CDM TECHNICAL REPORT: CDM-13-00

SEAWAY: A Multi-Agent Decision-Support System for Naval Expeditionary Logistic Operations

Anthony Wood
Kym Jason Pohl
Justin Crawford
Mark Lai
John Fanshier
Ken Cudworth
Tom Tate
Hisham Assal
Shawn Pan

CDM Technologies Inc.
2975 McMillan Avenue (Suite 272), San Luis Obispo, California 93401

and

Jens Pohl
Collaborative Agent Design Research Center
California Polytechnic State University, San Luis Obispo, California 93407

Abstract

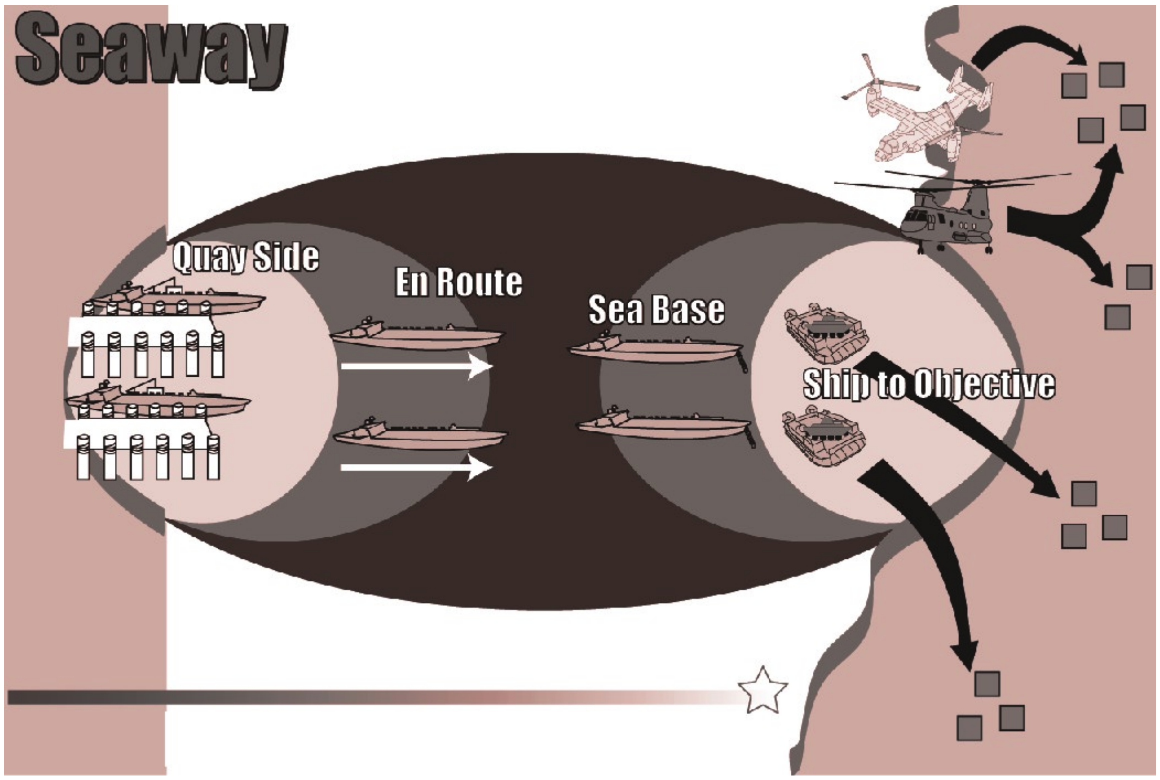
This report describes work performed by CDM Technologies Inc. in conjunction with the Collaborative Agent Design (CAD) Research Center of California Polytechnic State University (Cal Poly), San Luis Obispo, for the Office of Naval Research (ONR), on the SEAWAY experimental system for planning, gaming and executing maritime logistic operations from a sea base. SEAWAY incorporates three fundamental concepts that distinguish it from existing (i.e., legacy) command and control applications. First, it is a collaborative system in which computer-based agents assist human operators by monitoring, analyzing and reasoning about events in near real-time. Second, SEAWAY includes an ontological model of the sea base that represents the behavioral characteristics and relationships among real world entities such as sea base ships, in-bound supply ships, supplies and equipment, infrastructure objects (terrain, intermediate embarkation ports, supply points, roads, and rivers), and abstract notions. This object model provides the essential common language that binds all SEAWAY components into an integrated and adaptive decision-support system. Third, SEAWAY provides no ready made solutions that may not be applicable to the problems that will occur in the real

world. Instead, the agents represent a powerful set of tools that together with the human operators can adjust themselves to the problem situations that cannot be predicted in advance. In this respect, SEAWAY is an adaptive logistic command and control system that supports planning, execution and training functions concurrently.

SEAWAY is an experimental maritime logistic decision-support system that is intended to provide near real-time adaptive command and control in sustaining joint forces from the sea during contingencies. It is based on satisfying the dynamic requirements of joint forces operating ashore, with the ability to provide: offload planning and dynamic re-planning; visibility on all items en route by sea and warehoused at the sea base; track and respond to the dynamic logistic support requirements cycle originating with the supported force ashore; coordinate and control the ship-to-shore ship-to-objective, and ship-to-unit delivery of supplies ashore through a near real-time transport composite operational picture; track supplies and execute reorder; and, provide a full range of warehousing and cargo churning functions aboard the ships of the sea base.

Acknowledgements

The work on the SEAWAY project described in this report is being sponsored by the Logistics Program Office of the Office of Naval Research (ONR), Washington (DC).



SEAWAY within a distributed sea-basing environment.

SEAWAY: A Multi-Agent Decision-Support System for Naval Expeditionary Logistic Operations from a Sea Base

Table of Contents

1. Sea-Basing Concepts and Operations

1.1 Sea-Based Logistics

The concept of sea-based logistics deals with the operational and tactical sustainment of forces operating on and from the sea. The fundamental premise of this concept is the support of forces ashore in foreign countries from a sea base of ships that is independent of political and/or military constraints. The forces are essentially naval in character, but organized and equipped to operate as an integral part of a Joint Task Force (JTF). To enable the vision of Operational Maneuver From The Sea (OMFTS) and Ship To Objective Maneuver (STOM), sea-basing must sustain distributed and maneuvering forces 10 or more miles inland without the need for logistic footprints on shore.

Four notions provide the framework for the employment of US naval expeditionary forces in the future. First, Navy planners continue to evaluate various approaches to extended sea-basing, from the use of L-class shipping as a contingency sea base to the use of a Mobile Offshore Base (MOB). Between these extremes there are serious issues for consideration, such as near real-time cargo tracking within the hull, theater cargo visibility, re-supply of the sea base, seaward security, coordination of near shore and sea-based activities, and modernization of the Command and Control (C2) tools that are available for forcible entry from the sea.

Marine and Army planners continue to conceptualize approaches that leverage the operational and tactical advantages that the sea provides to naval expeditionary forces. Like the sea-basing discussion, the OMFTS vision presents significant challenges including a heavy reliance on the helicopter umbilical linking the supported force and a sea base supplying a maneuver force accompanied by very light service support units. In theory, this situation demands cargo delivery directly to units at the right time, in the right location, in the right quantity, and packaged for immediate use (i.e., no containers and no dumps).

Maritime Prepositioning Forces (MPF) provide the third concept affecting the future of naval expeditionary forces. While MPF offers significant advances in terms of placing the heavy lift forward in the theater, its container-centered approach does not easily align itself with sea-basing. The use of an Intermediate Staging and Embarkation Port (ISEP) provides one interim solution. An ISEP near the operating area allows for the discharging and repackaging of MPF cargo for stowing onboard the sea base.

Precision logistics provides the fourth concept affecting the future of naval expeditionary warfare and sea-basing. Precision logistics ensures transparent visibility of all theater stocks of supplies and equipment. These logistic plans must establish a timely and predictable flow of supplies and equipment tailored to the dynamic requirements of the supported force. At the same time, the sea base must have the ability to compensate for substantial changes in the situation ashore. Finally, it must provide a base for the delivering helicopters, air cushion landing craft (LCAC), and other transport craft.

The final concept is the requirement to provide for logistic continuity of operations from the offshore (i.e., sea-basing) phase through the near shore (i.e., JLOTS) phase.

US military forces have never operated on a sustained basis from a sea base nor have they attempted to support an agile maneuvering force operating 50 miles inland using a helicopter umbilical. This agility requires the organization of a force that is largely without accompanying service support units. In other words, a force that will need to receive supplies at the right place, at the right time, in the right quantities, and packaged for immediate use. Clearly, the implications of these concepts and notions are potentially critical, and yet only partially understood. How will a MAGTF commander identify the relative opportunity cost of increasing the share of helicopters (e.g., V-22s) devoted to logistics and decreasing that available for assault lift? How will the PHIBGRU sea base commander know if he can actually execute the required logistic missions from the sea base within the time window prescribed by the MAGTF? These are just some of the questions that require further exploration and answers. The SEAWAY application that is the subject of this report is intended to serve the multiple roles of providing a gaming environment for concept exploration and requirements determination, a set of intelligent planning tools, and a collaborative decision-support system for operational execution.

1.1.1 Formation of the Sea Base

The following typical scenario provides some insight into the general circumstances that surround the formation and operation of a sea base.

On short warning political and social unrest degenerates into open civil war in the island archipelago of Nesia. Elements loyal to rebel forces see a key strait that is vital to ocean shipping and strategic response. Following a Joint Chiefs of Staff (JCS) alert, the regional ClNC directs that a forward deployed Amphibious Ready Group (ARG) with an embarked Marine Expeditionary Unit (MEU) of 2000 Marines, and 15 days of embarked sustainability be committed to reopen the strait. The ARG, comprising three amphibious assault ships (i.e., LHD, LSD, and LST), diverts as directed. The CINC also directs that a Carrier Battle Group (CVBG), consisting of one aircraft carrier and its escorts, be alerted for support if needed

Shortly thereafter, reconnaissance reports as well as situation reports (SlTREPS) from the landed MEU make it clear that mission requirements will greatly exceed the capabilities of this small Marine Air Ground Task Force (MAGTF) that is built around one reinforced infantry battalion (BLT) and a composite helicopter squadron. In fact, the ClNC now decides that a much larger MAGTF may well have to be committed for up to six months. The force will have to employ Operational Maneuver From The Sea (OMFTS), conducting operations in deep inland locations as much as 25 miles from the coast on six or seven separate islands flanking the strait. Finally, the dynamically changing conditions ashore preclude establishing a large permanent logistic footprint with traditional dumps. The result is a decision to form and employ a mobile sea base to support the MAGTF.

A Joint Task Force (JTF) Commander is dispatched by the regional ClNC to assume command of the force already operating ashore (i.e., the landed MEU) as well as planned reinforcements. In this case the regional Marine Force (MARFOR) provides a BGen and small nucleus JTF staff. The newly formed JTF is then reinforced with a Marine regimental headquarters (MARFOR), as well as an air alert force consisting of a reinforced infantry battalion with light weaponry (but very limited sustainability). The JTF is further strengthened with the arrival of an additional ARG with its embarked MEU. Additional rotary wing assets ferry to the operating area to provide tactical and operational mobility to the growing MAGTF. At the same time additional reinforcing of the Combat Service Support (CSS) and other combat elements arrive by strategic airlift at an Intermediate Staging and Embarkation Port (ISEP). At the ISEP they are transferred to V-22 helicopters and are lifted several hundred miles to the sea base. With the arrival of all these forces the MAGTF now consists of about 7000 Marines sustained from a mobile offshore sea base. The nucleus of that base consists of three LHD/LHA ships, three LST ships, and three LSD ships. The CVBG, operating in open waters outside the strait, provides sea control for the operation.

Since the amphibious ships composing the sea base are designed to project a force and then sustain it on a foreign (and even hostile) shore, they provide great flexibility in accessing, packaging, and delivering cargo ashore. They are also designed to provide the base support and maintenance facilities for the waterborne surface craft and air cushion landing craft (LCAC), as well as the airborne rotary wing assets (V-22) that will provide both troop mobility ashore and cargo delivery from the sea base to the scattered MAGTF elements ashore.

However, the access flexibility that must be maintained on the ships comes with the price of limited cargo quantity. In effect, the amphibious ships are stowed to facilitate offloading. In other words they are stowed inefficiently in order to allow quick access to any cargo item that is needed. Moreover, most of these ships were loaded to support a MEU for a limited six month deployment. Accordingly, 15 days of sustainability for a much smaller force is all that is initially available in the sea base. The stocks of supplies and equipment needed to support a much larger MAGTF engaged in OMFTS will be both far greater and very different than the stocks embarked initially to sustain a MEU. Because of these and other factors, there is a requirement to provide a steady stream of supplies to the sea base to support planned operations. The sea base can then capitalize on the flexible cargo transfer and shifting capabilities of the amphibious ships to access, prepare, load, deliver, and back-haul supplies and equipment responsively to meet the changing requirements of the force ashore.

The stream of supplies and equipment needed to supply the sea base involves the fleet's Combat Logistic Force (CLF) at sea, Maritime Prepositioning Force (MPF), crane ships (TAC), and merchant shipping under Military Traffic Management Command (MTMC) control. The CLF, already forward at sea, is likely to be the first source of supply to the sea base. MPF shipping will sail from its forward locations in the theater. After conducting an offload of vehicles at a close inshore location, the MPF shipping will proceed to the ISEP where selected supplies and equipment can be offloaded. As appropriate, the cargo offloaded from the MPF shipping at the ISEP can be loaded to one or more of the amphibious ships which will then return to the sea base. Finally, selected MTMC ships will be loaded with other supplies and equipment requested by the JTF commander at Continental US (CONUS) ports or from forward locations such Okinawa. On arrival in the theater, these ships can be offloaded in two different modes. First, their cargoes can be offloaded at the ISEP in the same fashion as the MPF shipping. The offloaded cargo will then be stowed on an amphibious ship that returns to the sea base to sustain the MAGTF. In the second case, if conditions allow, the MTMC ship can be offloaded by a crane ship directly onto the ships of the sea base in sheltered waters. Both options may be employed.

Depending on the locations of units, delivery ashore from the sea base can be performed by waterborne utility landing craft (LCU), waterborne air cushion landing craft (LCAC), and helicopter (V-22 and CH-53E). If operational conditions permit a very close in-shore (JLOTS) offload (i.e., less than four miles), then causeways may also be used. However, the initial operational situation calls for the MAGTF to employ OMFTS. This is a form of operation in which the bulk of support elements are retained on the sea base located more then 25 miles from the coast with little if any logistic structure established ashore. From a system perspective, this means that cargo will have to be delivered where it is needed, when it is needed, in the quantities required, and packaged so that the receiving unit can make immediate use of it. In practice this means that heavy equipment that cannot be transported by helicopter must be landed at a temporary landing point from which units move inland to the objectives. Once established ashore, the distances involved (i.e., 85 miles or more) and the requirement for the cargo to be usable on delivery mean that most supply missions will consist of palletized cargo delivered by helicopters.

1.1.2 Typical Command Relationships

A brief overview of a typical set of command relationships will serve to illustrate how SEAWAY assists in maritime contingencies. Although this description focuses on Navy and Marine forces, it could also feature a supported force of US Army units.

The Joint Force Commander (JFC) is the common superior for all components of the Joint Task Force (JTF). Those elements are referred to as the Navy Force Component (NAVFOR) and Marine Force Component (MARFOR). Each component is commanded by a senior Navy or Marine officer. The NAVFOR Commander (COMNAVFOR) commands: (1) the Carrier Battle Group (CVBG) which provides overall security to the sea base; (2) the Amphibious Group (PHIBGRU) made up of nine amphibious ships constituting the sea base; and, (3) the Maritime Prepositioning Force (MPF) when it is in the operational area. In turn, the Amphibious Group Commander (COMPHIBGRU) commands the sea base and is considered to be the principal user of SEAWAY.

The COMMARFOR commands the MAGTF which is composed of: (1) the Command Element (CE); (2) the Air Combat Element (ACE); (3) the Ground Combat Element (GCE); and, the Combat Service Support Element (CSSE). Although all four elements of the MAGTF are expected to use SEAWAY to some extent, the CSSE must be considered the primary user.

Once the JFC has issued his guidance and published his ***commander's intent*** to COMNAVFOR and COMMARFOR, each of these subordinate commanders in turn publish their intent. In the case of COMNAVFOR, SEAWAY is most concerned with his intent and guidance to COMPHIBGRU who commands the sea base. COMPHIBGRU will meet with his staff and the MAGTF CSSE commander and his staff (embarked on the sea base) to determine how best to support the guidance he has received. This guidance will focus on the provision of support to the MAGTF for a designated operational period. Concurrently the CSSE commander will have also received the MAGTF commander's intent, and will share that guidance with the COMPHIBGRU. The latter will issue his guidance to his subordinate elements, among which is the Assault Craft Unit (ACU) commander who will provide the air cushion landing craft (LCAC) to support the surface delivery missions. The ACU commander in turn examines the guidance and, using a SEAWAY terminal. provides a statement of LCAC supportability back to the PHIBGRU staff.

Meanwhile, the MAGTF commander has issued his intent. The CSSE commander, in coordination with the PHIBGRU staff, immediately begins translating the proposed scheme of maneuver into a scheme of logistic support (i.e., a Statement of Requirements for supplies and equipment deliveries with specified locations, quantities, delivery windows, packaging, and receiving units, for the period covered by the intent). In this context the CSSE performs at least three roles with respect to SEAWAY: (1) it is the single source for entering a statement of logistic intent into the system; (2) it is the single means by which additional support requests from units ashore are entered into SEAWAY; and, (3) it is the agency that prioritizes all requests. While the CSSE is busy addressing logistic intent, the ACE commander, on receiving his guidance, coordinates with his helicopter commander (embarked on the sea base). The helicopter commander then coordinates with all concerned with regard to his ability to meet the delivery requirements. Since he controls the principal delivery assets, and since these are also the primary mobility assets for the force ashore, this statement of helicopter availability is carefully coordinated with all concerned. In these activities, the helicopter commander and his staff are closely linked to the CSSE staff as well as the PHIBGRU staff. Each of these staff groups can rely on SEAWAY terminals to provide the necessary collaborative and decision-support capabilities.

The Intermediate Support and Embarkation Port (ISEP) is a subordinate element of the sea base and functions as a landlocked ship. The CSSE maintains a subordinate detachment at the ISEP in order to address MAGTF concerns and support issues with the ISEP staff. Again, SEAWAY is expected to serve as the principal coordination tool between this CSSE detachment and the ISEP staff, as well as between the ISEP and the PHIBGRU staff operating the sea base.

All of this coordination comes together in the TACtical LOGistic (TACLOG) coordination center. Here a joint staff of sailors and Marines develops the offload plan. Using SEAWAY, they integrate logistic intent, available inventories, forecast LCAC availability, forecast helicopter availability, the location of receiving friendly forces, and time distance factors. Other factors such as enemy capabilities, terrain concerns, and weather are also factored into the developing schedule of airborne and surface borne delivery missions.

1.2 The SEAWAY Application

To fully understand and realize the issues generated by the sea-basing concepts, there is a need for a software system that will allow the ramifications of various alternative strategies and decisions to be explored. SEAWAY is being designed in phases to provide that environment in the context of the following six primary capabilities:

- First, SEAWAY will provide continuous theater visibility of all inventory in-bound by sea, at the ISEP(s), and stowed on the sea base.

- Second, SEAWAY will allow commanders and their staffs to project the availability of key operational capabilities.

- Third, in response to formal input by the CSSE, SEAWAY will prepare a plan identifying the missions, mission types, mission profiles, and time required to provide the needed logistic support for the proposed maneuver scheme.

- Fourth, SEAWAY will allow staffs to collaboratively examine tradeoffs between changes in time, available transport assets, quantities of supplies, and scheme of maneuver options.

- Fifth, SEAWAY will maintain an approved offload execution plan and track near real-time changes.

- Sixth, SEAWAY will receive data from external feeds, including weather, enemy threats, and available transport assets, and propose delivery routes in collaborative interaction with users.

To accomplish these functions SEAWAY will integrate information in three overlapping spheres of activity as shown in Figure 1.

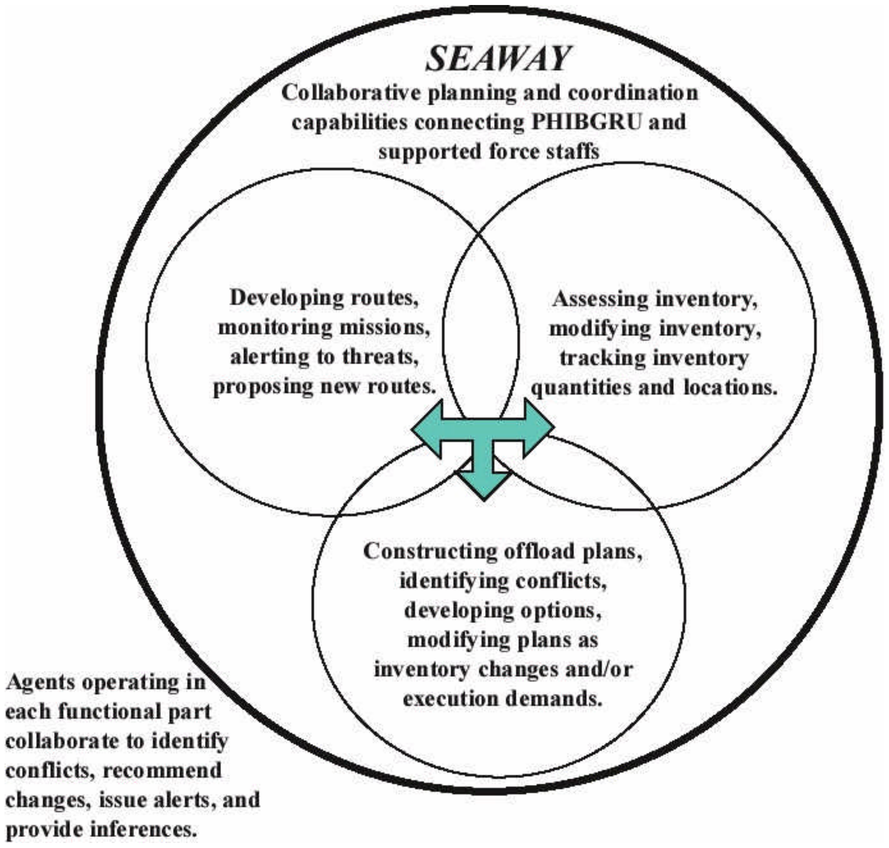

SEAWAY
Collaborative planning and coordination capabilities connecting PHIBGRU and supported force staffs
Developing routes, monitoring missions, alerting to threats, proposing new routes.
Assessing inventory, modifying inventory, tracking inventory quantities and locations.
Constructing offload plans, identifying conflicts, developing options, modifying plans as inventory changes and/or execution demands.
Agents operating in each functional part collaborate to identify conflicts, recommend changes, issue alerts, and provide inferences.

Figure 1: Spheres of sea-basing activities

Staff personnel can use SEAWAY as a modeling or evaluation tool for sea base studies. For example, if SEAWAY is initialized with data describing a typical operational situation and the force being supported, its agents will collaborate and assist users to examine the efficiency of logistic flows in terms of supporting the force ashore. In future phases it is proposed to add capabilities that will allow SEAWAY to also examine the efficiency of various hull options and their impact on the operation of a sea base.

1.2.1 SEAWAY Load Planning Logic

The SEAWAY program objective is to develop the logic and set of intelligent decision tools to be used by the logistician planning and supporting a force from the sea. In this respect SEAWAY begins with the requirements generated by the maneuver force ashore. This first cycle of activities is based on four principal functions performed by the force ashore:

Cycle A	**Direct:**	What? How many? When? To whom? In what package?
	Query:	What? When? In what quantity?
	Plan:	Match logistic and operational capabilities to intent.
	Report:	Operational situation. Threats. Execution options.

The second cycle of activity, referred to as the Delivery Cycle, is performed by the CSSE and involves the following functions:

Cycle B	**Prioritize:**	According to requirements.
	Select:	What is suitable and available.
	Plan:	According to projected offload timelines, execution options, and the tactical situation.
	Offload:	Deliver as planned, confirm delivery, and direct departure.
	Direct:	According to plan and updated tactical, logistical, and meteorological conditions.

The functions of the third or sea-basing cycle are related to operational planning, locating the requested materiel, repositioning, repackaging, and loading onto transport craft.

Cycle C	**Locate:**	Plan access to materiel and timeline.
	Churn:	Move and reposition according to scheduled offload requirements.
	Project:	Project future requirements in respect to plans ashore.
	Reorder:	Submit reorders automatically as materiel is delivered ashore.
	Plan:	Plan for future support needs.

The fourth or Loadout Cycle involves all of the functions related to the movement of materiel to the sea base and ISEP.

Cycle D	**Receive:**	Cargo lists and cargo.
	Sequence:	Order cargo according to ship characteristics and offload needs.

Load:	According to pre-stow plan and marshal sequence.
Record:	Document containers, major equipment, and pallet locations for future churning requirements.
Transmit:	Send stow plans and cargo manifest to sea base.

Primary Functions: The logistics functions to be initially addressed in the first phase of the SEAWAY project are: load planning; warehouse operations; continuous inventory; asset visibility (i.e., on-board ships, within the sea base, and during movement to the shore); and, decision-making (i.e., logistical, and in support of and response to tactical requirements and events). In subsequent phases the maintenance of equipment, vehicles and the sea base at large, housekeeping (i.e., for the sea base and rotating Marine Corps troops), feeding, and other functions will need to be addressed.

Background: The current mission of the LHD Amphibious Assault ship is primarily to deploy to the amphibious objective area, discharge its Marines and their equipment and then sustain them with food, ammunition and spare parts for a limited period of time. The load is typically planned to be discharged in serial fashion of Marines and their associated equipment on a first-on, last-off basis. The employment of the amphibious assault ship in sea-basing operations will require a change in supply and transportation operations. The amount and type of materiel to be loaded and the load plan need to take into consideration a longer time period for the sustainment of troops ashore.

In the context of future maritime contingencies, the load planning process is required to address two sequential components of a mission: the assault phase when the ship first arrives in the objective area; and, the warehousing and distribution phase during the sustainment of the forces ashore. Thus, the plan must be two-faceted. The last-on materiel will be that which is designated for the assault serials (i.e., the first materiel and troops to be discharged). Although the first discharge of Marines and materiel may not be an actual assault this terminology is commonly used and understood by those involved in sea-basing operations.

The materiel designated for the sustainment phase of the operation (i.e., the warehouse materiel) will be loaded next, immediately following the assault materiel. To the extent possible this materiel will be stowed in locations and in a manner that will not require it to be moved to other locations after the assault phase. The objective is to have the sustainment materiel stowed so that the containers (e.g. QUADCONs) can be readily opened and become the storerooms for smaller items such as spare parts (e.g., Class IX material). Pallets and other stored materiel stowed so that they are readily accessible for issue and close to the repackaging and consolidation site.

The sea base ships will also need to be replenished while on station. Functions relating to underway replenishment actions, and subsequent strike down and storage of the materiel, will need to be addressed in future phases of the SEAWAY program. The attendant processes involved in the storage and issuing of materiel, the movement of materiel within the sea base, repackaging and consolidation, and transportation, all need to be supported in the final version of the SEAWAY application.

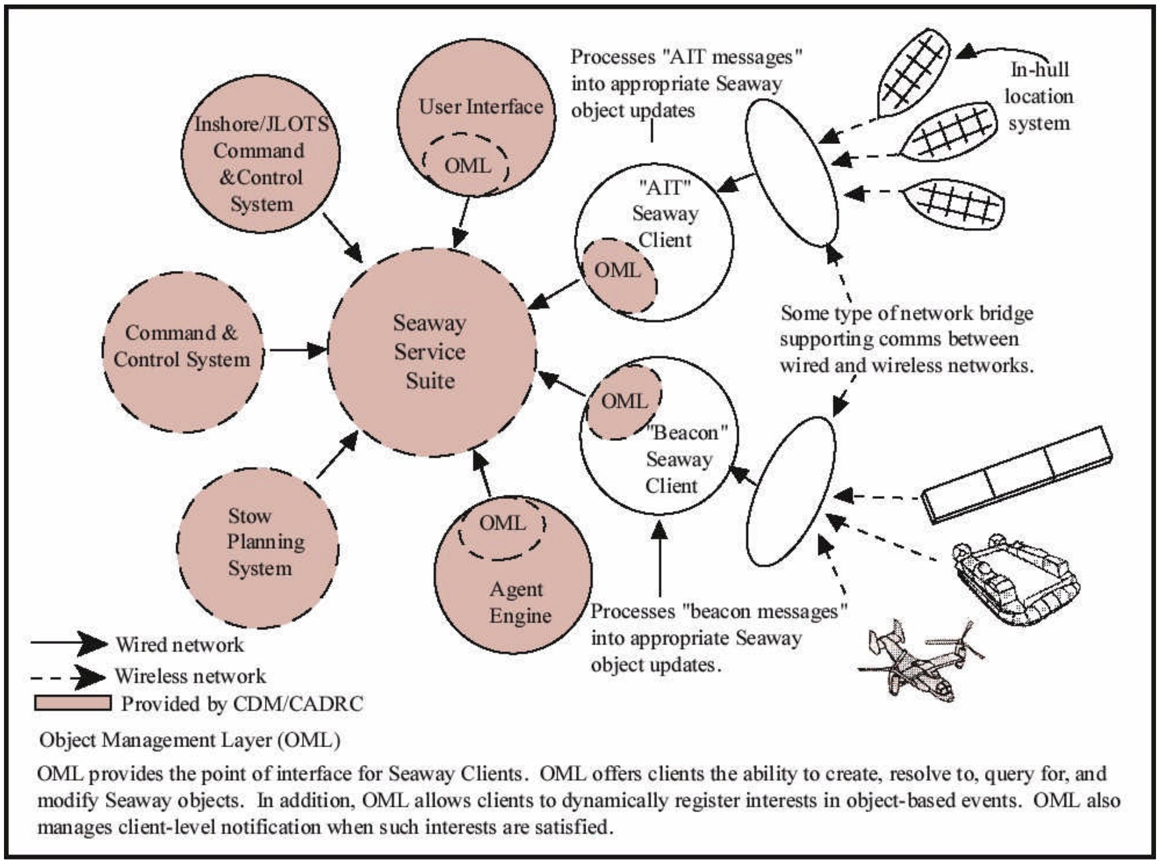
Load Planning for Sea-Basing Operations: The loading of a ship requires many different load planning functions to be performed. The culmination of these functions into an integrated document is the load plan.

> **Embarkation/Offload Plan:** The embarkation plan must support the plan for landing, the scheme of maneuver ashore, and the plan for landing supplies. Personnel, equipment, and supplies must be loaded in such a manner that they can be unloaded at the time and in the sequence required to support the operations ashore. The embarkation plan must provide for the highest possible degree of (military) unit self-sufficiency. When more than one ship is involved the embarkation plans must allow for the distributed loading of critical supplies and people across the ships, while maintaining the objective of integrated sequential debarkation.
>
> **Debarkation Plan:** The order in which the first elements are to be discharged is the most important part of the load plan, because regardless of the type or nature of the cargo, a plan cannot be generated without first identifying the order of discharge or debarkation. Thus, the debarkation plan is a reverse image of the embarkation plan.
>
> **Sea Base Sustainment Plan:** The sea base sustainment plan (i.e., typically the assault follow-on phase) consists of those troops, equipment, and materiel required for supporting the maneuvering forces ashore. It will consider the space required for the storage of materiel, the repackaging and consolidation of materiel, the movement of the materiel to loading points, as well as the transportation to shore. The plan must also consider the need for re-supply of the sea base and the movements of retrograde from the shore back to the sea base.

The load plan must allow for ***access and flow*** considerations. This includes the provision of aisles and areas for accessing the materiel and for movement within the ship to the repackaging or re-transit sites, such as elevators and conveyors. Tracking of the materiel during this phase is important to the continuous inventory and specific location identification requirements of SEAWAY.

Automated Identification Technology (AIT): SEAWAY will need to be able to track the movement of materiel both within ships and during its movement to Supply Points on the shore. Printed or electronic bar code labels (i.e., smart tags) attached to individual cargo items or loaded pallets can be automatically read by electronic scanning devices placed at strategic locations on ships. Data transmitted from these readers directly into SEAWAY will allow SEAWAY to maintain in its inventory the current on-board location of all tagged materiel.

This will require readers to be placed at the entrances and exits of all storerooms and magazines for the purpose of tracking the incoming and outgoing flow of materiel. In addition, readers need to be located in the elevator loading spaces to check materiel coming off and going onto elevators. Finally, readers will be required in the repackaging and loading areas to monitor the movement of materiel from the ship onto a transport craft and vice versa during retrograde. Once a particular cargo item has been automatically associated with a transport craft in this manner, the ability of the craft to beacon its current position allows SEAWAY to track the materiel as it moves from the sea base to the Supply Points ashore (Figure 2).


Processes "AIT messages" into appropriate Seaway object updates
In-hull location system
User Interface
OML
Inshore/JLOTS Command &Control System
"AIT" Seaway Client
OML
Seaway Service Suite
Command & Control System
Some type of network bridge supporting comms between wired and wireless networks.
OML
"Beacon" Seaway Client
Stow Planning System
OML
Agent Engine
Processes "beacon messages" into appropriate Seaway object updates.
Wired network
Wireless network
Provided by CDM/CADRC
Object Management Layer (OML)

OML provides the point of interface for Seaway Clients. OML offers clients the ability to create, resolve to, query for, and modify Seaway objects. In addition, OML allows clients to dynamically register interests in object-based events. OML also manages client-level notification when such interests are satisfied.

Figure 2: Integration of AIT and beaconing systems in SEAWAY

Joint Logistics Over The Shore (JLOTS): SEAWAY will provide for the smooth transition from offshore operations to near shore JLOTS operations. It will insure that the same coordinating and logistic situational awareness tools that were used in the sea-basing phase are then also applied to JLOTS execution, without interruption.

1.2.2 The SEAWAY Development Plan

The SEAWAY application is an ambitious project that embraces a multiplicity of functional requirements in a very complex and time constrained decision making environment. Furthermore, it is being designed to satisfy the requirements of an activity area that is evolving and is neither clearly understood nor comprehensively defined at this time. Therefore, it is expected that several new requirements for the SEAWAY application will be generated during the development and progressive utilization of early versions.

Accordingly, the development of SEAWAY is being undertaken in a phased sequence extending over several years, with each phase focusing on a particular primary utilization objective. For example, the objective of SEAWAY V.0 was to demonstrate the feasibility of applying an ontology-based, multi-agent, collaborative, decision-support approach to the system architecture of the SEAWAY application. The focus objective of SEAWAY V.1 is to provide a real-time, collaborative gaming environment for exploring sea-basing strategies and options. Since the functional capabilities completed during each development phase will be cumulative, the SEAWAY application will progressively support gaming, modeling, planning, and operational logistics command and control in an integrated and concurrent fashion.

The principal objectives, architectural innovations and elements of the multi-year SEAWAY development plan, with emphasis on the first two years, are summarized below.

· SEAWAY is being developed as a naval logistic decision-support system tool with which to evaluate various options for providing logistic support from the sea in maritime contingencies. Because naval expeditionary operations cover such a wide spectrum, SEAWAY is designed to work with any combination of shipping and logistic support concepts. Further, it is designed to support contingency decision making and evaluation for both offshore logistic-support operations (including sea-basing) and near shore (JLOTS) operations.

· The initial two-year development period, culminating with SEAWAY V.1, will be completed in September 2001. This development period is defined by two major demonstrations: a proof-of-concept demonstration (SEAWAY V.0 with basic functionality only) held at the naval Amphibious Base, Coronado in September 2000; and, an initial prototype system demonstration (SEAWAY V.1) to be held in October 2001, SEAWAY V.1 will support a gaming environment for purposes of evaluating the impacts and implications that various logistic-support decisions might exert on Naval forces in future contingencies.

· In a first-of-its-kind display of adaptive naval command and control capability, it is planned that the Fall 2001 gaming demonstration include inter-linking SEAWAY with IMMACCS (or a similar tactical command and control system) and ICODES, two other intelligent agent systems supporting naval expeditionary operations. Consideration is also being given to including SEAWAY as a component in limited objective experiments during 2001.

· The SEAWAY system architecture employs a technology commonly referred to as Artificial Intelligence (AI). In reality an object model of the maritime logistic environment is created and then constantly expanded and refined with information feeds from multiple sources, as operations proceed ashore and afloat. Expert, computer-based agents with specific tasks reason about the information inside this model of maritime logistic reality, make real-time recommendations, and continuously develop inferences for the sea base and MAGTF staffs. These agents can be thought of as a kind of tireless 'shadow staff' that acts as an assistant supporting planning and execution by the active joint and/or service staffs. Since SEAWAY is a distributed system with nodes that can be distributed ashore and afloat, it enables simultaneous collaborative planning between and among both the 'purple', 'blue' and 'green' teams.

· SEAWAY is really a tool kit designed to flexibly adjust to the demands of any maritime logistic situation. For example, among the most important of its potential contributions to the refinement of emerging joint and service concepts is the capability to portray a supported force/MAGTF scheme of maneuver and associated scheme of logistic support as an offload plan. Among other capabilities, this offload and support plan highlights the time needed for implementation and presents the sequence of helicopter, fixed wing, surface craft (e.g., LCAC), and other missions required for execution. The system can identify tradeoffs among key factors such as the time and/or the lift assets that must be devoted to logistics in order to execute a particular scheme of maneuver. It is anticipated that the prototype (i.e., SEAWAY V.1) will also provide an early capability to identify the cost of periodic changes to a basic offload support plan in terms of delays in realizing key capabilities ashore and delays in the receipt of vital equipment.

· The initial SEAWAY V.0 proof-of-concept demonstration in August 2000 utilized a scenario reflecting the operational capabilities that are actually projected for the next five years. While both the Maritime Prepositioning Force – Future (MPF-F) and the Mobile Offshore Base (MOB) are future options under discussion, the only option for the next five years which possesses the necessary selective cargo access, internal cargo churning capabilities, and air/surface offload capabilities required for logistic support from the sea is amphibious assault shipping. Buttressing this central role for amphibious shipping is the strong likelihood that any 'sea base' formed in the next five years is likely to be a 'contingent sea base' (i.e., a sea base that begins with a forward deployed ARG/MEU as its nucleus and subsequently expands to meet the requirement to support a larger force). Finally, during this interim period the capability to transfer containerized cargo directly from merchant ships to amphibious assault shipping at sea, is unlikely to be realized. Accordingly, the initial demonstration of SEAWAY V.0 in August of 2000 employed amphibious assault shipping in a scenario that included a nearby (e.g., 200 nautical miles) Intermediate Staging and Embarkation Port (ISEP). For purposes of the demonstration scenario, containerized cargo was offloaded and unpacked at the ISEP and then shuttled to the amphibious shipping providing logistic support to the force ashore.

· The August 2000 (SEAWAY V.0) proof-of-concept demonstration was designed to demonstrate the basic functionality of the underlying expert agent technology. It was also aimed at producing recommendations for shaping prototype system capabilities for the next development phase scheduled to be completed in September 2001. Accordingly, the demonstration used a scenario that begins at a point after the force has been established ashore using MPF and forward afloat forces. Selected functions from several capability categories were featured in this initial demonstration of basic SEAWAY functionality. A general description of each category follows.

1. The system demonstrated the basic capability to provide <u>useful</u> theater-wide cargo visibility. The goal here was theater-wide cargo transparency that provides timely support to planners and executors. This category involves a set of functions in which expert agents manipulate information on cargo en route, cargo on the amphibious ships offshore, and cargo debarked and waiting at the ISEP. In this role the agents are capable of performing functions such as: provide projections of when items are likely to arrive; responding to planner queries for materiel totals by item and by location; track totals distributed to the landing force; and, provide visibility of the items embarked on transport craft making deliveries to the supported force ashore.

2. The system demonstrated the basic capability to identify the accurate location of cargo within the holds of the amphibious ships making up the sea base, as well as the location of cargo items stowed ashore at the ISEP. Based on the current information held in SEAWAY the agents provide up-to-date reports and project timelines to achieve a 'ready for offload' status on the flight deck or on the well deck of the LHDs.

3. The system demonstrated the basic capability to develop and continuously update an offload plan (i.e., 'Future Plan') that accurately reflects the changing demands of the supported force engaged in operations ashore. The formal planning component within SEAWAY is initiated by receipt of a 'Statement of Requirements' entered by the CSSE. Subsequently, the appropriate commanders examine the automatically generated 'Future Plan'. The approved portions of the 'Future Plan' become the current 'Execution Plan'. On a continuous basis, agents will insert new missions into the 'Future Plan', identify impacts, monitor critical items identified as such by the JTF

commander, and develop alternative 'Future Plans' in response to planning ashore.

4. Using five separate stations the system demonstrated the basic capability for collaborative and distributed planning and coordination. This arrangement was designed to permit both the sea base staff, the staff of the supported force, and any other staffs to use SEAWAY as their 'assistant' in reaching decisions for logistically supporting the joint force. To satisfy this requirement, SEAWAY is being developed as a distributed system whose screens and agents will be widely available ashore and afloat. Further, the system includes a package of collaborative tools with visual, graphic, and audio capabilities. As the contingency shifts focus, these tools as well as system capabilities to maintain logistic situation awareness will allow SEAWAY to support a smooth logistic C2 transition from offshore support operations employing amphibious shipping, to near shore Joint Logistics Over The Shore (JLOTS) operations.

· Following the September 2000 demonstration of SEAWAY V.0, the Office of Naval Research (ONR) sponsor of the project authorized the development of a proof-of-concept logistic planning module (i.e., LOGGY) as an addition to SEAWAY. LOGGY is intended to assist the CSSE commander and his staff to rapidly evaluate and translate schemes of maneuver into schemes of logistic support, and then assess feasibility. In this role LOGGY will produce the supported force 'Statement of Logistic Requirements' which is the trigger initiating formal planning within SEAWAY. With the addition of LOGGY, the SEAWAY tool kit will contain tools tailored for the joint task force headquarters, the supporting Navy force, and the supported force ashore.

· A logical progression for further development, beyond SEAWAY V.1 in September 2001, would involve four one-year phases culminating in field testing of SEAWAY as a maritime logistic Command and Control (C2), decision-support system. The advantage of this phased program is that it permits SEAWAY to be used for concept exploration and acquisition decision making almost immediately.

Phase (1): ***<u>SEAWAY V.0 completed in September 2000</u>***
Demonstrated the feasibility of applying an ontology-based, multi-agent, collaborative, decision-support approach to the system architecture of the SEAWAY application.

Phase (2): ***<u>SEAWAY V.1 to be completed September 2001</u>***
Will field the prototype as a gaming tool for concept exploration and assessment of what level of sea-based

logistic support is feasible under differing circumstances. This phase is the beginning of a process that would use SEAWAY as a modeling and gaming tool to introduce metrics to the discussion of what really is required to execute sea basing and Objective Maneuver From The Sea (OMFTS).

Phase (3): ***SEAWAY V.2 to be completed September 2002***
This phase will add the modeling capability within the prototype to permit evaluation of various alternative ship options and combinations of ships in the support role. It would be combined with the extant gaming capability to introduce a flexible and powerful tool into acquisition discussions.

Phase (4): ***SEAWAY V.3 to be completed September 2003***
This phase will begin to address a second set of required capabilities, namely adaptive logistic decision support, and builds directly on the foundation provided by the gaming and modeling phases. During this phase it is proposed to test SEAWAY with the operating forces in limited experiments and exercises as a near real-time adaptive logistic C2 capability which can be flexibly tailored to meet future contingencies.

Phase (5): ***SEAWAY V.4 to be completed September 2004***
This phase will further develop the second set of required capabilities to support programmatic and investment decisions. Long lead times are required for ship acquisition and/or modification as well as in other major capital investment programs. To date a host of sea basing alternatives has been raised. However, it has not been possible to answer numerous queries without being able to subject proposed options to multiple aspect gaming and modeling. Such a modeling capability is urgently needed to aid in identifying the most effective assets to support future sea-based joint/naval expeditionary operations. Phase (5) would add this vital modeling capability to SEAWAY.

Phase (6): ***SEAWAY V.5 to be completed September 2005***
This phase will build on the existing SEAWAY V.4 capabilities to provide a prototype comprehensive, adaptive, agent-based logistic C2 system. Joint spokesmen have characterized the Timor humanitarian contingency as a harbinger of similar contingencies that

likely lie ahead. It was largely sea based. It was a coalition operation. It involved a series of quite unexpected logistic C2 demands on the friendly force and its afloat base. And it changed its form rapidly, often with little warning. Fortunately, Australia was close at hand to assist in offsetting the operational and logistic impacts. However, we cannot plan future naval expeditionary operations based on such an uniquely favorable situation. As USCINPAC spokesmen have emphasized, there is an urgent need to provide a flexible logistic C2 capability for joint and coalition operations like Timor. This near-real time logistic C2 capability must enable users to transparently view the theater logistic situation and to continuously evaluate planning and execution alternatives in relationship to the current and projected logistic picture. It must provide automated tools to assist in effectively allocating and re-allocating scarce resources. In addition, it must effectively respond to a wide range of missions from assault to humanitarian contingencies. Finally, because we cannot see the future, this adaptive logistic capability must provide a seamless logistic C2 capability from offshore sea-based operations through near shore JLOTS operations.

2. Technical Concepts and Overview

SEAWAY has been implemented as a three-tier architecture that distinguishes between information, logic and presentation. It utilizes an object-serving collaboration facility that is based on the Common Object Request Broker Architecture (CORBA) (Mowbray and Zahavi 1995). Specifically, SEAWAY is an adaptive, distributed, open architecture system that incorporates four notions that are fundamental to its decision-assistance capabilities.

Notion 1: Whereas legacy systems typically process data, SEAWAY processes information. The key to the assistance capabilities of SEAWAY is that the system has some 'understanding' of the information that it is processing. In SEAWAY every entity in the screen display of the sea base and battlefield (e.g., ship, road, building, truck, tank, enemy unit, etc.) as well as intangible entities such as weather, enemy threat, and so on, are represented as individual objects with characteristics and relationships to each other. Therefore, the operator interacts with a computer display that consists of hundreds of real world entities (objects) that all have some 'understanding' of each other's nature, interests and objectives, and a great deal of 'understanding' of their own characteristics and capabilities.

Notion 2: SEAWAY is a collection of powerful collaborative tools, not a library of predefined solutions. This approach is intended to overcome the deficiencies of legacy systems in which built-in solutions to predetermined problems often differ significantly from the complex operational situations encountered in the real world. SEAWAY is a collaborative decision-support system in which the operators interact with computer-based agents (i.e., decision making tools) to solve problems that cannot be precisely nor easily predetermined.

Notion 3: SEAWAY incorporates agents that are able to reason about the characteristics and the relationships of the many real world entities (objects), all of which embody meaning. In its first proof-of-concept demonstration (Sep.21, 2000) SEAWAY included agents that: apply their domain specific knowledge to tracking supplies; monitoring supply shortages; assessing the impact of weather conditions on sea-basing operations; assisting in the preparation of future plans; scheduling missions and sorties; and, initiating reach-back capabilities.

Notion 4: SEAWAY integrates planning, execution and training within one common command and control user environment. The computer-based agents and the SEAWAY users continuously collaborate as they interact with each other in rapidly changing sea-basing situations. In this respect SEAWAY reflects the complexity of the real world where problem solutions must be continuously reviewed as conditions change, and it becomes increasingly difficult and inconvenient to separate planning, re-planning, execution, and training functions into artificially discrete activities.

SEAWAY is one integrated system and not a confederation of loosely linked sub-systems. It consists of the following principal components:

- An Object Model (i.e., ontology) that facilitates the internal representation of information (rather than data). In particular, SEAWAY supports the dynamic formation of associations among objects at both the user and agent levels.

- An Agent Engine that supports the interaction of multiple computer-based agents in a collaborative human-computer decision making environment.

- An object-serving collaboration facility that manages the object-based interactions among the various components on a subscription basis. All SEAWAY components are clients of the object-serving collaboration facility and indicate their information interests by registering a subscription profile. Whenever, information that is within the subscription of one or more clients becomes available the object-serving collaboration facility automatically pushes this information into a cache memory area and sends an alert message to the client. In addition, clients may also query for information for which they have not subscribed.

- A hardware independent SEAWAY Object Browser (SOM) user-interface that facilitates operator interaction within the object-based information context and the collaborative agent assistance capabilities of SEAWAY. Through the Object Browser the operator may: enter requirements, transportation asset availability and environmental conditions (e.g., sea state); initiate the agent-assisted preparation of future plans; generate reports; and, gain access to the most detailed level of information about specific supply items..

- A Translator that is capable of mapping data received from external applications, such as the Integrated Computerized Deployment System (ICODES), to the object-based representation held within the SEAWAY Object Model (CADRC 1994, Pohl et al. 1997 (85-103)).

- A direct linkage through a shared ontology to an agent-based tactical command and control system (i.e., FCADS (Framework for Collaborative Agent Decision-Support)) that alerts SEAWAY to changes in battlefield conditions and enemy threat situations that may impact sea-basing operations.

Utilizing the ICDM (Integrated Cooperative Decision Making) framework (Myers and Pohl 1994, Pohl 1997), the SEAWAY application is based on a three-tier architecture that makes clear distinctions between information, logic, and presentation. These tiers are represented by the three major SEAWAY system components; namely: the object-serving collaboration facility (information tier), the Agent Engine (logic tier), and the SEAWAY Object Browser (presentation tier) (SOB in Figure 3). Included in the information tier is a translator providing information translation between SEAWAY and ICODES. Each of these tiers functions in an integrated fashion to form a comprehensive agent-based decision-support execution framework. This framework allows multiple human decision makers to solve complex problems in a collaborative fashion obtaining decision-support assistance from a collection of heterogeneous on-line agents.

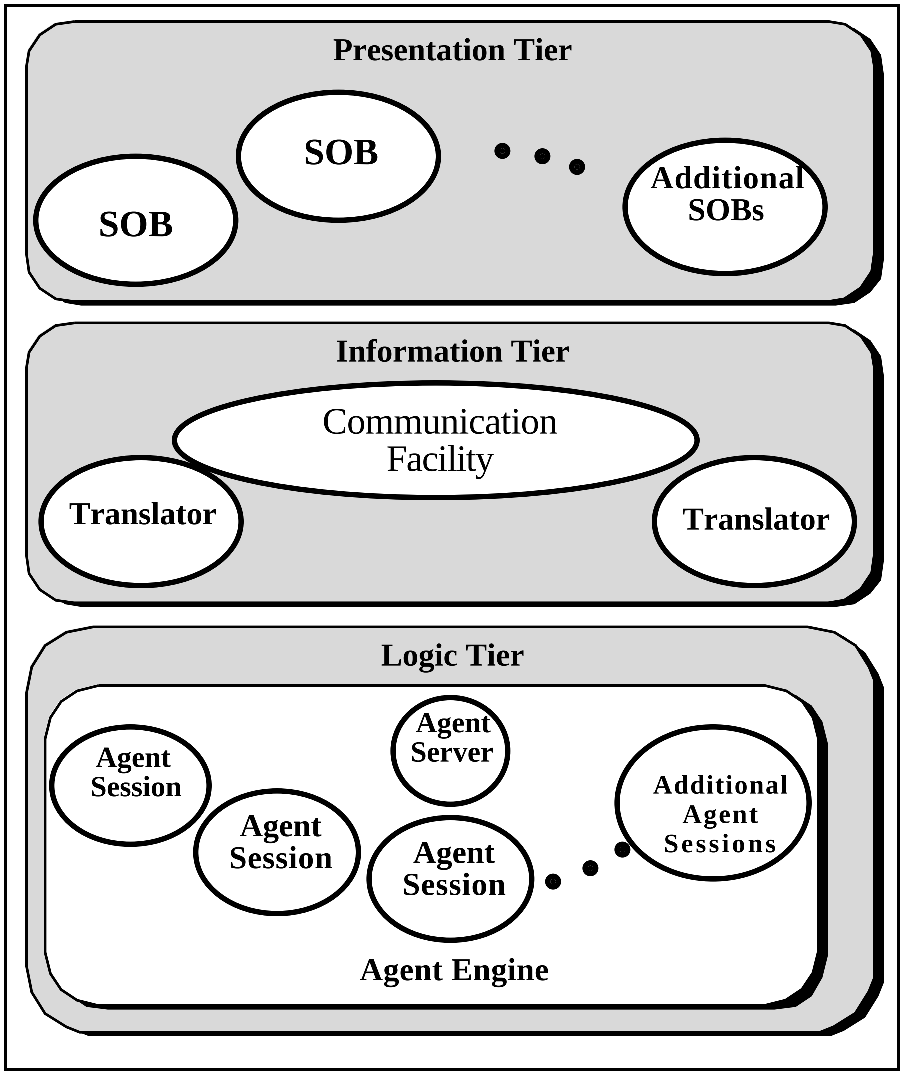


Figure 3: The three-tier architecture of SEAWAY

2.1 The Object-Serving Collaboration Facility

Conceptually, the collaboration facility represents a library of objectified information that clients utilize to both obtain and contribute information. The only difference is that clients can obtain this information in, not only a pull fashion, but can also have the collaboration facility push them information on a subscription basis. Physically, the collaboration facility exists as an enhanced distributed object server based on the Common Object Request Broker Architecture (CORBA) specification.

As the basis of the object-serving collaboration facility, distributed object servers are designed to service client requests for information. The knowledge of exactly where the information resides and how it can be retrieved is completely encapsulated inside the object server. This means that clients need not be concerned with which client has what information and in what form that information exists. This feature becomes instrumental in providing an environment where collaborative application components inter-operate in a de-coupled fashion.

Regardless of the native representation of the information, distributed object servers can be used to present information to clients in the form of objects. However, this does not discount the need for information to be modeled as high-level objects in their native form portraying behavior and conveying relationships. While on the surface this representational morphing capability of object servers seems promising, in practice this feature proves to be quite misleading. If the information is not represented at a high level upon its conception, such objectification amounts to little more than wrapping data in communicable object shells. These shells fail to convey any additional insight into the meaning or implication of the information than was present to begin with in its original form. Although in the future there may be potential for successful research efforts in this area, at present, unless information is originally modeled as objects, knowledge-oriented applications prove to gain little from the distributed object server feature.

However, applications such as SEAWAY that do model information as high-level objects stand to gain considerably from employing distributed object servers. Distributed object servers preserve a purely objectified representation of information as it moves throughout the system. This is due to the fact that the internal mechanisms of distributed object servers process information as objects themselves.

2.2 The Agent Engine

The Agent Engine represents the logic-tier of the underlying three-tier architecture of SEAWAY. Existing as a client of the object-serving collaboration facility the Agent Engine is capable of both obtaining and injecting information into the communication facility. Architecturally, the Agent Engine consists of an agent server capable of serving collections of agents (Figure 3). These collections, or Agent Sessions, exist as self-contained, self-managing agent communities capable of interacting with the communication facility to both acquire and inject information. As a client possessing and registering interests in events and information, agent activity is triggered by changes in the sea base environment. Regardless of whether agents are interacting with the collaboration facility or each other, interaction takes place in terms of objects. This again illustrates the degree to which an object representation is preserved as information is processed throughout SEAWAY.

2.3 The Client User Interface

Representing the third and final tier of the three-tier architecture employed by SEAWAY the Client User Interface exists as a culmination of instances of the SEAWAY Object Browser. Collectively, these interfaces provide human users with a means of viewing and manipulating the information and analysis provided by the other two tiers of the SEAWAY decision-support system. Understanding the importance of data presentation, the Client User Interface presents the user with this information and analysis in a robust and graphical manner.

As clients of the object-serving collaboration facility, users have the ability to interact with each other in a collaborative fashion. Specifically, by either injecting or obtaining information from the communication facility, users working on the same view have the potential of exchanging strategic or other kinds of information in a collaborative manner. All information and analysis remains localized within a particular view unless explicitly copied into another view through user interaction.

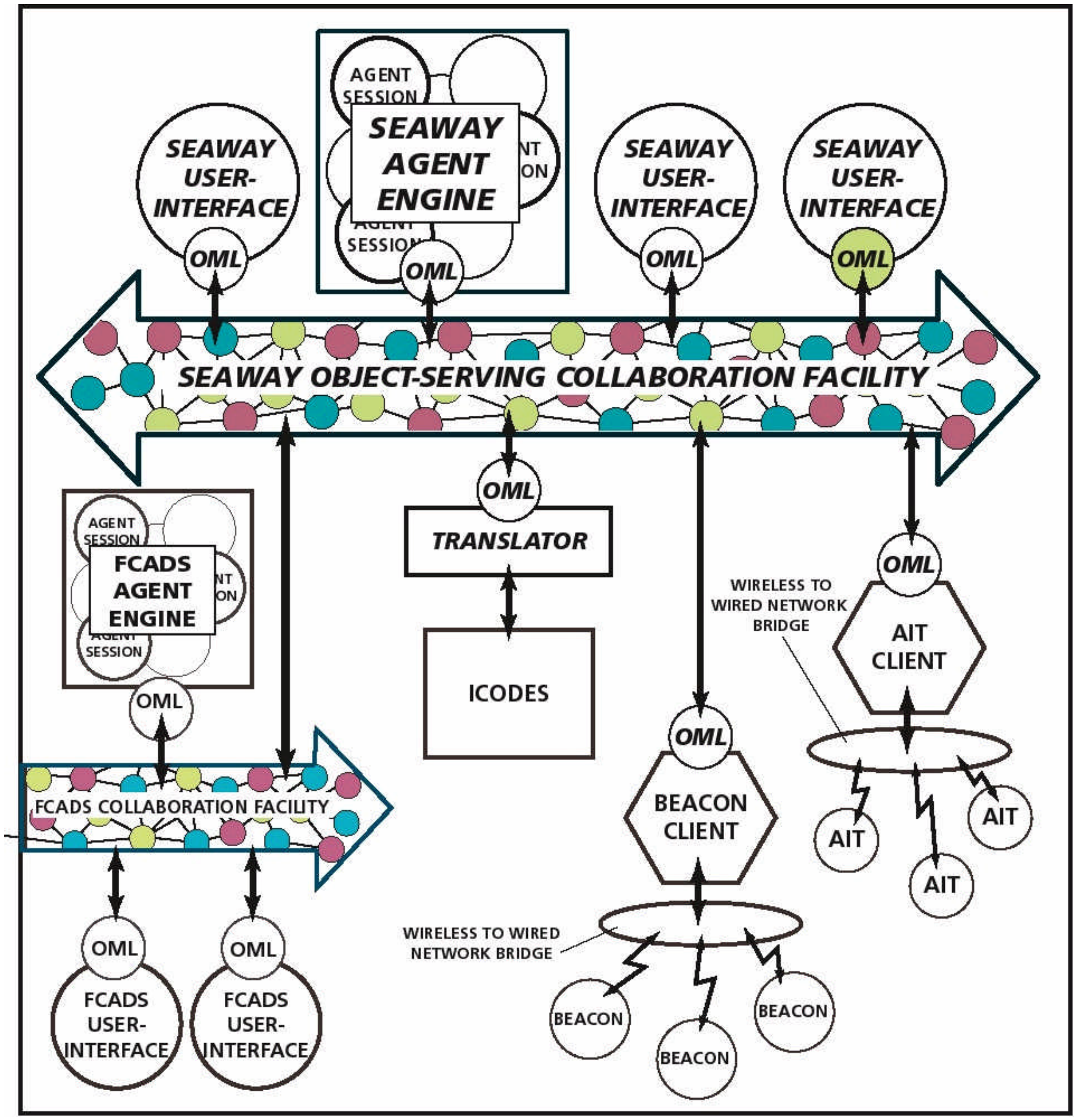
2.4 The SEAWAY Architecture

Based on the ICDM toolkit and framework the SEAWAY architecture is divided into two distinct components, a set of services and a set of clients who utilize these services. The services essentially manage the creation, modification, persistence, access, and change notification of shareable, ontology-based objects. Object creation and modification is managed by an Object Server, while persistence of these objects is provided by a Persistence Server.

For the proof-of-concept system (SEAWAY V.0, Sep.21, 2000) a simple serialized approach was taken. Access is provided through two mechanisms. Access for named objects is provided through the Object Server in conjunction with a Naming Service. All other types of access are provided via a constraint-based Query Server, and object change notification is provided to clients via a Subscription Server. Clients indicate their changing interests to the Subscription Server which in turn notifies them when such conditions occur. Collectively, these services provide clients with the ability to manipulate and collaborate as the information within the system (i.e., the problem state) changes.

The clients in the SEAWAY system can themselves be divided into two distinct sets. However, the distinctions between these sets come primarily from the application domain. One of the goals of the SEAWAY proof-of-concept system was to demonstrate the ability of applications supporting differing domains to inter-operate in a collaborative fashion. In the case of SEAWAY V.0 the two inter-operating domains are logistical command and control and tactical command and control. With this in mind, clients are divided into two groups, those assisting in logistical operations and those assisting in tactical operations. For each of these domains both a client graphical user interface and an agent engine, housing domain specific agents, are provided. However, both groups of clients share the same ontology and set of services. They simply present and analyze the information in their own domain-specific manner.

Subsequent versions of SEAWAY (e.g., SEAWAY V.1 planned for release in October, 2001) will pursue an architecture that increases the distinction between components supporting differing domains. That is, not only will each domain be supported by its own collection of clients and servers but these collections will operate on their own, potentially differing ontologies. Although operating on potentially different models of the ***world*** inter-domain collaboration will still take place via a domain mapping facility. Through the provision of such capabilities it is intended to allow wholly different domain-specific decision-support applications to inter-operate in an effective, collaborative manner.



Schematic system diagram of the SEAWAY architecture and its linkages to external systems such as ICODES, FCADS, and AIT and beacon data feeds.

3. The ICDM Toolkit and Framework

SEAWAY is being developed as an ICDM-based decision-support application. It is therefore relevant to briefly describe the technological basis of the ICDM toolkit and discuss its contribution to the design and development of agent-based, decision-support systems.

The initial development of the ICDM toolkit was undertaken by the Collaborative Agent Design Research Center (CADRC) at Cal Poly, San Luis Obispo, California, with the objective of providing a formalized architecture together with a set of development and execution tools that could be utilized to design, develop, and execute agent-based, decision-support applications. The core element of this process is the development of an ontological framework to provide a relationship-rich model of the various system and application domains in which the proposed application is required to operate. Based on a three-tier architecture, ICDM incorporates technologies, such as distributed-object servers and inference engines, to provide a framework for collaborative, agent-based decision-support systems that offer developmental efficiency and architectural extensibility.

Throughout the past decade the CADRC, more recently in conjunction with CDM Technologies Inc., has been intensely engaged in the design and development of agent-based, decision-support systems from a decidedly pragmatic standpoint (Pohl et al. 1997). As a result of these efforts, the CADRC and CDM Technologies have developed a manifesto of sorts describing a collection of criteria which we consider to be fundamental to the development of agent-based, decision-support systems (Pohl 1997).

First and foremost on this list of design criteria is the need for an object-based representation of information that will allow software modules (e.g., agents) to reason about events in the problem space. Within the context of a top-down symbolic approach (as distinct from a bottom-up connectionist approach (Minsky 1990)) information processed within the system must be described as objects having attributes, behavior, and most importantly relationships. Collectively, such symbolic descriptions form an application's information object model or ontology (Fowler and Scott 1997). This requirement not only applies to the modeling of information but, at times, is even portrayed in the representation of the agents themselves. Without such an objectified representation that allows critical informational relationships to be captured, determination of the meaning and implication of information becomes extremely difficult, if not impossible.

After several implementations it became clear that to take full advantage of such objectified representation, a supportive framework needed to be established. A framework that centers around objects. To satisfy this need the ICDM framework was developed and continues to evolve as an architecture, together with a set of development and execution tools that can be used to design, implement, and execute agent-based, decision-support applications.

The ICDM model is based on a three-tier architecture making clear and distinct separations between information, logic, and presentation (Gray et al. 1997). These tiers are represented by the three major components comprising the ICDM model: the ***information tier*** consisting of a collection of information management servers (i.e., Information Server, Subscription Server, etc.); the ***logic tier*** represented currently by an Agent Engine; and, the ***presentation tier*** or Client User Interface (Figure 4). Each of these components functions in an integrated fashion to form a comprehensive agent-based decision-support execution framework. This framework allows multiple human decision-makers to solve complex problems in a collaborative fashion obtaining decision-support assistance from a collection of heterogeneous on-line agents.

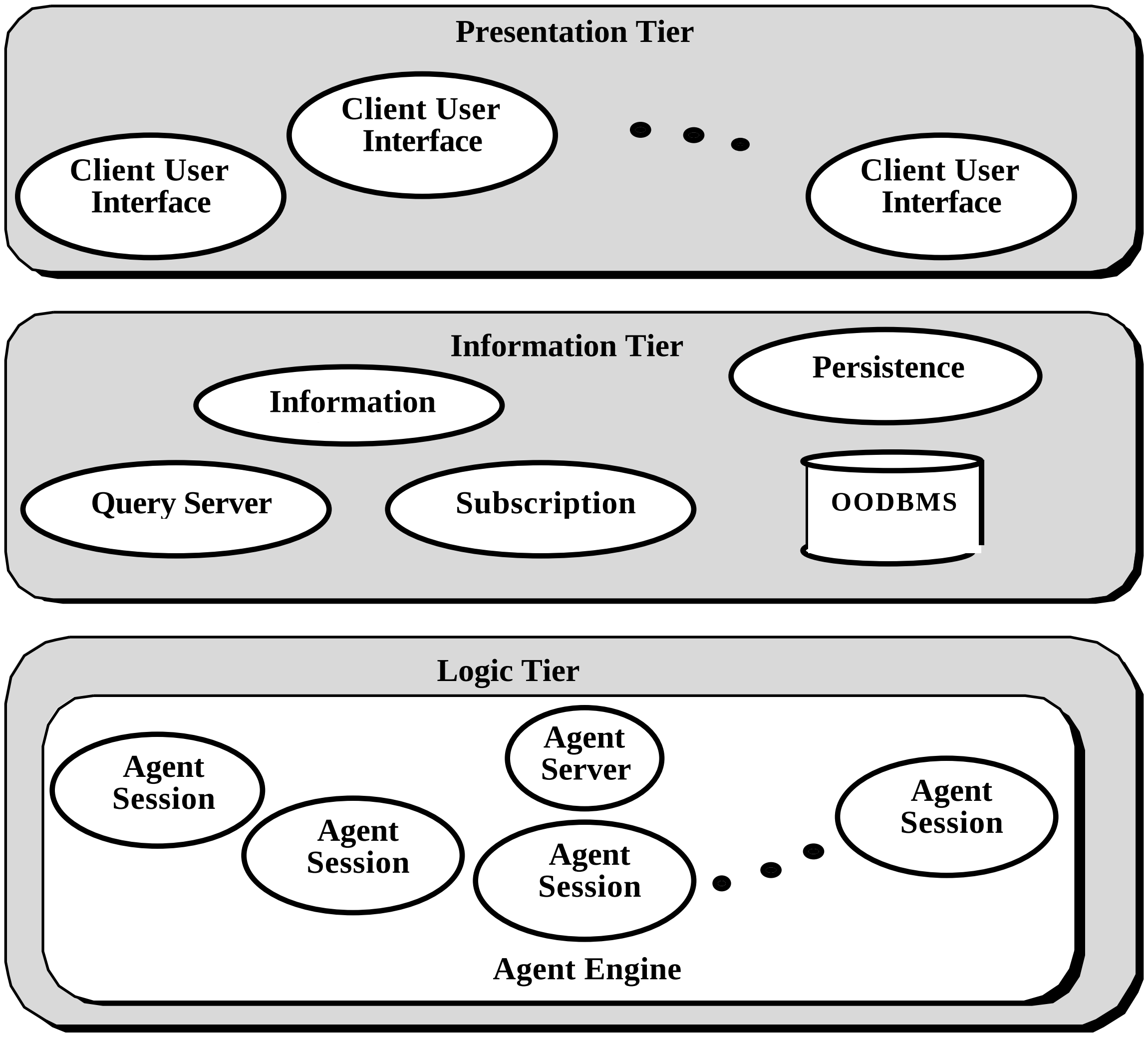


Figure 4: The three-tier architecture of the ICDM toolkit

3.1 Information Server

The core of the ***information tier*** is the Information Server. Conceptually, the Information Server represents a library of objectified information that clients utilize to both obtain and contribute information. The only difference is that clients can obtain this information in, not only a ***pull*** fashion (i.e., query service), but can also have the Information Server ***push*** them information on an interest basis (i.e., subscription service). Physically, the Information Server exists as a distributed object server based on the Common Object Request Broker Architecture (CORBA) (Mowbray and Zahavi 1995).

Distributed object servers, forming the basis of the Information Server, are designed to service client requests for information. The knowledge of exactly where the information resides and how it can be retrieved is completely encapsulated inside the object server. This means that clients need not be concerned with who has what information and in what form that information exists. This feature becomes instrumental in providing an environment where collaborative application components operate in a de-coupled manner via the Information Server.

Regardless of the native representation of information, distributed object servers can be used to present information to clients in the form of objects. However, this does not discount the need for information to be modeled as high-level objects in its native form, portraying behavior and conveying relationships. While on the surface this representational morphing capability of object servers seems promising, in practice it can be quite misleading. If the information is not represented at a high level upon its conception, such objectification amounts to little more than wrapping data in communicable object shells. These shells fail to convey any more insight into the meaning or implication of the information than was present to begin with in its original form. Although in the future there may be potential for successful research efforts in this area, at present, unless information is originally modeled as objects, knowledge-oriented applications prove to gain little from a distributed object server feature.

However, applications that do model information as high-level objects stand to gain considerably from employing distributed object servers. Under these conditions distributed object servers preserve the purely objectified representation of information as it moves throughout the system. This is due to the fact that the internal mechanisms of distributed object servers process information as objects themselves.

The ICDM model takes full advantage of these object-oriented facilities by integrating an Object-Oriented DBMS (OODBMS) into its information environment (Bancilhon et al. 1992). The OODBMS is the facility that the Information Server uses to store the application's objects. Employing an OODBMS to store the information objects has two significant advantages. First, an OODBMS retains the object-oriented representational nature of the information as it exists in its persistent form. Whenever there is representational degradation there is potential for loss of informational content and meaning. By utilizing both access and storage facilities that are capable of processing and manipulating information as objects, there is no degradation of representation as information flows throughout the application environment.

The second advantage of employing an OODBMS relates to the manner in which Information Server clients request information. Whether mining for information or posting a standing subscription, clients formulate their information requests in terms of objects. More specifically, these requests are described in terms of object attributes and object relationships. Such queries can range from simple existence criteria to more complex constraints incorporating both logical and relational operators.

Another method that can be used to obtain information from the ***information tier*** utilizes on the notion of subscription. Clients can dynamically register standing subscriptions that are again described in terms of the application's ontological system. For example, a client may request to be notified whenever the available inventory quantity of a particular supply item falls below a specified threshold value. Once registered, the Subscription

Server continually monitors this condition. When satisfied, the Subscription Server essentially ***pushes*** the results to whichever client has indicated an interest (i.e., registered an appropriate subscription). The alternative to this subscription mechanism would be to have interested clients perform the same query on an iterative basis until such a condition occurs. Each unsatisfied query will potentially decrease resources (i.e., computing cycles) available to other application components. If a client takes a more conservative approach by which the repeated query is made on a less frequent basis, the client risks being out of date with the current state of affairs until the next iteration is performed. With this in mind, the incorporation of a ***push information to interested clients*** mechanism becomes a very valuable facility in providing decision-support applications with an efficient, up-to-date execution environment.

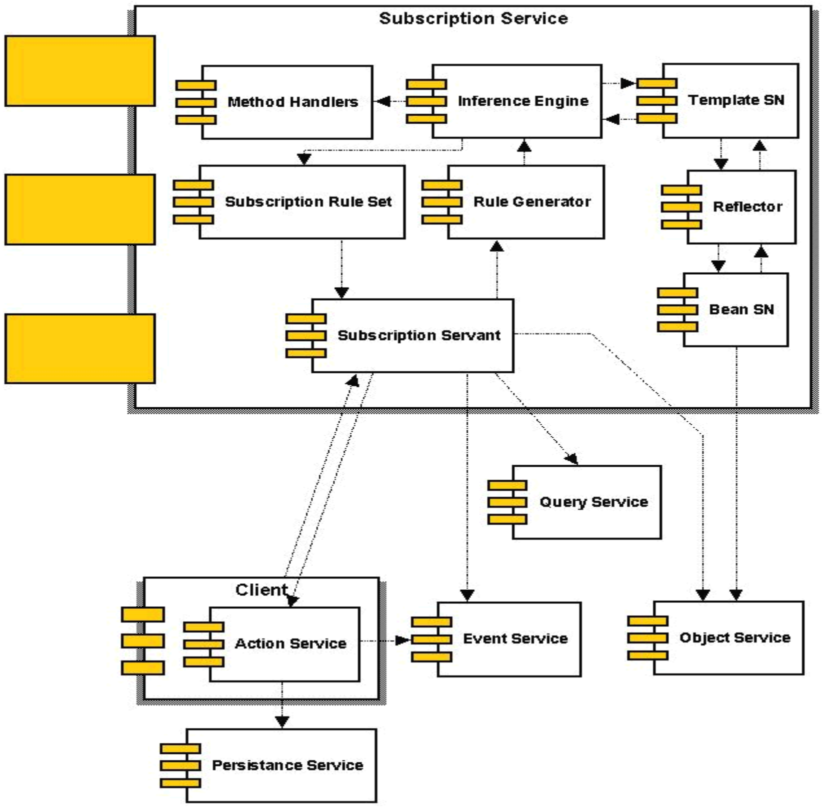


Figure 5: Inference engine-based subscription service architecture

A subscription service essentially monitors a dynamic set of client interests or subscriptions. In this sense, a subscription can be thought of as a ***standing*** or continuous query. Similar to individual queries, interests can be described in terms of information values and constrained events. Once posted, these interests are monitored by the subscription service on a continuous basis. If satisfied the subscription service notifies all interested parties of the situation. This notification can even be coupled with the ***pushing*** of contextual event information to appropriate subscribers. However, in many cases the relevant information a subscriber seeks is that the event has occurred and the subscriber may not in fact be concerned with actual event context. In this case it is more efficient to de-couple the notification of interest satisfaction and the conveyance of contextual information. If desired, such information can be obtained through a series of follow-on queries issued by the subscriber.

Combination of an object-based representation (i.e., ontology) with an inference engine provides a very powerful subscription capability within the necessary object-serving communication facility. We have had some experience with the development of subscription services utilizing the CLIPS rule-based inference engine developed by NASA (NASA 1992). However, many other inference engines are commercially available. CLIPS is offered, essentially free of charge (i.e., cost of documentation only) to US Government agencies and their contractors, in source code form. In all rule-base inference engines predicate logic is described in terms of rules, potentially complex patterns together with related actions. Each rule has a pattern describing a set of conditions and a subsequent action to execute upon its satisfaction. To efficiently manage the matching of potentially high volumes of patterns across equally high and ever-changing pools of information, CLIPS employs an extremely efficient scheme known as the RETE algorithm (Forgy 1982). As a result the varying degree of pattern satisfaction across large sets of rules and dynamic sets of information can be maintained in the context of a near real-time decision-support system. It is this efficient and extensive pattern matching ability that makes this mechanism an excellent candidate for forming the core of a robust, ontology-based subscription service.

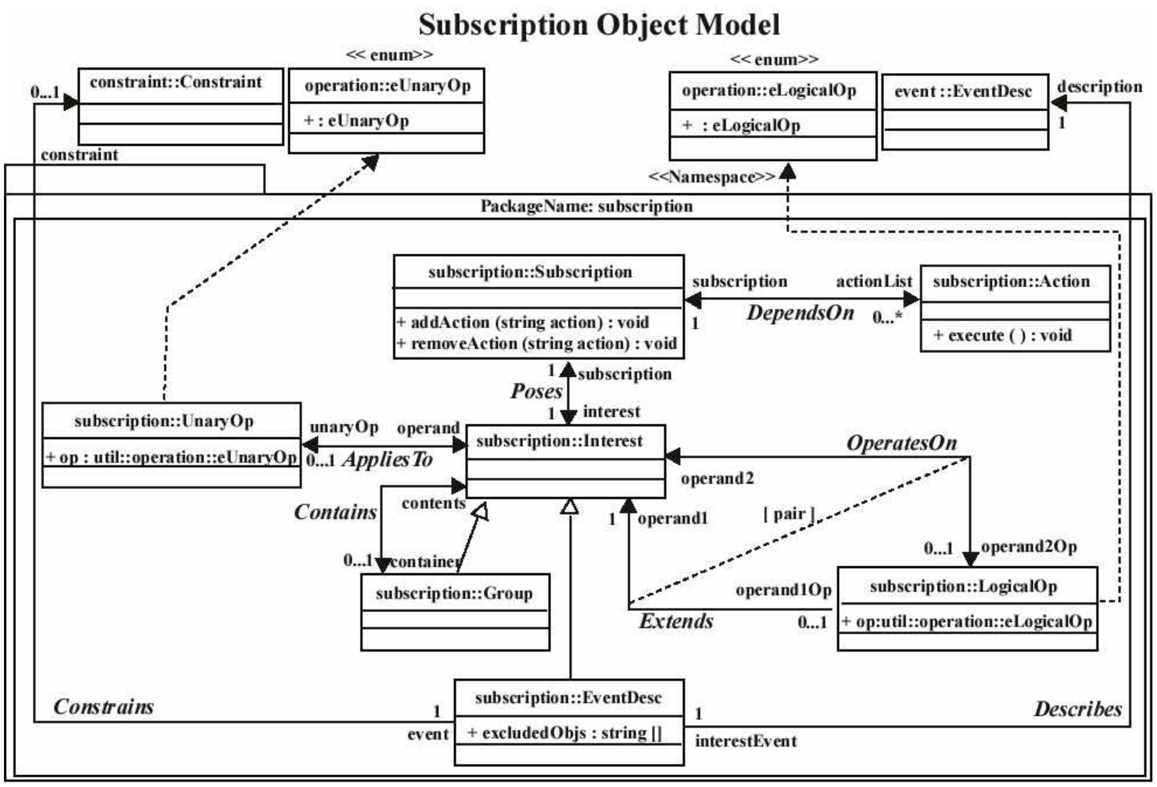


Figure 6: Subscription service domain ontology described

The basic components comprising a subscription service architecture are shown in Figure 5. Each component works in conjunction with the others to effectively manage the dynamic set of subscriptions. If implemented within a CORBA-based environment this architecture adheres closely to the application server design pattern (Ayers et al. 1999). In such a pattern both information and functionality are essentially ***served*** to a dynamic set of clients as sharable, collaborative objects. Following this pattern, the subscription service is presented to clients in the form of subscription objects that can be instantiated. These subscription objects embody the same set of qualities as any CORBA object, and can be represented in the ontology as a ***subscription/notification*** domain (Figure 6).

To invoke the subscription service a client may either instantiate a subscription object or instead chose to utilize an existing subscription registered by another client (i.e., subscription objects enjoy the CORBA quality of being sharable). Regardless of method, during this process the subscriber also associates a local action object to the subscription (Figure 7(a)). It is this action that the client wishes to have executed upon subscription satisfaction. Each time a new subscription is created the Rule Generator component of the subscription service constructs a corresponding CLIPS rule. The content of this rule represents the characteristics of the particular subscription. This rule is then placed under the management of the CLIPS inference engine that in turn monitors the pattern for satisfaction. If a match occurs the action portion of the rule (not to be confused with the client action object) triggers the associated client action.

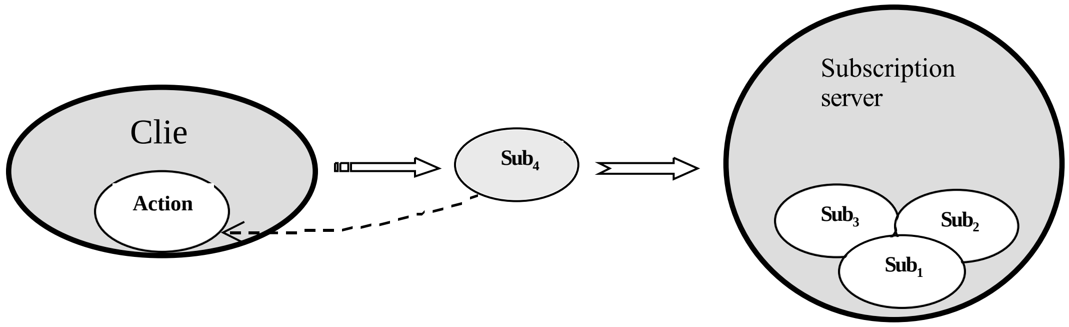


Figure 7(a): Subscription registration

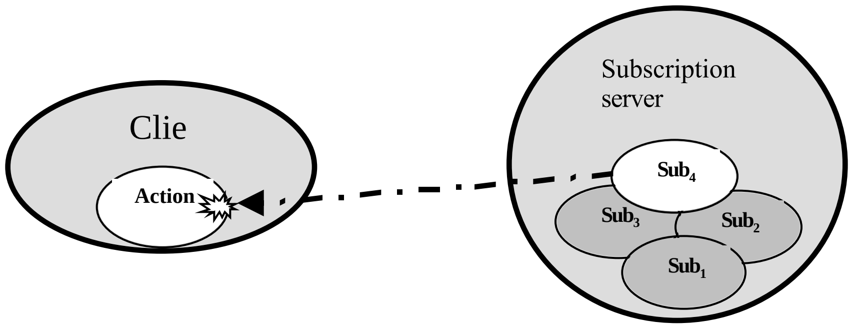


Figure 7(b): Client notification regarding interest satisfaction

In the subscription service architecture described here, client action objects exist as specialized CORBA objects. However, while subscription objects are implemented within the scope of the subscription service, action objects are implemented within the context of the subscribing client. This introduces a powerful capability inherent in CORBA-based architectures. In a CORBA-based paradigm the distinction between client and server is somewhat blurred. Each application component has the potential of being a client in one sense and a server in another. The notification mechanism outlined in this design makes significant use of this feature to permit asynchronous communication between the subscription service and its clientele. Once the subscription service identifies that an interest has indeed been satisfied the subscription service then becomes a client to the action server part of the subscriber it intends to notify. In a CORBA-like fashion, the subscription service remotely executes the ***notify*** method of the action object of each relevant client (Figure 7(b)). Once they are executing within the context of their action objects, notified subscribers may determine the most appropriate course of action to react to the event.

By employing an inference engine as the core of the subscription service two significant capabilities are achieved. First, subscribers in a decision-support application can now exploit the powerful and extensive pattern matching functionality inherent in a rule-based inference engine. The degree and complexity of subscriber interests are constrained only by the extent of the ontological system on which they operate. Second, a strong similarity can be drawn between subscriptions comprised of extensive interests together with specific actions and the powerful pattern/action architecture of an expert system rule. In this manner, the subscription service allows decision-support applications to be described as collections of ***distributed*** expert system rules with the inherent benefits of autonomous and opportunistic capabilities.

3.2 Agent Engine

The Agent Engine represents the ***logic tier*** of ICDM's underlying three-tier architecture. Existing as a client to the services of the ***information tier*** (i.e., access, query, subscription, and persistence) the Agent Engine is capable of both obtaining and injecting information. Architecturally, the Agent Engine consists of an agent server capable of serving collections of agents (Figure 4). These collections, or Agent Sessions, exist as self-contained, self-managing agent communities capable of interacting with the ***information tier*** to both acquire and inject information.

For the most part, the exact nature of agents and collaborative model employed is left to the application specification. However, regardless of the types of agents contained in an Agent Session, agent activity is triggered by changes in application information. This information may take the form of global objects managed by the ***information tier*** or local objects utilized in agent collaboration which are managed by the Agent Session itself. Regardless of whether agents are interacting with the ***information tier*** or each other, interaction takes place in terms of objects. This again illustrates the degree to which an object representation is preserved as information is processed throughout the application environment.

3.3 Agent Session Configuration

The grouping of agent analyses into heterogeneous collections of agents allows for a number of interesting configurations. These configurations determine the size, number, and individual scope of the agent sessions. While a wide variety of Agent Session configurations exist, we have found considerable success in formulating such a configuration based on two primary criteria.

The first criterion introduces the notion of a ***view***. A ***view*** is a conceptual perspective of reality. In other words, a ***view*** can be thought of as a single investigation into solving a problem whether it is based on fact or speculation. For example, a ***view*** may describe events and information relating to what is actually occurring in reality. Yet, another ***view*** may describe an alternative or desired reality. An illustration of this approach can be observed in the Integrated Marine Multi-Agent Command and Control System (IMMACCS) developed by the CADRC and CDM Technologies for the Marine Corps (Pohl et al. 1999). IMMACCS uses a single ***view*** to represent the information and events occurring in the battlespace. In a similar manner, IMMACCS employs any number of additional views to represent hypothetical investigations to determine suitable strategies for dealing with potential events or circumstances.

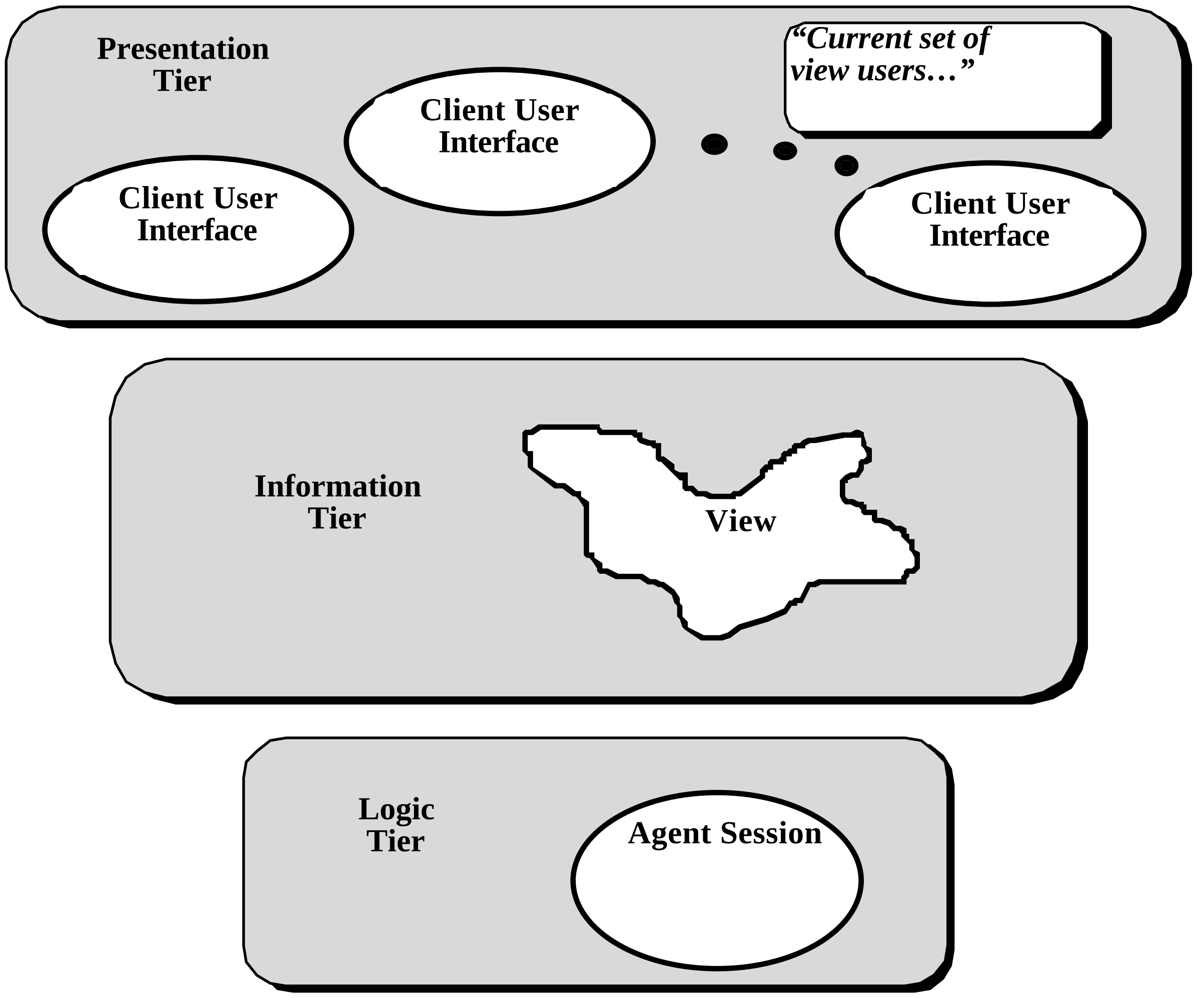


Figure 8: Multiple users can interact with a ***view*** that is analyzed by one Agent Session

Regardless of use, however, there is a one-to-one correspondence between a conceptual ***view*** and an Agent Session (Figure 8). This means that independent of exactly which version of reality a ***view*** represents, there exists a dedicated Agent Session providing users of that ***view*** with agent-based analysis and decision-support. Each agent of a particular Agent Session deals only with the ***view*** associated with its Agent Session. Organizing information analysis in this manner allows for an efficient and effective means of distinguishing information and activities relating to one ***view*** from those pertaining to another. Unless prompted by user intervention (i.e., user-directed movement of information between views), each set of information is completely disjoint from the other.

The second configuration criterion that we currently subscribe to determines the number and nature of agents contained in an Agent Session at any point in time. Our decision-support applications typically utilize a variety of agent types. Three of these agent types include Domain Agents, Object Agents, and Mediator Agents (Pohl 1995). Service-oriented Domain Agents embody expertise in various application-relevant domains (i.e., weapon selection and deconfliction for tactical command and control, tidal dynamics and trim and stability for ship load planning, and so on). The collection of Domain Agents populating an Agent Session at any point in time determines the variety of domain specific perspectives and the analytical depth available during the analysis of the associated ***view***. Under the configuration scheme utilized in most of our decision-support applications, users can add or remove these domain perspectives in a dynamic fashion.

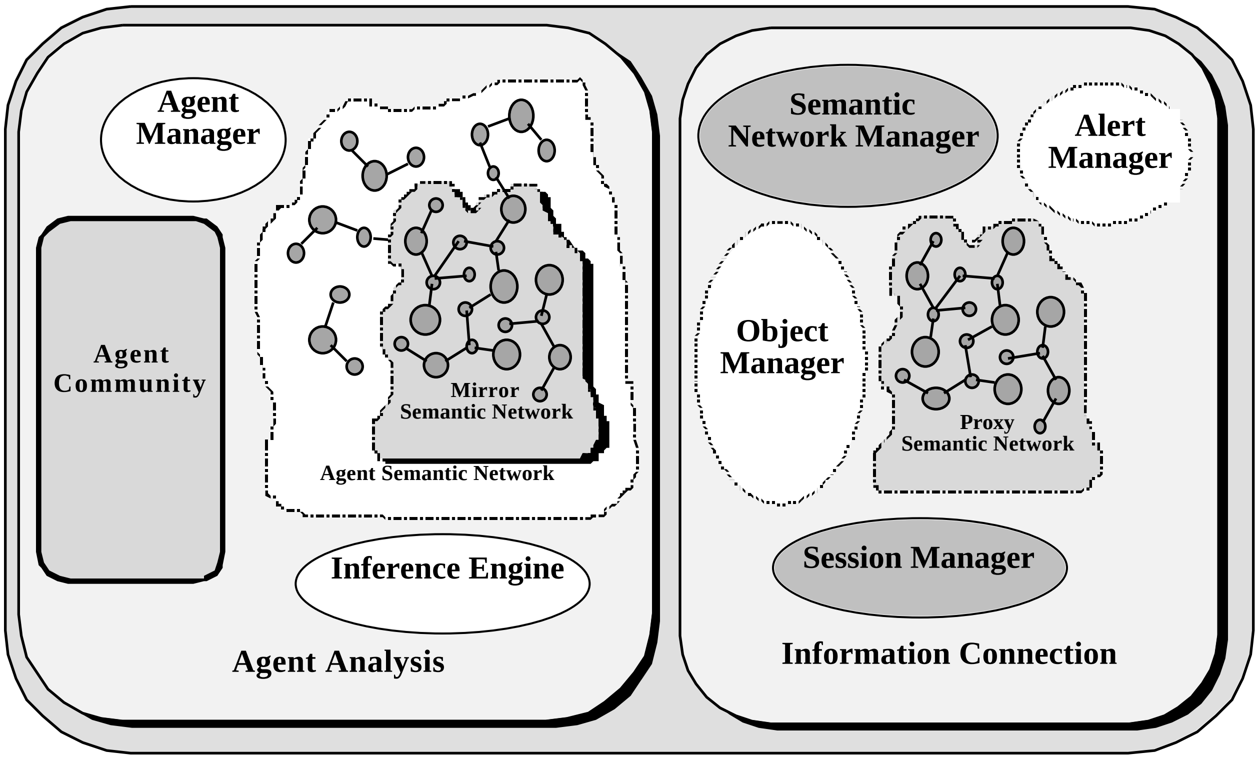


Figure 9: Agent Session architecture

Object Agents, on-the-other-hand extend the notion of high-level informational representation by essentially ***agentifying*** information through empowering information objects with the ability to act on their own behalf. Both human users and other agents as needed can initiate this agentification of information into Object Agents.

In an attempt to resolve conflicts arising between collaborating agents, Mediation Agents may be employed as third party mediators. It is the goal of these mediators to bring about consensus among agents that may have reached an impasse.

Under the current ICDM model each of these agent contingents is dynamically configurable by both the user(s) and the system itself. This approach to Agent Session configuration promotes the notion of offering assistance in the form of dynamically configurable tools rather than predefined solutions (Pohl 1997).

3.4 Agent Session Architecture

Architecturally, an Agent Session consists of several components including the Semantic Network and Semantic Network Manager, Session Manager, Inference Engine, and Agent Manager (Figure 9). These components operate in an integrated fashion to maintain a current information connection between the agents residing in the Agent Session and the associated ***view*** described in the ***information tier***.

3.5 Semantic Network

The Semantic Network consists of a collection of two sets of application specific information objects. The first set is used to support local collaboration among agents. Depending on the specific collaborative model employed, agents may use this local Semantic Network to propose recommendations to each other or request various services. This information is produced and modified by the agents and remains local to the Agent Session.

The second set of information is a sort of duplicate, mirror image of the ***view*** information stored in the ***information tier***. In actuality, this information exists as a collection of object-based interfaces allowing access to ***view*** information stored in the Information Server. Such interfaces are directly related to the application's ontological model. In other words, these interfaces, or proxies (Mowbray and Zahavi 1995), are represented in terms of the objects described in the information object model.

Through these interfaces, Information Server clients have the ability to access and modify objects contained in the ***information tier*** as though they are local to the client's environment. All communication between the object interfaces and their remote object counterparts is encapsulated and managed by the Information Server and completely transparent to the clients. This is a fundamental feature of distributed object servers on which the Information Server is based (Orfali et al. 1996).

3.6 Semantic Network Manager

As the primary manager of the two sets of information described above, the Semantic Network Manager focuses the majority of its efforts on the management of the bi-directional propagation of information between Information Server proxies and an equivalent representation understandable by the Inference Engine. Such propagation is accomplished through employing an Object Manager. The purpose of this manager is to essentially maintain mappings between the object proxies and their corresponding Inference Engine counterparts. The necessity of this mapping reveals a limitation inherent in most distributed object server and inference engine facilities. Most facilities supporting one of these two services require control over either the way client information is represented or the manner in which it is generated. This is due to the fact that both facilities require specific behavior to be present in each object they process. Examples of such facilities include IONA's ORBIX distributed object server (IONA 1996) and NASA's CLIPS inference engine (NASA 1992). Both of these facilities suffer from this limitation. Nonetheless, this dilemma can be solved through the use of an intermediate object manager that maintains mappings between the two sets of objects.

An additional responsibility of the Semantic Network Manager deals with the subscriptions, or interests, held on behalf of the agent community. That is, the Semantic Network Manager is responsible for maintaining the registration of a dynamically changing set of information interests held on behalf of the Agent Session agents. In addition, the Semantic Network Manager is responsible for processing notification(s) when these interests are subsequently satisfied. Such processing includes the propagation of information changes to the agent community that may in turn trigger agent activity.

To perform these two interest-related tasks the Semantic Network Manager employs the services of the Alert Manager. The Alert Manager provides an interface to the Subscription Server facility and is available to any ***information tier*** client wishing to maintain a set of information interests. Employment of the Alert Manager by subscribers has two distinct advantages. First, clients are effectively de-coupled from the specifics of the subscription interface. This allows the same application client to be compatible with a variety of object server implementations. Second, the Alert Manager interface allows subscribers to effectively decompose themselves into a dynamic collection of thread-based interest clients (Lewis and Berg 1996). In this respect, the Alert Manager extends the monolithic one-to-one relationship between the Subscription Server and its clients into one that supports a one-to-many relationship. Such decomposition of functionally related behavior into lightweight processes promotes the concepts of multi-processing in conjunction with resource conservation.

3.7 Inference Engine

The Inference Engine provides the link between changes occurring in the Semantic Network and agent activation. Recall that agent activation can occur when a change in the Semantic Network is of interest to a particular agent. In such a case, the Inference Engine, having knowledge of specific agent interests in addition to changes occurring in the Semantic Network is responsible for activating, or scheduling the action(s) the agent wishes to execute. This activation list forms the basis for the Agent Manager to determine which agent actions to execute on behalf of the currently scheduled agent.

3.8 Agent Manager

The Agent Manager is responsible for the management of the agent community housed in an Agent Session. This management includes the instantiation and destruction of agents as they are dynamically allocated and de-allocated to and from the agent community. In addition, the Agent Manager is responsible for managing the distribution of execution cycles allowing each agent to perform operations. Disbursement of execution cycles occurs in a round-robin fashion allowing agent analysis to be evenly distributed among relevant agents. Whether or not an agent utilizes its allotted cycles depends on whether it has any tasks or actions to perform.

3.9 Session Manager

As the overall manager of the Agent Session environment the Session Manager has two main responsibilities. The first of these responsibilities focuses on the initialization of each of the other Agent Session components, upon creation. When an Agent Session is created in response to the creation of a ***view***, the Session Manager is the first component to be activated. Once initialized, the Session Manager activates the Semantic Network Manager and Inference Engine. Continuing its efforts, the Session Manager then activates the Agent Manager. Upon startup, the Agent Manager initializes itself by allocating an appropriate initial set of agents. Depending on the application specifics, these agents may in turn perform a series of initial queries and subscriptions that will eventually propagate to the ***information tier*** via the Semantic Network Manager.

3.10 ICDM as a Development Toolkit

As a further formalization of the ICDM approach to agent-based, decision-support applications, the CADRC and CDM Technologies have developed a collection of design and development tools to accompany the execution framework discussed above. These tools essentially combine the roles of application designer and application developer into a single effort.

Decision-support applications can be designed and developed through a series of high-level models describing information structure and analytical logic. High-level classes can be identified through a series of Unified Modeling Language (UML) class diagrams, forming a comprehensive information object model (Fowler and Scott 1997). Such a model describes the application specific design and problem space as a collection of high-level objects complete with attributes and relationships. This is essentially the same high-level description of application information identified earlier as being crucial to agent-based, decision-support applications.

By the same token, it may be argued that much of the analytical reasoning applied to this information can be described in terms of a methodology suitable for representing object-based logic. Though currently at the theoretical stage, the methodology intended to be employed to serve this purpose attempts to represent logic as a series of rules (Hayes-Roth et al. 1983). Each of these rules identifies both a condition and a corresponding action to take upon the satisfaction of that condition.

This is where the advantages of using a high-level, object-based representation again become apparent. Both the condition and action components of these rules can be described in terms of the information object model of an application. In other words, conditions can be represented as a series of constrained references to object attributes strung together with logical and relational operators. At the same time, the corresponding rule action is itself described in terms of a series of basic manipulation functions (i.e., create, modify, delete) executed against the information object model. When the informational state described in the condition section of the rule occurs, the corresponding action component will modify or produce information thus creating an entirely new informational state. This new state may in turn trigger other rules to execute in a similar fashion. Although not all logic can be represented in this manner, experience has shown that this approach can be applied to a significant portion of analytical reasoning found in decision-support applications.

Using high-level design models that describe both domain information and domain logic as input, the ICDM Toolkit allows for the automatic generation of a significant portion of the supporting decision-support system. In other words, using the ICDM toolkit the information object model can be used as a basis for automatically generating basic (i.e., construction, attribute-level modification, and destruction) object-specific behavior. In a similar manner, the logic model can be used to automatically generate the condition and action components of the agent rules representing the logic-tier of a decision-support application. Figure 10 illustrates the basic set of components of an ICDM-based application, together with their development method.

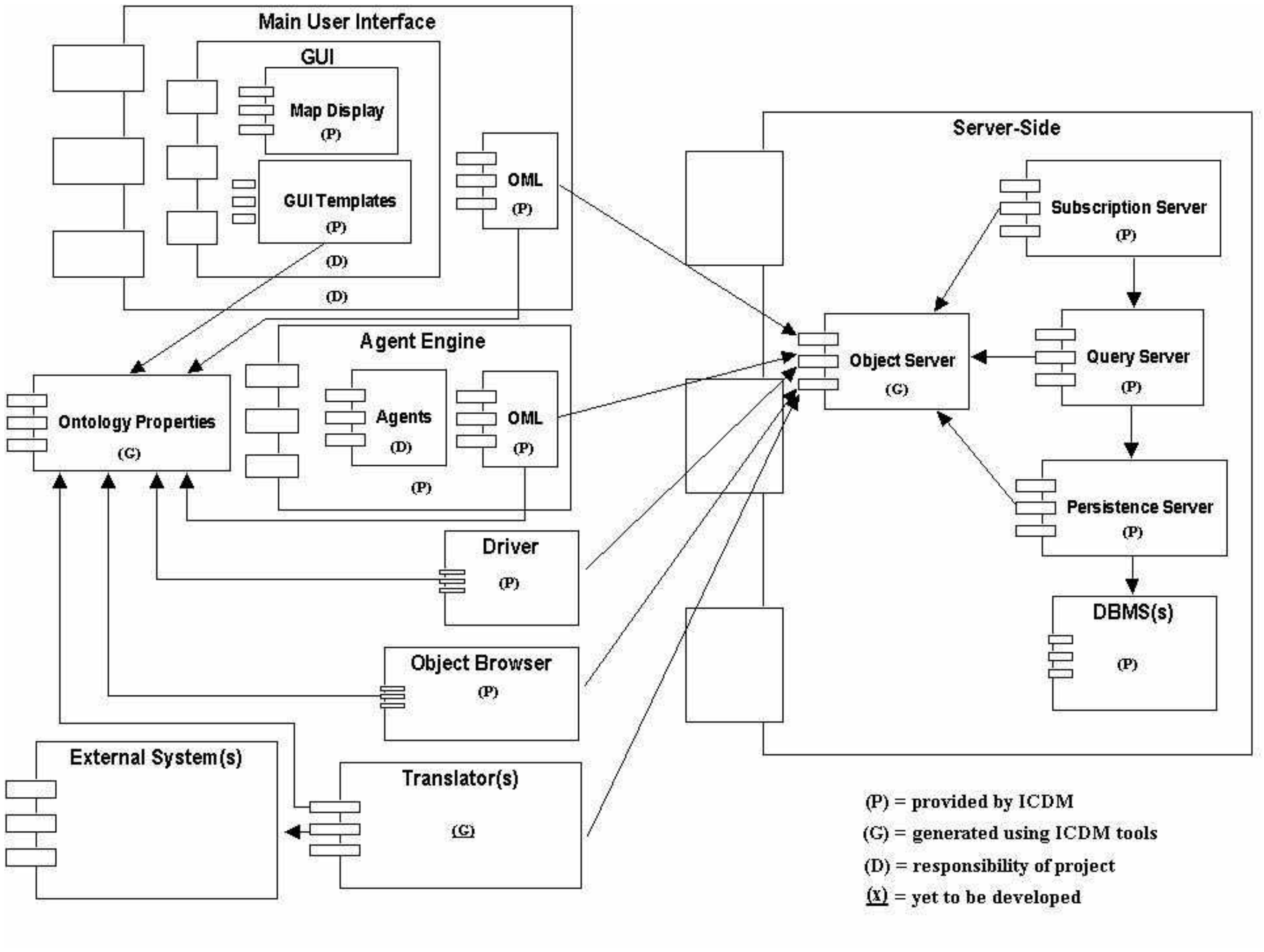


Figure 10: ICDM components and their development method

By elevating a significant proportion of the agent-based, decision-support application development effort to the level of conceptual design, such applications can be developed, maintained, and modified in a considerably more efficient and proficient manner than is possible with the normal (i.e., largely manual) implementation-specific approach. Further, this approach essentially eliminates the loss of intent that often occurs as application development moves from the designers to the program developers. Utilizing the ICDM model together with its design and development tools these roles essentially become synonymous.

4. General Description of the SEAWAY Ontology

The SEAWAY ontology is divided into several somewhat related domains (Figure 11). While some of these domains describe application-specific events and information (e.g., military logistical cargo transactions, military tactical Call-For-Fire (CFF), and so on) others describe more general, abstract notions (e.g., design, application, view). The goal in developing the SEAWAY ontology was to abstract general, cross-domain notions into high-level domain models. As such, these descriptions can be applied across several application domains. More domain-specific, concrete notions can then be described as extensions of these high level models.

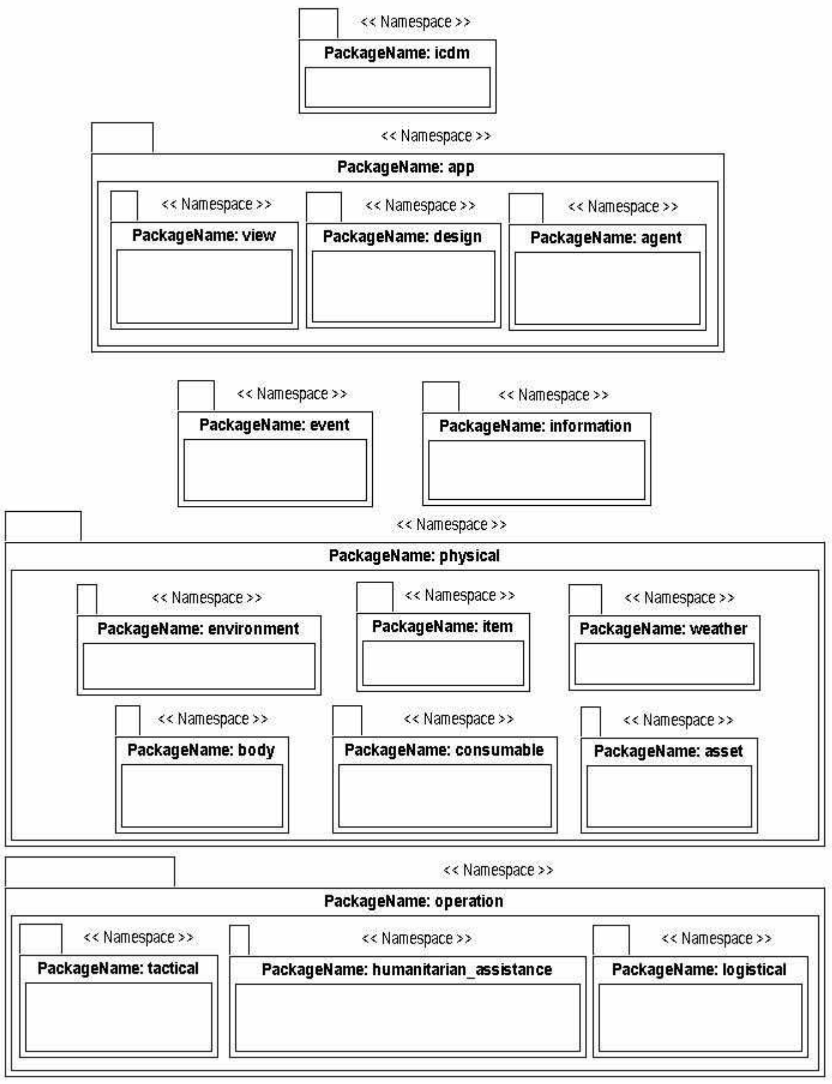


Figure 11: SEAWAY V.0 ontology domains

There are two primary meta-characteristics modeled in the SEAWAY ontology. Through inheritance, these meta-characteristics propagate to extended, more specific ontological components. The first meta-characteristic relates to the property of being ***trackable***. This characteristic is introduced at the 'physical.Mobile' level. Through inheritance, any entity that is a kind of 'physical.Mobile' receives the property of being ***trackable***.

The second meta-characteristic relates to the dispensability of an item. This property is represented at the 'physical.item.Item' level. Similar to the ***trackable*** characteristic, anything that is a kind of 'physical.item.Item' receives the quality of being dispersible or ***suppliable***. In addition, as an extension of 'physical.Mobile' such ***suppliable*** items are also ***trackable***. Together, these two meta-characteristics provide an effective foundation for assigning basic logistical and tactical functionality to the application-specific domains identified within the ontology.

While these meta-characteristics are only implicitly represented in the SEAWAY ontology, two additional notions are represented explicitly in the form of object classes; namely, ***Container*** and ***Empowerable***. A ***Container*** holds (i.e., contains) components. In fact, a ***Container*** can be thought of as a dynamic set of components. There is no inherent order imposed. However, while the contents of ***Containers*** may be very different all ***Container*** objects have a common derivation. An ***Empowerable*** object is one that can be dynamically embodied, or enhanced with additional information, knowledge, or capabilities. This is intentionally a very open-ended notion allowing for unconstrained exploration into various potential 'powers' that can be imparted to an object.

Within the framework provided by the two implicit meta-characteristics and explicitly represented notions, the SEAWAY V.0 ontology includes 17 domains, as follows:

Agent	Software module that has at least the ability to reason and communicate through an expressive language. In addition, ***Agents*** may have one or more of the following capabilities: maintain knowledge; act on their own initiative; collaborate with other agents to accomplish goals; and, use local information to manage local resources.
Application	Application software components and their relationships.
Asset	Equipment, supplies, weapons, munitions, and human resources such as military units. An ***Asset*** is ***suppliable*** and ***trackable***.
Body	Organization or people. A ***Body*** is ***suppliable*** and ***trackable***.
Consumable	Through inheritance from ***Item***, a consumable is ***suppliable*** and ***trackable***. As the name suggests, a consumable can also be depleted.
Design	Describes a potential solution to a particular problem. Such a solution may identify various actions and processes needed for the realization of the solution. In general a ***Design*** also supports a set of requirements.
Environment	The natural (e.g., mountains, lakes, rivers, oceans, etc.) and artificial (e.g., buildings, roads, man-made obstructions, etc.) surroundings.

Humanitarian Assistance	A kind of ***Operation*** that provides assistance to mostly civilians who have been exposed to some crisis (e.g., natural disaster) and require coordinated relief services, equipment, and supplies.
Information	Reports, orders, military intelligence, or any knowledge communicated or received concerning a particular fact or circumstance.
Item	A separate article. Being a kind of 'physical.Mobile' an ***Item*** also has the property of being dispersible. That is, an ***Item***, along with any of its derivations, is not only ***trackable*** but also ***suppliable***.
Logistical	Dealing with the procurement, distribution or movement of equipment, supplies, weapons, munitions, and personnel.
Mobile	Any physical entity that moves or can be moved. As such, a ***Mobile***, and any of its derivations are ***trackable***.
Operation	A set of tasks or processes designed to achieve one or more objectives.
Physical	Anything pertaining to materials, whether natural, man-made, or organic.
Tactical	A kind of ***Operation*** that is of a warfighting nature, involving always friendly and enemy forces, usually also civilians, and often allied forces.
View	A particular version of reality. Reality itself can be considered a ***View***. Existing as a ***Container*** a ***View*** contains all information, entities, events, and so on. needed to describe itself. ***Views*** may be shared among any number of users. ***View*** components may be copied between ***Views***. Also, ***Views*** may be merged with other ***views*** employing various forms of conflict identification and consequential resolution to deal with the inconsistencies that may arise.
Weather	The state of the atmosphere in respect to wind, temperature, precipitation, cloudiness, barometric pressure, etc.

Appendix A provides a detailed description of the SEAWAY V.0 ontology in terms of domains, objects, and inter-object relationships.

5. The SEAWAY Agents

While there are many definitions of computer-based agents, it is generally agreed that agents are application software modules or processes that have the ability to collaborate through their use of communication capabilities and reason about events through their ability to utilize an expressive language. According to Wooldridge and Jennings (1995) "... *agents are computer systems, situated in some environment, that are capable of flexible autonomous actions* ...".

Building on their definition in conjunction with the notion of the ***Agent*** domain represented in the SEAWAY ontology (see Section 4) the agents in SEAWAY incorporate ***situated***, ***autonomous***, and ***flexible*** behavioral characteristics. They are ***situated*** since they receive sensory information from the sea base, from the battlefield and from the in-bound supply line, in addition to the information coming through other components of the system, and perform acts that may change that environment (e.g., creating alerts, making suggestions, and formulating recommendations). SEAWAY agents are ***autonomous*** because they act without the direct intervention of human users, although they allow the latter to interact with them at any time. They also have control over their own actions and internal state.

In respect to ***flexibility***, SEAWAY agents possess at least three qualities that promote flexible behavior. They are ***responsive***, since they perceive their environment through an object model that describes many of the relationships and associations that exist in the sea-basing environment. They are ***proactive*** because they can take the initiative in making suggestions or recommendations (e.g., transport selection for a particular sortie, or route selection for moving supplies or equipment to satisfy a request) and they do that in an opportunistic fashion. For example, when a request for supplies is initiated, the Future Plan agent spontaneously, without the explicit request of the user, determines the impact of the request on the current schedule of sorties and missions and revises the Future Plan accordingly.

A fundamental quality is the ability of agents to communicate (i.e., ***socialize***) with each other and with human users to work on their own problems or assist others with their problems. Conceptually, SEAWAY provides a framework for this to be managed by a coordination agent that looks at suggestions made by different agents and determines if conflicts exist. If there is a conflict, negotiation can be initiated among the conflicting agents in an attempt to reach a consensus solution (Jennings et al. 1998).

An agent is a collection of rules that monitors specific conditions and generates alerts when these conditions are satisfied. An ***alert*** is one mechanism through which agents communicate information to others. The information that agents operate on comes into the Agent Engine (see Sections 2.2 and 3.2) through the Information Server representing the object-serving collaboration facility (see Sections 2.1 and 3.1). The general design of an agent consists of the following components:

1. The conditions that trigger the agent. This is the functional specification of the agent.
2. The objects and their attributes that are involved in these conditions. This is the part of the representation that is used by the agent.

3. The logic that defines the relationships among these objects and attributes.

5.1 Common Agent Language

The object-serving collaboration facility provides the main repository of information in the system, whether it comes directly from sensors (e.g., beacons or smart tags), through other systems, or is entered manually by the user. The representation is object-oriented and a subscription mechanism is provided to notify other components in the system when objects change.

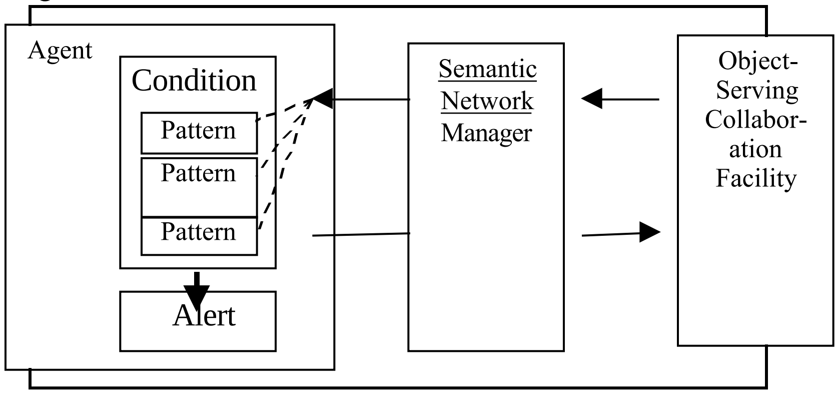


Figure 12: The agent ***alert*** mechanism

The Agent Engine acts on a subset of the object instances, subscribing to the subset of objects and attributes that it needs[1]. The Agent Session Manager receives the event notifications from the Information Server and processes the information in them. The Semantic Network Manager then reflects the changes into the agent environment, while the Agent Manager controls the execution of the agents (Figure 12).

Since the agents are programmed in the CLIPS (Version 6.05) expert system shell (NASA 1992, Giarratano and Riley 1994) that utilizes the RETE algorithm, they do not perform any search or checking for the required information. The RETE algorithm takes the incoming information and automatically allocates it to the rule patterns that provide a match (Forgy 1982). When all of the information for one rule is available (i.e., when all of the predicate patterns in the rule are matched with new information received from the Information Server) the rule is placed on the agenda to execute (i.e., fire) at the earliest opportunity. In this regard, the agents simply wait for all of the information slots to be filled, rather than having to check for new information.

5.2 Opportunistic Agent Execution

One element of autonomy in agent applications is the ability of agents to perform tasks whenever these may be appropriate. This requires agents to be continuously looking for an opportunity to execute. In this context ***opportunity*** is typically defined by the existence of sufficient information. For example, to identify a shortage of supplies (e.g., munitions or fuel) in the sea base, some agent has to monitor the consumption of the particular supply item until there is a shortage and then issue a warning.

[1] A small utility generates the subscription list based on what the agents actually use in their reasoning.

The implementation of agents in CLIPS provides a natural way for opportunistic execution to occur. The RETE algorithm matches objects with rules and keeps track of the partial matches of each rule. When there is a full match (i.e., all of the predicate patterns of the rule have objects to match) the rule is activated and placed on the agenda for execution. If any of the objects that are involved in the match of an activated rule are deleted, this match becomes partial again and the rule is de-activated (i.e., removed from the agenda). This mechanism provides an environment in which rules fire whenever they have an opportunity, rather than in a sequential fashion.

5.3 Functional Specifications

The requirements for agents are defined in terms of two elements: conditions; and, actions. The conditions are the specifications of the situation that the agent monitors, while the actions are the alerts that should be generated when these conditions are true.

- ***Conditions:*** The conditions are specified in terms of objects, attributes and the relationships among them. Each condition is formed by a pattern of object, attributes, values, and Boolean tests. Patterns are grouped by logical connectors, such as AND, OR, and NOT. The more patterns and relationships are specified, the more specific these conditions become.

- ***Actions:*** The right hand side (RHS) of a rule represents the action to be taken when the conditions are satisfied. The most general type of action is to generate an alert. There are a number of alerts in the object model that deal with different types of information. Alerts include warnings, violations, and recommendations.

A formal description of the functional specification of agents may take several forms. A graphical representation of agents is proposed in Figure 13 and utilized in Figure 14, to describe the design of agents and the communication of agent requirements with the object model. An agent is described in terms of rules. Each rule consists of two parts: a pattern group; and, an action group. The pattern group contains a number of patterns (i.e., at least one) that are connected by a logical operator. The default operator is AND. Each pattern represents an object that is of interest to this rule.

The object pattern is identified by a class name and possibly an instance name. Any number of attributes of this object may be part of the pattern as long as they have logical implications for the rule. Constraints can be defined to limit the values of attributes or to define a relationship among a group of attributes. They can be defined for single patterns or across patterns. A special type of constraint is implicitly defined by using the same slot variable in two different slots. This type of constraint is referred to as a ***binding*** constraint and can be replaced by an ***equality*** constraint in the regular form.

The action group contains a number of actions. An action is typically a change to the object repository (create, update, or delete an object.) Some computation may be needed to determine the values of the change. Computations include mathematical functions, object-access functions, and any user-defined functions.

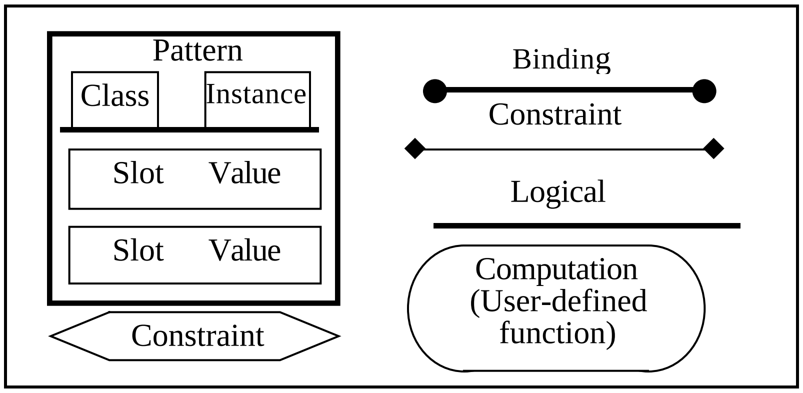


Figure 13: Graphical representation of agents

5.4 Agent Engine Implementation

The creation and operation of SEAWAY agents are managed by an Agent Engine (see Section 3.2) that has a number of components. The responsibilities of the Agent Engine include the creation of agent sessions, managing the update changes in the object repository, and managing agent communications. The principal components of the Agent Engine include:

- ***Agent Session Server:*** Every ***View*** in SEAWAY is associated with an agent session. The agent session server is the component that creates new sessions whenever new ***Views*** are created, and associates these ***Views*** with their sessions.

- ***Agent Session Manager:*** The session manager is responsible for all agent session operations, and the management of their relationships to ***Views***. An agent session is created as a result of the creation of a ***View*** and is attached to that ***View*** in a one-to-one relationship. If the ***View*** is deleted the session manger is responsible for deleting the agent session.

- ***Object Manager:*** The difference in internal representation of objects, in both the user-interface and the Agent Engine, requires a module that is dedicated to the management of change on both sides. The Object Manager is a module that handles the alert notifications coming from the Information Server to update the Agent Engine objects. It also manages the changes in the object repository that come as a direct result of the work of the Agent Engine. There are three basic operations that are performed on objects: create; update; and, delete. Each object is identified in the ontology by a unique object name, and the Information Server uses this name to identify objects in the Agent Engine.

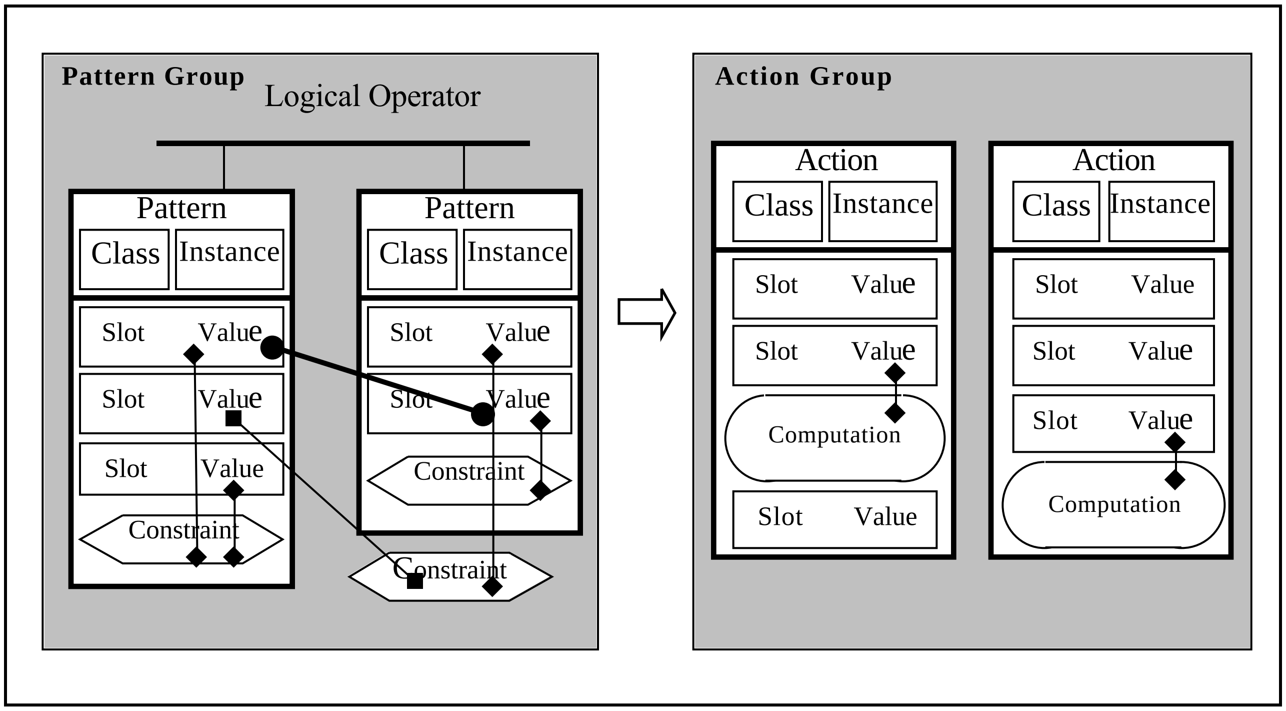


Figure 14: Agent Engine architecture (utilizing the graphical symbols of Figure 13)

- *Agent Manager:* The Agent Manager is a CLIPS module that handles the scheduling of agents for execution. It has knowledge about each agent such as its name, type and the alerts that it is capable of generating. Every agent is identified by an agent identity object, and the Agent Manager relies on this object to cycle through all of the agents. Several aspects of this component require further explanation, as follows:

  1. **Scheduling Strategies –** The Agent Manager can be configured to apply a selected strategy for executing agents, based on: the length of time of execution; the number of rules on the agent's agenda; the order of agents in the list; or, at random. Various user-selectable strategies can alter the order in which the Agent Manager focuses agents. Whichever strategy is in use selects the next agent for execution. No matter what order the agents are selected in, all agents are guaranteed one execution opportunity every cycle. The Agent Manager can be easily extended to include additional strategies by copying an appropriate existing strategy and modifying it.

  2. **Iteration Limits of the Execution Cycle –** For various reasons, the number of control cycle iterations during execution may be limited. The Agent Manager currently supports cycle limits ranging from one to infinity (i.e., no limit). If a fixed cycle limit is set, then the Agent Manager will stop execution following the specified cycle number.

It is also possible to limit the number of rules each agent may execute when focused. The Agent Manager currently supports rule limits ranging from one to infinity (i.e., no limit). If a fixed rule limit is set, then the Agent Manager will allow a module to run no more than the specified number of rules. If an agent agenda runs dry before firing the specified number of rules, then that agent terminates execution and returns to the Agent Manager. If a fixed rule limit is not specified, then the Agent Manager will use the current number of activations for a particular agent as the rule limit for that agent during the current control cycle.

3. **Debugging Reports –** The Agent Manager offers a reporting capability for debugging purposes that may be activated by the user. These reports provide the following information:
   - Initial option values and data file used, if any.
   - Agent status (e.g., executing or idle) as a text phrase.
   - Cycle (i.e., iteration) count.
   - Termination notice when the Agent Manager shuts down.
4. **Agent Status Indication –** The Agent Manager has the ability to indicate when each agent starts executing and when it becomes idle, by setting a status flag in the agent identity object. It can also report the number of rules currently on the agenda for each agent.

5.5 Agents Supported in SEAWAY V.0

This section describes the different agents collaborating within the SEAWAY system. These agents are divided into two domain-related categories namely, tactical and logistical. Each subsection below describes a particular agent in terms of domain, interests, and consequential actions.

5.5.1 Friendly Fire Agent

Domain: Tactical Command and Control

Purpose: Monitors tactical activity pertaining to current or inadvertently planned friendly fire (i.e., a friendly firing on another friendly unit). This capability exists as an '*app.agent.ServiceAgent*' agent.

Event Dependencies and Subsequent Actions:

Event: Existence of an '*operation.tactical.CallForFire*' targeting a '*physical.Physical*' with a friendly force code.

Action: Creates an '*app.agent.Warning*' agent report referencing item and call-for-fire.

Event: Existence of an '*operation.tactical.FireOrder*' targeting a '*physical.Physical*' with a friendly force code.

Action: Creates an '*app.agent.Warning*' agent report referencing item and fire-mission.

Event: Existence of an '*operation.tactical.FireOrder*' where the munitions effects could indirectly endanger a '*physical.Physical*' with a friendly force code.

Action: Creates an '*app.agent.Warning*' agent report referencing item and item.

5.5.2 Engagement Agent

Domain: Tactical Command and Control

Purpose: Monitors tactical activity pertaining to current or planned engagements between any combination of friendly, enemy, or neutral except friendly on friendly, which is handled by the Friendly Fire agent. This capability exists as an '*app.agent.ServiceAgent*' agent.

Event Dependencies and Subsequent Actions:

Event: Existence of an '*operation.tactical.CallForFire*' targeting a '*physical.Physical*' with an enemy or neutral force code.

Action: Creates an '*app.agent.Warning*' agent report referencing item and call-for-fire.

Event: Existence of an '*operation.tactical.FiringOrder*' targeting a '*physical.Physical*' with an enemy or neutral force code.

Action: Creates an '*app.agent.Warning*' agent report referencing item and fire-mission.

Event: Existence of an '*operation.tactical.FiringOrder*' where the munitions effects could indirectly damage a '*physical.Physical*' with an enemy or neutral force code.

Action: Creates an '*app.agent.Warning*' agent report referencing item and fire-mission.

5.5.3 Enemy Encroachment Agent

Domain: Tactical Command and Control

Purpose: Monitors tactical activity pertaining to enemy encroachment on a circular boundary surrounding a particular friendly unit. This capability exists as an '*app.agent.GuardianAgent*' agent.

Event Dependencies and Subsequent Actions:

Event: Passage of an enemy '*physical.asset.Platform*' or an enemy '*physical.body.Body*' inside a guarded '*physical.Physical*' protective perimeter.

Action: Creates an '*app.agent.Warning*' agent report referencing item and enemy.

5.5.4 Critical Items List (CIL) Agent

Domain: Logistical Command and Control

Purpose: Monitors the state of '*physical.item.Item*' whose type appears in an '*operation.logistical.CriticalItemList*'. This capability exists as an '*app.agent.ServiceAgent*' agent.

Event Dependencies and Subsequent Actions:

Event: Creation of a '*physical.item.Item*' of a type appearing in an '*operation.logistical.CriticalItemList*'.

Action: Creates an '*app.agent.FYI*' agent report referencing item.

Event: Deletion of a '*physical.item.Item*' of a type appearing in an '*operation.logistical.CriticalItemList*'.

Action: Creates an '*app.agent.Warning*' agent report describing deleted item.

Event: Association of a '*physical.item.Item*' whose type appears in an '*operation.logistical.CriticalItemList*' to an '*operation.logisticalRolledUp.ItemInfo*' which in turn is associated to an '*operation.logistical.CargoTransaction*'. This would constitute the item's assignment to a supply mission thus potentially decreasing the satisfaction level of one or more '*operation.logistical.CriticalItemList*'.

Action: Creates an '*app.agent.FYI*' agent report referencing item.

Event: Disassociation of a '*physical.item.Item*' whose type appears in an '*operation.logistical.CriticalItemList*' to an '*operation.logistical.RolledUpItemInfo*' which in turn is associated to an '*operation.logistical.CargoTransaction*'. This would constitute the item's unassignment to a supply mission thus potentially increasing the satisfaction level of one or more '*operation.logistical.CriticalItemList*'.

Action: Creates an '*app.agent.FYI*' agent report referencing item.

Event: If the inventory of an item in the ISEP or sea base falls below its minimum quantity on-hand constraint.

Action: Formulates an '*operation.logistical.RBItemOrder*' and creates an '*app.agent.FYI*' agent report containing reference to the reach-back item order.

5.5.5 Sea State Agent

Domain: Logistical Command and Control

Purpose: Monitors changes in the state of the sea alerting to potential impact on logistics activities. This capability exists as an '*app.agent.ServiceAgent*' agent.

Event Dependencies and Subsequent Actions:

Event: Modification of '*environment.BodyOfWater.seaState*'.

Action: Creates an '*app.agent.Warning*' agent report containing appropriate information.

5.5.6 Shipment Compliance Agent

Domain: Logistical Command and Control

Purpose: Monitors the compliance level of '*operation.logistical.CargoTransaction*' with respect to the requirement(s) they are attempting to support. This capability exists as an '*app.agent.ServiceAgent*' agent.

Event Dependencies and Subsequent Actions:

Event: If '*Operation.logistical.CargoTransaction.actionStatus*' is equal to '*ACTION_ABORTED*'.

Action: Creates an '*app.agent.Warning*' agent report containing reference to the shipment in question.

Event: If a shipment's ETA interval is not a subset of the shipment's requirement '*arrivalInterval*'.

Action: Creates an '*app.agent.Warning*' agent report containing reference to the shipment in question along with time-conflict information.

5.5.7 Future Plan Agent

The Future Plan agent is by far the most complex agent currently supported in SEAWAY. It considers the Statement of Requirements, the existing inventory in the sea base, the existing inventory in the ISEP, and the available transport assets, to automatically generate a Future Plan. Taking into account the earliest and latest arrival time (i.e., ETA and LTA) of each requested supply item, the location of the appropriate Supply Point, and the priority of the requested item, the agent develops a schedule of missions and sorties. The factors that need to be considered are numerous, as seen in the following characterization of the problem domain.

Inventory:	Characteristics (e.g., weight, size, description, etc.), quantities and locations of all available cargo items are known to SEAWAY. This includes the expected arrival times of cargo items that are expected to be available in the future.
Start Points:	Cargo is moved from supply ships in the sea base to Supply Points on the shore. In addition, cargo is off-loaded from containerized commercial (i.e., leased) ships at an Intermediate Support and Embarkation Port (ISEP) remote from the theater. This cargo is repackaged and transported in shuttle fashion from the ISEP to the sea base.
End Points:	Predetermined Supply Points ashore (i.e., in the theater) serve as drop-off stations and pick-up stations (for returned items only).
Conveyances:	Helicopters and sea surface craft defined by: type; speed; carrying capacity (i.e., footprint area and weight); and, conditions under which each conveyance is usable. Quantity of each type available for the next 24-hour period are held in SEAWAY.
Packaging:	Most cargo is repackaged (i.e., prior to transportation from sea base to Supply Points) onto pallets of standard size (40 in x 48 in) and variable heights. Large end-items are transported as single units (with nested smaller cargo items on-board whenever practicable). At the ISEP leased commercial supply ships with containers are unloaded and the cargo in containers is repackaged, because there are no adequate lifting facilities available for commercial containers in the sea base.
Distances:	Approximately: 50 nm from sea base to beach; 250 nm from sea base to ISEP; and, 30 nm from beach to Supply Points.
Routes:	Predefined routes between the sea base and Supply Points, and between the sea base and the ISEP. Routes typically include Way Points, however, no cargo is unloaded at Way Points.
Requirements:	A Statement of Requirements specifies all requests for cargo by:

description
quantity
ETA-LTA requested delivery window
priority
unit (military unit ID)
Supply Point (destination for drop-off)
priority

Requests will also include Standard Supply Packages (SSP) for water, food, ammunition, and fuel.

Constraints:

1. Environmental conditions (e.g., ceiling, visibility, wind speed, sea state) and enemy threat conditions (i.e., enemy activity in proximity of transportation routes).
2. Availability of conveyances for the next 24-hour period. However, actual usability of available conveyances may be further restricted by environmental and enemy threat conditions.
3. Carrying capacity of each conveyance, in terms of area (SF) and weight (ST). One Short Ton (ST) is equal to 2,000 lb.
4. ETA-LTA requested delivery window.
5. Current and expected cargo availability.
6. Route restrictions due to enemy threat conditions.

Assumptions:

1. All pick-ups (returned cargo items) are made at drop-off points (i.e., Supply Points).
2. Time taken to off-load pallets and end-items is variable. However, assumptions of 1 min per pallet off-load, and 3 min per end-item off-load are considered acceptable.
3. External helicopter loads reduce the helicopter speed by approximately 75% to 90 knots.
4. It can be reasonably assumed that enemy threat conditions occur on land only.
5. Friendly forces arrive on-shore with three days of supplies. The sea base must try to keep them at a level of five days of supplies.

Ground Rules:

1. Approved portions of the Future Plan constitute the current Execution Plan.
2. The Future Plan will be continuously regenerated as information changes occur in SEAWAY. However, the

Execution Plan must remain ***fixed*** during these Future Plan generations. Agent alerts will notify users of the inability to satisfy requirements and prompt users to disapprove or change portions of the Execution Plan so that another Future Plan can be automatically generated.

3. Missions requiring multiple sorties can be interrupted.
4. The planning period for a Future Plan is 12 hours.
5. Users must have the ability to ***fix*** a portion of the Future Plan, even though it has not been approved.
6. The highest priority cargo items define the priority of a mission.
7. All cargo requests/requirements are designated as being in support of the ***main effort*** or the ***secondary effort***.

Objectives/Metrics:

1. Satisfy ***main effort*** to cargo threshold levels by assigned priority first.
2. Satisfy ***secondary effort*** to threshold levels by assigned priority second.
3. Satisfy ***main effort*** to full levels by assigned priority third.
4. Satisfy ***secondary effort*** to full levels by assigned priority fourth.
5. Reduce ***secondary effort*** quantities to threshold level if necessary to satisfy other ***secondary effort*** requests to threshold level.
6. Give preference to Class III and Class V cargo for tactical deployments.
7. Deliver between requested ETA-LTA window.
8. Assume low penalty for delivery before requested ETA.
9. Assume (very) high penalty for delivery after requested LTA.
10. ***Minimize helicopter usage***. This must be treated as a major objective.

.

We are currently pursuing three quite different approaches for the design and implementation of the Future Plan agent. The first approach, which has been implemented in SEAWAY V.0, is based on a rule-based strategy utilizing the CLIPS expert system shell (NASA 1992). It essentially adopts a heuristic, constraint-based solution methodology that leans heavily on inter-agent collaboration principles (Myers and Pohl 1994, Pohl 1997, Pohl 1999).

The second approach is being undertaken externally and utilizes an operations research methodology (Kang and Gue 1997, Hamber 1998, Gue 2000). This approach focuses on the dynamic distribution aspects of the problem in terms of capacity expansion models (Luss 1982) and dynamic facility allocation (Sweeney and Tatham 1976, Hormozi and Khumawala 1996). The third approach is also being undertaken externally and utilizes an evolutionary computing methodology that essentially relies on evolutionary algorithms (Fogel 2000). Specifically, this approach attempts to apply a recently developed evolutionary computation paradigm, ***particle swarm optimization***, to the Future Plan generation problem (Kennedy and Eberhart 1995). The implementation and testing of theses two alternative approaches is planned for October 2001 in SEAWAY V.1.

In the rule-based approach, which constitutes the current implementation of the Future Plan agent in SEAWAY V.0, the principal constraints of this dynamic distribution and allocation problem are identified as: (1) inadequate number of transport craft (particularly helicopters) for ferrying the cargo to the Supply Points; (2) inability to deliver the requested supplies within the earliest/latest time slots stipulated in the Statement of Requirements; and, (3) unavailability in the sea-base of some requested supplies.

In addition, the following factors have to be accounted for in the knowledge base of the rule-based Future Plan agent:

- Need to repackage most of the cargo on the supply ships that are currently in the sea base, onto pallets prior for loading onto transport craft.
- Need to repackage all of the containerized cargo (i.e., most of the cargo) on the two ships that are en route to the sea base at a land-based Intermediate Support and Embarkation Point (ISEP), which is located about 200nm from the sea base remote from the battlefield. This is necessary because there are no adequate lifting facilities available in the sea base that are capable of handling the commercial containers on the mostly merchant ships that are used to re-supply the sea base. For this reason these in-bound ships will normally unload their cargo at the ISEP and not in the sea base.

Within this problem context the principal function of the Future Plan agent is to produce a schedule of when and in what increments the requested cargo items will reach their predetermined destinations (i.e., Supply Points) within the constraints of a given number of sorties by helicopters and surface craft, and the priorities assigned to the cargo items. The Supply Points (i.e., pick-up and drop-off points), cargo priorities, available helicopters, available surface craft (e.g., hovercraft, boats, etc), current location of each cargo item, destination of requested cargo items, earliest and latest requested delivery date/times, and alternative possible routes, are all known to SEAWAY and held at any time in the object instance store of the object-serving collaboration facility (see Section 3.1).

A Future Plan agent with comprehensive functionality would need to consider all of the following variables. In its current implementation the Future Plan agent considers most, but not all, of these factors in the automatic generation of a Future Plan:

Variable (1A):	current location of each requested cargo item
Variable (1B):	pick-up point location of each requested cargo item

Variable (1C):	date/time ready for pick-up at pick-up point for each requested cargo item
Variable (1D):	drop-off point location of each requested cargo item
Variable (1E):	earliest requested delivery date/time of each requested cargo item
Variable (1F):	latest requested delivery date/time of each requested cargo item
Variable (1G):	priority of each requested cargo item
Variable (1H):	weight of each requested cargo item
Variable (1K):	type (e.g., principal end-item or break-bulk) of each requested cargo item
Variable (2A):	current location of each available transport craft
Variable (2B):	earliest arrival date/time at each pick-up point of each available transport craft
Variable (2C):	cruising speed of each available transport craft
Variable (2D):	maximum speed of each available transport craft
Variable (2E):	range of each available transport craft
Variable (2F):	maximum cargo weight of each available transport craft
Variable (2G):	max. weight of suspended cargo for each available transport craft
Variable (2H):	operational status of each available transport craft
Variable (2K):	type (and category) of each available transport craft
Variable (3A):	location of each Supply Point (i.e., pick-up and drop-off points)
Variable (3B):	location of each Way Point
Variable (4A):	estimated time taken to package a break-bulk cargo item on a pallet, based on: · approximate horizontal distance from its current position to the packaging location · approximate vertical distance from its current position to the packaging location · approximate time taken to position it on the pallet · degree of difficulty factor based on cargo type · expected sea state during packaging operation (not applicable to ISEP)
Variable (4B):	estimated time taken to package any one of the three Standard Re-Supply Packages (i.e., infantry (I-SRP), tank (T-SRP), and artillery (A-SRP)): · average horizontal distance from its current position to the packaging location · average vertical distance from its current position to the packaging location · approximate time taken to position it on the pallet

- degree of difficulty factor based on cargo type
- expected sea state during packaging operation (not applicable to ISEP)

Variable (4C): estimated time taken to secure a principal end-item on a transport craft, based on:

- approximate horizontal distance from its current position to the craft
- approximate vertical distance from its current position to the craft
- approximate time taken to lash/hoist it on or under the craft
- degree-of-difficulty factor based on principal end-item type
- expected sea state during securing operation (not applicable to ISEP)

Variable (4D): estimated time taken to unload a pallet from a transport craft

Variable (4E): estimated time taken to unload a principal end-item from a transport craft

In terms of the characterization of agents used previously in this Section for the description of the other SEAWAY agents (i.e., domain, interests, and consequential actions) the rule-based version of the Future Plan agent is characterized in SEAWAY V.0 as follows:

Domain: Logistical Command and Control

Purpose: Supports the satisfaction of logistical requirements through supply mission planning. This capability exists as an '*app.agent.MentorAgent*' agent.

Event Dependencies and Subsequent Actions:

Event: Creation of an '*operation.logistical.Requirement*'.

Action: Plans and creates one or more supporting '*operation.Sorties*' in the appropriate '*operation.logistical.Plan*' Future Plan.

Event: Deletion of an '*operation.logistical.Requirement*'.

Action: Removes all supporting '*operation.logistical.CargoTransactions*' in the appropriate '*operation.logistical.Plan*' Future Plan.

6. Operating SEAWAY Through Its Client User Interface

The purpose of this section is to provide some examples of the functional capabilities of SEAWAY V.0 during typical sea-basing operations. It must be noted that SEAWAY does not incorporate any solutions to predefined problems. Rather, SEAWAY should be viewed as a set of tools that interact with each other and the users to collaboratively solve problems as they occur in the real world.

6.1 SEAWAY as a Set of Tools

The agents are the most powerful members of this tool set, for several reasons. First, they are able to communicate with each other and the users. This allows one agent to spontaneously enlist the services of one or more other agents in its inferencing tasks. Therefore, by virtue of their communication capabilities, agents are able to collaborate among themselves in an opportunistic manner. Also, through the ***alert mechanism*** agents are able to involve the human user in their collaborations. This provides an avenue for an agent to enlist the assistance of a human user in matters that are either not discernable through logical reasoning alone or require deeper domain knowledge than is available to the agent. In the current version of SEAWAY this potentially powerful agent capability is only marginally present.

Second, those agents that have deep knowledge in a particular domain (e.g., service agents such as the Friendly Fire and the Critical Items List (CIL) agents (see Section 5)) are able to represent the views of their domains and argue those views as they participate in the analysis and evaluation of the current state of the situation. In other words, the service agents ensure that all available viewpoints are represented throughout all collaborations. The use of the word ***available*** is intended to stress that these viewpoints are limited to the existence of an agent to represent a particular viewpoint.

Third, the agents work in parallel and undertake their tasks opportunistically. Both of these characteristics are essential qualities of a truly collaborative environment. As opposed to the rationalistic approach to problem solving (Pohl et al. 1997) which proceeds in an essentially sequential manner and may be undertaken by a single problem solver. Collaboration requires the participation of multiple parties whether human or computer-based or both. These parties should, and will, exercise their autonomy and initiative to contribute when they are able and willing to do so, in a largely unpredictable fashion.

In this respect even the current, initial version of SEAWAY incorporates the beginnings of what might be termed an ***adaptive*** quality. While the knowledge domains of individual service agents are clearly defined and their deep knowledge capabilities therefore entirely predetermined, the interactions of the agents are not at all predefined. In addition, the behavior of all agents is of course to a large extent governed by the events that occur in the real world problem environment, and that are at any particular point in time reflected in the object instance store of the object-serving collaboration facility (see Section 3.1). It can be argued that while the knowledge and reasoning capabilities of the agents are predictable, the interaction of these limited individual capabilities within an unconstrained event-driven environment are to some extent unpredictable. Therefore, SEAWAY incorporates at least the foundations of adaptive behavior.

Finally, it is worthy of note that despite their communication capabilities and opportunistic behavior the agents contain relatively simple inferencing mechanisms that operate on the complexities of the current state of the problem situation. This current state is represented by the current contents of the object instance store, within the contextual framework of the SEAWAY ontology (see Section 4 and Appendix). The relationships, and in particular the dynamic associations, among the object instances are mostly responsible for the level of complexity inherent in SEAWAY. The agents gain their more sophisticated capabilities and potentially adaptive qualities as they navigate through the complexities represented by the object model. In other words, the complexity of SEAWAY rests in the internal representation of the current state of the problem situation and not in the predetermined inferencing mechanisms (e.g., rules) that are embedded in each agent.

6.2 The Scenario Driver

The Scenario Driver is part of the ICDM toolkit (see Section 3.10, designated as Driver in Figure 10) and has been implemented in SEAWAY to provide the user with the ability to record and playback event sequences in either real-time or simulated time (i.e., faster or slower than real-time). It incorporates a simple user-interface (Figure 15) to facilitate the rapid recording of object-based events occurring within a ***view*** in the SEAWAY Client User Interface. Such events may include: the creation and deletion of objects (e.g., request for supplies, tracks, fire events, etc.); the changing of values of object attributes (e.g., unavailability of a transport craft); and, the movement of a track from one location to another.

Once a sequence of events has been recorded it can be played back at varying speeds, up to 50 times the recorded speed. Events may also be entered into the Scenario Driver through a manual process (i.e., off-line) and later played back on-line.

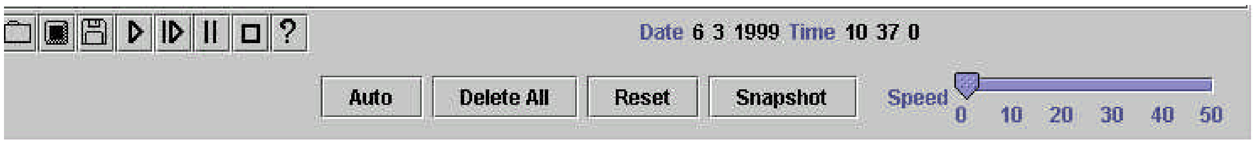


Figure 15: The Scenario Driver control panel

6.2.1 Implementation Design

The functional objectives of the Scenario Driver were based on the desire to provide a means of injecting events and activities into SEAWAY from a source that could be conveniently controlled by users. This desire arose for several reasons. First, SEAWAY recognizes that in the dynamically changing environment of a military mission, planning and execution functions must be supported in an integrated fashion. Not only does this require the ability to simulate events during planning activities, but it also implies the need for playback of simulated plans during execution activities. Second, early during the design of SEAWAY it was postulated that the resulting system would be very useful for training purposes. In particular, it was agreed that SEAWAY should be able to support gaming sessions in which several users assume different roles under a combination of interactive and simulated scenarios. In fact, this will be the focus of SEAWAY V.1, with the near term objective of supporting the gaming requirements of a Limited Objective Exercise (LOE) in October 2001. Third, during the earliest design stages the SEAWAY development team considered the possibility of utilizing a facility such as the Scenario Driver for regression testing purposes during the later stages of software development.

The following functional objectives were established to guide the design of the Scenario Driver:

- Ability to record and playback scenarios in simulated or real-time.
- Ability to step through simulated scenarios under user control, with pause or continue, and stop or restart, capabilities.
- Ability to control the playback speed of a simulated scenario.
- Ability to take a snap-shot of a current ***view*** so as to be able to restore that ***view*** at a later time.
- Ability to playback multiple scenarios that are synchronized in time.

The Scenario Driver is implemented as a multi-threaded application that can be used in stand-alone mode connected to the object-serving collaboration facility, or embedded within the SEAWAY Client User Interface. The class diagram shown in Figure 16, reflects the relationships among the various driver components. The user-interface consists of a driver control panel (Figure 15) that provides a toolbar, a slider, and a choice-box. The toolbar provides the interface to control the scenario (i.e., start, stop, pause, continue, step, record, etc.). The slider allows the playback speed of the scenario to be controlled, and the choice-box displays a list of the currently open scenarios. The '*DriverControlPanel*' class serves as the controller (i.e., it takes all requests entered through the user-interface and causes the current driver thread(s) to perform the necessary tasks.

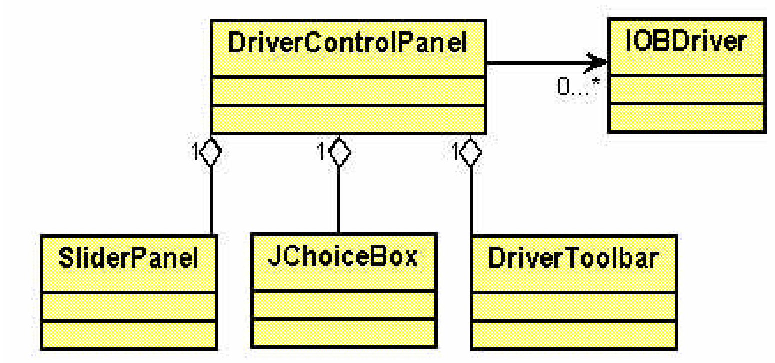


Figure 16: Scenario Driver class diagram

Each Scenario Driver is a single thread that reads from an input scenario file. The thread parses and arranges this file in chronological order, then executes the content of the file, and finally releases control once the scenario has been completely executed or if the driver is interrupted.

6.2.2 Utilizing the Scenario Driver

As discussed previously, the ability of the Scenario Driver to record and playback events in the SEAWAY Client User Interface is useful for testing, training, and planning purposes. While the types of events generated by the Scenario Driver in these simulations usually involve the movement of tracks, the Driver is equally capable for injecting any kind of objects and object attribute changes into sea-basing ***views***.

From the user's perspective the Scenario Driver is made up of several components.

- The playback control panel which features the following options:
  1. The ***Open*** button for opening saved scenario files.
  2. The ***Record*** button to begin recording a scenario file.
  3. The ***Save*** button to terminate recording/saving a scenario file.
  4. The ***Play*** button to start the playback of an open scenario file.
  5. The ***Step*** button to playback an open scenario file incrementally.
  6. The ***Pause*** button to pause a playing scenario file.
  7. The ***Stop*** button to stop a playing scenario file.
  8. The ***?*** or Help button to invoke the on-line Help system.
- The ***Date*** and ***Time*** group which displays the current date and time.
- The auto/manual toggle switch (i.e., ***Auto*** button) which allows the user to select manual or automatic mode for the time and date information of events while recording a scenario. The ***Auto*** setting records the events in real-time, while the ***Manual*** setting allows the time and date to be specified as the event is being recorded. This is useful if an event is scheduled or expected to occur at a specific time.
- The ***Delete All*** button which deletes all the objects in the ***view***. This is a useful, but dangerous capability. If not used carefully, all objects in a ***view*** could be inadvertently deleted.
- The ***Reset*** button which resets the ***view*** to the state of the snapshot file.
- The ***Snapshot*** button which records the current state of a ***view*** and assists in returning a ***view*** to an initial state.
- The scenario playback speed control which allows the playback speed to be increased up to 50 times the normal speed.

6.2.3 The Script Preparation Process

The creation of a scenario can be accomplished by either recording events as they actually occur during the operation of the SEAWAY system or by scripting and recording events off-line (e.g., for demonstration purposes). Scenarios may be recorded at any time during the operation of SEAWAY. Since in SEAWAY V.0 the Scenario Driver is not embedded in the Client User Interface, the driver is initiated on any of the SEAWAY user-stations by double-clicking on the ***Container Driver*** icon. Following this sequence the Scenario Driver Control Panel is displayed along the bottom of the Client User Interface window, allowing the user to click on the ***Record*** button to commence recording the activities as they occur in the ***view***.

Preparing the Script: The recording of a specific scenario for demonstration purposes requires some planning prior to the execution and recording of activities for subsequent playback. The following step-by-step process is highly recommended:

Step 1: Determine the purpose of the demonstration.

Step 2: Prepare written scripts that fully develop the objects and relationships that will be required in support of the demonstration.

Step 3: Start the Client User Interface and subscribe to a ***view***. If there are any objects that need to be created in overall support of the scenario script they should be created using the Client User Interface template forms, prior to the next step.

Step 4: Click on the ***Record*** button in the Scenario Driver Control Panel which is displayed at the bottom of the Client user Interface window.

Everything is now ready for the creation, deletion, movement, and modification of attribute values, of objects through their templates in accordance with the scenario scripts.

Saving the Completed Script: After all of the object manipulations have been completed, the user must click on the ***Save*** button and enter a filename when prompted to save the scenario script.

Testing the Scenario Script: To test the scenario it is necessary for the user to first click on the ***Reset*** button and answer in the affirmative when asked if the ***view*** should be reset. The user then clicks on the ***Open*** button, navigates to and selects the saved scenario file, and clicks on ***Open***. Finally, the user clicks on the ***Play*** button, and the Scenario Driver will start to inject the pre-recorded scenario script into SEAWAY.

6.2 A Simulated Demonstration Scenario

The following simulated SEAWAY demonstration scenario utilizes both predefined sequences of events injected by the Scenario Driver and a Translator to an external application that provides SEAWAY with the cargo inventories on ships (both in the sea base and in-bound) and the ISEP, and user input through the Client User Interface. The external application is ICODES (Integrated Computerized Deployment System) the system of record for ship load planning in the Army, Marine Corps and Navy. ICODES is in itself a multi-agent decision-support application developed by the Collaborative Agent Design Research Center (Pohl et al. 1997). ICODES V.3 has been installed at over 50 ports used by the US Army world-wide, since 1997, and ICODES V.5 is slated to be fielded in early 2001 to Army, Marine Corps, and Navy cargo specialists and embarkation officers.

Demonstration Scenario Context: The situational context is an Amphibious Ready Group (ARG) sea-basing operation off the coast of Somalia. In the sea base, approximately 50nm off-shore there is a Carrier Battle Group (CVBG) with three loaded amphibious assault ships (i.e., supply ships). An Intermediate Staging and Embarkation Port (ISEP) has been established 200nm in the rear and partially stocked, and two loaded Military Traffic Management Command (MTMC) merchant supply ships are in-bound to the ISEP. The amphibious assault ships are the Essex, the Wasp, and the Bataan, and the in-bound ships are the Cape Taylor and the Sgt. William R. Button.

FCADS (Framework for Agent Decision-Support), a tactical command and control system, is also linked to SEAWAY in such a manner that agents in FCADS can communicate with agents and the Client User Interface in SEAWAY. In this way SEAWAY can be advised of enemy threat situations that could impact supply missions and sorties.

The demonstration scenario utilizes a SEAWAY server and three SEAWAY clients. The server is represented by a computer without a user-station (i.e., monitor terminal), but can be accessed through any of the client user-stations. The SEAWAY clients consist of a computer and a monitor terminal and represent the sea-based Tactical Logistics Coordination Center (TACLOG), the Joint Task Force and Marine Air Ground Task Force (JTF/MAGTF) staff, and the Amphibious Group (PHIBGRU) staff, respectively. In a real world sea-basing operation there would be additional SEAWAY client stations for at least the ISEP, the Air Combat Element (ACE), several Combat Service Support (CSS) detachments in the sea base and on-shore, and a link to CINC headquarters. The ICODES application can be accessed on any of the SEAWAY client stations, and so also can the Translator and the Scenario Driver. The linkage between SEAWAY and FCADS is transparent to the users.

Summary of Scenario Events and Activities: The Amphibious Ready Group (ARG) has already arrived and the landing force has gone ashore. SEAWAY has been partially initialized with inventory cargo that is currently on two of the amphibious assault ships and one of the in-bound MTMC merchant ships. Cargo inventories currently resident on the LHD-5 Bataan and the William R. Button will be loaded into SEAWAY during the demonstration scenario, so also will the current inventory in the ISEP. A Statement of Requirements already exists in SEAWAY and all transport craft and their current location are known. However, the subset of transport craft that are actually available for logistics operations will be determined during the demonstration.

Concurrently, FCADS agents are tracking friendly and enemy unit positions and engagements. Alerts will be posted in the SEAWAY Client User Interface relating to threats to the sea-basing operation from friendly fires and related enemy engagements.

The demonstration sequence will step through the incremental development of a complete inventory picture within SEAWAY, the establishment of the three components that are required for the generation of a Future Plan (i.e., cargo inventory, Statement of Requirements, and availability of transport craft), and the implementation of an Execution Plan, as follows:

Step (1): Review of the initialized sea base status.

Step (2): Loading of the LHD-5 Bataan cargo using Translator.

Step (3): Review of the Critical Items List based on a CIL agent alert.

Step (4): Setting of minimum cargo quantity thresholds.

Step (5): Loading of supply inventory in the ISEP using Translator.

Step (6): Loading of the William R. Button cargo using Translator.

Step (7): Acknowledgement of a Sea State agent alert.

Step (8): Establishment and review of transport craft availability.

Step (9): Review and update of the Statement of Requirements.

Step (10): Review of the Future Plan automatically generated by the Future Plan agent.

Step (11): Authorization of the Future Plan.

Step (12): Tactical alerts from FCADS recommending a ***Mission Abort***.

Detailed Demonstration Scenario Sequence: Each of the three SEAWAY client stations, that form part of the demonstration scenario, display the same map of a region in the vicinity of Somalia (Figure 17). Symbols representing ships in the sea base, the ISEP, and Supply Points on-shore are clearly visible.

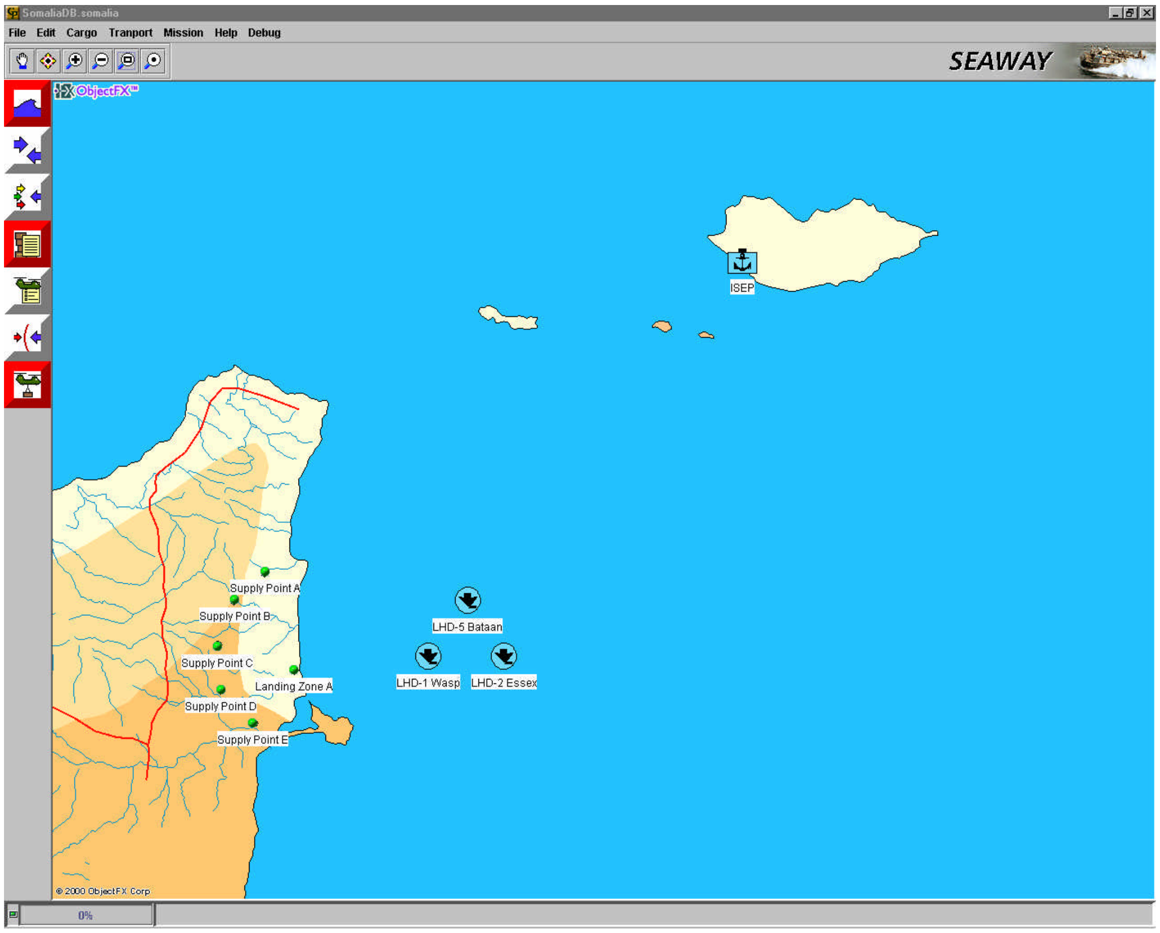


Figure 17: Typical user interface on a SEAWAY client

1. Display the 'Define Theater' Window:

TACLOG station:	click on:	'File' menu option
	select:	Define Theater

Point out the 3 sea-base ships, the ISEP, and the 2 in-bound ships.

2. Display the 'Define In-Bound Ship ETA' Window:

TACLOG station:	click on:	'Cargo' menu option
	select:	Define In-Bound Ship ETA

Point out the arrival times of the 2 in-bound ships.

3. Bring up ICODES:

Minimize the SEAWAY Browser on the JTF/MAGTF station**.**

JTF/MAGTF station:	double click:	'ICODES V.5' icon
	click on:	'Loadout Plan' menu option
	select:	Open Plan
	select:	LHDs.ilp
	click on:	'Open' button

Point out the loadout plans of the 3 sea-base ships, then close ICODES.

4. Bring up the Translator and load the cargo list of the LHD-5 Bataan:

PHIBGRU station:	double click:	'Translator' icon

Minimize the Translator (Command) Window.

PHIBGRU station:	click on:	'Choose File' button
	double click:	cargo_files
	select:	final_bataan.itc
	click on:	'Open' button

Return to Translator Window.

	click on:	Cargo List For
	select:	LHD-5 Bataan
	click on:	'OK' button
	click on:	'OK' button in Closing Window

5. Demonstrate the Critical Items List (CIL) Agent:

Point out the CIL Agent alert and view some alerts.

PHIBGRU station:	double click:	CIL Agent icon

Do not acknowledge these alerts, then close the Window.

6. Explain how to set 'Minimum Quantity Thresholds' for cargo items:

TACLOG station:	click on:	Cargo option in SEAWAY
	select:	Define Cargo Priorities from menu

View some of the cargo items and show how to set minimum quantities.

	select:	(any cargo item)
	click on:	'Edit' button
	click on:	'Cancel' button and close Window

7. Display the 'Item Status' Window and demonstrate the drill-down capability:

TACLOG station:	click on:	Cargo option in SEAWAY
	select:	Item Status from menu
	select:	155NM HE for drill-down

8. Bring up ICODES:

Minimize the SEAWAY Browser on the JTF/MAGTF station.

JTF/MAGTF station: double click: 'FINAL ISEP' icon

ICODES brings up the ISEP automatically.

Zoom in on the staging area with the most cargo.

Point out the cargo in the staging areas of the ISEP, then close ICODES.

9. Bring up the Translator and load the cargo from the ISEP into SEAWAY:

PHIBGRU station: double click: 'Translator' icon

Minimize the Translator (Command) Window.

PHIBGRU station:	click on:	'Choose File' button
	double click:	cargo_files
	select:	final_isep.itc
	click on:	'Open' button

Return to Translator Window.

	click on:	Cargo List For
	select:	ISEP
	click on:	'OK' button
	click on:	'OK' button in Closing Window

10. Note additional CIL Agent alerts.

11. Display the 'Item Status' Window to show cargo in sea-base and ISEP:

TACLOG station:	click on:	Cargo option in SEAWAY
	select:	Item Status from menu
	select:	155MM PROP CHG WB M4

12. Bring up ICODES:

Minimize the SEAWAY Browser on the JTF/MAGTF station.

JTF/MAGTF station:	double click:	'FINAL CONUS.ilp' icon
	click on:	'Loadout Plan' menu option

	select:	Open Plan
	select:	CONUS.ilp
	click on:	'Open' button

Point out the loadout plans of the Taylor and Button ships, then close ICODES.

13. Bring up the Translator and load the cargo lists of the in-bound ships:

PHIBGRU station:	double click:	'Translator' icon

Minimize the Translator (Command) Window.

PHIBGRU station:	click on:	'Choose File' button
	double click:	cargo_files
	select:	final_button.itc
	click on:	'Open' button

Return to Translator Window.

	click on:	Cargo List For
	select:	William R. Button
	click on:	'OK' button
	click on:	'OK' button in Closing Window

14. Note additional CIL Agent alerts.

15. Display the 'Item Status' Window to show cargo in sea-base, ISEP, and in-bound ships:

TACLOG station:	click on:	Cargo option in SEAWAY
	select:	Item Status from menu
	select:	RIFLE 5.56MM and close Window

16. Demonstrate the Sea State Agent:

PHIBGRU station:	click on:	Mission option in SEAWAY
	select:	Define Sea State from menu
	type:	4
	click on:	'OK' button

Point out the Sea State Agent alert on the TACLOG station.

TACLOG station:	double click:	Sea State Agent icon

Acknowledge and post all sea State Agent alerts, then close Sea State Agent Window.

17. Demonstrate how to add/edit Transport availability:

TACLOG station:	click on:	Transport option in SEAWAY
	select:	Define Transport from menu
	select:	(any Transport)
	click on:	'Edit' button

type: Reserved
click on: 'Cancel' button and close Window

18. Display the 'Transport Availability' Window:

TACLOG station: click on: Transport option in SEAWAY
select: Transport Availability from menu
select: LCAC for drill-down, then close Window

19. Display the 'Requirements Template' Window for Statement of Requirements:

TACLOG station: click on: Mission option in SEAWAY
select: Requirements Template from menu

View all of the current requirements in the Statement of Requirements
Close Window.

20. Display the 'Pickup Request' Window:

JTF/MAGTF station: click on: Mission option in SEAWAY
select: Pickup Request Template from menu

View all current pickup requests.
Drill down to describe pickup request of 2 damaged LAVs.
Close Window.

21. Demonstrate the Future Plan generation capabilities:

Explain: ***inventory***, ***requirements***, and ***transport capabilities*** are now available.

TACLOG station: click on: Mission option in SEAWAY
select: Future Plan from menu

View all Future Plan components: Sorties, etc.
Close Window.

22. Display the 'Plan Summary' Window:

TACLOG station: click on: Mission option in SEAWAY
select: Plan Summary from menu

View the LCAC Sorties and the CH-53E Sorties (bringing back the damaged LAVs).
Close Window.

23. Demonstrate how to authorize a Future Plan to become an Execution Plan:

TACLOG station: click on: Mission option in SEAWAY
select: Authorization Template from menu
select: Plan 0
click on: 'Authorize' button and close Window

24. Show that an Execution Plan now exists:

TACLOG station:	click on:	Mission option in SEAWAY
	select:	Execution Plan from menu

25. Display the 'Sortie Completion' Window:

TACLOG station:	click on:	Mission option in SEAWAY
	select:	Sortie Completion from menu

Complete some Sorties.
Close Window.

26. Demonstrate the tactical C2 interface with FCADS:

Already zoomed in on tactical area on all stations: TACLOG; JTF/MAGTF; PHIBGRU.
Bring Scenario Driver to the front on the JTF/MAGTF station.

JTF/MAGTF station:	click on:	'Open' symbol button
	select:	enemyMove.ds
	click on:	'Submit' button

Run driver script either 'all at once' or 'step by step'.

JTF/MAGTF station:	click on:	'Run' or 'Step' symbol button

Point out agent alerts on TACLOG station.

Double click on ***Encroachment Agent***, acknowledge and post all alerts.
Close Window.

Double click on ***Engagement Agent***, acknowledge and post all alerts.
Close Window.

Double click on ***Fires Agent***, acknowledge and post all alerts.
Close Window.

6.4 The White Board and Messaging Facilities

Transparently to users, but external to the SEAWAY application, the client user interface provides access to whiteboard and message-passing facilities. These facilities allow users to communicate with drawings (e.g., free-format graphical annotation of the current SEAWAY map display) and text messages. The White Board and Messaging (WBM) application was written in Java and builds directly onto the commercially available Java Shared Data Toolkit (Sun Microsystems 1999).

The whiteboard component of the WBM application allows multiple users to identify points of interest on a map (Figure 18). The user can draw lines, circles, dots, and place text notes on the drawing, or superimpose any of these on a backdrop map. The use of different pen colors allows users to distinguish their annotations from one another. The whiteboard also has an eraser pen that can be used to modify drawings. A ***Refresh*** button clears all drawings off the whiteboard. The principal features may be summarized as follows:

- all users see the same image and annotations;
- ability to draw lines, dots and circles, and add text notes;
- availability of multiple pen colors and widths;
- ability to refresh and erase.

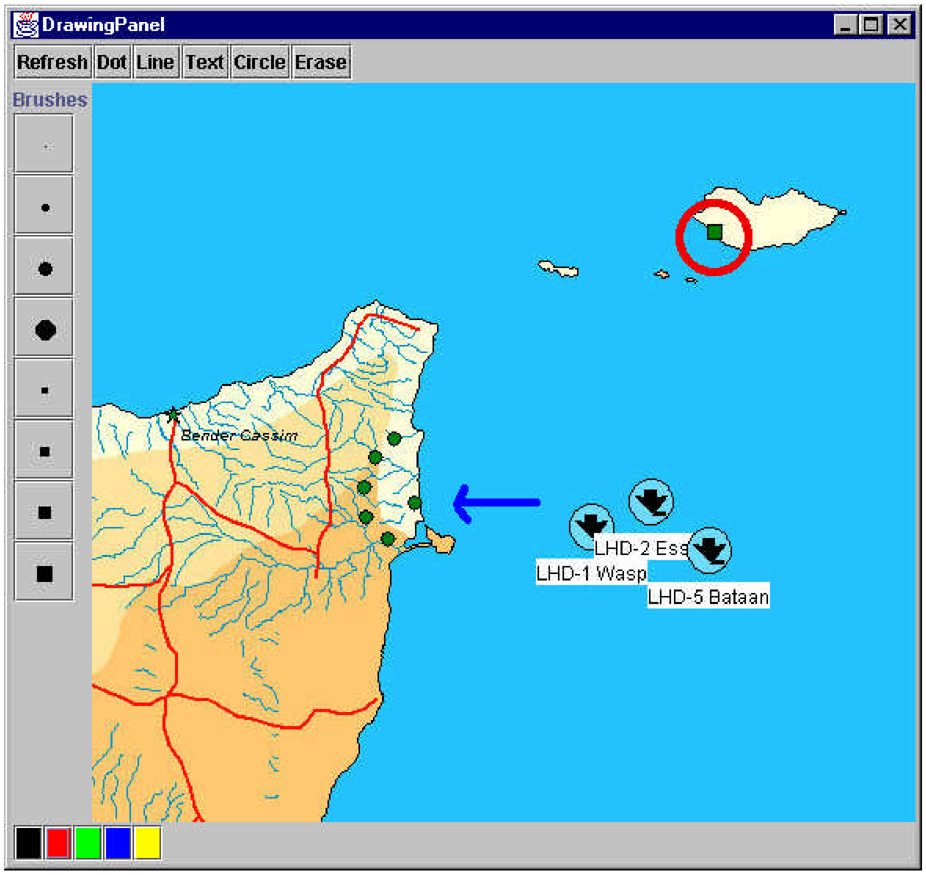
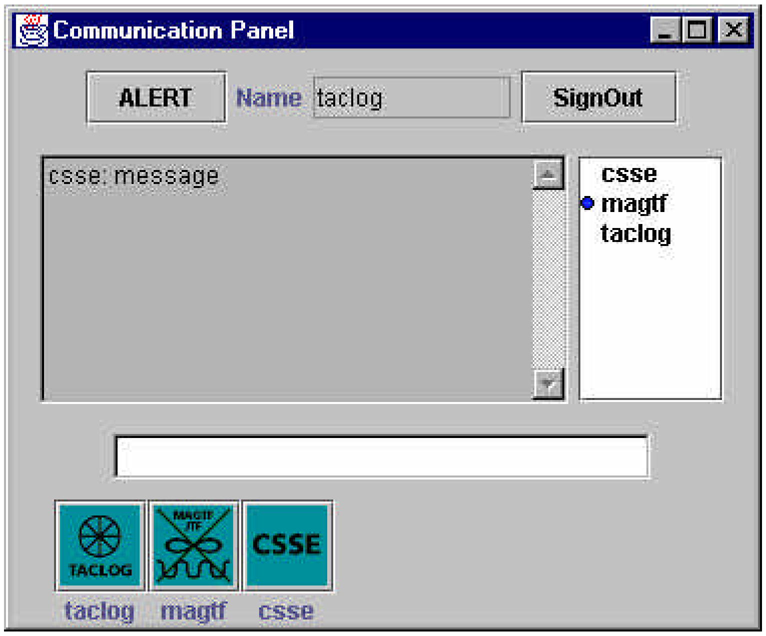


Figure 18: Typical annotated map utilizing the whiteboard component of WBM

The messaging or chat component of the WBM application allows multiple users to communicate with one another with text messages. The text message from any user is either broadcast to everyone, or recipients may be identified by selecting the names of specific recipients in a widow that list the names of all connected users. This window is located to the immediate right of the text entry window (Figure 19). A blue dot next to the name of a user identifies outgoing message recipients. For high priority messages, a blinking red alert can be sent to a user by clicking on the ***icon*** button of that user, located at the bottom of the text entry window. The main features of the chat component are listed below:

- listing of users that are currently connected;
- ability to broadcast message;
- ability to send message to multiple, specified recipients;
- automatic display of sender's name with received message;
- blinking red alert window for high priority messages.



csse: message

Figure 19: User interface of the chat component of the WBM application

6.5 Typical Examples of SEAWAY Operational Sequences

The following SEAWAY client user interface displays are based on the simulated demonstration scenario described in Section 6.3, above. The setting for these sequences is a region encompassing most of Somalia and the adjoining ocean. An explanation of each sequence is provided within an expanded caption.

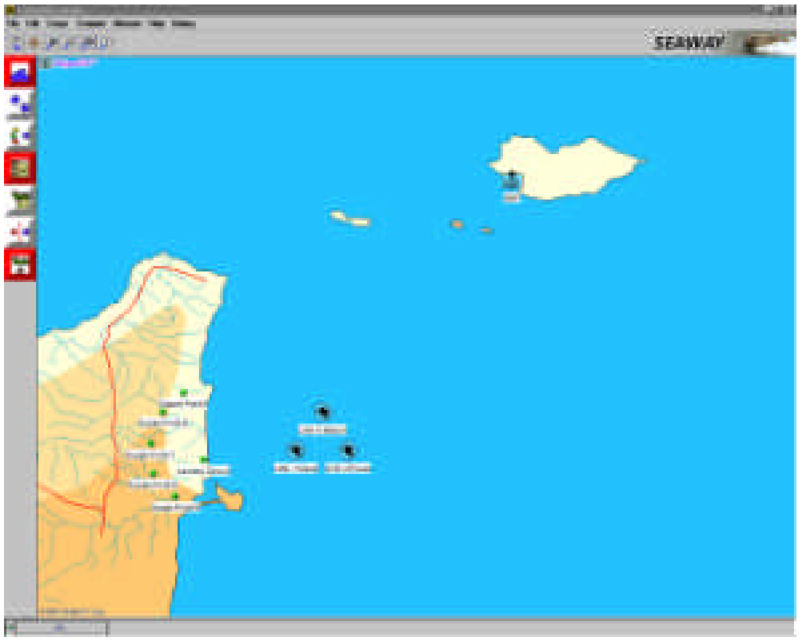
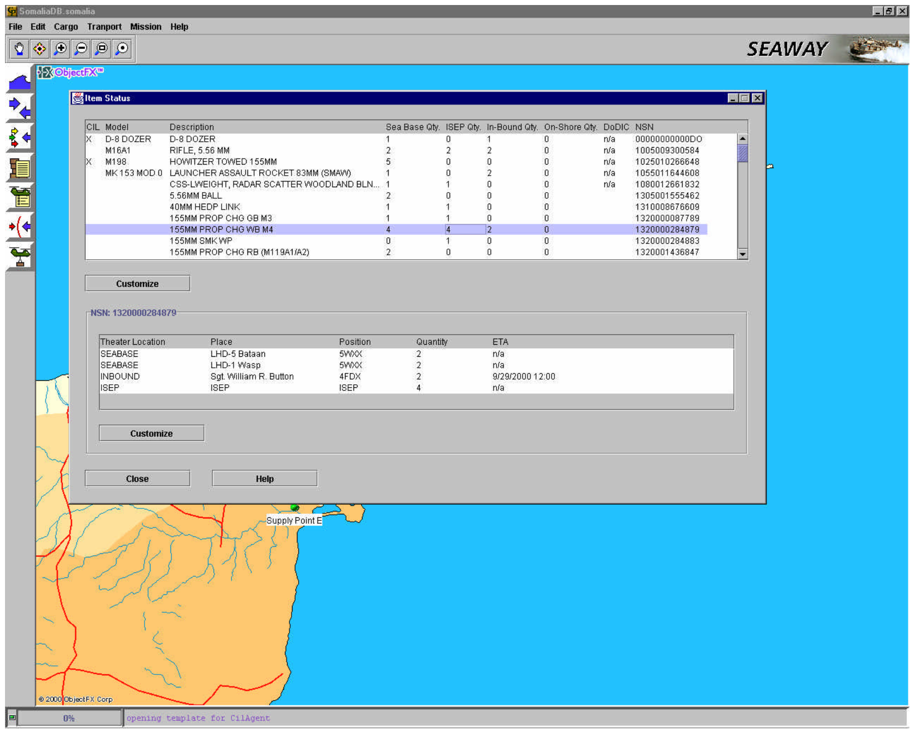


Figure 20: The SEAWAY V.0 client user interface
(Somalia setting of the simulated demonstration scenario).

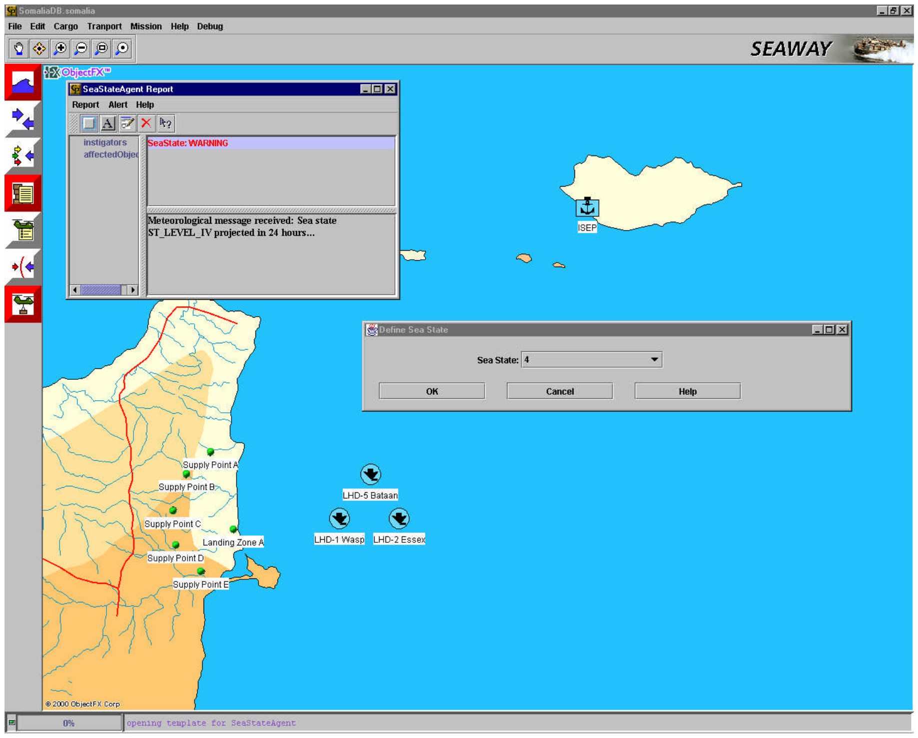


CIL	Model	Description	Sea Base Qty.	ISEP Qty.	In-Bound Qty.	On-Shore Qty.	DoDIC	NSN
X	D-8 DOZER	D-8 DOZER	1	0	1	0	n/a	00000000000DO
	M16A1	RIFLE, 5.56 MM	2	2	2	0	n/a	1005009300584
X	M198	HOWITZER TOWED 155MM	5	0	0	0	n/a	1025010266648
	MK 153 MOD 0	LAUNCHER ASSAULT ROCKET 83MM (SMAW)	1	0	2	0	n/a	1055011644608
		CSS-LWEIGHT, RADAR SCATTER WOODLAND BLN...	1	1	0	0	n/a	1080012661832
		5.56MM BALL	2	0	0	0		1305001555462
		40MM HEDP LINK	1	1	0	0		1310008676609
		155MM PROP CHG GB M3	1	1	0	0		1320000087789
		155MM PROP CHG WB M4	4	4	2	0		1320000284879
		155MM SMK WP	0	1	0	0		1320000284883
		155MM PROP CHG RB (M119A1/A2)	2	0	0	0		1320001436847

NSN: 1320000284879

Theater Location	Place	Position	Quantity	ETA
SEABASE	LHD-5 Bataan	5WXX	2	n/a
SEABASE	LHD-1 Wasp	5WXX	2	n/a
INBOUND	Sgt. William R. Button	4FDX	2	9/29/2000 12:00
ISEP	ISEP	ISEP	4	n/a

Figure 21: Objectified map with superimposed ***Item Status Report***
(drill-down detail for selected supply item).



SeaState: WARNING

Meteorological message received: Sea state ST_LEVEL_IV projected in 24 hours...

Figure 22: Sea State alert and explanation window
(note that the CIL agent and the Future Plan agent are also alerting).

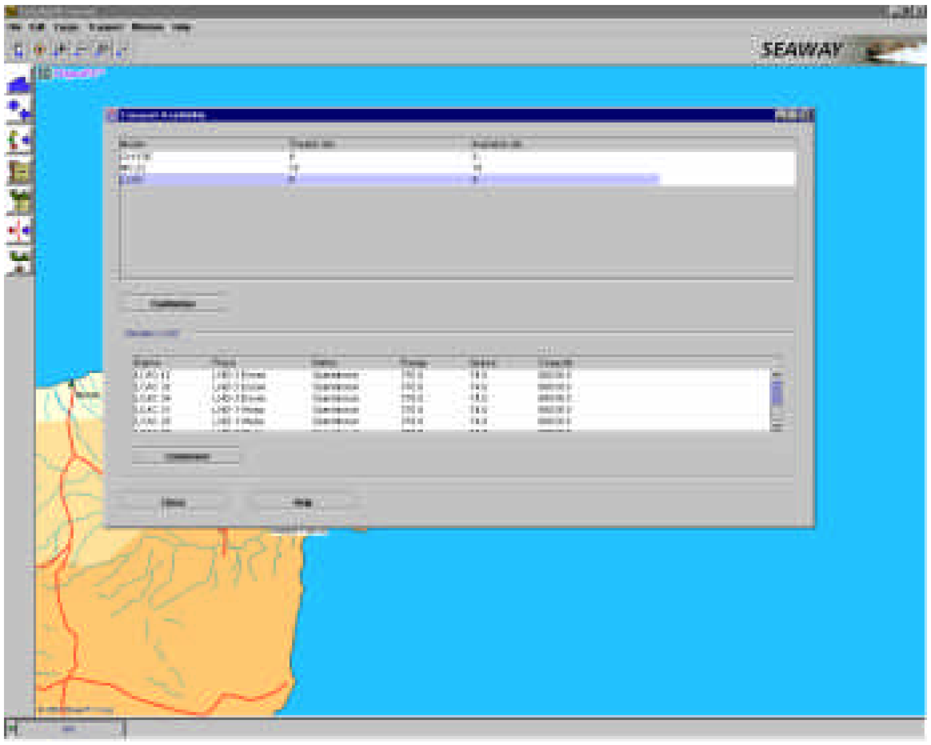


Figure 23: Objectified map with superimposed ***Transport Availability Report*** (drill-down detail of LCAC showing current location and operational status).

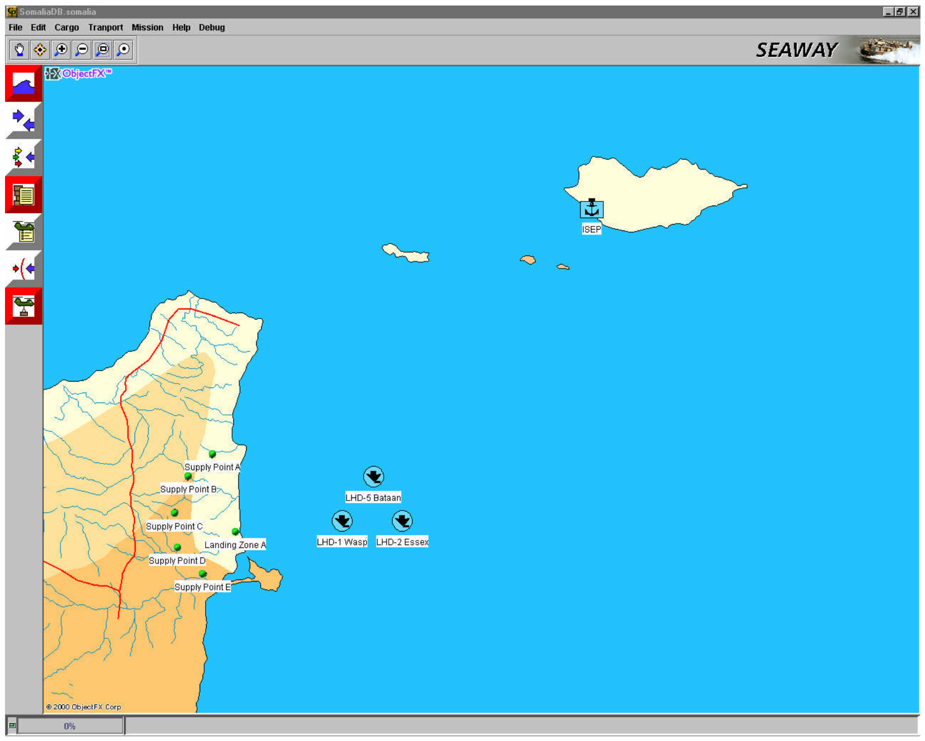


Figure 24: ***Requirements Template*** indicating current supply requests (drill-down detail for requesting unit at a particular Supply Point).

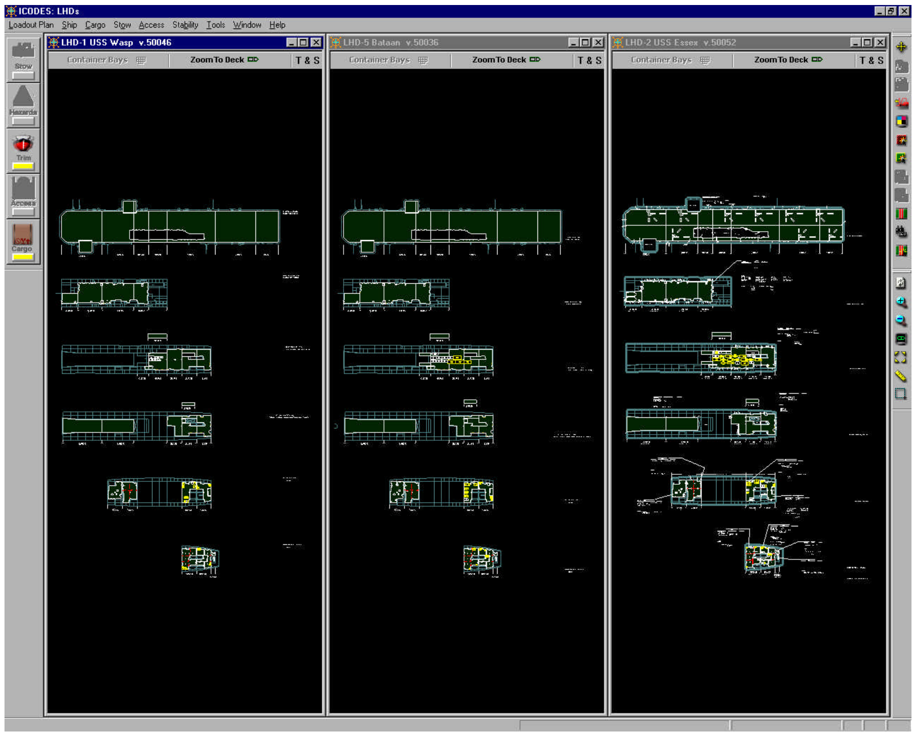
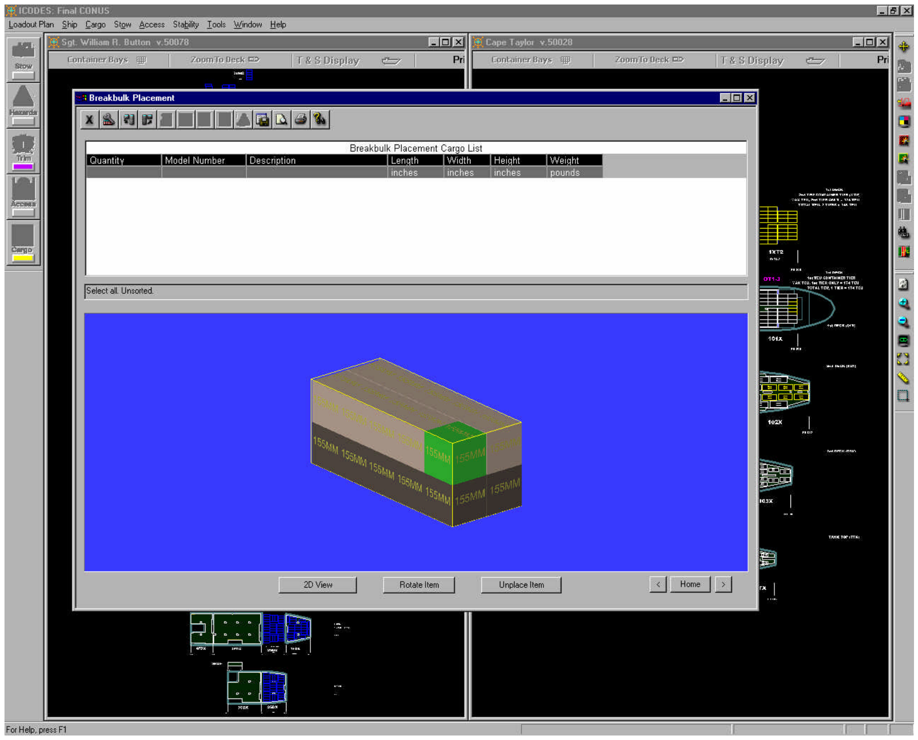


Figure 25: Seamless transition to ICODES showing the current stowed cargo layout on the three sea-based amphibious ships.

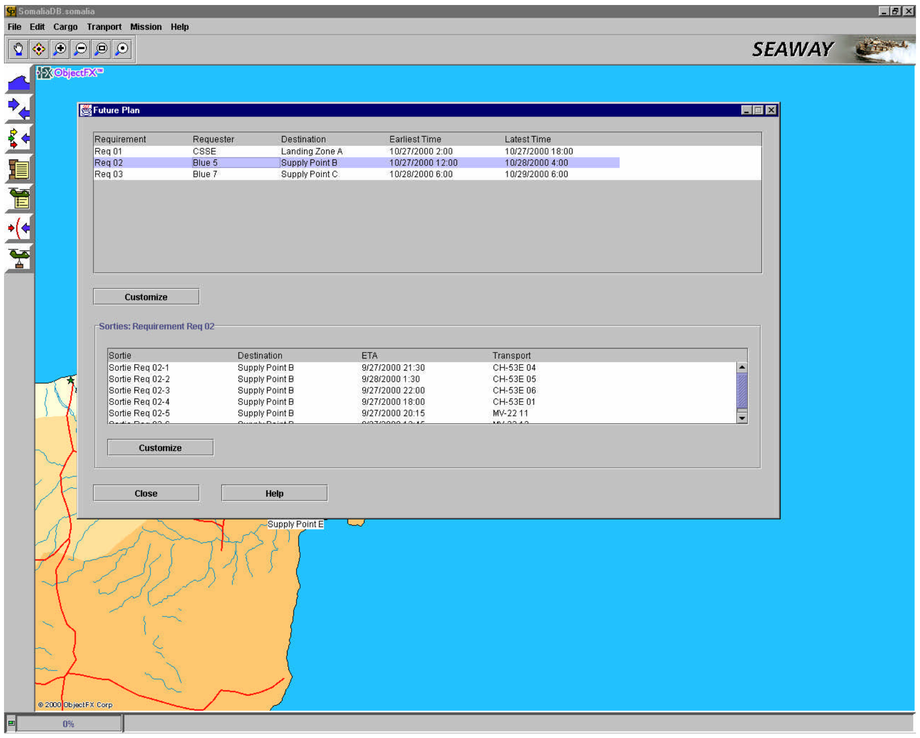


Breakbulk Placement Cargo List

Quantity	Model Number	Description	Length	Width	Height	Weight
			inches	inches	inches	pounds

Select all. Unsorted.

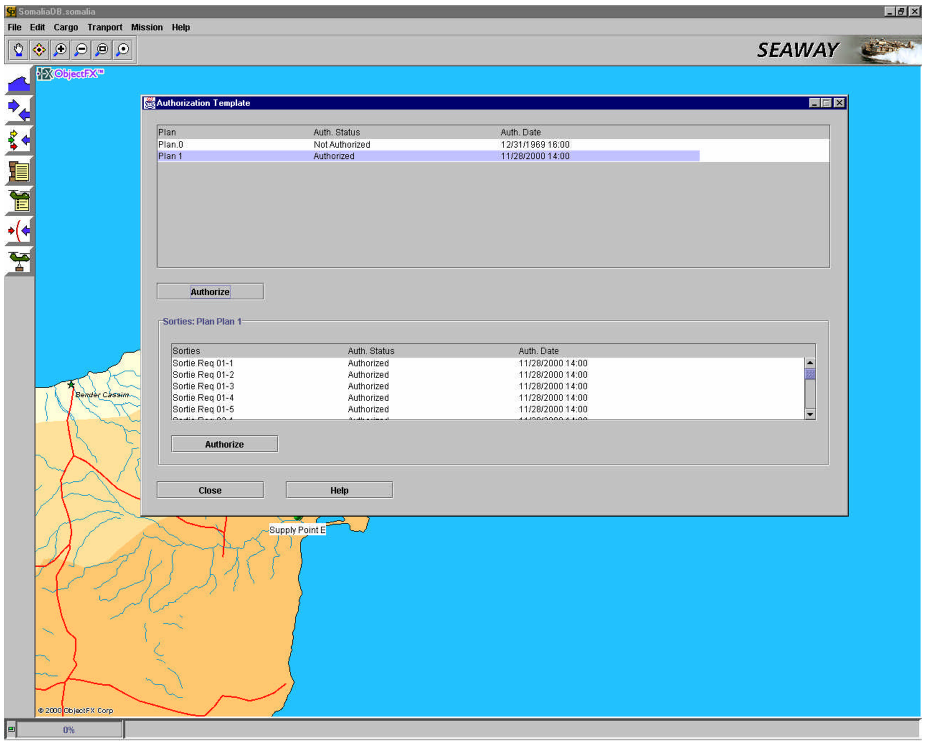
Figure 26: ICODES provides the ability to locate any specific supply item down to the level of the contents of a container.



Requirement	Requester	Destination	Earliest Time	Latest Time
Req 01	CSSE	Landing Zone A	10/27/2000 2:00	10/27/2000 18:00
Req 02	Blue 5	Supply Point B	10/27/2000 12:00	10/28/2000 4:00
Req 03	Blue 7	Supply Point C	10/28/2000 6:00	10/29/2000 6:00

Sortie	Destination	ETA	Transport
Sortie Req 02-1	Supply Point B	9/27/2000 21:30	CH-53E 04
Sortie Req 02-2	Supply Point B	9/28/2000 1:30	CH-53E 05
Sortie Req 02-3	Supply Point B	9/27/2000 22:00	CH-53E 06
Sortie Req 02-4	Supply Point B	9/27/2000 18:00	CH-53E 01
Sortie Req 02-5	Supply Point B	9/27/2000 20:15	MV-22 11

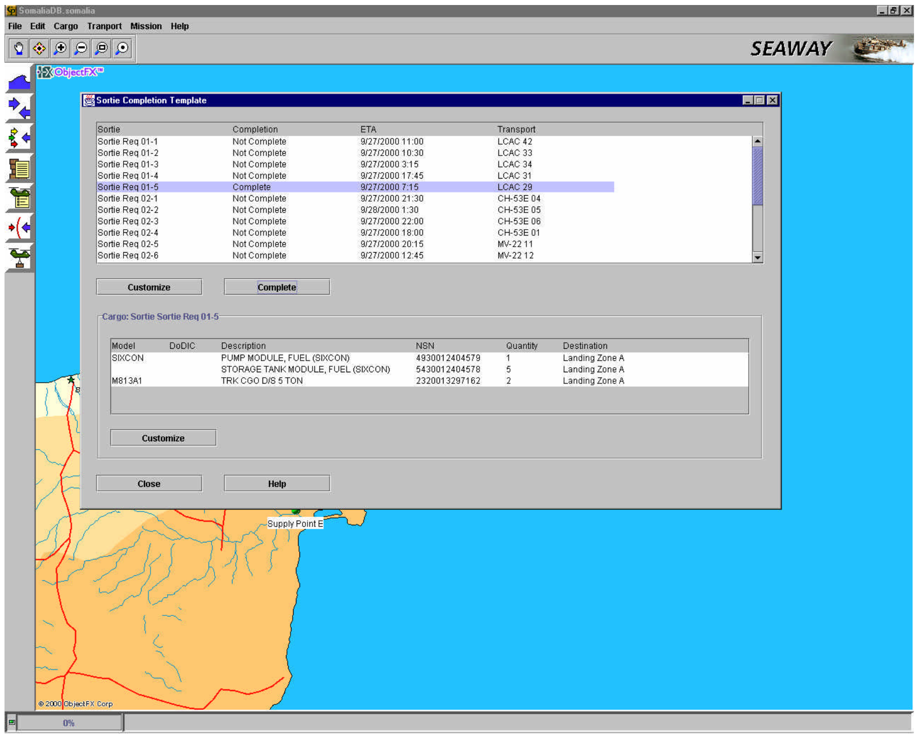
Figure 27: Objectified map with superimposed ***Future Plan Report*** (drill-down detail by Sortie showing ETA and proposed transport craft).



Plan	Auth. Status	Auth. Date
Plan.0	Not Authorized	12/31/1969 16:00
Plan 1	Authorized	11/28/2000 14:00

Sorties	Auth. Status	Auth. Date
Sortie Req 01-1	Authorized	11/28/2000 14:00
Sortie Req 01-2	Authorized	11/28/2000 14:00
Sortie Req 01-3	Authorized	11/28/2000 14:00
Sortie Req 01-4	Authorized	11/28/2000 14:00
Sortie Req 01-5	Authorized	11/28/2000 14:00

Figure 28: Objectified map with superimposed ***Authorization Template*** (any authorized portions of the Future Plan constitute the Execution Plan).



Sortie	Completion	ETA	Transport
Sortie Req 01-1	Not Complete	9/27/2000 11:00	LCAC 42
Sortie Req 01-2	Not Complete	9/27/2000 10:30	LCAC 33
Sortie Req 01-3	Not Complete	9/27/2000 3:15	LCAC 34
Sortie Req 01-4	Not Complete	9/27/2000 17:45	LCAC 31
Sortie Req 01-5	Complete	9/27/2000 7:15	LCAC 29
Sortie Req 02-1	Not Complete	9/27/2000 21:30	CH-53E 04
Sortie Req 02-2	Not Complete	9/28/2000 1:30	CH-53E 05
Sortie Req 02-3	Not Complete	9/27/2000 22:00	CH-53E 06
Sortie Req 02-4	Not Complete	9/27/2000 18:00	CH-53E 01
Sortie Req 02-5	Not Complete	9/27/2000 20:15	MV-22 11
Sortie Req 02-6	Not Complete	9/27/2000 12:45	MV-22 12

Cargo: Sortie Sortie Req 01-5

Model	DoDIC	Description	NSN	Quantity	Destination
SIXCON		PUMP MODULE, FUEL (SIXCON)	4930012404579	1	Landing Zone A
		STORAGE TANK MODULE, FUEL (SIXCON)	5430012404578	5	Landing Zone A
M813A1		TRK CGO D/S 5 TON	2320013297162	2	Landing Zone A

Figure 29: Objectified map with superimposed ***Sortie Completion Report*** (drill-down detail by Sortie for each supply item delivered).

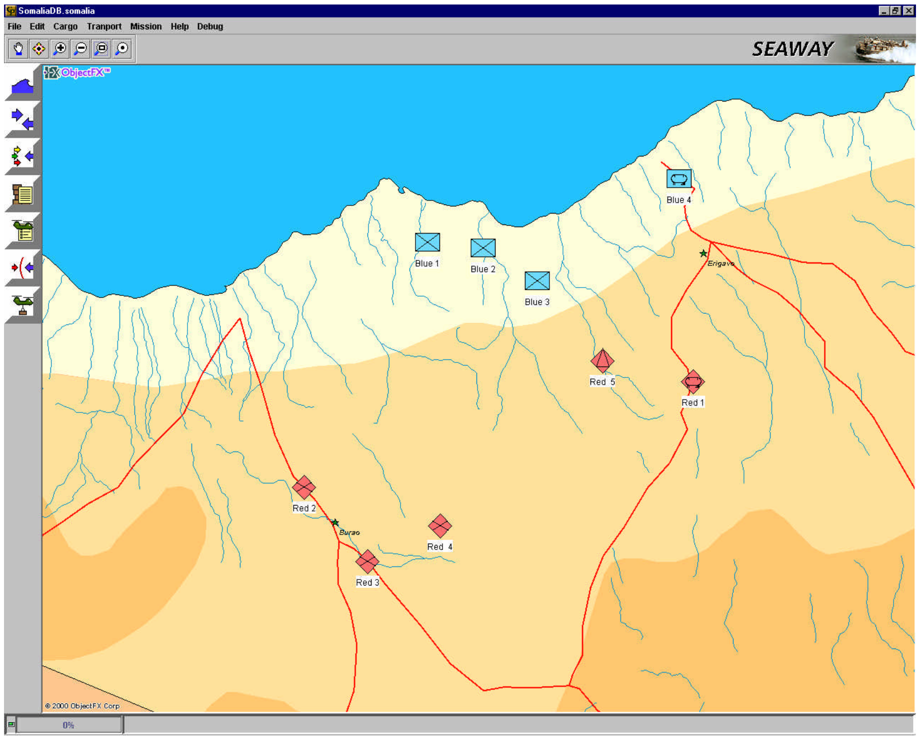
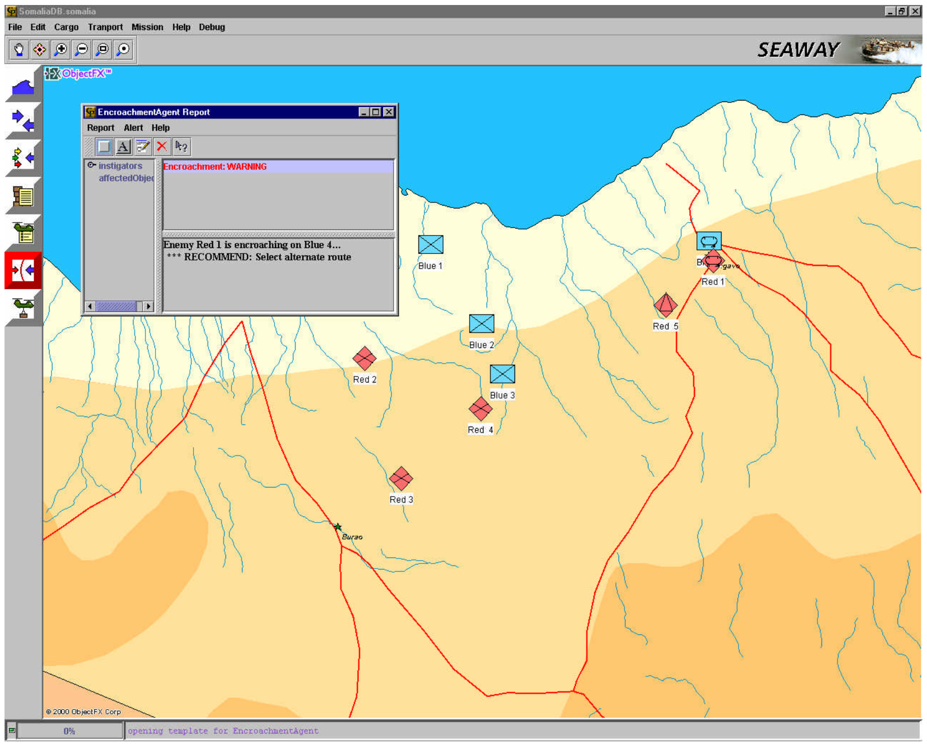


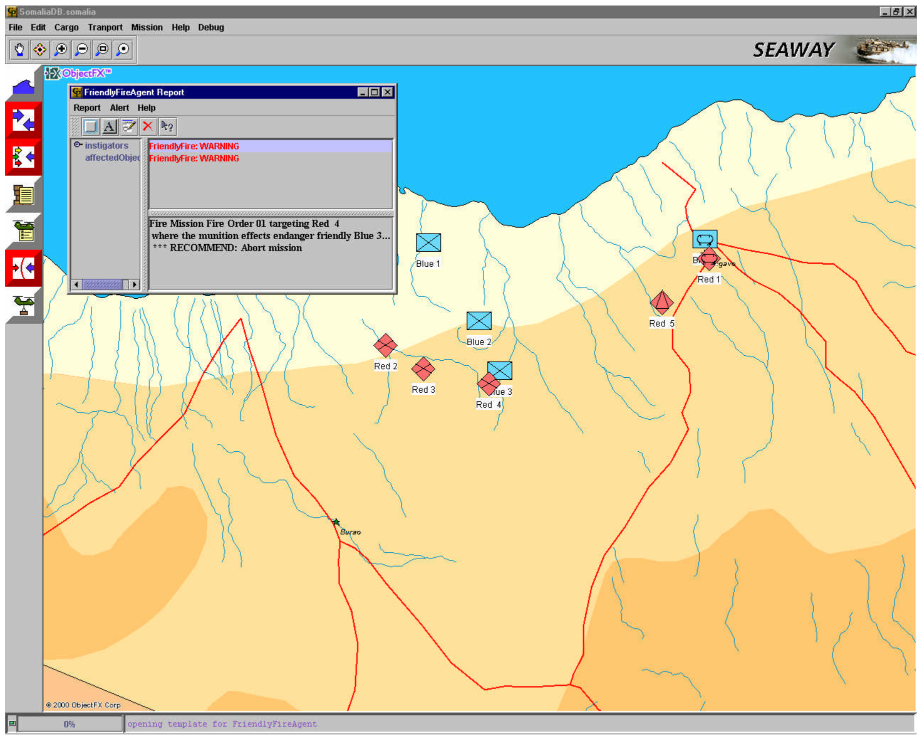
Figure 30: Seamless transition to ***Common Tactical Picture*** generated by FCADS (showing current location of friendly and enemy units).



Encroachment: WARNING

Enemy Red 1 is encroaching on Blue 4...
 *** RECOMMEND: Select alternate route

Figure 31: ***Encroachment Agent Alert*** indicating enemy unit encroaching on Supply Point and recommending the selection of an alternative route).



FriendlyFire: WARNING
FriendlyFire: WARNING

Fire Mission Fire Order 01 targeting Red 4
 where the munition effects endanger friendly Blue 3...
 *** RECOMMEND: Abort mission

Figure 32: ***Friendly Fire Agent Alert*** indicating fire mission in proximity of Supply Point and recommending "Abort Mission".

6.6 Using SEAWAY as a Training Tool

As discussed earlier in Section 6.1, SEAWAY represents a set of tools that collaborate with each other and human users to solve problems. Among these the agents are the most sophisticated and useful tools, principally due to their ability to spontaneously and opportunistically communicate with the world that is external but nevertheless related to their immediate knowledge domains. However, ***the communication from agent to agent is not direct***. In several respects this must be viewed as one of the most powerful features of the SEAWAY architecture. First, this obviates the need for one agent to anticipate what another agent might contribute to the problem solving process. In other words, each agent operates independently within the community of agents. Similar to our human world in which each person operates with a high degree of freedom, in SEAWAY the agents are driven by the information that they receive and send to the object-serving collaboration facility (see Sections 2.1 and 3.1) . This creates an opportunistic and potentially adaptive environment, because the impact on the agent community of the information that passes through the Information Server is only partially predictive.

Second, the absence of any direct linkages between agents greatly simplifies the design of each agent, by essentially eliminating the need to consider interdependencies among agents. Each agent constitutes a clearly defined set of capabilities that the agent can and will exercise whenever the appropriate information becomes available. It is obviously much easier to design these capabilities in isolation, without having to consider their potential influence and impact on other parts of the system. In SEAWAY each agent is an observer that watches the world represented in the object instance store and contributes to this world subject to its interests and capabilities.

Third, the SEAWAY architecture considers every component, whether agent or user-interface, to be a client to the object-serving collaboration facility. This important concept leads to an entirely ***open architecture*** that allows components to be readily added or deleted. Therefore, SEAWAY can be viewed as an extensible environment in which no component (i.e., with the exception of the object-serving collaboration facility) is indispensable, nor is any potential new component unacceptable. There is, however, a need for a common language, in the same way as that need exists in our human world. In SEAWAY this need is satisfied by the ontology, implemented as an object model.

These fundamental characteristics allow SEAWAY to function equally well as a planning, execution and training environment, and support these functions concurrently. The same functionality that is available during execution operations is available during training. In other words, the map interface, the template interface, agent analysis, and domain databases all function in the same manner, with the same capabilities, during a training session as they will during actual operations. However, unlike execution mode in which SEAWAY is mostly stimulated by external data feeds from the sea base and the battlefield, a training session may be driven by: the Scenario Driver; trainee users; trainer users; or, any combination of all of these data feeds.

Script Driven: Whether for demonstration or training purposes a scenario file or multiple scenario files may be created to depict actual battlefield situations. The creation, destruction, modification, or movement of sea base entities and supply requests are scripted, recorded, and then used to stimulate the system. Operators, acting as a single or multiple SEAWAY clients then react and contribute to the situation. Driving SEAWAY in this manner, with live operators interacting with the system, allows tactics, techniques, and procedures to be tested, as well as providing the operators valuable opportunities for refining their decision-making skills. Another positive aspect of using SEAWAY for training is that the operators are honing their skills on the same system that they are using for planning and execution.

Gaming Mode: SEAWAY may also be used in a highly dynamic mode allowing multiple operators to engage each other in a ***gaming*** session. While this mode of operation is only marginally supported in SEAWAY V.0, it is planned to become the principal focus of SEAWAY V.1 in October 2001. This mode of operation can also be augmented with Scenario Driver scripts inserted by an independent party to add uncertainty of operations (e.g., weather effects or changes in transport craft availability).

7. Definition of SEAWAY Terms

The following definitions apply to the specific meaning of terms used in the SEAWAY Client User Interface and in any documentation relating to the SEAWAY application.

authorization template	A SEAWAY ***message template*** that provides information about one or more decisions relating to either the selection of an alternative course of action generated by SEAWAY, or a course of action devised by a user external to SEAWAY.
back-haul	Pick-up of ***cargo*** from a ***supply point*** for transportation to some other ***supply point***.
breakbulk	Supply items that are not large end-items (e.g., ammunition, food, lumber, most items that are stowed in containers, etc.).
cargo	Any combination of ***supplies***, ***equipment*** and ***personnel***.
commodities	Subcategories of some ***supply classes***.
completion template	A SEAWAY ***message template*** that provides information about the completion of a ***sortie*** or ***mission***.
container	A specialized ***cargo*** holding unit (i.e., ***stow area***) that is capable of containing certain types of ***supplies*** and/or ***equipment*** (e.g., an A1M1 tank cannot be accommodated in a ***container***). SEAWAY supports three types of ***containers***, as follows: Milvan (i.e., commercial container); sea-shed; and, pallet.
conveyance	A facility with movement and storage (temporary) capabilities that can be used to transport ***cargo***. SEAWAY V.1 will recognize only sea and air ***conveyances***. Sea ***conveyances*** include supply ships, tactical ships, and sea surface ***transport*** craft. Air ***conveyances*** include helicopters and fixed-wing aircraft.
day of ammunition	Also referred to as DOA. A measurement equal to the amount of ammunition required per day, used to translate the narrative statement of Commander's Intent into a quantified statement of Logistic Intent.

day of supplies	Also referred to as DOS. A measurement equal to the amount of supplies required per day, used to translate the narrative statement of Commander's Intent into a quantified statement of Logistic Intent.
equipment	Principal end-items (e.g., trucks, tanks, weapons, etc.) that are used by ***units***.
execution plan	The 'authorized' component elements of all existing ***future plans***. The ***execution plan*** constitutes the current execution activity.
future plan	All component elements of a ***plan*** that are currently designated as 'not authorized' or 'under revision'. SEAWAY must be able to support multiple Future Plans. Any portion of any Future Plan may be authorized to form part of the Execution Plan (i.e., there is only one Execution Plan). Therefore, while each Future Plan is separate, the authorized portions of any Future Plan constitute the current Execution Plan. The generation of a Future Plan is always initiated by a user.
in-bound	All cargo items that are currently on ships in-bound to the sea-base or ISEP (these are usually commercial ships leased by MTMC).
lighterage	See ***transport***.
location	Currently all locations for ***points*** are expressed either as 8-digit grid positions or as latitude/longitude positions.
message template	A SEAWAY user-interface window of predefined format that allows a user (i.e., CSSE) to enter information relating to a specific functional sequence. Wherever possible the entered information will be in the form of values that are either numerical or selected from a set of enumerations, rather than free-format text.
mission	The delivery of supplies to a single ***supply point***. ***Missions*** are unit-based and location-based . In other words, a single ***mission*** may involve more than one ***sortie*** but cannot involve more than one ***offload point***.
mission mode	A code used in the Logistic Intent to indicate the nature of the delivery stream (e.g., 'C' indicates that the deliveries must be scheduled as a continuous flow until

	the requirement has been met; 'S' indicates that all items must be delivered simultaneously (or nearly so) demanding multiple ***transport*** craft operating in parallel).
mission window	A period beginning with the earliest acceptable delivery date/time (dtg) and ending with the latest acceptable delivery date/time (dtg).
offload point	A ***supply point*** where ***cargo*** is required to be delivered. Since SEAWAY will not deal with the distribution of supplies within the battlefield, it can be assumed that all ***offload points*** are CSSE locations.
on-hand	All cargo items that are currently in the sea-base (i.e., does not include items that are located at the ISEP or items that are in-bound).
on-shore	All cargo already delivered to ***supply points*** or currently on ***transport*** craft en route to ***supply points***.
package	***Supplies*** (i.e., one or more supply items) and/or ***equipment*** packaged together.
packaging	Indicates the form in which the ***cargo*** will be delivered (e.g., 'P' indicates that the ***cargo*** will be palletized.
pallet	A metal or wooden tray (i.e., 40 in by 48 in) weighing approximately 100 lb. Used to package cargo for loading onto ***transport*** craft.
personnel	Troops that compose a military force.
phase	Portion or subset of a ***plan***.
plan	A basic entity (i.e., object) in SEAWAY that describes a course of action for fulfilling the requirements contained in a ***statement of requirements***. ***Plans*** are normally divided into ***phases***.
plan template	The collective information in SEAWAY that defines a course of action (i.e., a ***plan***) for fulfilling a ***statement of requirements***. Specifically, a ***plan template*** defines authorization and completion status, ***time delivery windows***, ***sorties***, ***missions***, ***routes*** and ***cargo***, by ***phase***.

point	A definable location on the surface of the earth or in the air at which either ***cargo*** is picked up or dropped off, or a change of course of a ***conveyance*** may take place. SEAWAY will support the following types of ***points***: ***supply points*** (where ***cargo*** may be picked up or dropped off (i.e., ***offload point***)) ***way-points*** (where ***conveyances*** may change direction to circumvent some potential obstacle)
priority	A set of integer numbers from 0 to 4 indicating the importance of the ***cargo*** to be delivered. A ***sortie*** can have more than one type of ***cargo*** aboard. However, its place in the delivery queue is determined by the highest ***priority*** of the ***cargo*** onboard.
reach-back	The process of ordering/requesting ***cargo*** that is not currently available in the ***sea-base***, from land-based ***warehouses*** or commercial suppliers.
request template	A SEAWAY ***message template*** that specifies ***cargo*** requirements that constitute additions, deletions or changes to the current ***statement of requirements*** (i.e., an amendment to the ***statement of requirements***). SEAWAY V.1 will assume that all ***request template*** messages have been validated by the CSSE prior to entry into SEAWAY.
requirements template	Also referred to as a ***statement of requirements***. A SEAWAY ***message template*** that specifies ***cargo*** requirements by ***plan***, ***phase*** and ***unit***. Each ***statement of requirements*** will require the preparation of a new ***plan***. Therefore, only one ***statement of requirements*** is entered into SEAWAY for each ***plan***.
retrograde	Movement of supplies, equipment and personnel from the shore back to the sea base.
route	The physical path taken by a ***sortie***. A ***route*** consists of route-legs between ***supply points*** and/or ***way-points***.
sea-base	General term referring to all of the vessels at sea that are serving the supply needs of a task force (i.e., typically supply ships).

SIXCON	Fuel and fuel pump container modules (i.e., 96 in by 80 in by 48 in (height)), about one sixth of the size of a standard 20 ft MILVAN container. A SIXCON consists of six modules (i.e., five fuel modules of fuel (900 gal each) and one module for the pump) that can be stacked together in three stacks of two, to be the same size as a 20 ft container (i.e., 8 ft by 8 ft by 20 ft). A SIXCON holding 4500 gal of fuel and one pump unit can be transported by truck or helicopter. Each fuel module weighs about 9550 lb and a pump unit weighs about 4000 lb.
sortie	Transportation of ***cargo*** to one or more ***supply points***. Therefore, a ***sortie*** typically involves multiple ***missions***. Each ***sortie*** is a single round trip from ***pickup point*** (normally, but not necessarily the sea-base) to shore and return to the sea-base. However, multiple ***sorties*** may be required to fulfill the supply requests for a single ***offload point*** and therefore a single ***mission***. In this respect ***sortie***s refer to transportation and delivery.
standard load	A proposed SEAWAY innovation consisting of a set of standard pallet loads composed of various mixes of ***cargo*** that would be used to speed up operations aboard the sea-base. The mix of ***cargo*** for each ***standard load*** would need to be established.
standard supply package	These supply packages provide sustainability for a set period of time (i.e., days of supplies). Standard Supply Packages (SSPs) consist of specified quantities of fuel, munitions, water, and food, tailored by ***unit*** type. The content of each SSP type is constant. For example, over 200 pallets are required for a one-day supply of ammunitions to an artillery battery. SEAWAY will support separate SSPs for food, water, ammunitions, and fuel.
statement of requirements	The combination of requirements contained in the original ***requirements template*** and any subsequent requests that may have been received for the ***future plan*** under consideration through the entry of ***request template*** messages.
stow area	A physical space that is capable of accommodating the storage of ***cargo***. A ***stow area*** may be located on a ***conveyance***, at a ***supply point***, in a ***warehouse***, or within another ***stow area***.

supplies	Items (i.e., supply items) that are consumed or used in support of personnel and equipment (e.g., food, fuel, ammunition, munitions and spare parts).
supply class	All ***cargo*** items belong to one of nine standard ***supply classes***.
supply point	A basic entity (i.e., object) in SEAWAY that describes a physical location where ***cargo*** (i.e., ***supplies***, ***equipment*** and/or ***personnel***) can be picked up or dropped off. There are two types of ***supply points***, as follows: ***pickup point*** – where ***cargo*** is loaded onto a ***conveyance*** for subsequent delivery; ***offload point*** – where ***cargo*** is unloaded and delivered.
time delivery window	Every ***plan***, ***phase***, ***sortie***, and ***mission*** has an 'earliest' and 'latest' execution time.
transport	Any transport craft such as helicopters (V-22, CH53E and CH46), Landing Craft Air Cushion (LCAC), Landing Craft Utility (LCU), barges, Side-Loading Warping Tugs (SLWT), and causeways.
unit	A military organizational entity consisting of a certain number of ***personnel*** and ***equipment***. Typical examples of standard military ***units*** in ascending order of size are squad, platoon, company, battalion, regiment, and division.
warehouse	A storage facility (e.g., building(s), tent(s), open yard(s)) on land. In SEAWAY ***warehouses*** are considered as supply sources for ***reach-back*** purposes only. Therefore, generally SEAWAY will not monitor supply types and quantities in ***warehouses***. However, SEAWAY will monitor supply types and quantities in ***warehouses*** located at the ISEP that are associated with supplying the sea-base
way-point	A designated point in space that indicates a course change. ***Way-points*** may be inserted in ***routes*** (in between ***supply points*** or ***way-points***) to circumvent enemy actions or other adverse conditions (e.g., storms,

plumes, etc.) that may exist between ***supply points*** and/or ***way-points***.

8. Glossary of Acronyms

ACE	Air Combat Element
ACU	Assault Craft Unit
AI	Artificial Intelligence
ARG	Amphibious Ready Group
A-SRP	Altillery-Standard Re-supply Package
BLT	Battalion
C2	Command and Control
CE	Command Element
CFF	Call For Fire
CIL	Commander's Critical Items List (It should be noted that cargo items that are on the commander's Critical Items List (CIL) are not necessarily high priority items. Rather they are items that need to be monitored because they are of critical importance to the commander's plan and intent.)
CINC	Commander IN Chief
CLF	Combat Logistics Force
COMMARFOR	MARFOR Commander
COMNAVFOR	NAVFOR Commander
COMPHIBGRU	PHIBGRU Commander
CONUS	Continental United States of America
CVBG	Carrier Battle Group
CSSE	Combat Service Support Element
CTP	Common Tactical Picture
DBMS	Database Management System
DOA	Day Of Ammunition
DOS	Day Of Supplies
EWTGLANT	Expeditionary Warfare Training Group Atlantic
EWTGPAC	Expeditionary Warfare Training Group Pacific
GCE	Ground Combat Element
ICODES	Integrated Computerized Deployment System
IMMACCS	Integrated Marine Multi-Agent Command and Control System
ISEP	Intermediate Support and Embarkation Port
I-SRP	Infantry-Standard Re-Supply Package
JCS	Joint Chiefs of Staff
JFC	Joint Force Commander
JTF	Joint Task Force

LAV	Light Armored Vehicle
LCAC	Landing Craft Air Cushion
LCU	Landing Craft Unit
LFORM	Landing Force Operations Ready Materiel
LHD	Landing Helo Dock (amphibious assault ship)
MAGTF	Marine Air Ground Task Force
MARFOR	Marine Force Component (of a Joint Task Force)
MOB	Mobile Off-Shore Base (very large modular pontoon platform)
MPF	Maritime Prepositioning Force
NAVFOR	Navy Force Component (of a Joint Task Force)
OMFTS	Objective Maneuver From The Sea
OODBMS	Object-Oriented Database Management System
PHIBGRU	Amphibious Group
QUADCON	Fuel and fuel pump container modules (i.e., 96 in by 80 in by 48 in (height)), about one ninth of the size of a standard 20 ft MILVAN container. A QUADCON consists of four modules (i.e., three fuel modules of fuel (900 gal each) and one module for the pump) that can be stacked together in two stacks of two. (see also ***SIXCON***)
SIXCON	Fuel and fuel pump container modules (i.e., 96 in by 80 in by 48 in (height)), about one sixth of the size of a standard 20 ft MILVAN container. A SIXCON consists of six modules (i.e., five fuel modules of fuel (900 gal each) and one module for the pump) that can be stacked together in three stacks of two, to be the same size as a 20 ft container (i.e., 8 ft by 8 ft by 20 ft). A SIXCON holding 4500 gal of fuel and one pump unit can be transported by truck or helicopter. Each fuel module weighs about 9550 lb and a pump unit weighs about 4000 lb.
SRP	Standard Re-Supply Package
SSP	Standard Supply Package
STOM	Ship to Objective Maneuver
TACLOG	Tactical Logistics Coordination Center (sea-based)
TDW	Time Delivery Window
T-SRP	Tank-Standard Re-Supply Package
WBM	White Board and Messaging facility

9. References and Bibliography

Ayers D., H. Bergsten, M. Bogovich, J. Diamond, M. Ferris, M. Fleury, A. Halberstadt, P. Houle, P. Mohseni, A. Patzer, R. Phillips, S. Li, K. Vedati, M. Wilcox, and S. Zeiger (1999); 'Professional Java Server Programming'; Wrox Press, Birmingham, UK.

Bancilhon F., C. Delobel and P. Kanellakis (eds.) (1992); 'Building an Object-Oriented Database Systems'; Morgan Kaufmann, San Mateo, California.

CADRC (1994); 'ICODES Proof-of-Concept System: Final Report'; Contract #: N47408-93-7347, Naval Civil Engineering Laboratory (Port Hueneme, California), Collaborative Agent Design (CAD) Research Center, Cal Poly, San Luis Obispo, California.

Fogel D. (2000); 'What is Evolutionary Computation?'; IEEE Spectrum, February.

Forgy C. (1982); 'Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem'; Artificial Intelligence, Vol.19 (pp.17-37).

Fowler M. and K. Scott (1997); 'UML Distilled: Applying the Standard Object Modeling Language'; Addison-Wesley, Reading, Massachusetts.

Giarratano J. and G. Riley (1994); 'Expert Systems: Principles and Programming'; 2nd edition, PWS Publishing Company, Boston, Massachusetts.

Gray S. and R. Lievano (1997); 'Microsoft Transaction Server 2.0'; SAMS Publishing, Indianapolis, Indiana.

Gue K. (2000); 'A Dynamic Distribution Model for Combat Logistics'; Research Report, Department of Systems Management, Naval Postgraduate School, Monterey, California, January (research funded by Office of Naval Research, US Navy).

Hamber R. (1998); 'TloaDS manual – Phase I'; Technical Report, Naval Facilities Engineering Services Center, Port Hueneme, California.

Hayes-Roth F., D. Waterman and D. Lenat (eds.) (1983); 'Building Expert Systems'; Addison-Wesley, Reading, Massachusetts.

Hormozi A. and B. Khumawala (1996); 'An Improved Algorithm for Solving a Multi-Period Facility Location Problem'; IIE Transactions, vol.28 (pp. 105-114).

IONA (1996); 'Orbix Web: Programming Guide'; IONA Technologies Ltd., Dublin, Ireland.

Jennings N., K. Sycara and M. Wooldridge (1998); 'A Roadmap of Agent Research and Development'; Autonomous Agents and Multi-Agent Systems, Vol.1 (pp.7-38).

Kang K. and K. Gue (1997); 'Sea Based Logistics: Distribution Problems for Future Global Contingencies'; in Proc. 1997 Winter Simulation Conference.

Kennedy J. and R. Eberhart (1995); 'Particle Swarm Optimization'; in Proc. IEEE International Conference on Neural Networks, IEEE Service Center, Piscataway, New Jersey, (pp.1942-1948).

Lewis B. and D. Berg (1996); 'Threads Primer: A Guide to Multithreaded Programming'; SunSoft Press, Mountain View, California.

Luss H. (1982); 'Operations Research and Capacity Expansion Problems: A Survey'; Operations Research, vol.30(5), (pp. 907-947).

Minsky M. (1990); 'Logical vs. Analogical or Symbolic vs. Connectionist or Neat vs. Scruffy'; in Winston and Shellard (eds.) Artificial Intelligence at MIT, vol.1, MIT Press, Cambridge, Massachusetts, (pp. 218-243).

Mowbray T. and R. Zahavi (1995); 'The Essential CORBA: Systems Integration Using Distributed Objects'; John Wiley and Sons, Inc., New York, New York.

Myers L. and J. Pohl (1994); 'ICDM: Integrated Cooperative Decision Making - in Practice'; 6th IEEE International Conference on Tools with Artificial Intelligence, New Orleans, Nov. 6-9.

NASA (1992); 'CLIPS 6.0 Reference Manual'; Software Technologies Branch, Lyndon B. Johnson Space Center, Houston, Texas.

Orfali R., D. Harkey and J. Edwards (1996); 'The Essential Distributed Objects Survival Guide'; John Wiley and Sons, Inc., New York, New York.

Penmetcha K., A. Chapman and A. Antelman (1997); 'CIAT: Collaborative Infrastructure Assessment Tool'; in Pohl J (ed.) Advances in Collaborative Design and Decision-Support Systems, focus symposium: International Conference on Systems Research, Informatics and Cybernetics, Baden-Baden, Germany, August 18-22, (pp. 83-90).

Pohl J., A. Chapman, K. Pohl, J. Primrose and A. Wozniak (1997); 'Decision-Support Systems: Notions, Prototypes, and In-Use Applications'; Technical Report, CADRU-11-97, Collaborative Agent Design (CAD) Research Center, Cal Poly, San Luis Obispo, California, January.

Pohl J. (1997); 'Human-Computer Partnership in Decision-Support Systems: Some Design Guidelines'; in Pohl J. (ed.) Advances in Collaborative Design and Decision-Support Systems, focus symposium: International Conference on Systems Research, Informatics and Cybernetics, Baden-Baden, Germany, August 18-22, (pp.71-82).

Pohl J. (1999); 'Collaborative Decision-Support and the Human-Machine Relationship'; Office of Naval Research Workshop, Collaborative Agent Design (CAD) Research Center, San Luis Obispo, California, April 20-22, (pp.21-46).

Pohl J., M. Porczak, K.J. Pohl, R. Leighton, H. Assal, A. Davis, L. Vempati and A. Wood, and T. McVittie (1999); 'IMMACCS: A Multi-Agent Decision-Support System'; Technical Report, CADRU-12-99, Collaborative Agent Design (CAD) Research Center, Cal Poly, San Luis Obispo, California, August.

Pohl K. (1995); 'KOALA: An Object-Agent Design System'; in Pohl J (ed.) Advances in Cooperative Environmental Decision Systems, focus symposium: International Conference on Systems Research, Informatics and Cybernetics, Baden-Baden, Germany, August 14-18, (pp. 81-92).

Pohl K. (1997); 'ICDM: A Design and Execution Toolkit for Agent-Based, Decision-Support Applications'; in Pohl J. (ed.) Advances in Collaborative Design and Decision-Support Systems, focus symposium: International Conference on Systems Research, Informatics and Cybernetics, Baden-Baden, Germany, August 18-22 (pp.101-110).

Shulman A. (1991); 'An Algorithm for Solving Dynamic Capacitated Plant Location Problems with Discrete Expansion Sizes'; Operations Research, vol.39(3), (pp.423-436).

Sun Microsystems (1999); 'Java Shared Data Toolkit User Guide'; Version 2.0(FCS), October 4, JavaSoft, 2550 Garcia Avenue, Mountain View, CA 94043.

Sweeney D. and R. Tatham (1976); 'An Improved Long-Run Model for Multiple Warehouse Location'; Management Science, vol.22(7), (pp. 748-758).

Wooldridge M. and N. Jennings (1995); 'Intelligent Agents: Theory and Practice'; The Knowledge Engineering Review, 10(2) (pp.115-152).

10. Appendix: SEAWAY V.0 Ontology

Seaway Ontology Report

Date : October 11, 2000
Time : 5:32:59 PM

Enumerations:

- app.eDay
- app.eMonth
- app.eProtocol
- app.Component.eHealthStatus
- app.design.Action.eActionStatus
- app.design.Design.eAuthStatus
- app.view.Component.eSpecificity
- information.Requirement.eReqStatus
- operation.Sortie.eAuthStatus
- operation.Sortie.eExeStatus
- operation.logistical.eSupplyClass
- operation.logistical.CargoTransaction.eCargoTransType
- operation.logistical.Requirement.eEffortType
- physical.Physical.eForceCode
- physical.Physical.eAffiliation
- physical.asset.eStowAccessibilityType
- physical.asset.AirWeapon.eAirWeaponType
- physical.asset.Aircraft.eAircraftType
- physical.asset.ArmoredVehicle.eArmoredVehicleType
- physical.asset.FixedWing.eFixedWingMilAirType
- physical.asset.MilitaryAircraft.eMilitaryAircraftType
- physical.asset.RotaryWing.eRotaryWingType
- physical.asset.RotaryWing.eRotaryWingMilAirType
- physical.asset.Sensor.eSensorType
- physical.asset.Tank.eLoaderType
- physical.asset.Vehicle.eVehicleType
- physical.asset.VehicleTrailer.eVehTrailerType
- physical.asset.Vessel.eVesselType
- physical.asset.Vessel.eHullType
- physical.asset.Weapon.eWeaponType
- physical.body.Body.eLoyalty
- physical.body.CombatServiceSupportUnit.eCSSUnitType
- physical.body.CombatSupportUnit.eCombatSupUnitType

- physical.body.CombatUnit.eCombatUnitType
- physical.body.GroundUnit.eGroundUnitType
- physical.body.MilitaryOrganization.eEchelon
- physical.body.Organization.eOrgSize
- physical.body.Organization.eOrgType
- physical.body.Person.ePersonStatus
- physical.body.SpecialOps.eSpecialOpsType
- physical.consumable.Cartridge.eCartridgeType
- physical.consumable.Missile.eMissileType
- physical.consumable.POL.ePOLType
- physical.consumable.Pyrotechnics.ePyrotechnicType
- physical.consumable.UnguidedMunitions.eUnguidedMunType
- physical.environment.BodyOfWater.eSeaState
- physical.item.Item.eItemStatus
- physical.item.Item.eItemLocation
- physical.item.ItemContainer.eContainerType
- physical.weather.WeatherSystem.eWeatherSystemType

Structs:

- app.DateTimeInterval
- app.s2DDims
- app.s3DDims
- app.sDate
- app.sDuration
- app.sPosition
- app.sPriority
- app.sTime
- operation.logistical.sItemDeliveryStatusByNSN
- operation.logistical.sItemPopByLocation
- operation.logistical.sItemQuantity

Classes:

- Container
- ICDMBase
- ICDMObject
- app.Application
- app.Calender
- app.Clock
- app.Component
- app.DateTime

- app.Suite
- app.agent.AgentReport
- app.agent.BaseAgent
- app.agent.Emergency
- app.agent.Explanation
- app.agent.FYI
- app.agent.Facilitator
- app.agent.Guardian
- app.agent.Mentor
- app.agent.ObjectAgent
- app.agent.Recommendation
- app.agent.ServiceAgent
- app.agent.Warning
- app.design.Action
- app.design.Component
- app.design.Design
- app.design.Phase
- app.view.Component
- app.view.Empowerable
- app.view.SubView
- app.view.View
- information.ImageLayer
- information.Information
- information.Layer
- information.MapLayer
- information.Report
- information.Request
- information.RequestForReport
- information.Requirement
- operation.Activity
- operation.ActivityPt
- operation.Sortie
- operation.Trip
- operation.logistical.CargoTransaction
- operation.logistical.ConsolidatedReqReport
- operation.logistical.CriticalItemList
- operation.logistical.CyclicRequirement
- operation.logistical.ItemAvailabilityReport
- operation.logistical.PickupRequirement
- operation.logistical.Plan
- operation.logistical.Requirement

- operation.logistical.TransportAvailabilityReport
- operation.logistical.envItemPopulationReport
- operation.logistical.itemStatusReport
- operation.tactical.CallForFire
- operation.tactical.Deployment
- operation.tactical.FiringOrder
- operation.tactical.FiringSolution
- operation.tactical.Plan
- physical.ArtificialMobile
- physical.Mobile
- physical.NaturalMobile
- physical.Physical
- physical.asset.AirWeapon
- physical.asset.Aircraft
- physical.asset.ArmoredVehicle
- physical.asset.Asset
- physical.asset.Equipment
- physical.asset.FixedWing
- physical.asset.GuidanceSystem
- physical.asset.MilitaryAircraft
- physical.asset.Navigation
- physical.asset.Platform
- physical.asset.Radar
- physical.asset.RotaryWing
- physical.asset.Sensor
- physical.asset.Tank
- physical.asset.Vehicle
- physical.asset.VehicleTrailer
- physical.asset.Vessel
- physical.asset.Weapon
- physical.body.Body
- physical.body.CivilianOrganization
- physical.body.CombatServiceSupportUnit
- physical.body.CombatSupportUnit
- physical.body.CombatUnit
- physical.body.GroundUnit
- physical.body.MilitaryOrganization
- physical.body.Organization
- physical.body.Person
- physical.body.SpecialOps
- physical.consumable.Ammunition

- physical.consumable.Cartridge
- physical.consumable.Construction
- physical.consumable.Consumable
- physical.consumable.GeneralSupportItems
- physical.consumable.GuidedMunitions
- physical.consumable.HandGrenade
- physical.consumable.MedicalMaterial
- physical.consumable.Missile
- physical.consumable.Munitions
- physical.consumable.POL
- physical.consumable.PersonalDemandItems
- physical.consumable.Pyrotechnics
- physical.consumable.RepairItems
- physical.consumable.Subsistence
- physical.consumable.UnguidedMunitions
- physical.environment.ArtificialEnvironment
- physical.environment.Base
- physical.environment.BodyOfWater
- physical.environment.BuiltupArea
- physical.environment.Environment
- physical.environment.ISEP
- physical.environment.LogInventoryReserve
- physical.environment.NaturalEnvironment
- physical.environment.SeaBase
- physical.environment.Space
- physical.environment.StorageZone
- physical.environment.Zone
- physical.item.Item
- physical.item.ItemContainer
- physical.item.ItemInfo
- physical.item.RolledUpItemInfo
- physical.weather.WeatherSystem

Enumeration : app.eDay

Scope	Allowed Values
package	UNK_DAY, SUN, MON, TUE, WED, THU,

	FRI, SAT

Enumeration : app.eMonth

Scope	Allowed Values
package	UNK_MONTH, JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC

Enumeration : app.eProtocol

Scope	Allowed Values
package	UNK_PROTOCOL, HTTP, FILE, FTP, MAIL, NEWS

Enumeration : app.Component.eHealthStatus

Scope	Allowed Values
class	UNK_HEALTH_STATUS, HEALTHY, POOR_HEALTH

Enumeration : app.design.Action.eActionStatus

Scope	Allowed Values
class	UNK_ACTION_STATUS, ACTION_PENDING_EXECUTION, ACTION_EXECUTING, ACTION_ABORTED, ACTION_COMPLETED,

	ACTION_POSTPONED

Enumeration : app.design.Design.eAuthStatus

Scope	Allowed Values
class	UNK_AUTH_STATUS, AUTHORIZED, PENDING_AUTHORIZATION, REJECTED_AUTHORIZATION

Enumeration : app.view.Component.eSpecificity

Scope	Allowed Values
class	UNK_SPECIFICITY, NON_ENCYCLOPEDIC, ENCYCLOPEDIC

Enumeration : information.Requirement.eReqStatus

Scope	Allowed Values
class	UNK_REQ_STATUS, REQ_UNSUPPORTED, REQ_SUPPORTED

Enumeration : operation.Sortie.eAuthStatus

Scope	Allowed Values
class	UNK_AUTH_STATUS, AUTHORIZED, PENDING_AUTHORIZATION, REJECTED_AUTHORIZATION

Enumeration : operation.Sortie.eExeStatus

Scope	Allowed Values
class	UNK_EXE_STATUS, PENDING_EXECUTION, EXECUTING, COMPLETED_EXECUTION, ABORTED_EXECUTION

Enumeration : operation.logistical.eSupplyClass

Scope	Allowed Values

package	UNK_SUPPLY_CLASS, CLASS_I, CLASS_II, CLASS_III, CLASS_IV, CLASS_V, CLASS_VI, CLASS_VII, CLASS_VIII, CLASS_IX

Enumeration : operation.logistical.CargoTransaction.eCargoTransType

Scope	Allowed Values
class	UNK_CARGO_TRANS_TYPE, DELIVERY, PICKUP

Enumeration : operation.logistical.Requirement.eEffortType

Scope	Allowed Values
class	UNK_EFFORT_TYPE, PRIMARY, SUPPORTIVE

Enumeration : physical.Physical.eForceCode

Scope	Allowed Values
class	UNKPND:Unknown Unknown Pending , AIRHOS:Air Hostile , AIRPND:Air Unknown Pending , AIRFRD:Air Friend , SUBHOS:Sub Hostile , SUBPND:Sub Unknown Pending , SUBFRD:Sub Friend , SURHOS:Surf Hostile , SURPND:Surf Unknown Pending , SURFRD:Surf Friend , AIRUAF:Air Unknown Assumed Friend , AIRUAH:Air Unknown Assumed Hostile, AIRNEU:Air Neutral, SUBUAF:Sub Unknown Assumed Friend, SUBUAH:Sub Unknown Assumed Hostile, SUBNEU:Sub Neutral, SURUAF:Surf Unknown Assumed Friend,

	SURUAH:Surf Unknown Assumed Hostile, SURNEU:Surf Neutral, UNKUAH:Unknown Unknown Assumed Hostile, UNKUAF:Unknown Unknown Assumed Friend, UNKNEU:Unknown Neutral, LNDFRD:Land Friend, LNDUAF:Land Unknown Assumed Friend, LNDHOS:Land Hostile, LNDUAH:Land Unknown Assumed Hostile, LNDPND:Land Unknown Pending, LNDNEU:Land Neutral, AIREVU:Air Evaluated Unknown, SUBEVU:Sub Evaluated Unknown, SUREVU:Surf Evaluated Unknown, LNDEVU:Land Evaluated Unknown, UNKEVU:Unknown Evaluated Unknown, UNU033, UNU034, UNU035, UNU036, UNU037, UNKHOS:Unknown Hostile, UNKFRD:Unknown Friend, RES040, RES041, RES042, RES043, RES044, RES045, RES046, RES047, RES048, RES049, RES050, RES051, RES052, RES053, RES054, RES055, RES056, RES057, RES058, RES059, RES060, RES061, RES062, RES063,

	RES064, RES065, RES066, RES067, RES068, RES069, RES070, RES071, RES072, RES073, RES074, RES075, RES076, RES077, RES078, RES079, RES080, RES081, RES082, RES083, RES084, RES085, AIRALL:All Air, SUBALL:All Sub, SURALL:All Surf, LNDALL:All Land, UNKALL:All Unknown, ALLHOS:All Hostile, ALLPND:All Unknown Pending, ALLFRD:All Friend, ALLUAF:All Unknown Assumed Friend, ALLUAH:All Unknown Assumed Hostile, ALLNEU:All Neutral, ALLUNK:All Evaluated Unknown, UNU098, ALLALL:All All

Enumeration : physical.Physical.eAffiliation

Scope	Allowed Values
class	UNK_AFFILIATION:Unknown Affiliation, PENDING:Pending Unknown, FRIEND:Friend, NEUTRAL:Neutral, HOSTILE:Hostile, ASSUMED:Assumed Friend,

	SUSPECT:Suspect Hostile, JOKER:Joker Hostile, FAKER:Faker Hostile

Enumeration : physical.asset.eStowAccessibilityType

Scope	Allowed Values
package	UNK_STOW_ACCESSIBILITY_TYPE, RORO, LOLO

Enumeration : physical.asset.AirWeapon.eAirWeaponType

Scope	Allowed Values
class	UNK_AIR_WEAPON, MISSILE_IN_FLIGHT, DECOY

Enumeration : physical.asset.Aircraft.eAircraftType

Scope	Allowed Values
class	UNK_AIRCRAFT_TYPE, MILITARY_AIRCRAFT, WEAPON_AIRCRAFT, CIVIL_AIRCRAFT

Enumeration : physical.asset.ArmoredVehicle.eArmoredVehicleType

Scope	Allowed Values
class	UNK_ARM_VEH_TYPE, TANK_VEHICLE, PERSONNEL_CARRIER, INFANTRY_VEHICLE, C2V_ACV, CSS_VEHICLE, LIGHT_ARMORED_VEHICLE

Enumeration : physical.asset.FixedWing.eFixedWingMilAirType

Scope	Allowed Values
class	UNK_FIXED_WING_MIL_AIR_TYPE, BOMBER_FXD, FIGHTER_FXD, TRAINER_FXD,

	ATTACK_STRIKE_FXD, VS_TOL_FXD, TANKER_FXD, CARGO_AIRLIFT_FXD, ELECTRONIC_COUNTERMEASURES_FXD, MEDEVAC_FXD, RECONNAISANCE_FXD, PATROL, UTILITY_FXD, COMMUNICATIONS_FXD, SEARCH_AND_RESCUE_FXD, AIRBORNE_COMMAND_POST_FXD, DRONE_FXD, ANTI_SUBMARINE_FXD, SPECIAL_OPERATIONS_FORCES_FXD

Enumeration : physical.asset.MilitaryAircraft.eMilitaryAircraftType

Scope	Allowed Values
class	UNK_MIL_AIR_TYPE, FIXED_WING_MILITARY, ROTARY_WING_MILITARY, LIGHTER_THAN_AIR_MILITARY

Enumeration : physical.asset.RotaryWing.eRotaryWingType

Scope	Allowed Values
class	UNK_ROTARY_WING_TYPE, CH_46E, CH_53E, MV_22

Enumeration : physical.asset.RotaryWing.eRotaryWingMilAirType

Scope	Allowed Values
class	UNK_ROTARY_WING_MIL_AIR_TYPE, ATTACK_ROT, ANTI_SUBMARINE_ROT, UTILITY_ROT, MINE_COUNTERMEASURES_ROT, SEARCH_AND_RESCUE_ROT, RECONNAISANCE_ROT, DRONE_ROT, CARGO_AIRLIFT_ROT, TRAINER_ROT,

	MEDEVAC_ROT, SPECIAL_OPERATIONS_FORCES_ROT, AIRBORNE_COMMAND_POST_ROT, TANKER_ROT, ELECTRONIC_COUNTERMEASURES_ROT

Enumeration : physical.asset.Sensor.eSensorType

Scope	Allowed Values
class	UNK_SENSOR_TYPE, RADAR_DETECTOR, ACTIVE_RADAR, PASSIVE_RADAR, ACTIVE_SONAR, PASSIVE_SONAR, NAVGPS, CAMERA, SIGINT

Enumeration : physical.asset.Tank.eLoaderType

Scope	Allowed Values
class	UNK_LOADER_TYPE, MANUAL_LOADER, AUTOMATIC_LOADER

Enumeration : physical.asset.Vehicle.eVehicleType

Scope	Allowed Values
class	UNK_VEHICLE_TYPE, ARMORED_VEHICLE, UTILITY_VEHICLE, ENGINEER_VEHICLE, TRAIN_LOCOMOTIVE, CIVILIAN_VEHICLE

Enumeration : physical.asset.VehicleTrailer.eVehTrailerType

Scope	Allowed Values
class	UNK_VEH_TRAILER_TYPE, TRAILER

Enumeration : physical.asset.Vessel.eVesselType

Scope	Allowed Values
class	UNK_VESSEL_TYPE, COMBATANT_VESSEL, NON_COMBATANT_VESSEL, NON_MILITARY_VESSEL, OWN_VESSEL, EMERGENCY_VESSEL, HAZARD_VESSEL

Enumeration : physical.asset.Vessel.eHullType

Scope	Allowed Values
class	UNK_HULL_TYPE, CSNP, CSP, SLWT, LCM_8, LCU_1600, LCAC, LSV, LARC_V, LARC_LX, MCS, LHD, AK, AKR

Enumeration : physical.asset.Weapon.eWeaponType

Scope	Allowed Values
class	UNK_WEAPON_TYPE, MISSILE_LAUNCHER, SINGLE_ROCKET_LAUNCHER, MULTI_ROCKET_LAUNCHER, ANTITANK_ROCKET_LAUNCHER, AUTOMATIC_WEAPON, GRENADE_LAUNCHER, MORTAR, HOWITZER, ANTITANK_GUN, DIRECT_FIRE_GUN, AIR_DEFENSE_GUN

Enumeration : physical.body.Body.eLoyalty

Scope	Allowed Values
class	UNK_LOYALTY, COMPLETELY_LOYAL, MOSTLY_LOYAL, SOMEWHAT_LOYAL, NOT_LOYAL

Enumeration : physical.body.CombatServiceSupportUnit.eCSSUnitType

Scope	Allowed Values
class	UNK_CSSU_TYPE, ADMINISTRATIVE_CSSU, MEDICAL_CSSU, SUPPLY_CSSU, TRANSPORTATION_CSSU, MAINTENANCE_CSSU, SPECIAL_C2_HQ_CSSU

Enumeration : physical.body.CombatSupportUnit.eCombatSupUnitType

Scope	Allowed Values
class	UNK_CSU_TYPE, NBC_CSU, INTEL_CSU, LAW_CSU, SIGNAL_CSU, INFO_UNIT_WARFARE_CSU, LANDING_SUP_CSU, ORDNANCE_DISPOSAL_CSU

Enumeration : physical.body.CombatUnit.eCombatUnitType

Scope	Allowed Values
class	UNK_CU_TYPE, AIR_DEFENSE_CU, ARMOR_CU, ANTI_ARMOR_CU, AVIATION_CU, INFANTRY_CU, ENGINEER_CU, FIELD_ARTILLARY_CU, RECONNAISSANCE_CU, MISSILE_CU,

	INTERNAL_SECURITY_FORCES_CU

Enumeration : physical.body.GroundUnit.eGroundUnitType

Scope	Allowed Values
class	UNK_GROUND_UNIT_TYPE, COMBAT_UNIT, COMBAT_SUPPORT_UNIT, COMBAT_SERVICE_SUPPORT_UNIT, SPECIAL_C2_HQ_UNIT

Enumeration : physical.body.MilitaryOrganization.eEchelon

Scope	Allowed Values
class	UNK_ECHELON, TEAM, CREW, SQUAD, SECTION, PLATOON, DETACHMENT, COMPANY, BATTERY, TROOP, BATTALION, SQUADRON, ELEMENT, REGIMENT, GROUP, BRIGADE, DIVISION, TASK_FORCE, CORPS, FORCE, ARMY, ARMY_GROUP, FRONT, REGION

Enumeration : physical.body.Organization.eOrgSize

Scope	Allowed Values
class	UNK_ORG_SIZE, SMALL,

	MEDIUM, LARGE, MASSIVE

Enumeration : physical.body.Organization.eOrgType

Scope	Allowed Values
class	UNK_ORG_TYPE, POLICE, FIRE, MEDICAL, PROFESSIONAL, EDUCATIONAL, GOVERNMENTAL, SOCIAL, RELIGIOUS, POLITICAL, RECREATIONAL, CRIMINAL, PARAMILITARY

Enumeration : physical.body.Person.ePersonStatus

Scope	Allowed Values
class	UNK_PERSON_STATUS, ALIVE, DEAD, WOUNDED

Enumeration : physical.body.SpecialOps.eSpecialOpsType

Scope	Allowed Values
class	UNK_SPECIAL_OPS_TYPE, SOF, AVIATION, NAVAL_SOF, GROUND_SOF, SUPPORT_SOF

Enumeration : physical.consumable.Cartridge.eCartridgeType

Scope	Allowed Values
class	UNK_CARTRIDGE_TYPE, SMALL_ARM, MORTAR,

	CANNON, GRENADE, MACHINEGUN

Enumeration : physical.consumable.Missile.eMissileType

Scope	Allowed Values
class	UNK_MISSILE_TYPE, ANTI_TANK, BALLISTIC, CRUISE, TACTICAL, AIR_TO_AIR, AIR_TO_SURFACE, SURFACE_TO_AIR, SURFACE_TO_SURFACE

Enumeration : physical.consumable.POL.ePOLType

Scope	Allowed Values
class	UNK_POL_TYPE, JP4, JP5, JP8, DIESEL, KEROSENE, UNLEADED, GASOLINE

Enumeration : physical.consumable.Pyrotechnics.ePyrotechnicType

Scope	Allowed Values
class	UNK_PYRO_TYPE, SIGNAL, SCREEN, ILLUMINATION

Enumeration : physical.consumable.UnguidedMunitions.eUnguidedMunType

Scope	Allowed Values
class	UNK_UNGUIDED_MUNITIONS_TYPE, UNGUIDED, BOMB, MINE,

	ROCKET, CLAYMORE, HAND_GRENADE

Enumeration : physical.environment.BodyOfWater.eSeaState

Scope	Allowed Values
class	UNK_SEASTATE, ST_LEVEL_I, ST_LEVEL_II, ST_LEVEL_III, ST_LEVEL_IV, ST_LEVEL_V, ST_LEVEL_VI, ST_LEVEL_VII, ST_LEVEL_VIII, ST_LEVEL_IX

Enumeration : physical.item.Item.eItemStatus

Scope	Allowed Values
class	UNK_ITEM_STATUS, INTACT, SCHEDULED, RESERVED, DISABLED, DESTROYED

Enumeration : physical.item.Item.eItemLocation

Scope	Allowed Values
class	UNK_ITEM_LOCATION, INBOUND, ISEP, ISEP_TO_SEABASE, SEABASE, SEABASE_TO_SHORE, SHORE

Enumeration : physical.item.ItemContainer.eContainerType

Scope	Allowed Values
class	UNK_CONTAINER_TYPE, PALLET,

	FIFTY_CUBE, QUADCON, PALCON, MODULE_CONTAINER, CASE_CONTAINER

Enumeration : physical.weather.WeatherSystem.eWeatherSystemType

Scope	Allowed Values
class	UNK_WEATHER_SYSTEM_TYPE, SHOWER, RAIN_STORM, BLIZZARD, HURRICAN, TORNADO, TYPHOON, SNOW_STORM, HAIL_STORM

Struct : app.DateTimeInterval

Attributes:

Name	Type	Visibility	Default	Access	Units
app.DateTimeInterval.lowerBound	unsignedlong	public		read & write	storage = s display = s
app.DateTimeInterval.upperBound	unsignedlong	public		read & write	storage = s display = s

Struct : app.s2DDims

Attributes:

Name	Type	Visibility	Default	Access	Units
app.s2DDims.width	float	public		read & write	storage = m display = ft
app.s2DDims.length	float	public		read & write	storage = m display = ft

Struct : app.s3DDims

Attributes:

Name	Type	Visibility	Default	Access	Units
app.s3DDims.width	float	public		read & write	storage = m display = ft
app.s3DDims.length	float	public		read & write	storage = m display = ft
app.s3DDims.height	float	public		read & write	storage = m display = ft

Struct : app.sDate

Attributes:

Name	Type	Visibility	Default	Access	Units
app.sDate.month	app.eMonth				

Struct : app.sDuration

Attributes:

Name	Type	Visibility	Default	Access	Units
app.sDuration.interval	unsignedlong	public		read & write	storage = s display = s

Struct : app.sPosition

Attributes:

Name	Type	Visibility	Default	Access	Units
app.sPosition.time	long	public		read only	storage = s display = s
app.sPosition.latitude	double	public		read & write	storage = lat display = lat
app.sPosition.longitude	double	public		read & write	storage = lon display = lon
app.sPosition.altitude	double	public		read & write	storage = m display = ft

Struct : app.sPriority

Attributes:

Name	Type	Visibility	Default	Access	Units
app.sPriority.level	long	public		read & write	

Struct : app.sTime

Attributes:

Name	Type	Visibility	Default	Access	Units
app.sTime.seconds	unsignedlong	public		read & write	storage = s display = s

Struct : operation.logistical.sItemDeliveryStatusByNSN

Attributes:

Name	Type	Visibility	Default	Access	Units
operation.logistical.sItemDeliveryStatusByNSN.NSNCriteria	string	public		read & write	
operation.logistical.sItemDeliveryStatusByNSN.itemReqs	Key	public		read & write	
operation.logistical.sItemDeliveryStatusByNSN.numRequested	unsignedlong	public	0	read & write	
operation.logistical.sItemDeliveryStatusByNSN.numDelivered	unsignedlong	public	0	read & write	
operation.logistical.sItemDeliveryStatusByNSN.numRemaining	unsignedlong	public	0	read & write	

Struct : operation.logistical.sItemPopByLocation

Attributes:

Name	Type	Visibility	Default	Access	Units
operation.logistical.sItemPopByLocation.inboundItems	Key	public		read & write	
operation.logistical.sItemPopByLocation.ISEPItems	Key	public		read & write	
operation.logistical.sItemPopByLocation.ISEPToSeabaseItems	Key	public		read & write	
operation.logistical.sItemPopByLocation.seabaseItems	Key	public		read & write	
operation.logistical.sItemPopByLocation.seabaseToShoreItems	Key	public		read & write	
operation.logistical.sItemPopByLocation.ashoreItems	Key	public		read & write	
operation.logistical.sItemPopByLocation.relatedStorageZones	Key	public		read & write	

Struct : operation.logistical.sItemQuantity

Attributes:

Name	Type	Visibility	Default	Access	Units
operation.logistical.sItemQuantity.NSN	string	public		read only	
operation.logistical.sItemQuantity.DoDIC	string	public		read only	
operation.logistical.sItemQuantity.availableQty	long	public		read & write	

Class : Container

Inheritance:

ICDMBase

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
ContainerObject	ICDMObject	objects	AGGREGATION	0...	true

Class : ICDMBase

Inheritance:

Attributes:

Name	Type	Visibility	Default	Access	Units
ICDMBase.objectKey	Key	protected		read only	
ICDMBase.created	long	protected		read only	storage = sec display = date
ICDMBase.updated	long	protected		read & write	storage = sec display = date

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

---

Class : ICDMObject

Inheritance:

ICDMBase

Attributes:

Name	Type	Visibility	Default	Access	Units
ICDMObject.objectName	string	public		read only	
ICDMObject.owner	string	public		read & write	

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
ContainerObject	Container	Container_role	ASSOCIATION	1	false

Class : app.Application

Inheritance:

ICDMObject

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
AppSuite	app.Suite	Suite_role	ASSOCIATION	1	false
AppComp	app.Component	components	AGGREGATION	1...*	true

Class : app.Calender

Inheritance:

ICDMObject

Attributes:

Name	Type	Visibility	Default	Access	Units
app.Calender.curDate	app.sDate				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
ViewCal	app.view.View	View_role	ASSOCIATION	1	false

Class : app.Clock

Inheritance:

ICDMObject

Attributes:

Name	Type	Visibility	Default	Access	Units
app.Clock.curTime	unsignedlong	public		read & write	

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
ViewClock	app.view.View	View_role	ASSOCIATION	1	false

Class : app.Component

Inheritance:

ICDMObject

Attributes:

Name	Type	Visibility	Default	Access	Units
app.Component.healthStatus	app.Component.eHealthStatus				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
AppComp	app.Application	Application_role	ASSOCIATION	1	false

Class : app.DateTime

Inheritance:

ICDMObject

Attributes:

Name	Type	Visibility	Default	Access	Units
app.DateTime.date	app.sDate				
	app.sTime				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

---

Class : app.Suite

Inheritance:

ICDMObject

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
AppSuite	app.Application	applications	AGGREGATION	2...*	true

Class : app.agent.AgentReport

Inheritance:

information.Report

Attributes:

Name	Type	Visibility	Default	Access	Units
app.agent.AgentReport.content	string	public		read & write	
app.agent.AgentReport.acknowledged	boolean	public	false	read & write	

Associations:

Association	Associated Class	Associated Role	Type	Multiplicity	Navigabl

Name		Name			e
ReportsOn	app.agent.BaseAgent	reportingAgent	ASSOCIATION	1	true
RepExpl	app.agent.Explanation	explanation	AGGREGATION	1	true
RepComp1	app.view.Component	affectedObjs	ASSOCIATION	0...	true
RepComp2	app.view.Component	instigators	ASSOCIATION	0...	true

Class : app.agent.BaseAgent

Inheritance:

app.view.Component

Attributes:

Name	Type	Visibility	Default	Access	Units
app.agent.BaseAgent.agentId	string	public		read only	
app.agent.BaseAgent.activity	long	public		read & write	

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
ReportsOn	app.agent.AgentReport	reports	ASSOCIATION	0...	true

Class : app.agent.Emergency

Inheritance:

app.agent.AgentReport

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : app.agent.Explanation

Inheritance:

information.Information

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
RepExpl	app.agent.AgentReport	report	ASSOCIATION	1	false

Class : app.agent.FYI

Inheritance:

app.agent.AgentReport

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

---

Class : app.agent.Facilitator

Inheritance:

app.agent.BaseAgent

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

---

Class : app.agent.Guardian

Inheritance:

app.agent.ObjectAgent

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : app.agent.Mentor

Inheritance:

app.agent.Guardian

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : app.agent.ObjectAgent

Inheritance:

app.agent.BaseAgent

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
OAEmpow	app.view.Empowerable	empoweredObjs	ASSOCIATION	0...	true

Class : app.agent.Recommendation

Inheritance:

app.agent.AgentReport

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : app.agent.ServiceAgent

Inheritance:

app.agent.BaseAgent

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : app.agent.Warning

Inheritance:

app.agent.AgentReport

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : app.design.Action

Inheritance:

app.design.Component

Attributes:

Name	Type	Visibility	Default	Access	Units
app.design.Action.actionStatus	app.design.Action.eActionStatus				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
ActComp	app.design.Component	actionComps	AGGREGATION	0...	true

Class : app.design.Component

Inheritance:

information.Report

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
DesComp	app.design.Design	compDesign	ASSOCIATION	1	false
ActComp	app.design.Action	compAction	ASSOCIATION	1	false
PhaseComp	app.design.Phase	compPhase	ASSOCIATION	1	false

Class : app.design.Design

Inheritance:

information.Report

Attributes:

Name	Type	Visibility	Default	Access	Units
app.design.Design.authorizationStatus	app.design.Design.eAuthStatus				
	app.sTime				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
DesComp	app.design.Component	designComps	AGGREGATION	0...	true

Class : app.design.Phase

Inheritance:

app.design.Component

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
PhaseComp	app.design.Component	phaseComps	AGGREGATION	0...	true

---

Class : app.view.Component

Inheritance:

ICDMObject

Attributes:

Name	Type	Visibility	Default	Access	Units
app.view.Component.Specificity	app.view.Component.eSpecificity				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
ViewMngdComp	app.view.View	mngdCompView	ASSOCIATION	1	false
ViewRefComp	app.view.View	refCompViews	ASSOCIATION	0...	true
RepComp1	app.agent.AgentReport	affectedRep	ASSOCIATION	0...	true
RepComp2	app.agent.AgentReport	instigatorsRep	ASSOCIATION	0...	true

---

Class : app.view.Empowerable

Inheritance:

app.view.Component

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
OAEmpow	app.agent.ObjectAgent	ObjectAgent_role	ASSOCIATION	1	false

---

Class : app.view.SubView

Inheritance:

app.view.View

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
ViewSubView	app.view.View	View_role	ASSOCIATION	1	false

Class : app.view.View

Inheritance:

ICDMObject

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
ViewMngdComp	app.view.Component	managedComps	AGGREGATION	0...	true
ViewRefComp	app.view.Component	referencedComps	ASSOCIATION	0...	true
ViewSubView	app.view.SubView	subViews	AGGREGATION	0...	true
ViewCal	app.Calender	viewCalender	AGGREGATION	1	true

ViewClock	app.Clock	viewClock	AGGREGA TION	1	true

Class : information.ImageLayer

Inheritance:

information.MapLayer

Attributes:

Name	Type	Visibility	Default	Access	Units
information.ImageLayer.imageURL	string	public		read & write	

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : information.Information

Inheritance:

app.view.Component

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : information.Layer

Inheritance:

information.Information

Attributes:

Name	Type	Visibility	Default	Access	Units
information.Layer.layerCenterPoint	app.sPosition				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : information.MapLayer

Inheritance:

information.Layer

Attributes:

Name	Type	Visibility	Default	Access	Units
information.MapLayer.scale	long	public		read & write	

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

---

Class : information.Report

Inheritance:

information.Information

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
RFRReport	information.RequestForReport	RFRs	ASSOCIATION	0...	true

---

Class : information.Request

Inheritance:

information.Information

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

---

Class : information.RequestForReport

Inheritance:

information.Request

Attributes:

Name	Type	Visibility	Default	Access	Units
information.RequestForReport.RFRsatisfied	boolean	public	false	read & write	

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
RFRReport	information.Report	requestedReport	ASSOCIATION	0...	true

Class : information.Requirement

Inheritance:

information.Request

Attributes:

Name	Type	Visibility	Default	Access	Units
information.Requirement.reqStatus	information.Requirement.eReqStatus				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
TR	operation.Trip	supportingTrips	ASSOCIATION	0...	true
AR	operation.Activity	supportingActivities	ASSOCIATION	0...	true

Class : operation.Activity

Inheritance:

app.design.Action

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
AR	information.Requirement	activityReq	ASSOCIATION	1	true

Class : operation.ActivityPt

Inheritance:

app.design.Phase

Attributes:

Name	Type	Visibility	Default	Access	Units
operation.ActivityPt.ETA	unsignedlong	public		read & write	storage = sec display = sec
operation.ActivityPt.ETD	unsignedlong	public		read & write	storage = sec display = sec

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
ActPtEnv	physical.environment.Environment	activityPtEnv	ASSOCIATION	1	true

Class : operation.Sortie

Inheritance:

operation.Trip

Attributes:

Name	Type	Visibility	Default	Access	Units
operation.Sortie.authorizationStatus	operation.Sortie.eAuthStatus				
	app.sTime				
	operation.Sortie.eExeStatus				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : operation.Trip

Inheritance:

app.design.Action

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
TPOrg	physical.body.Organization	crew	ASSOCIATION	1	true
TPPlat	physical.asset.Platform	transport	ASSOCIATION	1	true
TR	information.Requirement	TripReqs	ASSOCIATION	0...	true

Class : operation.logistical.CargoTransaction

Inheritance:

operation.Activity

Attributes:

Name	Type	Visibility	Default	Access	Units
operation.logistical.CargoTransaction.cargoTransType	operation.logistical.CargoTransaction.eCargoTransType				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
CTCII	physical.item.Item	cargo	ASSOCIATION	0...	true

Class : operation.logistical.ConsolidatedReqReport

Inheritance:

information.Report

Attributes:

Name	Type	Visibil	Defa	Access	Uni

		ity	ult		ts
operation.logistical.ConsolidatedReqReport.locationCriteria	Key	public		read & write	
operation.logistical.ConsolidatedReqReport.timeIntervalCriteria	Key	public		read & write	
operation.logistical.ConsolidatedReqReport.itemsByNSN	operation.logistical.sItemDeliveryStatusByNSN				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : operation.logistical.CriticalItemList

Inheritance:

information.Information

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
CILII	physical.item.ItemInfo	criticalItems	AGGREGATION	0...	true

Class : operation.logistical.CyclicRequirement

Inheritance:

operation.logistical.Requirement

Attributes:

Name	Type	Visibility	Default	Access	Units
operation.logistical.CyclicRequirement.cycleInterval	app.sDuration				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : operation.logistical.ItemAvailabilityReport

Inheritance:

information.Report

Attributes:

Name	Type	Visibility	Default	Access	Units
operation.logistical.ItemAvailabilityReport.NSNCriteria	string	public		read & write	
operation.logistical.ItemAvailabilityReport.in24Hrs	operation.logistical.sItemQuantity				
	operation.logistical.sItemQuantity				
	operation.logistical.sItemQuantity				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : operation.logistical.PickupRequirement

Inheritance:

operation.logistical.Requirement

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
PRE	physical.environment.Environment	pickupLocation	ASSOCIATION	1	true

Class : operation.logistical.Plan

Inheritance:

app.design.Design

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

---

Class : operation.logistical.Requirement

Inheritance:

information.Requirement

Attributes:

Name	Type	Visibility	Default	Access	Units
operation.logistical.Requirement.priority	unsignedint	public	1	read & write	
operation.logistical.Requirement.effortType	operation.logistical.Requirement.eEffortType				
	app.DateTimeInterval				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
RE1	physical.environment.Environment	itemDestination	ASSOCIATION	1	true
RB1	physical.body.Body	requestor	ASSOCIATION	1	true
RB2	physical.body.Body	reqTarget	ASSOCIATION	1	true
RE2	physical.environment.Environment	reqDestination	ASSOCIATION	1	true
ReqII	physical.item.ItemInfo	requiredItems	AGGREGATION	0…	true

---

Class : operation.logistical.TransportAvailabilityReport

Inheritance:

information.Report

Attributes:

Name	Type	Visibilit	Default	Access	Units

		y			
operation.logistical.TransportAvailabilityReport.in12Hrs	operation.logistical.sItemQuantity				
	operation.logistical.sItemQuantity				
	operation.logistical.sItemQuantity				
	operation.logistical.sItemQuantity				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : operation.logistical.envItemPopulationReport

Inheritance:

information.Report

Attributes:

Name	Type	Visibility	Default	Access	Units
operation.logistical.envItemPopulationReport.NSNCriteria	string	public		read & write	
operation.logistical.envItemPopulationReport.itemDistributions	operation.logistical.sItemPopByLocation				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : operation.logistical.itemStatusReport

Inheritance:

information.Report

Attributes:

Name	Type	Visibility	Default	Access	Units
operation.logistical.itemStatusReport.NSNCriteria	string	public		read & write	
operation.logistical.itemStatusReport.locationsCriteria	Key	public		read & write	

operation.logistical.itemStatusReport.operationalQtys	int	public		read & write	
operation.logistical.itemStatusReport.nonOperationalQtys	int	public		read & write	

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : operation.tactical.CallForFire

Inheritance:

information.Request

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
CFFBody	physical.body.Body	cffRequestor	ASSOCIATION	1	true
CFFPhysical	physical.Physical	cffTargets	ASSOCIATION	0...	true

Class : operation.tactical.Deployment

Inheritance:

operation.Activity

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : operation.tactical.FiringOrder

Inheritance:

app.design.Action

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
FMPhysical	physical.Physical	fmTargets	ASSOCIATION	0...	true

Class : operation.tactical.FiringSolution

Inheritance:

app.design.Action

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
FOFS	physical.asset.Weapon	fmSolution	ASSOCIATION	1	true
FSMunition	physical.consumable.Munitions	fmMunition	ASSOCIATION	1	true

---

Class : operation.tactical.Plan

Inheritance:

app.design.Design

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

---

Class : physical.ArtificialMobile

Inheritance:

physical.Mobile

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

---

Class : physical.Mobile

Inheritance:

physical.Physical

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.Mobile.footPrintArea	float	public		read & write	storage = m2 display = ft2
physical.Mobile.course	float	public		read & write	storage = deg display = deg
physical.Mobile.bearing	float	public		read & write	storage = deg display = deg

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

---

Class : physical.NaturalMobile

Inheritance:

physical.Mobile

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

---

Class : physical.Physical

Inheritance:

app.view.Empowerable

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.Physical.forceCode	physical.Physical.eForceCode				

	physical.Physical.eAffiliation				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
PhysicalPos	app.sPosition	location	COMPOSITION	1	true
FMPhysical	operation.tactical.FiringOrder	FiringOrder_role	ASSOCIATION	1	false
CFFPhysical	operation.tactical.CallForFire	CallForFire_role	ASSOCIATION	1	false
PhysicalEnv	physical.environment.Environment	physicalEnv	ASSOCIATION	1	true
PS	physical.environment.Space	physicalSpaces	AGGREGATION	0...	true

Class : physical.asset.AirWeapon

Inheritance:

physical.asset.MilitaryAircraft

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.asset.AirWeapon.airWeaponType	physical.asset.AirWeapon.eAirWeaponType				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : physical.asset.Aircraft

Inheritance:

physical.asset.Platform

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.asset.Aircraft.aircraftType	physical.asset.Aircraft.eAircraftType				
	app.s2DDims				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
AircraftRadar	physical.asset.Radar	radar	ASSOCIATION	1	true

Class : physical.asset.ArmoredVehicle

Inheritance:

physical.asset.Vehicle

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.asset.ArmoredVehicle.armoredVehicleType	physical.asset.ArmoredVehicle.eArmoredVehicleType				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : physical.asset.Asset

Inheritance:

physical.item.Item

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.asset.Asset.supplyClass	operation.logistical.eSupplyClass				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
BodyAsset	physical.body.Body	Body_role	ASSOCIATION	1	false

Class : physical.asset.Equipment

Inheritance:

physical.asset.Asset

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.asset.Equipment.function	string	public		read & write	

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : physical.asset.FixedWing

Inheritance:

physical.asset.MilitaryAircraft

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.asset.FixedWing.fixedWingMilAirType	physical.asset.FixedWing.eFixedWingMilAirType				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : physical.asset.GuidanceSystem

Inheritance:

physical.asset.Navigation

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.asset.GuidanceSystem.searchFootprint	float	public		read only	storage = m2 display = ft2

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
GuidedMunGuidSys	physical.consumable.GuidedMunitions	GuidedMunitions_role	ASSOCIATION	1	false

Class : physical.asset.MilitaryAircraft

Inheritance:

physical.asset.Aircraft

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.asset.MilitaryAircraft.militaryAircraftType	physical.asset.MilitaryAircraft.eMilitaryAircraftType				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : physical.asset.Navigation

Inheritance:

physical.asset.Equipment

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.asset.Navigation.life	long	public		read only	storage = s display = s
physical.asset.Navigation.positionAccuracy	float	public		read only	storage = m display = ft
physical.asset.Navigation.altitudeAccuracy	float	public		read only	storage = m display = ft
physical.asset.Navigation.updateRate	long	public		read only	storage = s display = s

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
PlatformNav	physical.asset.Platform	Platform_role	ASSOCIATION	1	false

Class : physical.asset.Platform

Inheritance:

physical.asset.Asset

Attributes:

Name	Type	Visibility	Default	Access	Units

physical.asset.Platform.series	string	public		read only	
physical.asset.Platform.fuelCapacity	float	public		read only	storage = ltr display = gal
physical.asset.Platform.maxSpeed	float	public		read only	storage = km/hr display = mi/hr
physical.asset.Platform.maxRange	float	public		read only	storage = km display = mi
physical.asset.Platform.maxCargoWeight	float	public		read & write	storage = kg display = ton
physical.asset.Platform.combatRadius	float	public		read & write	storage = km display = mi
physical.asset.Platform.fuelLevel	float	public		read & write	storage = ltr display = gal
physical.asset.Platform.typicalCruisingSpeed	float	public		read only	storage = km/hr display = mi/hr

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
PlatformNav	physical.asset.Navigation	navigationSystem	ASSOCIATION	1	true
PlatformPOL	physical.consumable.POL	platformPOL	ASSOCIATION	0...	true
PlatformWeapon	physical.asset.Weapon	weaponSystems	ASSOCIATION	0...	true
TPPlat	operation.Trip	travelPlans	ASSOCIATION	1	true

Class : physical.asset.Radar

Inheritance:

physical.asset.Sensor

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
AircraftRadar	physical.asset.Aircraft	Aircraft_role	ASSOCIATION	1	false

Class : physical.asset.RotaryWing

Inheritance:

physical.asset.MilitaryAircraft

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.asset.RotaryWing.rotaryWingType	physical.asset.RotaryWing.eRotaryWingType				
	physical.asset.RotaryWing.eRotaryWingMilAirType				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : physical.asset.Sensor

Inheritance:

physical.asset.Equipment

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.asset.Sensor.sensorType	physical.asset.Sensor.eSensorType				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : physical.asset.Tank

Inheritance:

physical.asset.ArmoredVehicle

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.asset.Tank.hullType	string	public		read only	
physical.asset.Tank.loaderType	physical.asset.Tank.eLoaderType				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

---

Class : physical.asset.Vehicle

Inheritance:

physical.asset.Platform

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.asset.Vehicle.vehicleType	physical.asset.Vehicle.eVehicleType				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
UtilVehVeh	physical.asset.VehicleTrailer	slaveTrailer	ASSOCIATION	1	true

---

Class : physical.asset.VehicleTrailer

Inheritance:

physical.asset.Vehicle

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.asset.VehicleTrailer.VehicleTrailerType	physical.asset.VehicleTrailer.eVehTrailerType				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
UtilVehVeh	physical.asset.Vehicle	masterVehicle	ASSOCIATION	1	true

---

Class : physical.asset.Vessel

Inheritance:

physical.asset.Platform

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.asset.Vessel.vesselType	physical.asset.Vessel.eVesselType				
	physical.asset.Vessel.eHullType				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
SeaBaseVessel	physical.environment.SeaBase	seabase	ASSOCIATION	1	true

Class : physical.asset.Weapon

Inheritance:

physical.asset.Equipment

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.asset.Weapon.weaponType	physical.asset.Weapon.eWeaponType				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
PlatformWeapon	physical.asset.Platform	Platform_role	ASSOCIATION	1	false
FOFS	operation.tactical.FiringSolution	FiringSolution_role	ASSOCIATION	1	false

Class : physical.body.Body

Inheritance:

physical.item.Item

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.body.Body.loyalty	physical.body.Body.eLoyalty				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
OrgBody	physical.body.Organization	organizations	ASSOCIATION	0...	true
BodyConsumable	physical.consumable.Consumable	BodyConsumables	ASSOCIATION	0...	true
BodyAsset	physical.asset.Asset	bodyAssets	ASSOCIATION	0...	true
CFFBody	operation.tactical.CallForFire	reqCFF	ASSOCIATION	1	true
RB1	operation.logistical.Requirement	initiaitedReqs	ASSOCIATION	0...	true
RB2	operation.logistical.Requirement	bodyReqsToFill	ASSOCIATION	0...	true

Class : physical.body.CivilianOrganization

Inheritance:

physical.body.Organization

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : physical.body.CombatServiceSupportUnit

Inheritance:

physical.body.GroundUnit

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.body.CombatServiceSupportUnit.CSSUnitType	physical.body.CombatServiceSupportUnit.eCSSUnitType				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : physical.body.CombatSupportUnit

Inheritance:

physical.body.GroundUnit

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.body.CombatSupportUnit.combatSupUnitType	physical.body.CombatSupportUnit.eCombatSupUnitType				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : physical.body.CombatUnit

Inheritance:

physical.body.GroundUnit

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.body.CombatUnit.combatUnitType	physical.body.CombatUnit.eCombatUnitType				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : physical.body.GroundUnit

Inheritance:

physical.body.MilitaryOrganization

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.body.GroundUnit.groun	physical.body.GroundUnit.eGro				

dUnitType	undUnitType				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : physical.body.MilitaryOrganization

Inheritance:

physical.body.Organization

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.body.MilitaryOrganization.numCombatPersonnel	long	public	0	read & write	
physical.body.MilitaryOrganization.numberCasualities	long	public	0	read & write	
physical.body.MilitaryOrganization.numSupportPersonnel	long	public	0	read & write	
physical.body.MilitaryOrganization.numHQPersonnel	long	public	0	read & write	
physical.body.MilitaryOrganization.echelon	physical.body.MilitaryOrganization.eEchelon				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : physical.body.Organization

Inheritance:

physical.body.Body

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.body.Organization.numMembers	long	public	0	read & write	
physical.body.Organization.orgSize	physical.body.Organization.eOrgSize				
	physical.body.Organization.eOrgType				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
OrgBody	physical.body.Body	membersOfInterest	ASSOCIATION	0...	true
TPOrg	operation.Trip	Trip_role	ASSOCIATION	1	false

Class : physical.body.Person

Inheritance:

physical.body.Body

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.body.Person.personStatus	physical.body.Person.ePersonStatus				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

---

Class : physical.body.SpecialOps

Inheritance:

physical.body.MilitaryOrganization

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.body.SpecialOps.specialOpsType	physical.body.SpecialOps.eSpecialOpsType				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

---

Class : physical.consumable.Ammunition

Inheritance:

physical.consumable.Munitions

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.consumable.Ammunition.numRounds	unsignedint	public		read & write	
physical.consumable.Ammunition.caliber	float	public		read & write	storage = mm display = in
physical.consumable.Ammunition.effectiveCasualtyRadius	float	public		read only	storage = m display = ft
physical.consumable.Ammunition.safetyRadius	float	public		read only	storage = m display = ft

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : physical.consumable.Cartridge

Inheritance:

physical.consumable.Ammunition

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.consumable.Cartridge.muzzleVelocity	float	public		read only	storage = m/s display = ft/s
physical.consumable.Cartridge.cartridgeType	physical.consumable.Cartridge.eCartridgeType				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : physical.consumable.Construction

Inheritance:

physical.consumable.Consumable

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.consumable.Construction.supplyClass	operation.logistical.eSupplyClass				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : physical.consumable.Consumable

Inheritance:

physical.item.Item

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.consumable.Consumable.rolledUpQuantity	unsignedlong	public		read & write	

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
BodyConsumable	physical.body.Body	Body_role	ASSOCIATION	1	false

Class : physical.consumable.GeneralSupportItems

Inheritance:

physical.consumable.Consumable

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.consumable.GeneralSupportItems.supplyClass	operation.logistical.eSupplyClass				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : physical.consumable.GuidedMunitions

Inheritance:

physical.consumable.Munitions

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
GuidedMunGuidSys	physical.asset.GuidanceSystem	guidenceSystem	ASSOCIATION	1	true

---

Class : physical.consumable.HandGrenade

Inheritance:

physical.consumable.UnguidedMunitions

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.consumable.HandGrenade.delayTime	long	public		read only	storage = s display = s
physical.consumable.HandGrenade.numberFragments	long	public		read only	

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : physical.consumable.MedicalMaterial

Inheritance:

physical.consumable.Consumable

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.consumable.MedicalMaterial.supplyClass	operation.logistical.eSupplyClass				
	app.sDate				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

---

Class : physical.consumable.Missile

Inheritance:

physical.consumable.GuidedMunitions

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.consumable.Missile.missileType	physical.consumable.Missile.eMissileType				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : physical.consumable.Munitions

Inheritance:

physical.consumable.Consumable

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.consumable.Munitions.supplyClass	operation.logistical.eSupplyClass				
	app.sDate				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
FSMunition	operation.tactical.FiringSolution	FiringSolution_role	ASSOCIATION	1	false

Class : physical.consumable.POL

Inheritance:

physical.consumable.Consumable

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.consumable.POL.POLType	physical.consumable.POL.ePOLType				
	operation.logistical.eSupplyClass				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
PlatformPOL	physical.asset.Platform	Platform_role	ASSOCIATION	1	false

Class : physical.consumable.PersonalDemandItems

Inheritance:

physical.consumable.Consumable

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.consumable.PersonalDemandItems.supplyClass	operation.logistical.eSupplyClass				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : physical.consumable.Pyrotechnics

Inheritance:

physical.consumable.Ammunition

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.consumable.Pyrotechnics.pyrotechnicType	physical.consumable.Pyrotechnics.ePyrotechnicType				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : physical.consumable.RepairItems

Inheritance:

physical.consumable.Consumable

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.consumable.RepairItems.supplyClass	operation.logistical.eSupplyClass				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : physical.consumable.Subsistence

Inheritance:

physical.consumable.Consumable

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.consumable.Subsistence.supplyClass	operation.logistical.eSupplyClass				
	app.sDate				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : physical.consumable.UnguidedMunitions

Inheritance:

physical.consumable.Munitions

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.consumable.UnguidedMunitions.unguidedMunType	physical.consumable.UnguidedMunitions.eUnguidedMunType				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : physical.environment.ArtificialEnvironment

Inheritance:

physical.environment.Environment

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : physical.environment.Base

Inheritance:

physical.environment.BuiltupArea

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
BRUII	physical.item.RolledUpItemInfo	minItemQtys	ASSOCIATION	0...	true

Class : physical.environment.BodyOfWater

Inheritance:

physical.environment.NaturalEnvironment

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.environment.BodyOfWater.seaState	physical.environment.BodyOfWater.eSeaState				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : physical.environment.BuiltupArea

Inheritance:

physical.environment.ArtificialEnvironment

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.environment.BuiltupArea.population	long	public		read & write	

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : physical.environment.Environment

Inheritance:

physical.Physical

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.environment.Environment.area	float	public		read & write	storage = m2 display = m2
physical.environment.Environment.secured	boolean	public	false	read & write	

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
ActPtEnv	operation.ActivityPt	envActivityPts	ASSOCIATION	0...	true
RE1	operation.logistical.Requirement	receptionReqs	ASSOCIATION	0...	true
RE2	operation.logistical.Requirement	envReqsToFill	ASSOCIATION	0...	true
PhysicalEnv	physical.Physical	envContents	ASSOCIATION	0...	true
PRE	operation.logistical.PickupRequirement	itemPickupReqs	ASSOCIATION	0...	true

Class : physical.environment.ISEP

Inheritance:

physical.environment.Base

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

---

Class : physical.environment.LogInventoryReserve

Inheritance:

physical.environment.Base

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

---

Class : physical.environment.NaturalEnvironment

Inheritance:

physical.environment.Environment

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

---

Class : physical.environment.SeaBase

Inheritance:

physical.environment.Base

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
SeaBaseVessel	physical.asset.Vessel	housingVessels	ASSOCIATION	0...	true

Class : physical.environment.Space

Inheritance:

physical.environment.Environment

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.environment.Space.spaceDims	app.s3DDims				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
PS	physical.Physical	spaceOwner	ASSOCIATION	1	false

Class : physical.environment.StorageZone

Inheritance:

physical.environment.Zone

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.environment.StorageZone.maxWeightCapacity	float	public		read only	storage = kg display = ton

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : physical.environment.Zone

Inheritance:

physical.environment.Space

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

Class : physical.item.Item

Inheritance:

physical.ArtificialMobile

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.item.Item.model	string	public		read only	
physical.item.Item.itemStatus	physical.item.Item.eItemStatus				
	app.s3DDims				
	physical.item.Item.eItemLoca tion				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
III	physical.item.ItemInfo	itemInfos	ASSOCIATION	0...	true
CTCII	operation.logistical.CargoTransaction	cargoTransactions	ASSOCIATION	0...	true

Class : physical.item.ItemContainer

Inheritance:

physical.item.Item

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.item.ItemContainer.storageDimensions	app.s3DDims				
	physical.item.ItemContainer.eContainerType				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
ICII	physical.item.ItemInfo	contentInfo	AGGREGATION	0...	true

Class : physical.item.ItemInfo

Inheritance:

information.Information

Attributes:

Name	Type	Visibility	Default	Access	Units

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
ICII	physical.item.ItemContainer	itemContainer	ASSOCIATION	1	false
III	physical.item.Item	item	ASSOCIATION	1	true
CILII	operation.logistical.CriticalItemList	CILs	ASSOCIATION	1	false
ReqII	operation.logistical.Requirement	itemRequirement	ASSOCIATION	1	false

Class : physical.item.RolledUpItemInfo

Inheritance:

physical.item.ItemInfo

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.item.RolledUpItemInfo.quantity	unsignedlong	public		read & write	

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable
BRUII	physical.environment.Base	minItemQtysBase	ASSOCIATION	1	true

Class : physical.weather.WeatherSystem

Inheritance:

physical.NaturalMobile

Attributes:

Name	Type	Visibility	Default	Access	Units
physical.weather.WeatherSystem.area	float	private		read & write	storage = m2 display = ft2
physical.weather.WeatherSystem.weatherSystemType	physical.weather.WeatherSystem.eWeatherSystemType				

Associations:

Association Name	Associated Class	Associated Role Name	Type	Multiplicity	Navigable

11. Keyword Index