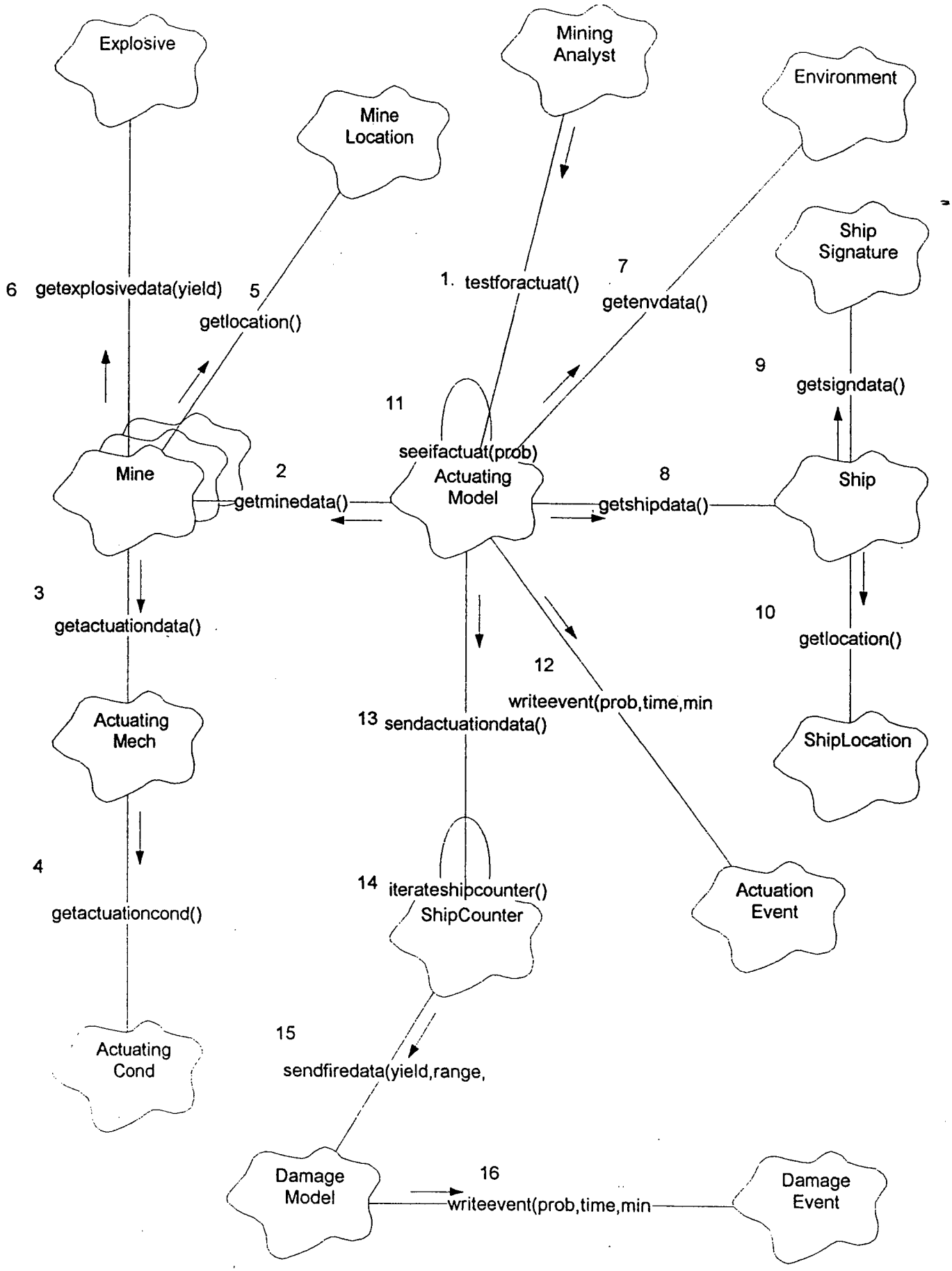
Scenario: Create a Non-Firing Actuation Event.

1. The Mining Analyst invokes the Actuating Model object.
2. The Actuating Model calls the mines for mine data.
3. Each mine calls its actuating mechanism for actuation mechaninsm data.
4. The actuating mechanism gets its actuation conditions.
5. The mine gets its location.
6. The mine gets data on its explosive yield.
7. The actuation model gets environmental data.
8. The actuation model calls the ship for ship data.
9. The ship gets signature data.
10. The ship gets its location
11. The actuation model determines if an actuation occurs and with what probability.
12. The actuation model writes an actuation event.
13. The acutation model sends actuation data to the ship counter.
14. The ship counter increments itself by one.


Explosive
Mining Analyst
Environment
MineLocation
ShipSignature
6 getexplosivedata(yield)
5 getlocation()
1. testforactuat()
7 getenvdata()
9 getsigndata()
11 seeifactuat(prob)
Mine
2 getminedata()
Actuating Model
8 getshipdata()
Ship
3 getactuationdata()
10 getlocation()
12 writeevent(prob,time,mine
Actuating Mech
13 sendactuationdata()
ShipLocation
4 getactuationcond()
Actuation Event
14 iterateshipcounter()
ShipCounter
Actuating Cond

Scenario: Determine If Mine Fires

1. The Mining Analyst invokes the Actuating Model object.
2. The Actuating Model calls the mines for mine data.
3. Each mine calls its actuating mechanism for actuation mechaninsm data.
4. The actuating mechanism gets its actuation conditions.
5. The mine gets its location.
6. The mine gets data on its explosive yield.
7. The actuation model gets environmental data.
8. The actuation model calls the ship for ship data.
9. The ship gets signature data.
10. The ship gets its location
11. The actuation model determines if an actuation occurs and with what probability.
12. The actuation model writes an actuation event.
13. The acutation model sends actuation data to the ship counter.
14. The ship counter increments itself by one.
15. The ship counter sends firing data to the damage model..
16. The damage model writes a damage event with the damage data.

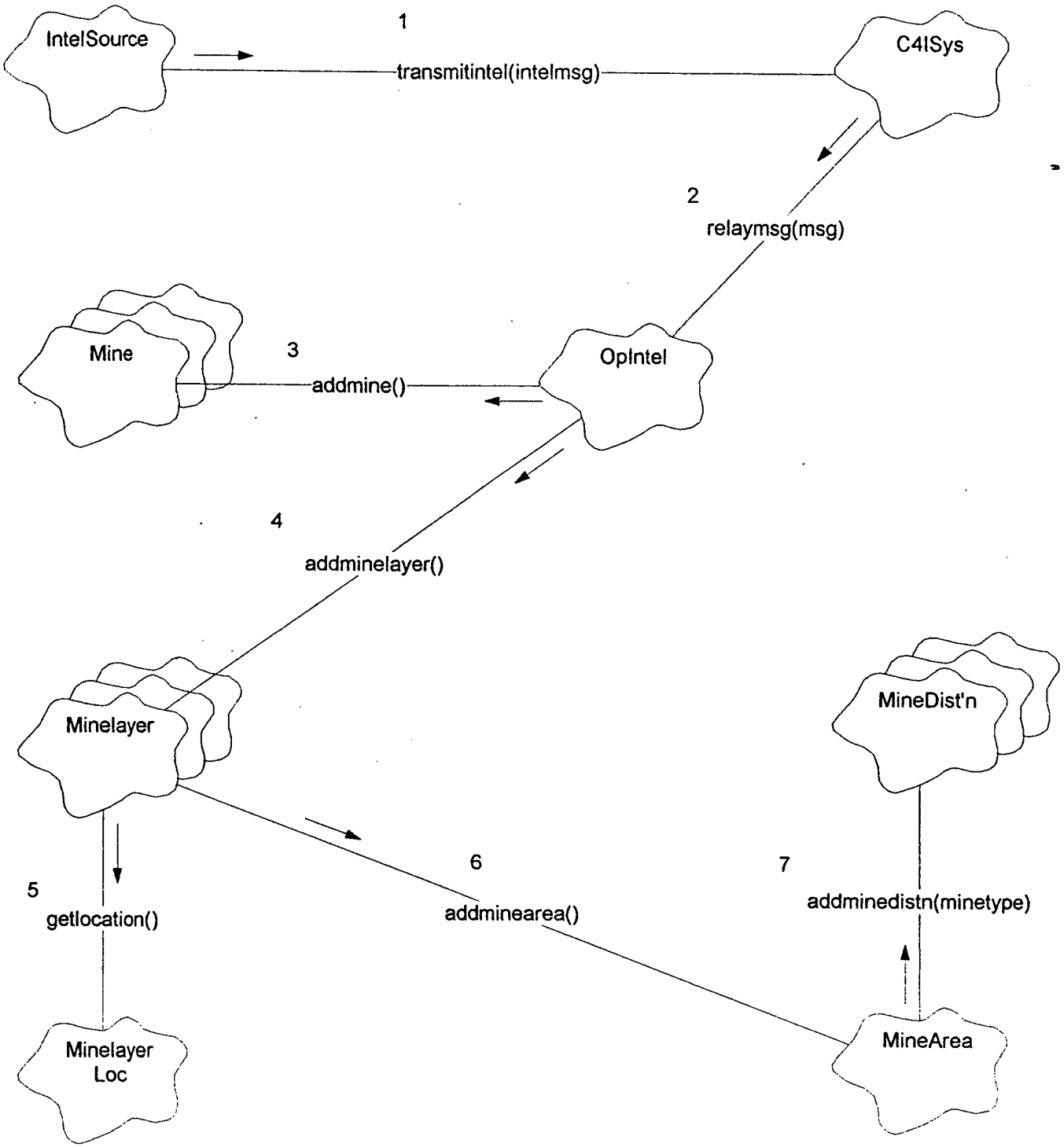


File: c:\rose\semmiw.mdl Tue Nov 14 18:16:32 1995 Object Diagram: <Top Level> / Create a Damage Event

Use Case: Specify Mine Distribution

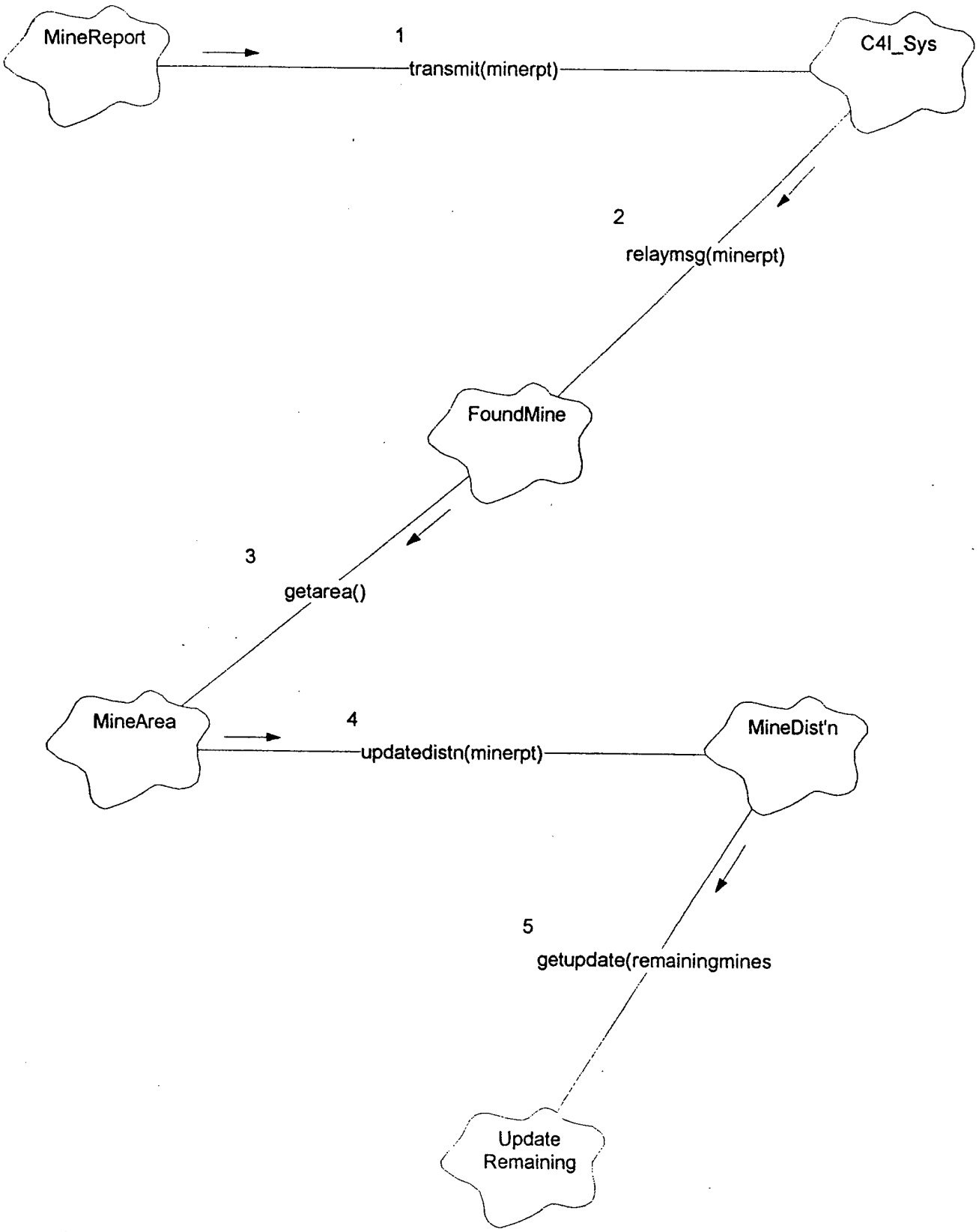
Scenario: Build Mine Distribution Prior

1. The Intelligence Source transmits an intelligence message to the C4I system.
2. The C4I System relays the message to the Operational Intelligence (OpIntel) object.
3. The OpIntel object creates mine objects for any reported mines.
4. The OpIntel object creates minelayer objects for any reported minelayers.
5. The minelayer creates locations for its reported positions.
6. The minelayer object passes the locations to the MineArea object, which builds a mine area.
7. The Mine Area object creates Mine Distribution Objects for each Threat Mine Type reported.


IntelSource
1
transmitintel(intelmsg)
C4ISys
2
relaymsg(msg)
Mine
3
addmine()
OpIntel
4
addminelayer()
Minelayer
MineDist'n
5
getlocation()
6
addminearea()
7
addminedistn(minetype)
Minelayer Loc
MineArea

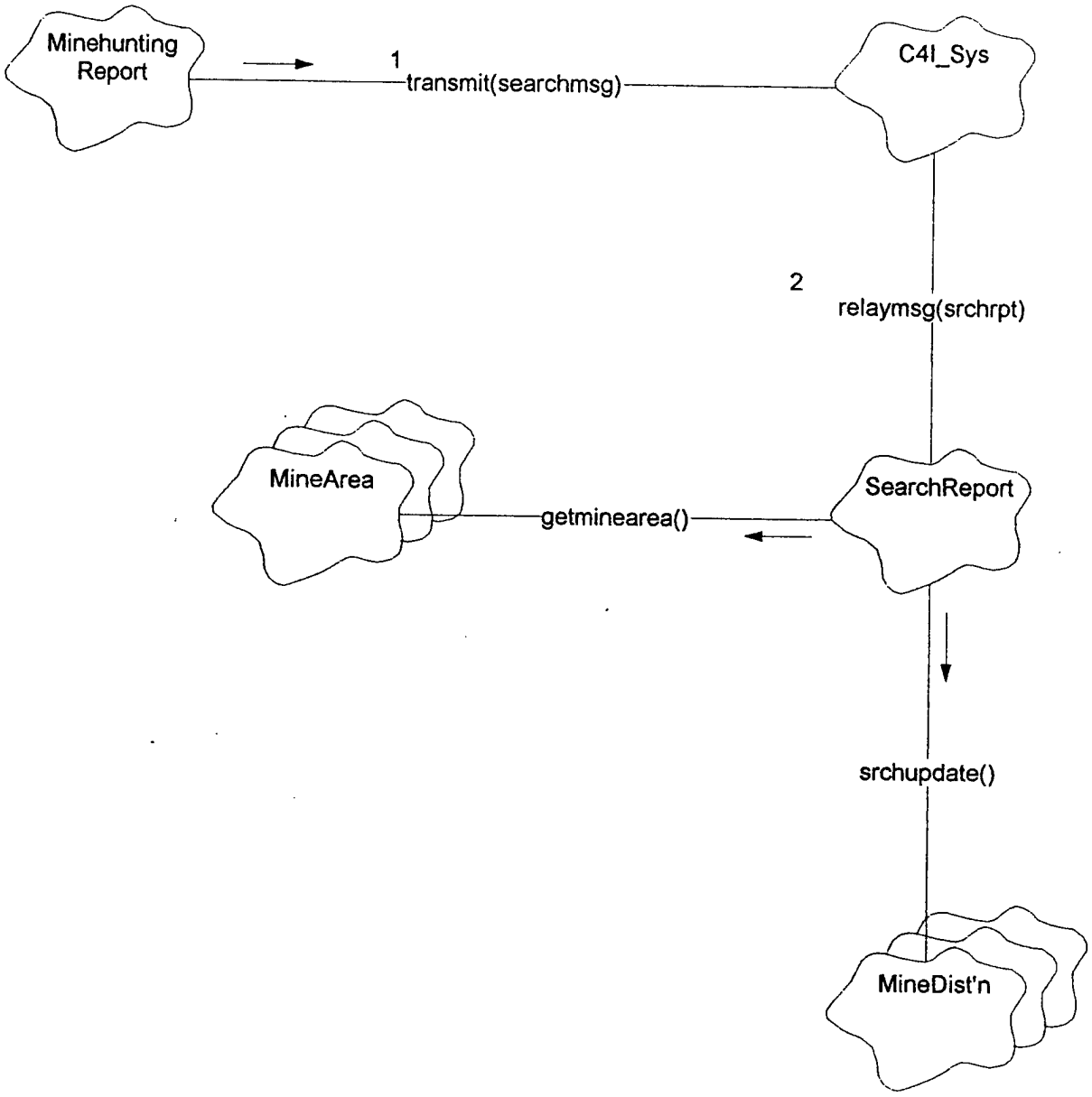
Scenario: Update Mine Distribution Based on Found Mine

1. The Mine Report object transmits a mine report to the C4I system.
2. The C4I system relays the Mine Report to the Found Mine Object.
3. The Found Mine object gets the mine area in which the found mine is located.
4. The Mine Area updates the minedistributions corresponding to the types that match the found mine.
5. The Mine Distribution calls the Update Remaining object to update the remaining number of mines.


MineReport
1
transmit(minerpt)
C4I_Sys
2
relaymsg(minerpt)
FoundMine
3
getarea()
MineArea
4
updatedistn(minerpt)
MineDist'n
5
getupdate(remainingmines
Update
Remaining

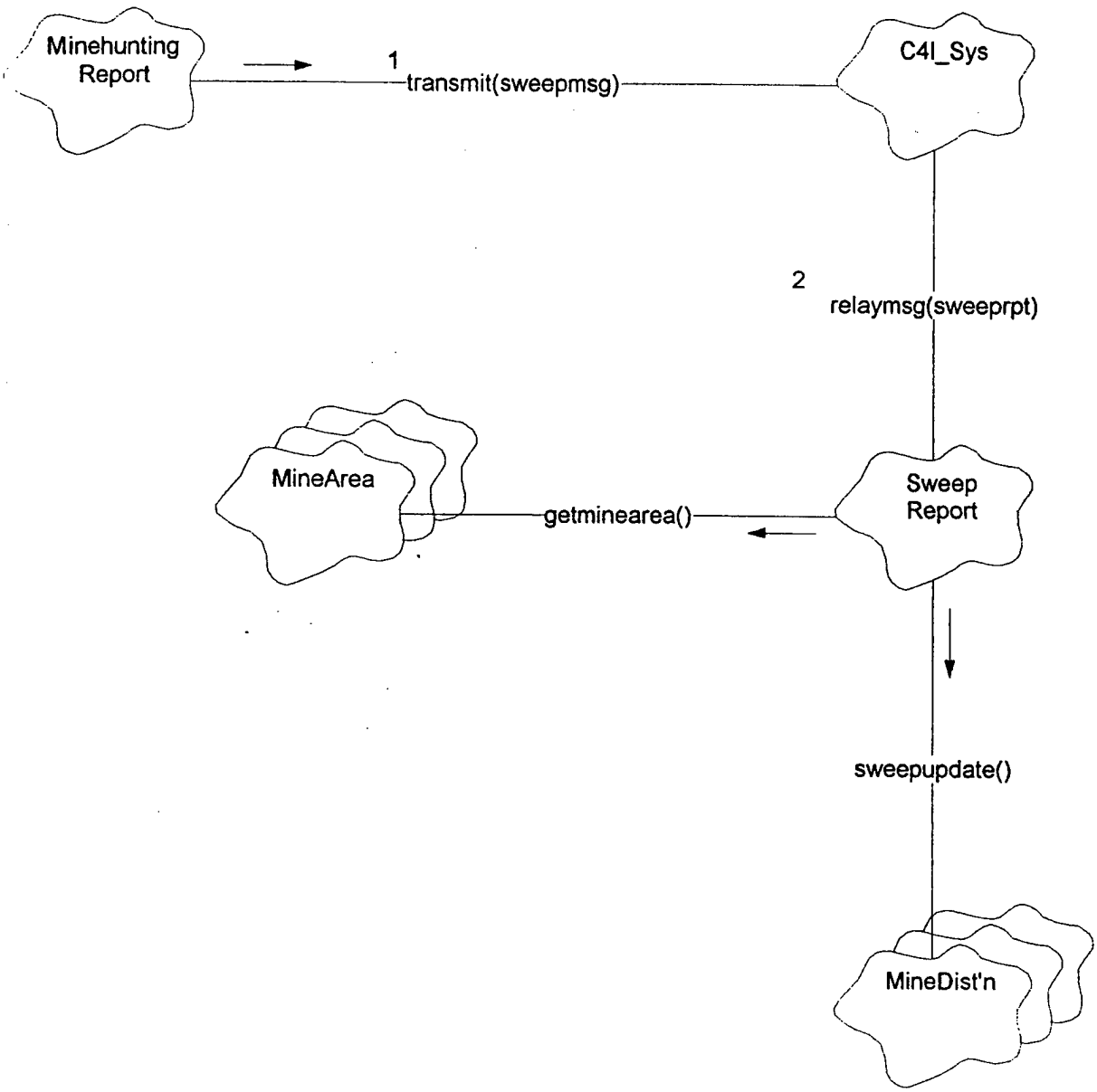
Scenario: Update Mine Distribution Based on Search

1. The Minehunting Report object transmits the search message to the C4I System.
2. The C4I system relays the search message to the Search Effort object.
3. The Search Effort object gets the mine area corresponding to the search.
4. The mine area calls the mine distributions for updates based on the search.
5. The mine distributions update themselves based on the search data.


Minehunting Report
1
transmit(searchmsg)
C4I_Sys
2
relaymsg(srchrpt)
MineArea
getminearea()
SearchReport
srchupdate()
MineDist'n

Scenario: Update Mine Distribution Based on Sweeping

1. The Minehunting Report object transmits the sweep message to the C4I System.
2. The C4I system relays the sweep message to the Sweep Effort object.
3. The Sweep Effort object gets the mine area corresponding to the sweep effort.
4. The mine area calls the mine distributions for updates based on the sweep effort.
5. The mine distributions update themselves based on the sweep data.


Minehunting Report
1
transmit(sweepmsg)
C4I_Sys
2
relaymsg(sweeprpt)
MineArea
getminearea()
Sweep Report
sweepupdate()
MineDist'n

Chapter 4: Estimated Number of Mines Remaining

The purpose of this chapter is to define a class of mathematical models that will form the basis for describing prior and posterior mine distributions in the SEMMIW. These models are key components of SEMMIW because all prediction models, threat models, and other measures of effectiveness draw their basic data from the probability distributions of mines in a minefield or from calculated changes in those distributions.

Having explicit models that permit us to define prior and posterior distributions of mines in a minefield is the key capability that sets SEMMIW apart from other minehunting and clearance decision aids. Current systems look at a minefield as a predefined operating area in which the presence of mines is suspected, without mathematically defining those suspicions. Mine hunting and mine clearance decision aids then calculate *conditional* probabilities of detection, that is, given that a mine is located in a certain area, they compute the probability that a particular search or sequence of searches would detect that mine. Computing conditional probabilities avoids the problem of having to specify *a priori* the probability that a mine is there in the first place.

The problem with using conditional probabilities alone is that one can never answer the questions, "How many mines are left in this area?" or "What is the probability that a ship will be damaged or sunk if it goes through that area?" In order to compute a distribution for mines remaining after search or clearance, one must have a distribution before the operation.

Once a prior can be expressed, then resources can be allocated in ways that optimize a specific objective, say, to minimize the expected number of mines in a swept channel. If the prior probability distribution is different in different parts of that channel, then an optimal use of scarce search time might call for multiple sweeps in some parts, single sweeps in others, and perhaps no sweep in some parts. The gain in effectiveness of the optimal plan (the one that considers the prior distribution) compared to a standard plan using the same resources becomes a convenient measure of contribution of the process of gathering and analyzing the data to create the prior distribution. As defined in [5], this process includes elements of *mapping, survey, and intelligence operations* and *surveillance operations*.

A probability distribution for mines in a minefield is an integer problem, i.e., in the limit, within some small subregion within the minefield, we can eventually say that there either is or is not a mine. In larger regions, we can define the probability of a specific number of mines being in that region. A general distribution of discrete events is described by a series

of probability values for each of the possible events. If we have a collection of events, then we describe the population by collecting the events and their probabilities together in ways that answer our questions. In general, we describe the collection of all possible events as a population of size N and each possible outcome as a sample. If the population and samples are appropriately defined so that all outcomes are equally likely, then each sample has probability 1/N. If we define an event as an ordered drawing of r objects at random from a population of n objects, a_1, a_2, ...a_n, then the number of those events is the number of different ways those selections can be made—and a selection of the same objects in a different order is a different event. We can write down the number of such orderings and the answer is different depending on whether repetitions are permitted; that is, if a_i is selected, can it be selected again in the same sample? The two cases are called sampling *with replacement* and *without replacement.* When sampling with replacement, each of the r draws can be any of the n objects and, by independent probability arguments, there are n^r orderings. If we sample without replacement, the first draw can be one of n objects, the second one of (n-1) objects, and so forth, giving the product n(n-1)...(n-r+1) different orderings.

In many problems, including the mine distribution problem, the order of objects is unimportant (or unobservable). But the mathematical descriptions of those unordered samples must be based on the probability definitions derived from the ordered cases [6].

This leads us to classes of distributions that describe discrete event samples without orderings, and three such classes in particular, the *Binomial* distribution, the *Poisson approximation* to the Binomial, and the Waiting Time, or *Negative Binomial* distribution. Each of these has its place in describing prior and posterior mine distributions, and they were first explored by Richardson and Corwin during the Suez Canal clearance operations in 1975 [5,6].

We will show that, under certain conditions, each of these three distributions is useful and sufficient to support a description of mine location distributions. Both the Binomial and Negative Binomial are necessary for quite different situations, but the Poisson Approximation can be used as a computationally efficient substitute for many cases in between.

The basic model for minefield in SEMMIW consists of objects called Mine Danger Areas, Mine Areas, and Mine Distributions. A Mine Danger Area is defined as the overall region in which a MCM commander must deploy his forces to search and destroy mines for the protection of the task force during an operation. A Mine Area is defined as a specific

area (large or small) within which a uniform Mine Distribution exists. A Mine Distribution is specified by a set of parameters which defines, for each mine type or class, the probabilities Pr{N=k} for k = 0,1,2,...K where the random variable N specifies the number of mines present. At first there may be only one relatively large Mine Area but, as the MCM assets are used to search and clear subsets of that area, then it may be useful to decompose into smaller areas, each with a different distribution based on the results of operations in those areas. For a beginning discussion, we will assume that the mine area and the search area are identical and that search effort is applied uniformly over the area. Later we will show how to handle a prior distribution when the area must be subdivided to represent different levels of search or clearance effectiveness.

We introduce the concept of *knowledge quality* to describe the information available for defining a prior distribution of mines. We define three levels of knowledge quality by comparing the prior information provided by mapping, survey, intelligence, and surveillance operations with the mine distribution information provided by a small mine reconnaissance effort.

We describe prior information as *high quality* when finding some mines with a small amount of search effort provides less information about the total mine distribution than our prior intelligence. For example, suppose we have a mine area that was previously surveyed and found to be clear of mines. Later we observe a hostile minelayer with a known inventory of K mines operating within the mine area and thereby obtain a very good notion that the area contains mines and a reasonable estimate, K, of the maximum number. If we now conduct reconnaissance and detect k mines, our estimate of the expected number of undetected mines remaining goes down. The more effective the search, the lower will be the estimate of remaining mines. Thus, we define *high* quality prior information as a situation in which, for any search of a *given* effectiveness, the more mines we find the *lower* is our estimate of the number of mines remaining undetected. This is because finding mines is very close to just fixing the positions of mines about which we had good prior knowledge.

In contrast, we view prior information as *low quality* when any reasonable reconnaissance effort will dominate our total information. For example, suppose we pick any littoral area in the world without regard to prior knowledge about the presence of mines. We might estimate the distribution of mines based on estimates derived from the world population of deployed mines and the total littoral ocean area. If we now search with a low probability of detection and find a few mines, then our estimate of the total number of mines will go up to account not only for the ones we found but also for the expected number of

undetected mines. As before, the more effective the search, the lower will be the *a posteriori* estimate of remaining mines. But we specifically define *low* quality prior information as a situation in which, for any search of a *given* effectiveness, the more mines we find the *higher* is our estimate of the number of mines remaining undetected. This is because, by finding mines, we are adding observations to a distribution estimate whose mean was low and whose variance was large.

For situations in between, we call prior information *medium quality* when a small amount of search effort provides about the same information as our prior intelligence. In this case, as in the others, the more effective the search, the lower will be the estimate of remaining mines. But, for any search of a *given* effectiveness, the estimate of the number of mines remaining undetected is the same, *no matter how many mines we find.* This situation corresponds to the situation in which the probability that an object exists in one subregion is independent of other, non-overlapping subregions. It is this independence that prohibits the use of detections in one part of a mine area to modify the probability distribution of mines in another part.

We now show that the three distributions, Binomial, Poisson, and Negative Binomial correspond respectively to the high quality, medium quality, and low quality prior information. It is essential that we use these different models because, without them, we are unable properly to use the information imparted by the number of mines found by the reconnaissance effort when it is appropriate, i.e., in the low- and high-quality prior information situations. The mathematical development was first reported in Richardson et. al. [6] and to Belkin et. al. in Appendix A to [5]. Dr. Belkin assisted in the current derivations.

4.1. The Expected Number of Remaining Mines

We suggest that one appropriate measure of MCM effectiveness is the expected number of remaining mines in an defined subregion. The uncertainty in that estimate may also be important, in order to define confidence intervals and thus define the probability that a vessel may encounter an undetected mine, but we will extend the discussion to those issues later. For now, let us concentrate on the expected value. Suppose we know for certain that the number of mines is known to be N=K. Define the expected number of remaining mines as $\bar{R}$. Before any search or clearance, $\bar{R}$=K. If we search and find k mines, then the expected number of remaining mines is $\bar{R}$ = K-k. Usually, however, one does not know the number of

targets in the area. In this case, it is necessary to specify a probability distribution for the number of mines. Then, on the basis of the number of mines found and the probability of detection in the region, a posterior distribution is found for the number of mines remaining undetected. The mean of this posterior distribution is $\bar{R}$, the expected number of mines remaining undetected.

In order to define $\bar{R}$ mathematically, let

N = number of mines in the section (N is a random variable),

$f(n) = Pr\{N = n\}$ for $n = 0, 1, 2, \ldots$.

Assume that the location of the mines are independent and identically distributed in the area and that all mines have the same detectability characteristics. Suppose that a search is performed in the area which results in

p = conditional probability that a given mine will be detected,

k = number of targets actually detected.

Since all of the mines have independent and identically distributed locations and the same detection characteristics, they all have the same probability of detection p and the detection of one mine is independent of the detection of any other mine.

Let

R = number of mines remaining undetected

$g(r,p,k) = Pr\{ R = r \mid \text{number of mines detected} = k \}$

From equation (6.6.8) of reference [7], we find that

$$g(r,p,k) = \frac{f(r+k)\, b(k; r+k, 1-p)}{\sum_i f(i+k)\, b(k; i+k, 1-p)} \quad \text{for } k = 1, 2, \ldots, \qquad (1)$$

where the binomial distribution b(k; r+k, 1-p) is the probability of failure to detect exactly k mines out of r+k, given probability of detection p.

From g, one can calculate

$$\bar{R}(p,k) = \sum_{r=0}^{\infty} r\, g(r,p,k) \quad . \tag{2}$$

Calculating $\bar{R}$ in a straightforward manner using equations (1) and (2) can be complicated and tedious. However, in the classes of distributions we propose for SEMMIW, the calculation of $\bar{R}$ is greatly simplified when search is uniform throughout the mine area, or when the mine area is divided into two subregions. In this section we show how to calculate $\bar{R}$ for the three classes of distributions on N, the number of mines: the Binomial, Poisson, and Negative Binomial distributions.

1. **Binomial.** We use the Binomial distribution for the high quality prior information situation. Let $0 < \alpha < 1$ be fixed and K be a positive integer, which values can be derived directly from analysts' estimates of the mean and maximum number of mines. Then,

$$\Pr\{N = n\} = \binom{K}{n} \alpha^n (1-\alpha)^{K-n} \quad \text{for } n = 1, 2, \ldots, K$$

and

$$E[N] = \alpha K \quad .$$

For $0 \le p \le 1$ and $k = 0, 1, \ldots, K$,

$$g(r,p,k) = \frac{\binom{K}{k+r}\alpha^{k+r}(1-\alpha)^{K-k-r}\binom{k+r}{r}p^k q^r}{\sum_{i=0}^{K-k}\binom{K}{k+i}\alpha^{k+i}(1-\alpha)^{K-k-i}\binom{k+i}{i}p^k q^i} \quad , \; k + r \le K$$

$$= \frac{\frac{K!}{(K-k-r)!\, r!k!}(\alpha p)^k(\alpha q)^r(1-\alpha)^{K-k-r}}{(\alpha p)^k(1-\alpha)^{K-k}\binom{K}{k}\sum_{r=0}^{K-k}\binom{K-k}{r}\left(\frac{\alpha q}{1-\alpha}\right)^r}$$

$$= \binom{K-k}{r}\frac{(\alpha q)^r(1-\alpha)^{-r}}{\left[1+\frac{\alpha q}{1-\alpha}\right]^{K-k}}$$

$$= \binom{K-k}{r}\frac{(\alpha q)^r(1-\alpha)^{K-k-r}}{(1-\alpha p)^{K-k}} \quad \text{for } 0 \le r \le K-k \tag{3}$$

$$\bar{R}(p,k) = \frac{(1-p)\alpha}{(1-\alpha p)} \quad (K-k) \quad .$$

Note that g and $\bar{R}$ depend on both p and k and, for constant p, $\bar{R}$ decreases with increasing k, giving us the desired behavior.

2. **Poisson.** We use the Poisson distribution for the medium quality prior information situation, under the following argument. As our uncertainty increases and the mean αK stays constant, using the Binomial model, the maximum number of mines K must increase and α must decrease. When K gets very large and α very small, the relationship between k and $\bar{R}$ vanishes and it is convenient to use an approximation to the Binomial distribution which is due to Poisson and which is derived in [8]. In this case,

$$\Pr\{N = n\} = \frac{\delta^n e^{-\delta}}{n!} \quad \text{for } n = 0, 1, 2, ...,$$

where $\delta > 0$ is the parameter (equivalent to αK in the Binomial distribution) which specifies the distribution. In fact,

$$E[N] = \delta \quad .$$

Since there is only one parameter for this distribution, it requires only an estimate of the mean number δ of mines in the mine area.

For $0 \le p \le 1$ and $k = 0, 1, 2, ...,$

$$g(r,p,k) = \frac{[(1-p)\delta]^r e^{-(1-p)\delta}}{r!} \quad \text{for } r = 1, 2, ..., \qquad (4)$$

$$\bar{R}(p,k) = (1-p)\delta \quad .$$

Note that, as expected, g and $\bar{R}$ do not depend on k.

3. **Negative Binomial.** We use the Negative Binomial distribution for the case of low quality prior information. We expect the mean number of mines to be very small but for there to be no reasonable upper bound on the number of mines.

If there is uncertainty about the mean number of mines in a region, one can proceed by assuming that if the mean were known to be δ, then the distribution of the number of mines is Poisson with mean δ. However, the mean δ is uncertain and this uncertainty is specified by means of a gamma distribution.

The gamma density

$$f(x;\nu,\beta) = \frac{1}{\Gamma(\nu)} \beta^{\nu} x^{\nu-1} e^{-\beta x} \text{ for } x \geq 0$$

depends on the two parameters $\nu > 0$ and $\beta > 0$. Note that ν need not be an integer.

The prior distribution on the number of mines N_A in an area A is given by

$$\Pr\{N_A = n\} = \int_0^{\infty} e^{-xA} \frac{(xA)^n}{n!} \frac{1}{\Gamma(\nu)} \beta^{\nu} x^{\nu-1} e^{-\beta x} \, dx$$

$$= \binom{\nu+n-1}{n} \left(\frac{\beta}{\beta+A}\right)^{\nu} \left(\frac{A}{\beta+A}\right)^{n}$$

$$= \binom{\nu+n-1}{n} (1-\alpha)^{\nu} \alpha^{n} \quad , \text{ for } n = 0,1,..., \tag{5}$$

with $\alpha = A/(\beta+A)$.

Suppose that the area A is swept and k mines are detected. Then, the posterior distribution on the number r of undetected mines in A is given by

$$g(r|k) = \frac{\binom{\nu+k+r-1}{k+r} \left(\frac{A}{\beta+A}\right)^{k+r} \left(\frac{\beta}{\beta+A}\right)^{\nu} \binom{k+r}{k} p^k q^r}{\sum_{j=0}^{\infty} \binom{\nu+k+j-1}{k+j} \left(\frac{A}{\beta+A}\right)^{k+j} \left(\frac{\beta}{\beta+A}\right)^{\nu} \binom{k+j}{k} p^k q^j} \text{ for } r = 0,1,... \tag{6}$$

The above expression simplifies to

$$g(r|k) = \binom{\nu+k+r-1}{r} \left(\frac{qA}{\beta+A}\right)^{r} \left(\frac{pA+\beta}{\beta+A}\right)^{\nu+k} \text{ for } r = 0,1,... \tag{7}$$

This is again negative binomial in form.

Thus we have shown that if the prior distribution on the number of mines in A is negative binomial of the form in equation (5), then the posterior distribution of undetected mines in A given that sweeping detected k mines is of the same negative binomial form with the substitutions:

$$\begin{cases} \nu \rightarrow \nu+k \\ \beta \rightarrow \beta+pA \\ A \rightarrow qA \end{cases} \quad . \tag{8}$$

The negative binomial distribution with parameters ν and α has mean $\alpha\nu/(1-\alpha)$. A simple calculation based on equation (8) then gives

$$\bar{R} = \frac{(K+k)(1-p)\alpha}{1-(1-p)\alpha} \quad . \tag{9}$$

Note that g and $\bar{R}$ again depend on both p and k but, for constant p, $\bar{R}$ increases with increasing k, giving us the desired behavior for the low quality information situation. The strength of the relationship between $\bar{R}$ and k increases as K decreases.

Notice that in each of the above cases, the posterior distribution for the number of targets remaining undetected is of the same class of the prior distribution. Only the parameters of the distribution change. In statistical language, these families of distributions are called conjugate families.

4.2. Combined Distributions

It also may be useful to express the prior as a linear combination of these distribution. For instance, one intelligence source may report that a particular area has been mined but, an evaluation of the source reduces the probability that the report is true. If the report is true (with probability β) then the Binomial Distribution is the correct model but if the report is false (with probability $1-\beta$) then the Negative Binomial Distribution may be more appropriate. This combined distribution may be expressed as,

$$\Pr\{N=n\} = \beta\binom{K}{n}\alpha^n(1-\alpha)^{K-n} + (1-\beta)\binom{K+n-1}{n}\alpha^n(1-\alpha)^K \quad \text{for } n = 0, 1, \ldots \quad .$$

4.3. Subdividing Mine Areas

The methods above are useful when the search effectiveness is spread evenly over the region of interest. In real life searches, however, the Mine Area is much larger than the search swath of one sensor or sweep and, although one can develop approximations for probability of detection over a wide area search, that search is rarely congruent with the Mine Area. Even if it were, it is often useful to model the detection and remaining mine distribution at a resolution appropriate to the resolution of the data.

4.4. Swept Channel Model

An important application is the determination of effectiveness in a *swept channel*. In this model the MCM activity specifies a channel through the mine area which all friendly forces will use to transit the area. While the prior distribution may be specified to apply to the entire area, the posterior distribution of mines in the swept channel is of interest.

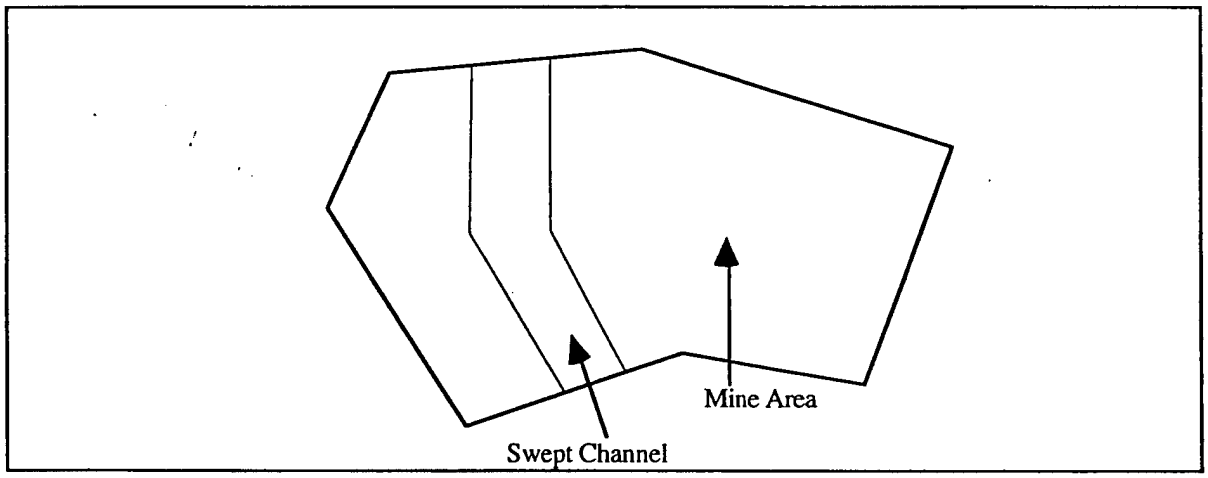


Figure 2. Swept Channel Defined as a Subregion of the Mine Area.

4.5. Model I — Binomial Distribution

In this case the total number K of mines laid in the minefield is assumed to be known. Suppose, in general, A is partitioned into disjoint subregions A_i and that K_i is the number of mines deployed in A_i. Then it is assumed that the joint distribution on the K_i is multinomial with

$$\alpha_i = \text{probability that a given mine is in subregion } A_i$$

$$= \frac{A_i}{A} .$$

Finally, it is assumed that minesweeping operations in A_i independently detect mines with probability p_i.

Consider that case in which A is divided into two subregions A_1 and A_2. Suppose that sweeping operations in A_1 have detected k_1 mines and that no sweeping has yet been conducted in A_2. Then define

$g_2(r|k_1)$ = conditional probability that r mines lie in A_2 given that k_1 mines were detected in A_1 .

Then

$$g_2(r|k_1) = \frac{\binom{K-r}{k_1} p_1^{k_1} (1-p_1)^{K-k_1-r} \binom{K}{r} a_2^r (1-\alpha_2)^{K-r}}{\sum_{j=0}^{K-k_1} \binom{K-j}{k_1} p_1^{k_1} (1-p_1)^{K-k_1-j} \binom{K}{j} \alpha_2^j (1-\alpha_2)^{K-j}} \quad \text{for } r = 0,1,\ldots,K-k_1 \quad . \qquad (10)$$

The expression in (10) can be simplified to

$$g_2(r|k_1) = \binom{K-k_1}{r} \left[\frac{\alpha_2}{p_1\alpha_2 + q_1}\right]^r \left[\frac{q_1(1-\alpha_2)}{p_1\alpha_2 + q_1}\right]^{K-k_1-r}, r = 0,1,\ldots,K-k_1 \quad , \qquad (11)$$

where $q_1 = 1-p_1$. This demonstrates that the posterior density for the mines in A_2 is again binomial in form with

$$K \rightarrow K - k_1$$

$$\alpha_2 \rightarrow \frac{\alpha_2}{p_1\alpha_2 + q_1} \quad . \qquad (12)$$

Observe that if mine detection conditions in A_1 are poor (p_1 near zero), then α_2 is relatively unaffected by the transformation in equation (12), although the maximum number of mines in A_2 is now known to be $K - k_1$. The sweeping of A_1 provides little new information about how the undetected mines are distributed. On the other hand, if mine detection conditions in A_1 are very good (p_1 near 1), then the new value of α_2 is close to 1. In this case, most of the undetected mines must now lie in A_2.

Suppose now that sweeping is also conducted in A_2 and k_2 mines are detected. Define

$g_2(r|k_1,k_2)$ = posterior distribution on number of mines r and A_2 given k_1 mines detected in A_1 and k_2 mines detected in A_2 .

An expression analogous to that in equation (10) can be written down for $g_2(r|k_1,k_2)$ but now using $g_2(r|k_1)$ as the binomial prior on the number of mines in A_2. It can be shown that $g_2(r|k_1,k_2)$ is again binomial in form with parameters that can be obtained by applying the following transformations in succession:

$$\begin{cases} K \rightarrow K - k_1 \\ \alpha_2 \rightarrow \dfrac{\alpha_2}{p_1\alpha_2 + q_1} \end{cases} \quad \text{(I)}$$

$$\begin{cases} K \rightarrow K - k_2 \\ \alpha_2 \rightarrow \dfrac{\alpha_2 q_2}{1 - \alpha_2 p_2} \end{cases} \quad \text{(II)} \qquad (13)$$

We observe that the order in which A_1 and A_2 are swept is unimportant. Therefore, it should be the case that $g_2(r|k_1,k_2)$ is independent of the order in which the substitutions in equation (13) above are applied. It is an easy matter to check that this is indeed the case.

Suppose that A_2 is an intended channel through the minefield. Sweeping may have been applied unevenly or with varying detection effectiveness over the remainder of the minefield (i.e., over A_1). The appropriate value for p_1 in calculating $g_2(r|k_1,k_2)$ is then spatial average of detection probability over A_1.

4.6. Model II — Poisson Distribution

In this case the total number of mines laid in the minefield is assumed to unknown. A spatial Poisson process model is assumed with a known density parameter δ. This Poisson process model defines both the number of mines laid and the spatial distribution of the mines. It is again assumed that minesweeping operations in A_i independently detect mines with probability p_i.

Because of the Poisson assumption, the numbers of mines in disjoint subregions of the minefield are independent. Therefore, minesweeping in one subregion A_1 provides no information about the number of mines in another subregion A_2. The conditional density of interest then is

$g_2(r|k_2)$ = conditional probability that r undetected mines remain in A_2 given that k_2 mines were detected in A_2 .

It is an easy matter to show that

$$g_2(r|k_2) = \frac{e^{-\delta A_2}\frac{(\delta A_2)^{k_2+r}}{(k_2+r)!}\binom{k_2+r}{k_2} p_2^{k_2} q_2^r}{\sum_{j=0}^{\infty} e^{-\delta A_2}\frac{(\delta A_2)^{k_2+j}}{(k_2+j)!}\binom{k_2+j}{k_2} p_2^{k_2} q_2^j} \quad r = 0,1,... \quad . \tag{14}$$

This expression can be simplified to give

$$g(r|k_2) = e^{-q_2\delta A}\frac{(q_2\delta A)^r}{r!}, \quad r = 0,1,... \tag{15}$$

This shows that the posterior distribution on undetected mines in A_2 is again Poisson but with reduced spatial density $q_2\delta$.

4.7. Model III — Negative Binomial Distribution

In this case, the total number of mines laid in the minefield is again assumed to unknown. A spatial Poisson process model is assumed but this time with an unknown density parameter δ. The parameter δ is assumed to have a gamma density. As in the binomial model and Poisson model cases, it is assumed that minesweeping operations in A_i independently detect mines with probability p_i.

Suppose now that as in our previous discussion in (5)-(8), the minefield region is divided into two subregions A_1 and A_2. Minesweeping in A_1, in addition to providing information about the number of mines in A_1, is also to provide information about the spatial density of mines throughout the minefield. Suppose that the sweeping operations in A_1 result in the detection of k_1 mines. Then it can be shown (using conditioning argument analogous to that in equation (6)) that the posterior probability density on the spatial density of mines is again gamma but with the substitutions:

$$\begin{cases} \nu \rightarrow \nu + k_1 \\ \beta \rightarrow \beta + p_1 A_1 \end{cases} \tag{16}$$

This posterior gamma density given the results of sweeping operations in A_1 determines the (prior) negative binomial distribution for the number of mines in A_2. This prior has the form shown in equation (5) with the values of ν and β obtained from equation (16) and with $A=A_2$. The posterior distribution on the number of undetected mines in A_2 given the results of sweeping operations in A_2 is again negative binomial with the parameter substitutions:

$$\begin{cases} \nu \rightarrow \nu + k_2 \\ \beta \rightarrow \beta + p_2 A_2 \\ A_2 \rightarrow q_2 A_2 \end{cases} \quad . \tag{17}$$

We have therefore shown that $g_2(r|k_1,k_2)$ as previously defined is negative binomial and we have specified a procedure for determining its parameters.

4.8. Track Segment Model

Another application for subdividing the mine area is where one requires a new distribution for the entire area based on a nonuniform search. A common and convenient model to represent nonuniform search is the *track segment* model. Suppose we can create a lateral range curve distribution as first described in [8]. The lateral range curve defines probability of detection for one object as a searcher passes by, as a function of the distance from searcher to object at the closest point of approach.

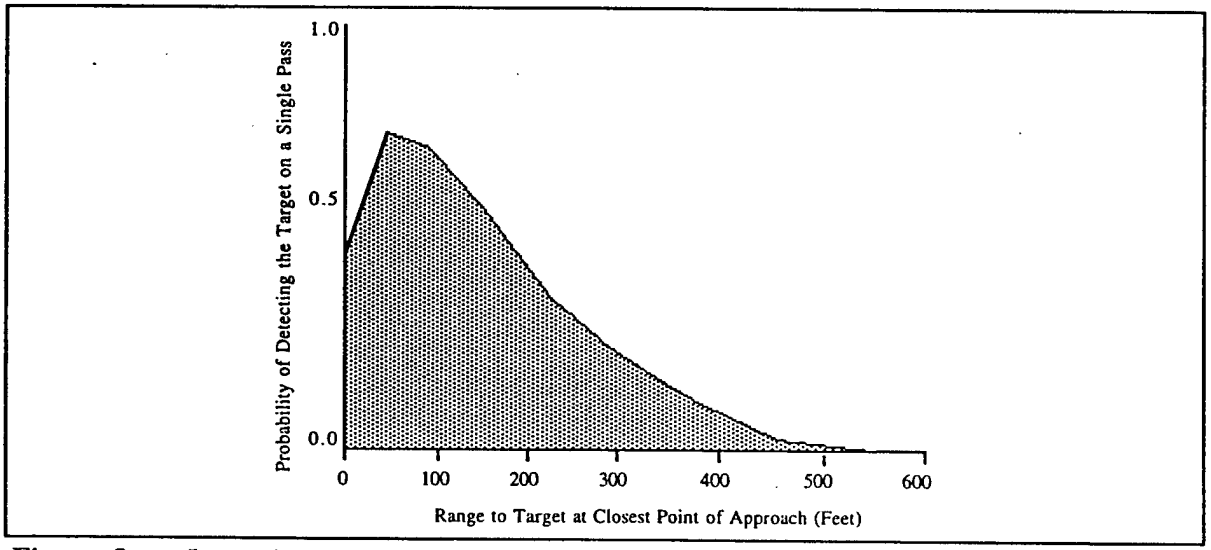


Figure 3. Lateral Range Curve Defined by Conditional Probability of Detection as a Function of Miss Distance (Range).

Let us approximate such a function by discretization, as shown in Figure 4.

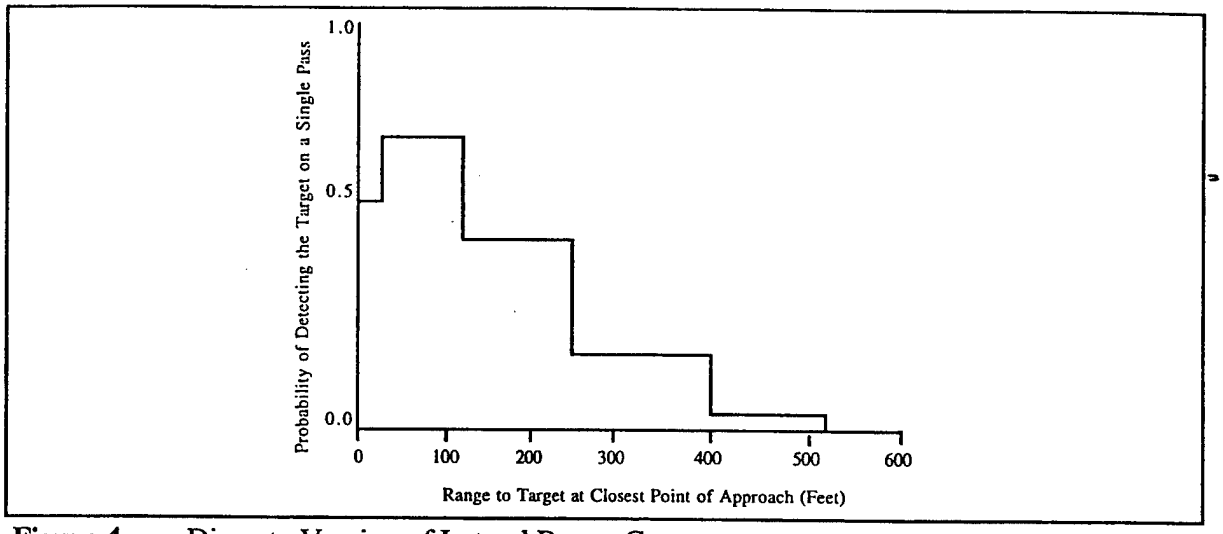


Figure 4. Discrete Version of Lateral Range Curve.

Now, when we describe the path of the searcher by a series of (connected) path segments, the probability of detection distribution can be very closely approximated by a collection of polygons defined by the discrete lateral range curve extended parallel to the path segments, as shown in Figure 5.[1]

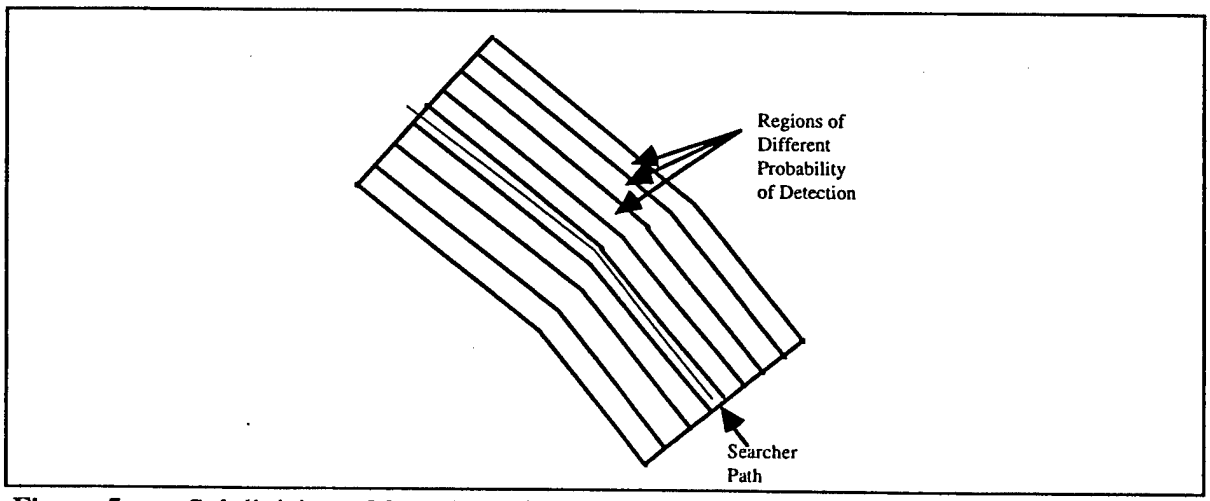


Figure 5. Subdivision of Search Region According to Discrete Lateral Range Curve.

In order to complete the modeling of the effect of search, we need to be able to subdivide the original mine area into J subregions corresponding to the polygons of equal probability of detection. The posterior distribution in every subregion is of interest.

---

[1]This is the model used in the MCM detection model in the MEDAL expeditionary warfare decision aid and in the MELIAN II Underwater Search program developed by Wagner Associates. MEDAL does not have a prior mine distribution capability; MELIAN II can represent the prior distribution for a single target but cannot currently represent prior information for a multiple target situation such as a mine area.

Let

k_i = the number of mines found in subregion i,

r_i = the number of mines remaining in subregion i,

p_i = the conditional probability of detection for a mine in subregion i, and

α_i = the fraction of the total mine area covered by subregion i.

$$A(r) = \Pr\{N = \sum_{j=1}^{J} (r_j + k_j)\}$$

Then,

$$\Pr\{r_1, r_2, \ldots r_J \mid k_1, k_2, \ldots k_J, p_1, p_2, \ldots p_J\} =$$

$$\frac{A(r) \prod_{i=1}^{J} \left\{ \binom{\sum_{j=1}^{J} (r_j+k_j)}{r_i+k_i} \alpha_i^{r_i+k_i} (1-\alpha_i)^{\sum_{j=1}^{J}(r_j+k_j)-r_i-k_i} \binom{r_i+k_i}{r_i} (1-p_i)^{r_i} p_i^{k_i} \right\}}{\sum_{s_1=0,\ldots} \sum_{s_2=0,\ldots} \ldots \left(A(s) \prod_{i=1}^{J} \left\{ \binom{\sum_{j=i}^{J} (s_j+k_j)}{s_j+k_j} \alpha_i^{s_i+k_i} (1-\alpha_i)^{\sum_{j=1}^{J}(s_j+k_j)-s_i-k_i} \binom{s_i+k_i}{s_i} (1-p_i)^{s_i} p_i^{k_i} \right\} \right)} \tag{18}$$

As search progresses, the polygons that represent the subregions of different search density become smaller and more numerous and the computations could become overwhelming in the cases of Binomial and Negative Binomial distributing. A useful approximation can be constructed using the Poisson approximation which then can be normalized to produce the total expected number of mines predicted by the appropriate distribution applied over the entire Mine Area.

4.9. Conclusion

Minehunting and minesweeping have as their general objectives to reduce the number of undetected or unexploded mines in a specified area to zero (within some tolerance). To measure the accomplishment of this objective statistically, given search results, it is necessary to specify prior distributions, which are built from intelligence and surveillance. The penalty for the use of incorrect priors can be either spending too much time searching or

sweeping, thus delaying important operations or, conversely, spending too little effort leaving an unacceptably high mine threat to friendly forces.

Reducing the uncertainties in posterior distributions and the accompanying penalties is the contribution of effective intelligence and surveillance, along with properly structured decision aids to develop measures and recommendations based on those priors.

References

[1] Booch, G., Object-Oriented Analsysis and Design with Applications, Second Edition, Benjamin/Cummings Pulications Company, Inc.

[2] White, I., Using the Booch Method, A Rational Approach, Benjamin/Cummings Publication, Inc.

[3] Van Belle, CAPT B.T. et. al., "Concept of Operations for Mine Countermeasures in the 21st Century", Mine Warfare Branch (N852) Report, September 1995.

[4] Feller, "An Introduction to Probability Theory and Its Applications", Vol. I, 3rd Ed., John Wiley & Sons, Inc. NY, 1968, p. 124.

[5] Richardson and Corwin, "General Analysis Plan for EOD Search Operations in the Suez Canal", Daniel H. Wagner Associates, Inc. Report to Explosive Ordnance Disposal Facility, April 1974.

[6] Richardson, et. al., "Operations Analysis During the 1974 Search for Unexploded Ordnance in the Suez Canal", Daniel H. Wagner Associates Report to Naval Explosive Ordnance Disposal Facility, Volume I, June 1975.

[7] Stone, L. D., Theory of Optimal Search, Academic Press, 1975.

[8] Koopman, B.O., "Search and Screening," OEG Report No. 56, The Summary Reports Group of the Columbia University Division of War Research 1946. (Available from the Center for Naval Analyses.)

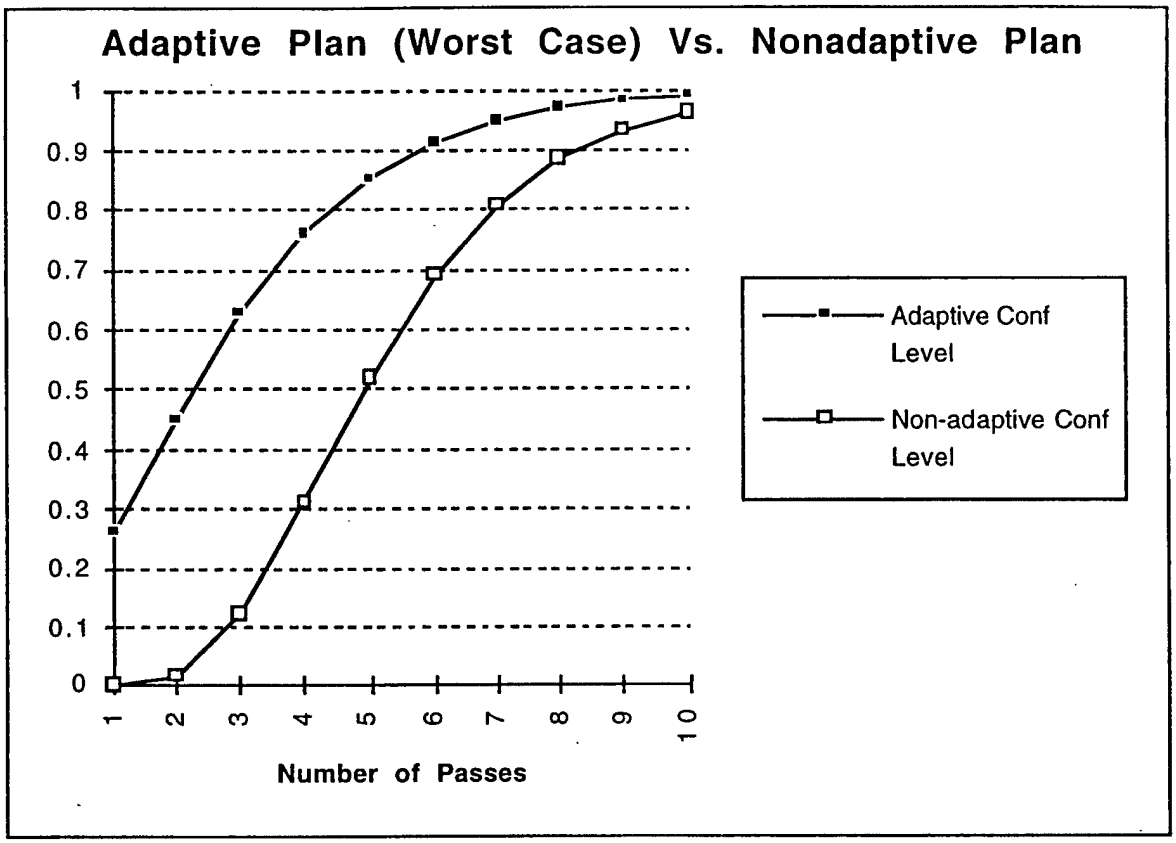
Number of Mines Detected	Probability No Mines Remain Undetected
0	.452
1	.489
2	.530
3	.574
4	.621
5	.672
6	.728
7	.788
8	.852
9	.924
10	1.000

5. Benefits of the Adaptive Plan

a. Smaller Expected Number of Passes Required to Achieve Equivalent Confidence Level:

Non Adaptive Plan	3.84
Adaptive Plan	1.79
Net Savings	53.6%

b. Increased Confidence Level. The graph below shows the confidence level resulting from the adaptive plan in the worst case (no mines detected) against the nonadaptive plan.


Adaptive Plan (Worst Case) Vs. Nonadaptive Plan
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
2
3
4
5
6
7
8
9
10
Number of Passes
Adaptive Conf Level
Non-adaptive Conf Level

Appendix B: C++ Header Code for SEMMIW

We have produced C++ header code for the classes defined in our Phase I effort. The Rational Rose/C++ tool produces .cpp and .h files for the classes in the model. We have included these files on a separate disk that has been delivered to the COTR. In this Appendix we have included as a sample the printed header code for the following SEMMIW classes:

- Forces::Force
- Forces::DiverTeam
- MineDangerArea::MineArea
- Threat::Mine Type

```
// CG 2.7 Version 2.00 950411 source generated Sat Nov 25, 1995 11:57:36
//## begin module.cm preserve=no
//          %X% %Q% %Z% %W%
//## end module.cm

//## begin module.cp preserve=no
//## end module.cp

//## Module: Force; Pseudo Package body
//## Subsystem: Forces
//## Source file: c:\rose\forces\force.cpp


//## begin module.additionalIncludes preserve=no
//## end module.additionalIncludes

//## begin module.includes preserve=yes
//## end module.includes

// Force
#include "Forces\Force.h"

//## begin module.additionalDeclarations preserve=yes
//## end module.additionalDeclarations


// Class Force
//## Constructors (implementation)
Force::Force()
  //## begin Force::Force%.hasinit preserve=no
  //## end Force::Force%.hasinit
  //## begin Force::Force%.initialization preserve=yes
  //## end Force::Force%.initialization
{
  //## begin Force::Force%.body preserve=yes
  //## end Force::Force%.body
}

Force::Force(const Force &right)
  //## begin Force::Force%copy.hasinit preserve=no
  //## end Force::Force%copy.hasinit
  //## begin Force::Force%copy.initialization preserve=yes
  //## end Force::Force%copy.initialization
{
  //## begin Force::Force%copy.body preserve=yes
  //## end Force::Force%copy.body
}


//## Destructor (implementation)
Force::~Force()
{
  //## begin Force::~Force%.body preserve=yes
  //## end Force::~Force%.body
```

```
}


//## Assignment Operation (implementation)
const Force & Force::operator=(const Force &right)
{
 //## begin Force::operator=%.body preserve=yes
 //## end Force::operator=%.body
}


//## Equality Operations (implementation)
int Force::operator==(const Force &right) const
{
 //## begin Force::operator==%.body preserve=yes
 //## end Force::operator==%.body
}

int Force::operator!=(const Force &right) const
{
 //## begin Force::operator!=%.body preserve=yes
 //## end Force::operator!=%.body
}



//## Get and Set Operations for Has Relationships (implementation)
const Mildesig Force::get_Designator() const
{
 //## begin Force::get_Designator%.get preserve=no
 return Designator;
 //## end Force::get_Designator%.get
}

void Force::set_Designator(const Mildesig value)
{
 //## begin Force::set_Designator%.set preserve=no
 Designator = value;
 //## end Force::set_Designator%.set
}

const UnboundedListByReference Force::get_hasassets() const
{
 //## begin Force::get_hasassets%.get preserve=no
 return hasassets;
 //## end Force::get_hasassets%.get
}

void Force::set_hasassets(const UnboundedListByReference value)
{
 //## begin Force::set_hasassets%.set preserve=no
 hasassets = value;
 //## end Force::set_hasassets%.set
}
```

```
// Additional Declarations
//## begin Force.declarations preserve=yes
//## end Force.declarations
```

```
// CG 2.7 Version 2.00 950411 source generated Sat Nov 25, 1995 11:57:36
//## begin module.cm preserve=no
//          %X% %Q% %Z% %W%
//## end module.cm

//## begin module.cp preserve=no
//## end module.cp

//## Module: Force; Pseudo Package specification
//## Subsystem: Forces
//## Source file: c:\rose\forces\force.h


#ifndef Force_h
#define Force_h 1

//## begin module.additionalIncludes preserve=no
//## end module.additionalIncludes

//## begin module.includes preserve=yes
//## end module.includes

// Asset
#include "Forces\Asset.h"

//## begin module.additionalDeclarations preserve=yes
//## end module.additionalDeclarations


//## Class: Force
//          Represents the Mine Warfare military force
//## InUnit: c:\rose\semmiw\semmiw.mdl
//## Category: Forces
//## Subsystem: Forces
//## Concurrency: Sequential
//## Persistence: Transient
//## Cardinality: n

class Force
{
 //## begin Force.initialDeclarations preserve=yes
 //## end Force.initialDeclarations

 public:
  //## Constructors (generated)
    Force();
    Force(const Force &right);

  //## Destructor (generated)
    ~Force();

  //## Assignment Operation (generated)
    const Force & operator=(const Force &right);
```

```
 //## Equality Operations (generated)
  int operator==(const Force &right) const;
  int operator!=(const Force &right) const;

 //## Get and Set Operations for Has Relationships (generated)
  const UnboundedListByReference get_hasassets() const;
  void set_hasassets(const UnboundedListByReference value);


 // Additional Public Declarations
  //## begin Force.public preserve=yes
  //## end Force.public


protected:
 //## Get and Set Operations for Has Relationships (generated)
  //## Documentation Force::Designator.has:
  //    The military designation of the particular mine warfare
  //    force.
  const Mildesig get_Designator() const;
  void set_Designator(const Mildesig value);


 // Additional Protected Declarations
  //## begin Force.protected preserve=yes
  //## end Force.protected


private:
 // Additional Private Declarations
  //## begin Force.private preserve=yes
  //## end Force.private


private:  //## implementation
 // Data Members for Has Relationships
  //## begin Force::Designator.has preserve=no  protected: Mildesig {1 -> 1V}
  Mildesig Designator;
  //## end Force::Designator.has

  //## begin Force::hasassets.has preserve=no  public: Asset {1 -> nR}
  UnboundedListByReference hasassets;
  //## end Force::hasassets.has


 // Additional Implementation Declarations
  //## begin Force.implementation preserve=yes
  //## end Force.implementation

};
```

```
#endif
```

```
// CG 2.7 Version 2.00 950411 source generated Sat Nov 25, 1995 11:57:36
//## begin module.cm preserve=no
//          %X% %Q% %Z% %W%
//## end module.cm

//## begin module.cp preserve=no
//## end module.cp

//## Module: DiverTeam; Pseudo Package body
//## Subsystem: Forces
//## Source file: c:\rose\forces\divrteam.cpp


//## begin module.additionalIncludes preserve=no
//## end module.additionalIncludes

//## begin module.includes preserve=yes
//## end module.includes

// DiverTeam
#include "Forces\DivrTeam.h"

//## begin module.additionalDeclarations preserve=yes
//## end module.additionalDeclarations


// Class DiverTeam
//## Constructors (implementation)
DiverTeam::DiverTeam()
  //## begin DiverTeam::DiverTeam%.hasinit preserve=no
  //## end DiverTeam::DiverTeam%.hasinit
  //## begin DiverTeam::DiverTeam%.initialization preserve=yes
  //## end DiverTeam::DiverTeam%.initialization
{
  //## begin DiverTeam::DiverTeam%.body preserve=yes
  //## end DiverTeam::DiverTeam%.body
}

DiverTeam::DiverTeam(const DiverTeam &right)
  //## begin DiverTeam::DiverTeam%copy.hasinit preserve=no
  //## end DiverTeam::DiverTeam%copy.hasinit
  //## begin DiverTeam::DiverTeam%copy.initialization preserve=yes
  //## end DiverTeam::DiverTeam%copy.initialization
{
  //## begin DiverTeam::DiverTeam%copy.body preserve=yes
  //## end DiverTeam::DiverTeam%copy.body
}


//## Destructor (implementation)
DiverTeam::~DiverTeam()
{
  //## begin DiverTeam::~DiverTeam%.body preserve=yes
  //## end DiverTeam::~DiverTeam%.body
```

```
}


//## Assignment Operation (implementation)
const DiverTeam & DiverTeam::operator=(const DiverTeam &right)
{
 //## begin DiverTeam::operator=%.body preserve=yes
 //## end DiverTeam::operator=%.body
}


//## Equality Operations (implementation)
int DiverTeam::operator==(const DiverTeam &right) const
{
 //## begin DiverTeam::operator==%.body preserve=yes
 //## end DiverTeam::operator==%.body
}

int DiverTeam::operator!=(const DiverTeam &right) const
{
 //## begin DiverTeam::operator!=%.body preserve=yes
 //## end DiverTeam::operator!=%.body
}


//## Other Operations (implementation)
Integer DiverTeam::add_a_diver()
{
 //## begin DiverTeam::add_a_diver%817318579.body preserve=yes
 //## end DiverTeam::add_a_diver%817318579.body
}

Integer DiverTeam::subtract_a_diver()
{
 //## begin DiverTeam::subtract_a_diver%817318580.body preserve=yes
 //## end DiverTeam::subtract_a_diver%817318580.body
}


//## Get and Set Operations for Has Relationships (implementation)
const Integer DiverTeam::get_Divercount() const
{
 //## begin DiverTeam::get_Divercount%.get preserve=no
 return Divercount;
 //## end DiverTeam::get_Divercount%.get
}

void DiverTeam::set_Divercount(const Integer value)
{
 //## begin DiverTeam::set_Divercount%.set preserve=no
 Divercount = value;
 //## end DiverTeam::set_Divercount%.set
}
```

```
const UnboundedListByReference DiverTeam::get_teamofdivers() const
{
 //## begin DiverTeam::get_teamofdivers%.get preserve=no
 return teamofdivers;
 //## end DiverTeam::get_teamofdivers%.get
}

void DiverTeam::set_teamofdivers(const UnboundedListByReference value)
{
 //## begin DiverTeam::set_teamofdivers%.set preserve=no
 teamofdivers = value;
 //## end DiverTeam::set_teamofdivers%.set
}


// Additional Declarations
//## begin DiverTeam.declarations preserve=yes
//## end DiverTeam.declarations
```

```
// CG 2.7 Version 2.00 950411 source generated Sat Nov 25, 1995 11:57:36
//## begin module.cm preserve=no
//          %X% %Q% %Z% %W%
//## end module.cm

//## begin module.cp preserve=no
//## end module.cp

//## Module: DiverTeam; Pseudo Package specification
//## Subsystem: Forces
//## Source file: c:\rose\forces\divrteam.h


#ifndef DivrTeam_h
#define DivrTeam_h 1

//## begin module.additionalIncludes preserve=no
//## end module.additionalIncludes

//## begin module.includes preserve=yes
//## end module.includes

// Diver
#include "Forces\Diver.h"
// EOD_Det
#include "Forces\EODDet.h"

//## begin module.additionalDeclarations preserve=yes
//## end module.additionalDeclarations


//## Class: DiverTeam
//          Represents a team of EOD divers
//## InUnit: c:\rose\semmiw\semmiw.mdl
//## Category: Forces
//## Subsystem: Forces
//## Concurrency: Sequential
//## Persistence: Transient
//## Cardinality: n

class DiverTeam : public EOD_Det
{
  //## begin DiverTeam.initialDeclarations preserve=yes
  //## end DiverTeam.initialDeclarations

  public:
    //## Constructors (generated)
      DiverTeam();
      DiverTeam(const DiverTeam &right);

    //## Destructor (generated)
      ~DiverTeam();

    //## Assignment Operation (generated)
```

```
  const DiverTeam & operator=(const DiverTeam &right);

//## Equality Operations (generated)
  int operator==(const DiverTeam &right) const;
  int operator!=(const DiverTeam &right) const;


//## Other Operations (specified)
  //## Documentation:
  //    Add 1 to the diver count
  //## Concurrency: Sequential
  //## Semantics:
  //    Divercount = Divercount + 1
  Integer add_a_diver();

  //## Documentation:
  //    Decrement the number of divers in the team by one.
  //## Concurrency: Sequential
  //## Semantics:
  //    divercount = divercount - 1
  Integer subtract_a_diver();


//## Get and Set Operations for Has Relationships (generated)
  //## Documentation DiverTeam::Divercount.has:
  //    Number of divers in the team.
  const Integer get_Divercount() const;
  void set_Divercount(const Integer value);

  const UnboundedListByReference get_teamofdivers() const;
  void set_teamofdivers(const UnboundedListByReference value);


 // Additional Public Declarations
  //## begin DiverTeam.public preserve=yes
  //## end DiverTeam.public


protected:
 // Additional Protected Declarations
  //## begin DiverTeam.protected preserve=yes
  //## end DiverTeam.protected



private:
 // Additional Private Declarations
  //## begin DiverTeam.private preserve=yes
  //## end DiverTeam.private



private:  //## implementation
```

```
  // Data Members for Has Relationships
    //## begin DiverTeam::Divercount.has preserve=no  public: Integer {1 -> 1V}
    Integer Divercount;
    //## end DiverTeam::Divercount.has

    //## begin DiverTeam::teamofdivers.has preserve=no  public: Diver {1 -> nR}
    UnboundedListByReference teamofdivers;
    //## end DiverTeam::teamofdivers.has


  // Additional Implementation Declarations
    //## begin DiverTeam.implementation preserve=yes
    //## end DiverTeam.implementation

};



#endif
```

```
// CG 2.7 Version 2.00 950411 source generated Sat Nov 25, 1995 11:58:49
//## begin module.cm preserve=no
//          %X% %Q% %Z% %W%
//## end module.cm

//## begin module.cp preserve=no
//## end module.cp

//## Module: MineArea; Pseudo Package body
//## Subsystem: MineDangerArea
//## Source file: c:\rose\mndngrra\minearea.cpp


//## begin module.additionalIncludes preserve=no
//## end module.additionalIncludes

//## begin module.includes preserve=yes
//## end module.includes

// MineArea
#include "MnDngrra\MineArea.h"

//## begin module.additionalDeclarations preserve=yes
//## end module.additionalDeclarations


// Class MineArea
//## Constructors (implementation)
MineArea::MineArea()
  //## begin MineArea::MineArea%.hasinit preserve=no
  //## end MineArea::MineArea%.hasinit
  //## begin MineArea::MineArea%.initialization preserve=yes
  //## end MineArea::MineArea%.initialization
{
  //## begin MineArea::MineArea%.body preserve=yes
  //## end MineArea::MineArea%.body
}

MineArea::MineArea(const MineArea &right)
  //## begin MineArea::MineArea%copy.hasinit preserve=no
  //## end MineArea::MineArea%copy.hasinit
  //## begin MineArea::MineArea%copy.initialization preserve=yes
  //## end MineArea::MineArea%copy.initialization
{
  //## begin MineArea::MineArea%copy.body preserve=yes
  //## end MineArea::MineArea%copy.body
}


//## Destructor (implementation)
MineArea::~MineArea()
{
  //## begin MineArea::~MineArea%.body preserve=yes
  //## end MineArea::~MineArea%.body
```

```
}


//## Assignment Operation (implementation)
const MineArea & MineArea::operator=(const MineArea &right)
{
 //## begin MineArea::operator=%.body preserve=yes
 //## end MineArea::operator=%.body
}


//## Equality Operations (implementation)
int MineArea::operator==(const MineArea &right) const
{
 //## begin MineArea::operator==%.body preserve=yes
 //## end MineArea::operator==%.body
}

int MineArea::operator!=(const MineArea &right) const
{
 //## begin MineArea::operator!=%.body preserve=yes
 //## end MineArea::operator!=%.body
}


//## Get and Set Operations for Has Relationships (implementation)
const UnboundedListByReference MineArea::get_hasmines() const
{
 //## begin MineArea::get_hasmines%.get preserve=no
 return hasmines;
 //## end MineArea::get_hasmines%.get
}

void MineArea::set_hasmines(const UnboundedListByReference value)
{
 //## begin MineArea::set_hasmines%.set preserve=no
 hasmines = value;
 //## end MineArea::set_hasmines%.set
}

const ThreatDistn MineArea::get_hasthreat() const
{
 //## begin MineArea::get_hasthreat%.get preserve=no
 return hasthreat;
 //## end MineArea::get_hasthreat%.get
}

void MineArea::set_hasthreat(const ThreatDistn value)
{
 //## begin MineArea::set_hasthreat%.set preserve=no
 hasthreat = value;
 //## end MineArea::set_hasthreat%.set
}
```

```
// Additional Declarations
//## begin MineArea.declarations preserve=yes
//## end MineArea.declarations
```

```
// CG 2.7 Version 2.00 950411 source generated Sat Nov 25, 1995 11:58:49
//## begin module.cm preserve=no
//          %X% %Q% %Z% %W%
//## end module.cm

//## begin module.cp preserve=no
//## end module.cp

//## Module: MineArea; Pseudo Package specification
//## Subsystem: MineDangerArea
//## Source file: c:\rose\mndngrra\minearea.h


#ifndef MineArea_h
#define MineArea_h 1

//## begin module.additionalIncludes preserve=no
//## end module.additionalIncludes

//## begin module.includes preserve=yes
//## end module.includes

// Area
#include "Gegraphy\Area.h"
// ThreatDistn
#include "MnDngrra\ThrtDstn.h"
// FoundMine
#include "MnDngrra\FondMine.h"

//## begin module.additionalDeclarations preserve=yes
//## end module.additionalDeclarations


//## Class: MineArea
//          Represents an area that either is known to be mined or
//          is suspected of being mined.
//## InUnit: c:\rose\semmiw\semmiw.mdl
//## Category: MineDangerArea
//## Subsystem: MineDangerArea
//## Concurrency: Sequential
//## Persistence: Transient
//## Cardinality: n

class MineArea : public Area
{
 //## begin MineArea.initialDeclarations preserve=yes
 //## end MineArea.initialDeclarations

 public:
  //## Constructors (generated)
    MineArea();
    MineArea(const MineArea &right);

  //## Destructor (generated)
```

```
  ~MineArea();

 //## Assignment Operation (generated)
  const MineArea & operator=(const MineArea &right);

 //## Equality Operations (generated)
  int operator==(const MineArea &right) const;
  int operator!=(const MineArea &right) const;

 //## Get and Set Operations for Has Relationships (generated)
  //## Documentation MineArea::hasmines.has:
  //    A mine area will have some (0 to N) found mines in it.
  const UnboundedListByReference get_hasmines() const;
  void set_hasmines(const UnboundedListByReference value);

  //## Documentation MineArea::hasthreat.has:
  //    A mine area has a composite mine threat distribution
  //    associated with it.
  const ThreatDistn get_hasthreat() const;
  void set_hasthreat(const ThreatDistn value);


 // Additional Public Declarations
  //## begin MineArea.public preserve=yes
  //## end MineArea.public



protected:
 // Additional Protected Declarations
  //## begin MineArea.protected preserve=yes
  //## end MineArea.protected



private:
 // Additional Private Declarations
  //## begin MineArea.private preserve=yes
  //## end MineArea.private



private:  //## implementation
 // Data Members for Has Relationships
  //## begin MineArea::hasmines.has preserve=no  public: FoundMine {1 -> 0..nR}
  UnboundedListByReference hasmines;
  //## end MineArea::hasmines.has

  //## begin MineArea::hasthreat.has preserve=no  public: ThreatDistn {1 -> 1V}
  ThreatDistn hasthreat;
  //## end MineArea::hasthreat.has


 // Additional Implementation Declarations
```

```
    //## begin MineArea.implementation preserve=yes
    //## end MineArea.implementation

};



#endif
```

```
// CG 2.7 Version 2.00 950411 source generated Sat Nov 25, 1995 11:59:05
//## begin module.cm preserve=no
//          %X% %Q% %Z% %W%
//## end module.cm

//## begin module.cp preserve=no
//## end module.cp

//## Module: MineType; Pseudo Package body
//## Subsystem: Threat
//## Source file: c:\rose\threat\minetype.cpp


//## begin module.additionalIncludes preserve=no
//## end module.additionalIncludes

//## begin module.includes preserve=yes
//## end module.includes

// MineType
#include "Threat\MineType.h"

//## begin module.additionalDeclarations preserve=yes
//## end module.additionalDeclarations


// Class MineType
//## Constructors (implementation)
MineType::MineType()
  //## begin MineType::MineType%.hasinit preserve=no
  //## end MineType::MineType%.hasinit
  //## begin MineType::MineType%.initialization preserve=yes
  //## end MineType::MineType%.initialization
{
  //## begin MineType::MineType%.body preserve=yes
  //## end MineType::MineType%.body
}

MineType::MineType(const MineType &right)
  //## begin MineType::MineType%copy.hasinit preserve=no
  //## end MineType::MineType%copy.hasinit
  //## begin MineType::MineType%copy.initialization preserve=yes
  //## end MineType::MineType%copy.initialization
{
  //## begin MineType::MineType%copy.body preserve=yes
  //## end MineType::MineType%copy.body
}


//## Destructor (implementation)
MineType::~MineType()
{
  //## begin MineType::~MineType%.body preserve=yes
  //## end MineType::~MineType%.body
```

```
}


//## Assignment Operation (implementation)
const MineType & MineType::operator=(const MineType &right)
{
 //## begin MineType::operator=%.body preserve=yes
 //## end MineType::operator=%.body
}


//## Equality Operations (implementation)
int MineType::operator==(const MineType &right) const
{
 //## begin MineType::operator==%.body preserve=yes
 //## end MineType::operator==%.body
}

int MineType::operator!=(const MineType &right) const
{
 //## begin MineType::operator!=%.body preserve=yes
 //## end MineType::operator!=%.body
}



//## Get and Set Operations for Has Relationships (implementation)
const Explosive * MineType::get_detonates() const
{
 //## begin MineType::get_detonates%.get preserve=no
 return detonates;
 //## end MineType::get_detonates%.get
}

void MineType::set_detonates(Explosive *const  value)
{
 //## begin MineType::set_detonates%.set preserve=no
 detonates = value;
 //## end MineType::set_detonates%.set
}

const ArmingMechanism * MineType::get_armedby() const
{
 //## begin MineType::get_armedby%.get preserve=no
 return armedby;
 //## end MineType::get_armedby%.get
}

void MineType::set_armedby(ArmingMechanism *const  value)
{
 //## begin MineType::set_armedby%.set preserve=no
 armedby = value;
 //## end MineType::set_armedby%.set
}
```

```
const SterilizingMechanism * MineType::get_hassteril() const
{
 //## begin MineType::get_hassteril%.get preserve=no
 return hassteril;
 //## end MineType::get_hassteril%.get
}

void MineType::set_hassteril(SterilizingMechanism *const  value)
{
 //## begin MineType::set_hassteril%.set preserve=no
 hassteril = value;
 //## end MineType::set_hassteril%.set
}

const ActuatingMechanism * MineType::get_firedby() const
{
 //## begin MineType::get_firedby%.get preserve=no
 return firedby;
 //## end MineType::get_firedby%.get
}

void MineType::set_firedby(ActuatingMechanism *const  value)
{
 //## begin MineType::set_firedby%.set preserve=no
 firedby = value;
 //## end MineType::set_firedby%.set
}

const char MineType::get_minetypename() const
{
 //## begin MineType::get_minetypename%.get preserve=no
 return minetypename;
 //## end MineType::get_minetypename%.get
}

void MineType::set_minetypename(const char value)
{
 //## begin MineType::set_minetypename%.set preserve=no
 minetypename = value;
 //## end MineType::set_minetypename%.set
}


// Additional Declarations
//## begin MineType.declarations preserve=yes
//## end MineType.declarations
```

```
// CG 2.7 Version 2.00 950411 source generated Sat Nov 25, 1995 11:59:05
//## begin module.cm preserve=no
//          %X% %Q% %Z% %W%
//## end module.cm

//## begin module.cp preserve=no
//## end module.cp

//## Module: MineType; Pseudo Package specification
//## Subsystem: Threat
//## Source file: c:\rose\threat\minetype.h


#ifndef MineType_h
#define MineType_h 1

//## begin module.additionalIncludes preserve=no
//## end module.additionalIncludes

//## begin module.includes preserve=yes
//## end module.includes

// ActuatingMechanism
#include "Threat\ActtngMc.h"
// SterilizingMechanism
#include "Threat\StrlzngM.h"
// ArmingMechanism
#include "Threat\ArmngMch.h"
// Explosive
#include "Threat\Explsive.h"

//## begin module.additionalDeclarations preserve=yes
//## end module.additionalDeclarations


//## Class: MineType
//          Template for a given type of mine
//## InUnit: c:\rose\semmiw\semmiw.mdl
//## Category: Threat
//## Subsystem: Threat
//## Concurrency: Sequential
//## Persistence: Transient
//## Cardinality: n

class MineType
{
  //## begin MineType.initialDeclarations preserve=yes
  //## end MineType.initialDeclarations

  public:
    //## Constructors (generated)
      MineType();
      MineType(const MineType &right);
```

```
  //## Destructor (generated)
    ~MineType();

  //## Assignment Operation (generated)
    const MineType & operator=(const MineType &right);

  //## Equality Operations (generated)
    int operator==(const MineType &right) const;
    int operator!=(const MineType &right) const;

  //## Get and Set Operations for Has Relationships (generated)
    //## Documentation MineType::detonates.has:
    //    Relates mine type to its explosive.
    const Explosive * get_detonates() const;
    void set_detonates(Explosive *const  value);

    //## Documentation MineType::armedby.has:
    //    Relates the mine to its arming mechanism.
    const ArmingMechanism * get_armedby() const;
    void set_armedby(ArmingMechanism *const  value);

    //## Documentation MineType::hassteril.has:
    //    Relates mine type to the mechanism that sterilizes it.
    const SterilizingMechanism * get_hassteril() const;
    void set_hassteril(SterilizingMechanism *const  value);

    //## Documentation MineType::firedby.has:
    //    Relates the mine type to the actuating mechanism.
    const ActuatingMechanism * get_firedby() const;
    void set_firedby(ActuatingMechanism *const  value);

    //## Documentation MineType::minetypename.has:
    //    The name of the mine type.
    const char get_minetypename() const;
    void set_minetypename(const char value);


  // Additional Public Declarations
    //## begin MineType.public preserve=yes
    //## end MineType.public


protected:
  // Additional Protected Declarations
    //## begin MineType.protected preserve=yes
    //## end MineType.protected


private:
  // Additional Private Declarations
    //## begin MineType.private preserve=yes
    //## end MineType.private
```

```
  private:  //## implementation
    // Data Members for Has Relationships
      //## begin MineType::detonates.has preserve=no  public: Explosive {1 -> 1R}
      Explosive *detonates;
      //## end MineType::detonates.has

      //## begin MineType::armedby.has preserve=no  public: ArmingMechanism {1 -> 1R}
      ArmingMechanism *armedby;
      //## end MineType::armedby.has

      //## begin MineType::hassteril.has preserve=no  public: SterilizingMechanism {1 -> 1R}
      SterilizingMechanism *hassteril;
      //## end MineType::hassteril.has

      //## begin MineType::firedby.has preserve=no  public: ActuatingMechanism {1 -> 1R}
      ActuatingMechanism *firedby;
      //## end MineType::firedby.has

      //## begin MineType::minetypename.has preserve=no  public: char {1 -> 1V}
      char minetypename;
      //## end MineType::minetypename.has


    // Additional Implementation Declarations
      //## begin MineType.implementation preserve=yes
      //## end MineType.implementation

};



#endif
```

Appendix A: Operational Benefits of MCM Planning Using Mine Distributions

1. In this simple example, based on good intelligence, there are 10 mines in the MDA. The probability an individual mine is in a lane of interest in the MDA is 0.2. The single pass probability of detection against an individual mine is 0.425. We assume that no mines are undetectable and that an appropriate measure of effectiveness for MCM effort is the number of passes required to achieve a specified goal.

2. The traditional minehunting criterion does not explicitly recognize the expected number of mines but, rather, attempts to achieve a minimum conditional probability of detection. In this example, we assume the mine hunter makes 4 passes to achieve a 90% probability of detection against one mine. In order to compare this nonadaptive plan against the adaptive plan, we need similar MOEs. We do that in this example by using the maximum number of mines that could be in the lanes (10) and compute the equivalent probability that no mines remain undetected. Under this assumption, after 4 passes there is a 31.4% chance that no mines remain undetected in the lane; equivalently, a 68.6% chance that at least 1 mine remains undetected in the lane.

3. An adaptive minehunting plan is based on the high-quality information prior mine distribution, using the mathematics in Chapter 4 of the Phase I Final Report. After each pass, the mine hunter re-evaluates the probability that at least one mine remains undetected in the lane, based on both the probability of detection *and the number of mines detected*. When that probability is less than 68.6% the mine hunter stops making passes.

4. The following tables illustrate the results of the adaptive plan.

Pass One. The search can stop if 2 or more mines are detected:

Number of Mines Detected	Probability of Detecting That Number of Mines	Probability At Least One Mine Remains
0	.411	.739
1	.382	.701
2 or more	.207	.660

Pass Two. After Pass Two, the probability that at least one mine remains undetected does not exceed 54.8%, no matter how many mines are detected in the pass. The mine hunter

can therefore stop making passes and still have at least the level of confidence that no mines remain undetected as under the non-adaptive plan.

The expected number of passes using adaptive plan is therefore 1.79, as shown below:

Event	Probability	Contribution to Expected Value
Stop After First Pass	.207	.207
Stop After Second Pass	.793	1.586

To compare the expected number of passes using the adaptive plan with the expected number of passes using the non-adaptive plan we must account for the probability that all 10 mines are detected in the lane on the first, second, or third pass, allowing the mine hunter to stop. The expected number of passes for the adaptive plan is 3.84, as shown below:

Pass on Which 10th Mine is Detected	Probability	Contribution to Expected Value
First	.0002	.0002
Second	.0178	.0356
Third	.1034	.3102
Fourth or later	.8786	3.5144

The probability that no mines remain undetected after the second pass depends on the number of mines detected in that pass. The confidence level vs. number of mines detected in the two passes is shown below: