CDM TECHNICAL REPORT: CDM-15-04

SILS MRAT: A Multi-Agent Decision-Support System for Shipboard Integration of Logistics Systems

Michael Zang
Jonathan Lee
Joyce Gaoiran
Adam Gray
Jered Gray
Charles Hayek
David Nau
Chad Pond
Michael Rutter
Zachary Speck

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. on subcontract to ManTech Advanced Systems International, Inc. (Fairmont, West Virginia), and under sponsorship of the Office of Naval Research (ONR). The principal aim of the SILS (Shipboard Integration of Logistics Systems) project is to provide a decision-support capability for Navy ships that integrates shipboard logistical and tactical systems within a near real-time, automated, computer-based shipboard readiness and situation awareness facility. Specifically, SILS is intended to provide the captain of a ship and his staff with an accurate evaluation of the current condition of the ship, based on the ability of all of its equipment, services and personnel to perform their intended functions.

The SILS software system consists of two main subsystems, namely: the SILS IE (Interface Engine) subsystem for information interchange with heterogeneous external applications, developed by ManTech Advanced Systems International; and, the SILS MRAT (Mission Readiness Analysis Toolkit) subsystem for intelligent decision-support with collaborative software agents, developed by CDM Technologies. This report is focused specifically on the technical aspects of the SILS MRAT subsystem.

The automated reasoning capabilities of SILS MRAT are supported by a knowledge management architecture that is based on information-centric principles. Such an

architecture utilizes a virtual model of the real world problem situation, consisting of data objects with characteristics and a rich set of relationships. Commonly referred to as an ontology, this internal information model provides a common vocabulary and ***context*** for software agents with reasoning capabilities. The concurrent need for incremental capability increases implies a steadily increasing data load from diverse operational (dynamic) and historical (static) data sources, ranging from free text messages and Web content to highly structured data contained in consolidated operational data stores, Data Warehouses, and Data Marts. In order to provide useful high-level capabilities the architecture is required to support the transformation of these data flows into information and knowledge relevant to the concerns and operational context of individual shipboard users. Accordingly, the system must be capable of not only storing data but also the relationships and higher level concepts that place the data into context. For this reason, to manage an increasing number of relationships and concepts over time, the SILS MRAT subsystem was designed to employ a formalized ontological framework.

There were four additional considerations in the selection of the overall SILS architecture. First, ***utility*** to support a useful level of automated information management (i.e., the ability to collaboratively analyze data, monitor dynamic operational context, formulate warnings and alerts, and generate recommendations). Second, ***flexibility*** to accommodate contributions from multiple team members that may employ differing technologies and implementation paradigms. Third, ***scalability*** to allow a progressive increase in the breadth and diversity of the data sources, the volume of data processed, the number of validated components, and the intelligence of the tools (i.e., agents). Fourth, ***adaptability*** to facilitate the tailoring of the information management capabilities to different data sources and existing data environments. The current SILS architecture addresses these desirable characteristics by partitioning the system into a lower-level data collection and integration layer, a higher-level information management layer (SILS MRAT), and a translation facility that is capable of mapping the data schema of the lower layer to the information representation (i.e., ontology) of the upper layer (SILS IE).

The higher-level information management layer provides a collaborative, distributed communication facility that supports the development of semi-autonomous modules of capability referred to as agents. The agents employ the formalized ontology supported by the communication facility to collaborate with each other and the human users in a meaningful manner.

Acknowledgements

The work on the SILS MRAT subsystem of the Shipboard Integration of Logistics Systems (SILS) program, described in this report, was performed under subcontract to ManTech Advanced Systems International, Inc. and is being sponsored by the Logistics Program Office of the Office of Naval Research (ONR), Washington (DC).

SILS MRAT: A Multi-Agent Decision-Support System for Shipboard Integration of Logistics Systems

Table of Contents

1. Introduction

There is a need for the integration of shipboard logistical and tactical systems within a near real-time, automated, computer-based shipboard readiness and situation awareness decision-support facility. Such a facility should be able to provide the captain of a ship and his staff with an accurate evaluation of the current condition of the ship, based on the ability of all of its equipment, services and personnel to perform their intended functions, and an overall assessment of the ability of the ship to undertake any given mission.

With a view of satisfying this need, the concept of SILS (Shipboard Integration of Logistics Systems) was conceived by Dr. Phillip Abraham of the Office of Naval Research. The prime contractor, ManTech Advanced Systems International, Inc. (Fairmont, West Virginia), in conjunction with CDM Technologies, Inc. (San Luis Obispo, California), Thomas Galie (NSWCCD, Philadelphia, Pennsylvania), and Chris Neff (CINCPACFLT, Hawaii), developed this concept into a project focused on the analysis and demonstration of agent-based decision-support software technology.

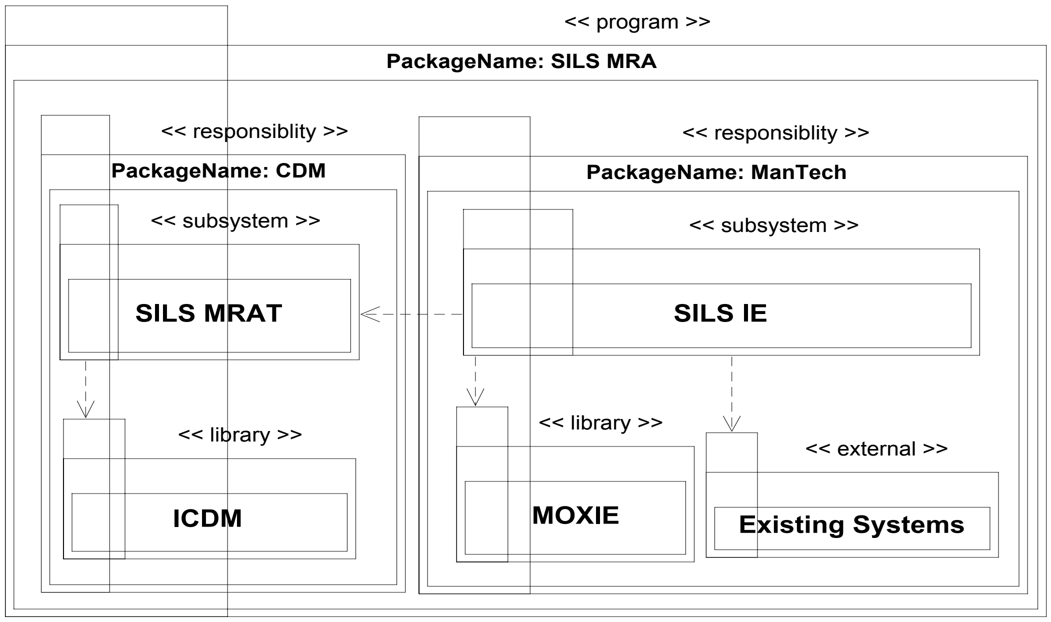


Figure 1: Development responsibilities

ManTech Advanced Systems International, Inc. was responsible for providing the SILS Interface Engine (SILS IE), a subsystem for information interchange between independently developed external systems. The SILS Interface Engine is a domain specific configuration based on a generic core known as the ManTech Object eXchange Interface Engine (MOXIE). The SILS Interface Engine is a key component of SILS as it is intended to provide most of the information upon which the decision-support activities are dependent. This top-level executable is generically described as the Interface Engine in the SILS Architectural Design Report (CDM 2001c). As a subcontractor to ManTech, CDM Technologies was responsible for providing the SILS Mission Readiness Analysis

Toolkit (SILS MRAT) an ICDM-based toolkit for collaborative decision-support. These development responsibility interrelationships are schematically depicted in Figure 1.

The Integrated Cooperative Decision Making (ICDM) toolkit provides a software development framework that facilitates the design and production of distributed, multi-agent, decision-support applications. Over the past several years, the evolving ICDM framework has been used with great success by CDM Technologies, Inc. and others for the design and development of large-scale applications for military customers. Apart from SILS MRAT for the US Navy, examples include the Integrated Marine Multi-Agent Command and Control System (IMMACCS) for the US Marine Corps, and the logistical ICODES and SEAWAY systems for the US Army and US Navy, respectively.

The core component of an ICDM-based application is a virtual representation of the real world entities and relationships that define the context of the application domain, in the form of an internal information model (i.e., ontology). The representation of information (i.e., data and relationships) allows the construction of software modules, referred to as agents, capable of performing tasks that require reasoning capabilities. Examples of such tasks include: the monitoring of events in dynamically changing situations; the detection of conflicts; the triggering of warnings and alerts; the formulation and evaluation of alternative courses of action; and, the collaborative assessment of situations. Typically, in ICDM-based applications the agents collaborate with each other and the human users in their monitoring, planning, and evaluation activities.

At the top level, SILS MRAT is comprised of: a collaborative, distributed, object-serving communication facility that houses a collection of executable domain object models; an agent engine housing a collection of collaborative agent federations; and, an end-user interface. These top-level subsystems and executable components (processes) are described in the SILS Architectural Design Report (CDM 2001c).

In summary, the essential objective of the SILS project is to provide responsive near real-time decision-support to a commanding officer and his principal department heads. It is also intended to demonstrate the capability of an information-centric, collaborative agent, software architecture to integrate existing systems deployed by the U.S. Navy and thus contribute to reducing the interoperability problems currently experienced by the Navy. Finally, SILS will act as a pilot for subsequent development of a deployable 'readiness node' at the ship level and as a basis of a more comprehensive, integrated fleet decision-support and readiness evaluation system.

2. Overview

The objective of SILS is to provide a comprehensive, near real-time decision aid to the commanding officer and the senior staff of a ship. Its general focus is on the planning and allocation of ship-related mission resources and the identification of changes in readiness status. More specifically SILS:

1. Provides agent-based shipboard decision-support to the commanding officer and senior enlisted personnel.
2. Allows users to view and develop the operational schedule and relate tasks to their required resources.
3. Provides users with agent-generated alerts and change notifications in response to changes in readiness status.
4. Allows users to customize and extend the status reporting features and mechanisms.
5. Integrates with shipboard information, control, and monitoring systems.

As a subset of SILS, MRAT is intended to serve two primary purposes. First, to assist the ship's captain and his officers with the logistically ship preparations for deployment. Second, upon deployment, to collaboratively assist in identifying the readiness levels and associated risks of major ship systems. The top-level goals of SILS MRAT therefore include the following:

1. Integrate information in a manner that will allow the captain to intuitively relate SILS information to his mission.
2. Present the captain with top-down readiness assessments of ship systems, schedules and personnel.
3. Provide 'at a glance' the current status and readiness posture of the ship.
4. Assist the captain in developing alternative courses of action.

Furthermore, SILS MRAT demonstrates the contributions that a collaborative decision-support system could make in assessing the readiness of a ship to perform a particular mission or task, by integrating information across individual ship information systems and personnel. In concept, SILS MRAT receives information from:

1. Sources internal to the ship such as the personnel, maintenance, logistics, supplies, and engineering readiness aspects of a ship's posture. Information sources can include embedded sensors, inputs from key personnel (e.g., the engineering officer), official ship performance characteristics, official usage data of various types, and the captain's instructions.
2. Sources external to the ship such as the environmental conditions, the ship's mission, and availability of logistical support and supplies.

Utilizing a hierarchal representation of the ship as a system of systems, software agents reason about this information. As mission discrepancies and/or potential failures are noted, the agents generate alerts, implication statements, projections, and/or alternatives. While these outputs will often identify a specific problem within a particular system, the system will also attempt to interpret the incoming information more broadly to support decision-making tasks related to the assessment of whether the ship is ready to perform a particular mission. Current and planned SILS MRAT agents include the following:

- **Mission Capability Agent:** Identifies high-level problems that affect the ship's overall ability to perform a mission.
- **Training and Performance Agent:** Identifies training and performance deficiencies.
- **Combat Systems Agent:** Monitors the health of the ship's combat systems.
- **HM&E Systems Agent:** Monitors the health of the ship's hull, as well as its mechanical, and electrical systems.
- **Navigation and Communication Agent:** Monitors the health of the ship's navigation and communication systems.
- **Damage Control Agent:** Monitors the damage status of the ship and the health of the ship's damage control systems.
- **Supply Agent:** Monitors the supply status of the ship.
- **Personnel Agent:** Monitors the manning status of the crew.
- **Environment Agent:** Monitors the external physical environment for conditions that may affect the mission.
- **Environmental Protection Agent:** Monitors the impact of operations on environmental regulations.
- **Rules of Engagement Agent:** Monitors operations for compliance with published rules of engagement.
- **Interface Agent:** Monitors the health of interfacing decision-support and information systems.

Top-level mission readiness assessments are presented with symbols providing immediate visibility of a ship's readiness status biased by mission type.

🟢 **Fully Mission Capable:** All major equipment and systems are fully capable of performing all required functions without reservations.

🟢 **Mission Capable** – All major equipment and systems are capable of performing all required functions with some reservations.

🟡 **Marginally Mission Capable:** All major equipment and systems are capable of performing all required functions with major reservations.

🔴 **Not Mission Capable in Selected Areas:** Not capable of performing selected major functions in a primary mission area.

Not Mission Capable: Major discrepancies exist in one or more key functional areas, making the ship incapable of accomplishing a primary mission.

The SILS MRAT interface is composed of a collection of application tools, primary among these is the Mission Readiness Assessment Tool depicted in Figure 2. This tool consists of a collection of panels and toolbars that contain views of the ship status and readiness conditions, and provide alerts, messages, and notifications to ship officers. The interface is designed to facilitate rapid assessment of problem conditions and assist in the decision-making process by exposing interrelated dependencies caused by seemingly disparate problems and providing facilities to assist in the development of appropriate courses of actions.

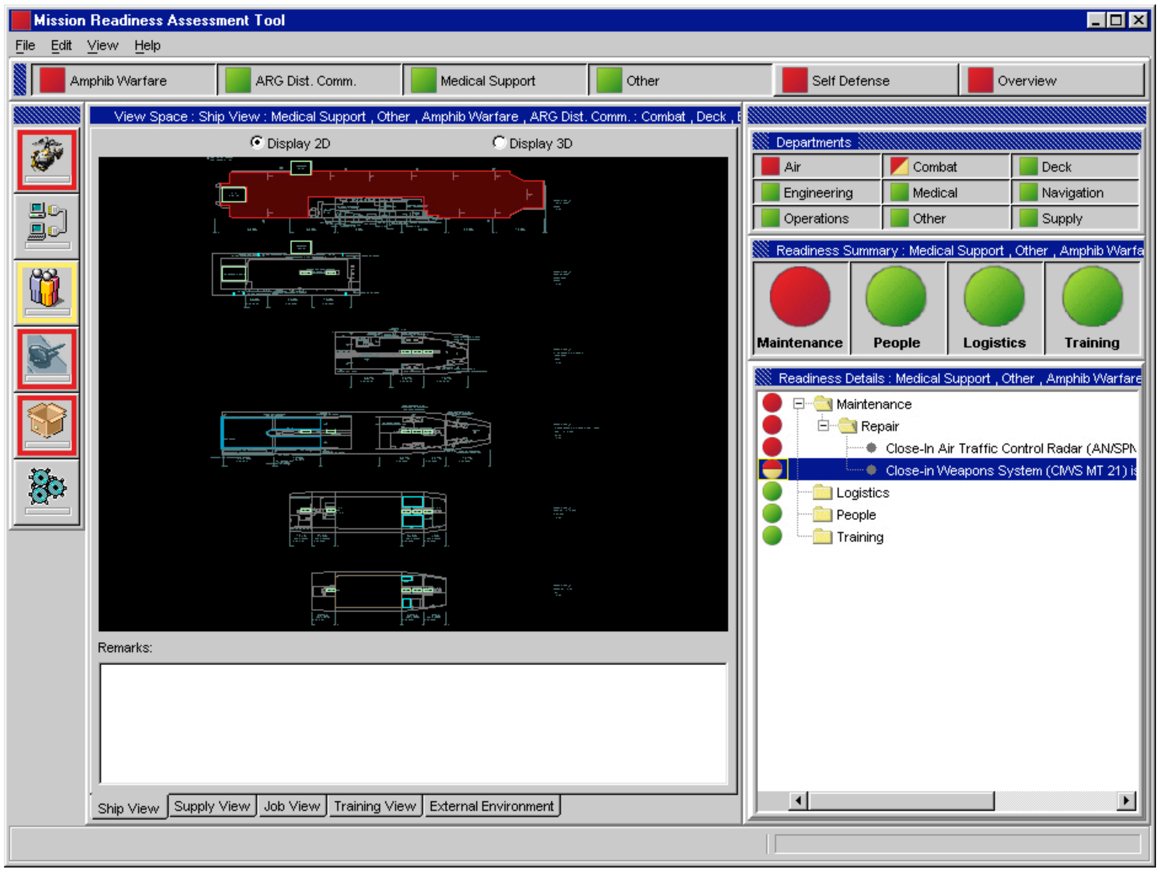

Mission Readiness Assessment Tool
File
Edit
View
Help
Amphib Warfare
ARG Dist. Comm.
Medical Support
Other
Self Defense
Overview
View Space : Ship View : Medical Support , Other , Amphib Warfare , ARG Dist. Comm. : Combat , Deck ,
Display 2D
Display 3D
Remarks:
Ship View
Supply View
Job View
Training View
External Environment
Departments
Air
Combat
Deck
Engineering
Medical
Navigation
Operations
Other
Supply
Readiness Summary : Medical Support , Other , Amphib Warfa
Maintenance
People
Logistics
Training
Readiness Details : Medical Support , Other , Amphib Warfare
Maintenance
Repair
Close-In Air Traffic Control Radar (AN/SPN
Close-in Weapons System (CIWS MT 21) is
Logistics
People
Training

Figure 2: Main screen of the Mission Readiness Assessment Tool (MRAT)

3. Operational Concepts and Typical Scenario

The SILS shipboard decision-support system has a significant logistics component integrated into a comprehensive decision aid for the ship's captain. The system is designed to support: planning and allocation of scarce mission resources; scheduling and integration of readiness-related activities; identification of opportunity costs; and, maintain a desirable level of distributed situation awareness among the ship's captain and senior enlisted personnel. In the latter role, this information-centric decision aid will essentially function as an intelligent integration facility for the principal existing data-centric shipboard systems that contribute to mission readiness.

Among the most important decisions made by the captain of a ship are those that determine the readiness of the ship for war. The captain is singularly responsible for executing this responsibility, which includes planning all preparations, allocating the finite resources under his control, requesting and justifying external resources, supervising the process, and continually assessing the ship's status to ensure that appropriate levels of readiness are attained and maintained.

The relative role that decision-support plays in the execution of these responsibilities will differ in the various phases of the deployment process. During pre-deployment (i.e., work up), while still under the Type Commander, the principal role of decision-support is in planning deliberate resource allocation. This includes evaluating the likely results of "shorting" some element of the mission in favor of fully allocating resources to another mission element. Such considerations will eventually lead to a command decision to accept shortfalls and/or to use some form of consequence analysis as support and justification in a request for external assistance. In as much as it is designed to be a distributed system, it may prove feasible to use SILS as a means of submitting the request with necessary justification and illustration.

Once the ship deploys to an operational area, the situation changes significantly. Here, external assistance is likely to be much more restricted and always subject to extended delays. Under these circumstances, it is vital to accurately characterize the elements involved in shifting resources and then weigh the consequences of the shift in a timely manner. This is a principal responsibility of command. Before deployment SILS will provide decision-support to the captain, and to a lesser extent the ship's officers, with the overall objective of characterizing the relationship between resources and mission performance on the one hand and likely consequences on the other hand.

A mission is a broad military tasking designed to accomplish a specified objective. As such, missions are composed of elements, which in turn consist of specific tasks to be performed to established standards. Ships can be judged to be fully "mission capable" (i.e., ready to perform all elements of the mission adequately) or "mission capable with exceptions". In the latter case, an exception results from one or more tasks that cannot currently be performed to standard. The assessment of the resources required to reach an acceptable performance level in the deficient areas, highlighting the potential to "under resource" another mission area, and identifying consequences in terms of the mission, are the focus of SILS.

A mission typically comprises phases and alternative methods of executing each phase. Decision assistance is desirable and appropriate for both mission phase planning and mission phase execution. Each phase has requirements based on where the phase will be executed and what elements will be utilized. Phase requirements can be broken down into training, supply, maintenance preparation, and so on. Each of these requirements can be characterized in terms of system readiness to complete the phase.

SILS is designed to link resources-to-tasks-to-mission elements during the formal resource planning and readiness work-up process, by first linking tasks to nominally required resources and then linking these to mission elements. Thereafter, rankings are drawn from the captain's mission analysis and assigned to the mission elements. In the case of the supply requirement, a decision-support advisor could consider the on-board assets, compare these to the required phase assets, and discover an asset delta (i.e., shortfall). The delta could be met in a number of ways. For example, it could be ignored or it could be satisfied by utilizing an external assets. On the other hand, it might be possible to change the training program or institute new procedures. Each of these options has risk levels associated with it.

The process will change when the ship is deployed. During work-up prior to deployment, the decision process is more formal, more deliberate, and without the urgency that pertains once the ship is deployed.

While, during work-up the ship might have been able to wait for a particular repair, once deployed a long delay could seriously impact the mission. A related factor in SILS decision-support is the system's role in nominating "best of feasible options", while the ship is deployed. The options available to a deployed ship usually differ significantly from those available while the ship is in port. Hence, useful decision-support will propose "best of feasible" solutions. The ranking will be a function of the captain's decision of which variable is most important (e.g., immediate fix, cheapest fix, etc.) and his assessment of the risks inherent in each option.

SILS is capable of identifying the risk involved in trading off the performance of one or more mission elements by shifting resources to improve performance elsewhere. Such a trade-off may be accomplished by characterizing the risk in terms of the effect that the resource shift exerts on the tasks associated with that particular mission element. This task-resource connection is a fundamental relationship within SILS. The kinds of resources that SILS is capable of taking into account during trade-off analysis include: time (i.e., for training, repair, etc.); funds and personnel (i.e., quantity, correct NEC, training status, etc.); and, outside assistance (i.e., supplies, services, etc.).

SILS is designed to automatically perform this type of analysis and offer the corresponding decision-support services while each phase of a mission is underway. As it detects shortfalls and deficiencies, it generates alerts and provides advice based on the assessment of conditions. In addition, SILS is able to support user-initiated, explicit requests for information on the status of specific mission elements and the feasibility assessment of a particular course of action.

3.1 Typical Mission Scenario

While performing distance support communications for the USS Peleliu of an Amphibious Readiness Group (ARG) in the North Arabian Sea, a routine inspection reveals a problem with the close-in air traffic control radar (AN/SPN-35B). This problem results in the suspension of all air operations. A technician on board the ship generates a maintenance action form (4790-2K) in the OMMS-NG system and a supply part requisition in the R-Supply (1250) system to initiate the repair procedure.

The ship receives notification to be prepared to conduct amphibious operations within the next 96 hours. This results in pre-mission checks, which determine that the Close-In Weapons System (CIWS MT 21) is not fully operational and requires a circuit card replacement. A maintenance technician generates a maintenance action form (4790-2K) and supply part requisition (1250) to replace the faulty circuit card.

The Supply Officer (SUPPO) receives notification of the part requisition for the CIWS repair and sees that the part is not on board, but is available on USS Comstock. While investigating the possible acquisition of the part from USS Comstock, he notes that all air operations have been suspended due to a pending repair. This unfortunately prevents any part requisition from arriving by air. The SUPPO then notices that the part may be repairable on board through the Gold Disk Program (2M/MTR). He checks the personnel roster and sees that the ship has a 2M technician on board with the correct NEC to perform the repair.

The circuit card is delivered to the 2M technician who generates the appropriate 4790-2K maintenance action form, performs the repair, and has the part delivered to the ship technician responsible for the CIWS system repair. The ship technician installs the card, verifies that the CIWS is operational, completes maintenance action form (4790-2K), and informs the 2M technician of the successful repair. The latter then completes the appropriate maintenance action form (4790-2K) for the circuit card repair.

The SUPPO notes the parts requisition for the close-in air traffic control radar (AN/SPN-35) repair and sees that the part is on board. He has the part taken to the ship technician responsible for the AN/SPN-35 repair, who then performs the repair and subsequently completes the required maintenance action form (4790-2K).

The Commanding Officer (CO) reviews the state of the ship biased for the pending amphibious assault. Noting that he apparently has no pending readiness issues, he checks the operational schedule and verifies that all pre-mission checks have been completed and that all prerequisite maintenance activities have been performed.

This scenario demonstrates the collaboration between agents and key ship personnel while showcasing the readiness assessment capability provided by SILS. It involves two concurrent system failures, namely the AN/SPN-35B air control radar and the CIWS MT 21 air defense system. The AN/SPN-35B failure effects the repair options of the more critical CIWS MT 21 failure.

3.2 The CO Assesses Ship Readiness

A routine PMS inspection reveals a problem with the AN/SPN-35B close-in air traffic control radar. A ship technician generates a 4790-2K Maintenance Action Form for AN/SPN-35 in the OMMS-NG maintenance system aboard ship and a 1250 Supply Part Requisition in the R-Supply system. These actions are automatically recorded by the SILS Interface Engine (i.e., due to the standing, source independent, agent subscriptions for maintenance actions and supply requisitions). The Interface Engine gathers the appropriate details from the source systems and posts them to the SILS MRAT Interface Domain, which in turn triggers the SILS MRAT Interface Agent to translate the posted information to the core Problem Domain for agent analysis. Shortly thereafter the CO receives notification that he is to be prepared to conduct amphibious operations within the next 96 hours.

The CO goes to a conveniently located computer workstation to check on the current state of his ship. He uses his personally configured SILS Dashboard (Figure 3) for his initial assessment and notes that the Overall Status Indicator is ‘red’, indicating ‘not mission capable’. In the Mission Status pane he can see that there are major readiness issues related to amphibious warfare. Additionally, the Department Status Pane indicates that the amphibious warfare problems are associated with the Air and Combat departments.

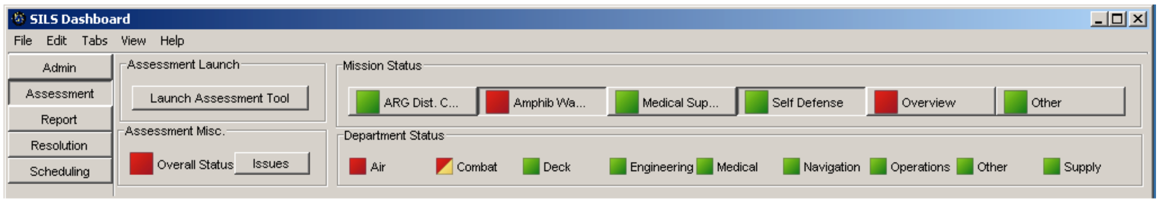


Figure 3: Commanding Officer’s SILS Dashboard

The CO decides further investigation is warranted and launches the SILS Mission Readiness Assessment Tool (MRAT) from his Dashboard, displayed in Figure 4. In the toolbars across the top of the application the CO selects “Amphib Warfare” in the Mission Bias toolbar and the “Air and Combat” departments in the Department Bias Toolbar. Then he selects the “Maintenance Mission Area” on the right.

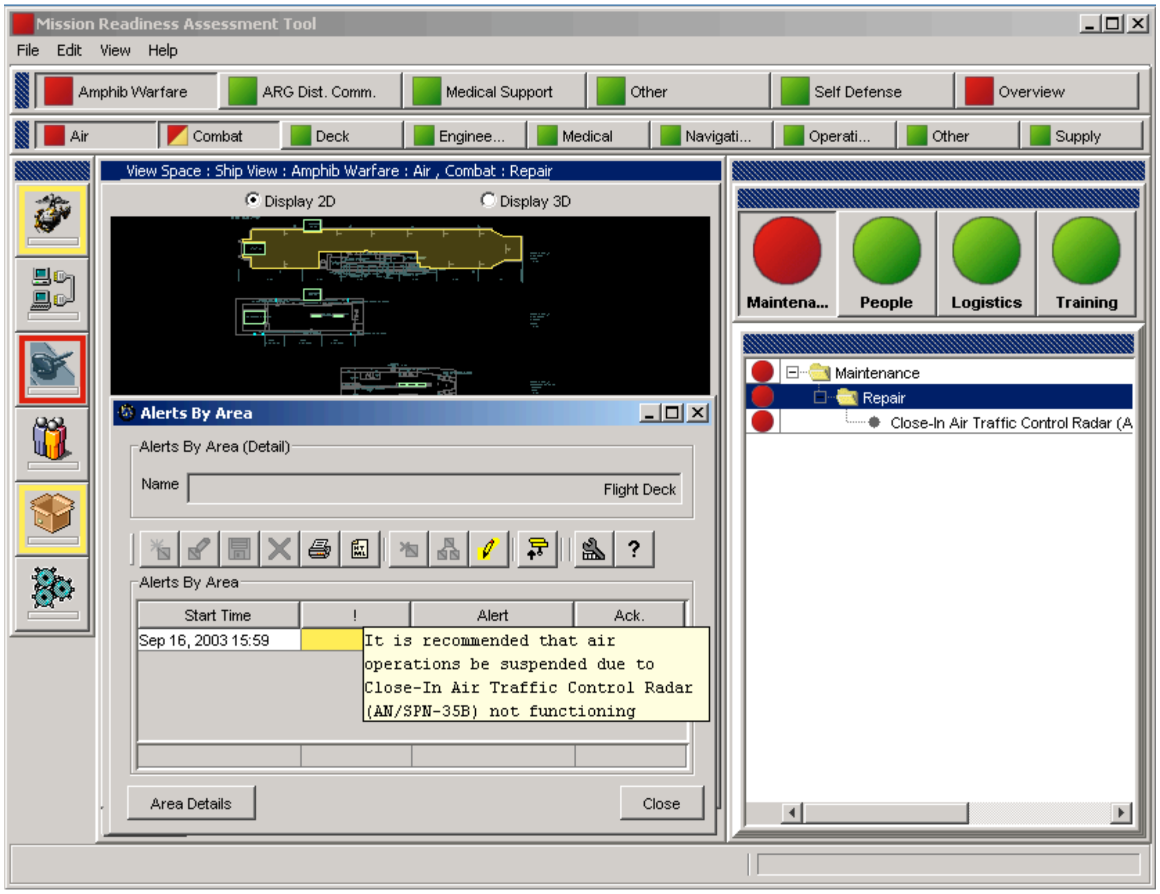
These selections impart a two level bias on the Readiness Observation Pane just below the Mission Area Pane, to narrow the information to show only those readiness observations related to Maintenance on Air or Combat department equipment that affect amphibious operations. The CO drills down the hierarchically categorized readiness observation tree and notes that the close-in air traffic control radar is down. He then checks the Problem Presentation Space to the left of the Readiness Observation Pane to review contextual information associated with the selected readiness observation. In the graphical Ship View he notes that the flight deck is depicted in yellow indicating associated agent alerts. He directly clicks on the graphical depiction of the flight deck to obtain a context sensitive menu from which he elects to display the alerts associated with the flight deck.

This action prompts the display of the “Alerts by Area” dialog. This shows an agent recommendation to suspend air operations due to the failure of the AN/SPN-35B radar. The CO concurs with the recommendation and elects to suspend air operations.

3.3 The SUPPO Initiates On Board Repairs

Through pre-mission checks in anticipation of an amphibious assault the Supply Officer (SUPPO) has determined that the CIWS MT 21 close-in weapons system is not fully operational. The ship technician generates a 4790-2K Maintenance Action Form for

CIWS in the OMMS-NG system and a 1250 Supply Part Requisition in the R-Supply system to replace the faulty circuit card. At his desk in the supply department, the SUPPO notes a Supply Agent state change on his personally configured SILS MRAT Dashboard.



Sep 16, 2003 15:59

It is recommended that air operations be suspended due to Close-In Air Traffic Control Radar (AN/SPN-35B) not functioning

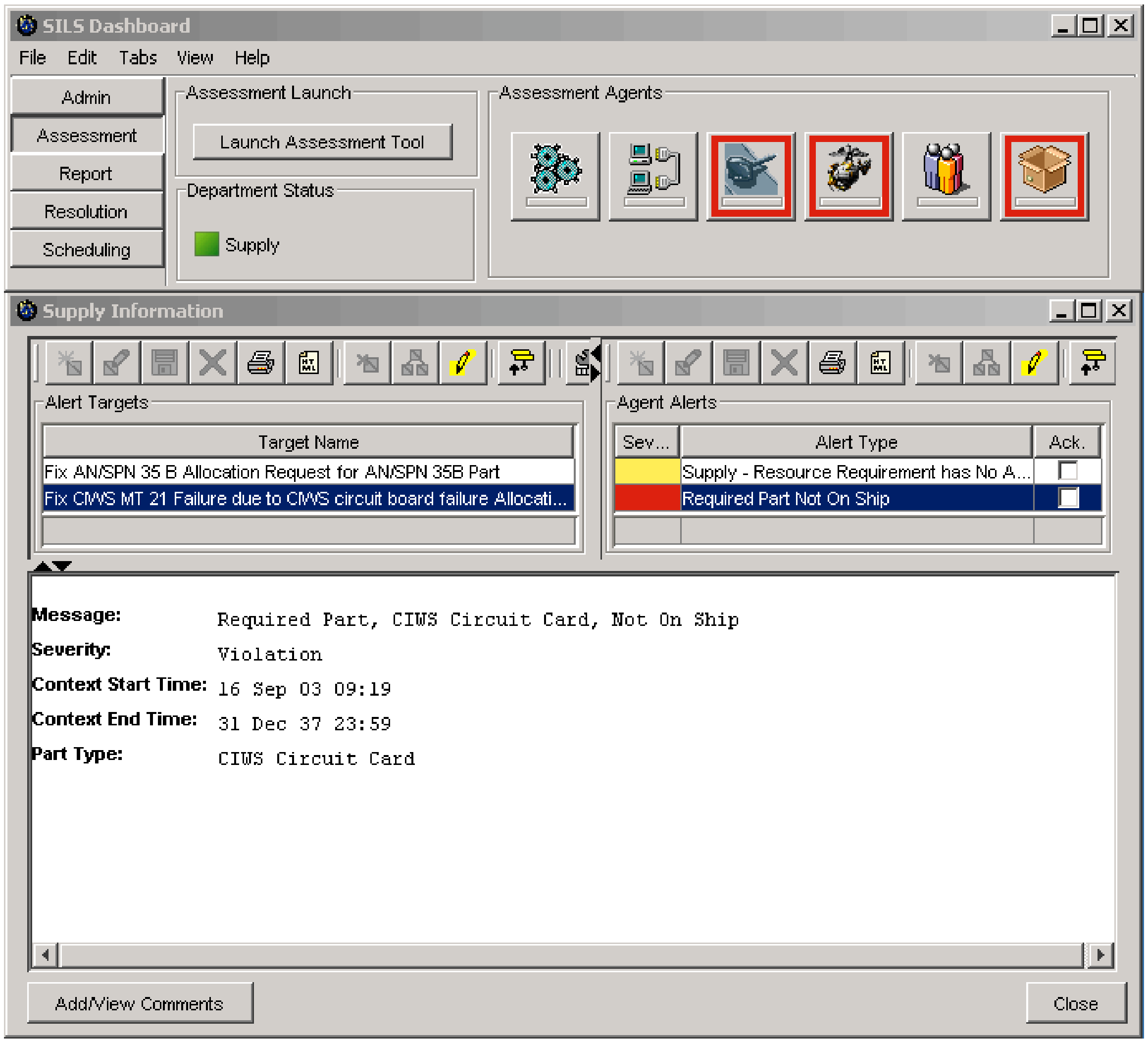
Figure 4: Commanding Officer's Assessment Tool

The SUPPO opens the Supply Agent Supply Information Dialog directly from the Dashboard by clicking on the Dashboard Supply Agent icon resulting in the display depicted in Figure 5. He launches the Assessment Tool, notes the Supply Agent state, and clicks on the Supply Agent to view the alert report. The report indicates two pending supply part requisitions. He looks at both issues and decides to focus first on the CIWS repair, as it represents a critical ship defense capability.

The SUPPO drills down the Maintenance Readiness Area hierarchy to the CIWS readiness issue, which he selects as a means of further biasing the problem presentation space in respect to his immediate concerns.

Subsequently the SUPPO returns to the SILS Dashboard, selects the "Resolution" tab and launches the Resolution Tool to review suggested approaches to resolving the CIWS problem. In the Resolution Tool, he clicks on the "Suggested Approaches" option to observe the possible impact of each alternative approach. Finally, closing the Resolution Tool, the SUPPO returns to the Assessment Tool, reviews the "View-Space Supply" view, and notes that the required part is not aboard ship but is available from the USS Comstock at a cost of $7,203.

The SUPPO looks at the "View Space-Ship" view, double-clicks on the "Flight Deck", selects "Alerts", and notes that air operations have been suspended. Next, the SUPPO looks at the "View Space-Job" view and notes that the required 1591 Gold Disk NEC is available on board. Given the high price and current problems with air operations the SUPPO decides to initiate on board repairs.



Target Name
Fix AN/SPN 35 B Allocation Request for AN/SPN 35B Part
Fix CIWS MT 21 Failure due to CIWS circuit board failure Allocati...

Sev...	Alert Type	Ack.
🟨	Supply - Resource Requirement has No A...	☐
🟥	Required Part Not On Ship	☐

```
Message:             Required Part, CIWS Circuit Card, Not On Ship
Severity:            Violation
Context Start Time:  16 Sep 03 09:19
Context End Time:    31 Dec 37 23:59
Part Type:           CIWS Circuit Card
```

Figure 5: Supply Officer views the Supply Agent's report

3.3.1 The 2M Technician Repairs the Circuit Card

The CIWS circuit card is delivered to the 2M technician, who generates the appropriate 4790-2K maintenance action form, performs the repair, completes the 4790-2K maintenance action form for the circuit card repair, and has the part delivered to the appropriate work center.

Thereafter the CO returns to the Dashboard, selects the "Scheduling" tab, launches the scheduling tool, reviews the scheduled tasks, and closes the "Scheduling Tool".

3.3.2 The Ship Technician Repairs the CIWS

The ship technician sees that the part is now available at the work center. He picks up the card and installs it in the CIWS, verifies that the CIWS is operational, and completes the 4790-2K maintenance action form for system repairs.

The SUPPO now checks the "View-Space Supply" view, biased by the CIWS readiness observation, and sees that the part is now available. He immediately notifies the ship technician, who performs the repair.

The CO checks the Assessment Tool, right clicks on the "Maintenance Readiness Summary", and notes that the "Close-in Weapons System (CIWS MT 21) is not fully operational" notice is no longer displayed.

3.3.3 The AN/SPN-35 is Repaired

The SUPPO notes the parts requisition for the AN/SPN-35 repair and sees that the part is on board. He has the part taken to the ship technician responsible for AN/SPN-35 repairs. The designated ship technician performs the repair and completes the required 4790-2K maintenance action form.

The CO notes that the ship is fully mission capable for all missions and re-establishes air operations.

3.3.4 The CO Reviews the State of the Ship

The CO reviews the state of the ship biased for the pending amphibious assault. He first notes that he has no pending readiness issues, and then checks the operational schedule to verify that all pre-mission checks have been completed and that the prerequisite maintenance activities have been performed.

4. Top Level Requirements

Succinctly stated, the design goals of SILS were aimed at increasing the situation awareness, responsiveness, and programming productivity of the captain and his command staff. The approach taken by the SILS project team to achieve these goals involved the development of a generic ontological model capable of representing the principal shipboard readiness factors and supporting collaborative agents with automatic reasoning capabilities.

4.1 Functional Requirements

The functional requirements of SILS were defined by members of the potential user community in terms of the following five assistance areas:

1. Portrayal and development of the operational schedule.
   - 1.1 Provision for drilling down within the operational schedule to identify sub-tasks.
   - 1.2 Provision for relating tasks to sub-tasks and sub-tasks to resources.
   - 1.3 Provision for portraying the operational schedule in terms of primary mission activities and supporting preparatory activities.
   - 1.4 Provision for designing proposed missions and integrating them into an existing operational schedule.
   - 1.5 Provision for providing the operational schedule in a format accessible to non-SILS users.
   - 1.6 Provision for presenting estimated completion percentages for parent-level tasks.

2. Provision of extensible status reporting features and mechanisms.
   - 2.1 Provision for users to dynamically change the format of the consolidated assessments within their status reports.
   - 2.2 Provision for users to establish new assessment requirements, implementing protocols, and supporting phenomena.
   - 2.3 Provision for scheduling, executing, and recording readiness status updates.
   - 2.4 Provision for individual departments to generate and post departmental status reports.
   - 2.5 Provision for individual users, particularly commanding officers, to develop consolidated ship status reports emphasizing particular areas of concern.
   - 2.6 Provision for tracing overall combat readiness assessments to their constituent components.

2.7 Provision to provide status reports in a format accessible to non-SILS users.

3. Provision of decision-support to the commanding officer and primary users.
    3.1 Ability to use the assessment capabilities to project readiness status.
    3.2 Ability to evaluate proposed missions in terms of key material and training requirements.
    3.3 Ability to relate readiness assessments to required actions.
    3.4 Ability to relate mission preparations to required actions.
    3.5 Ability to coordinate and access resources for related actions.
    3.6 Ability to relate standard assessment requirements to any of the principal missions.

4. Generation of alerts and change notifications.
    4.1 Ability to automatically alert users of significant changes in readiness status.
    4.2 Provision for users to filter alerts to reflect their particular interests.

5. Interaction with existing shipboard systems.
    5.1 Ability to establish, manage, and maintain connections to existing systems.
    5.2 Provision for accessing data affecting ship readiness in existing systems and mapping those data to the information model used by SILS.
    5.3 Provision for translating information held in the SILS information model to the data schema of existing systems, and pushing these data to those external systems.

4.2 Ancillary Requirements

In addition to the above functional requirements, the following desirable characteristics were established.

1. Free the captain and his officers from time-consuming tasks.
2. Provide the ship's officers with distributed and tailored situation awareness, and convenient access to information.
3. Enhance the knowledge levels of inexperienced officers through enhanced situation awareness.
4. Support of on-board training of selected personnel via an archival system.

5. System Architecture

The overview of the architecture of the SILS system provided in this section is based on the formal specifications provided by a system design model that was constructed at the beginning of the SILS development effort. This development artifact utilizes the Unified Modeling Language (UML) to provide a more concise, compact, and maintainable specification than can be achieved with ad hoc diagrams and English text.

One of the more powerful aspects of UML is that the graphical notations it employs are built on an underlying formal grammar. This supports the development of design views and diagrams to show specific aspects of the system, while contributing to an underlying self-consistent model that is much too complex to depict in any one picture. The approach allows software architecture specifications to exist at the multitude of different scopes and levels typically required. The main body of this document is organized around the top-level diagrams of the primary views that the design model uses to specify the various design aspects of the system.

The system design model leverages extensively on the large body of published software engineering patterns. Many of the concepts conveyed by a particular design diagram are indicated by a pattern reference. Readers unfamiliar with the pattern may wish to consult the associated reference material (Buschmann et. al. 1996; Buschmann et. al. 2000; Fowler 1997) for a full understanding of the design. Software engineering patterns provide a useful means for capturing proven solutions to software engineering problems in a generic system-independent manner.

A well-documented pattern provides a unique descriptive name, describes a software engineering problem in regard to a specific context, and presents a well-proven generic scheme for its solution. Patterns in software engineering are mostly associated with lower-level software design through the classic reference design patterns of the Gang of Four who introduced the software development community to the pattern concept (Gamma et al. 1994). However, design patterns are equally applicable across the spectrum of scale and abstraction within the discipline.

5.1 Logical View

The logical view provides the structure to manage and comprehend the source level artifacts used to develop the system. The topmost level of the logical design specifically targets the issues of code reuse across families of similar systems by providing a structure that allows the core technology to be identified, captured, and evolved, independently of any particular project or software system. For this purpose, the source level design artifacts are partitioned into four interdependent layers, as specified by the Relaxed Layers pattern depicted in Figure 6. The unique aspects of a specific system design are grouped into the ICDM System layer. The artifacts contained in this layer leverage the subsystems and service libraries provided by the underlying ICDM Framework layer and must be considered in relation to the framework, to be fully understood.

The general design artifacts applicable to a wide-range of decision-support systems have been abstracted from existing systems over the years into the ICDM Framework, ICDM toolkit, and ICDM guidelines. The toolkit provides the development and build environment including the code generators, which transform ICDM System layer artifacts into the subsystem targets specified in the ICDM Framework layer. The ICDM guidelines provide informal descriptions of the ideal characteristics of a sound decision-support system and capture the overarching vision. They serve as a backdrop against which system design decisions may be evaluated.

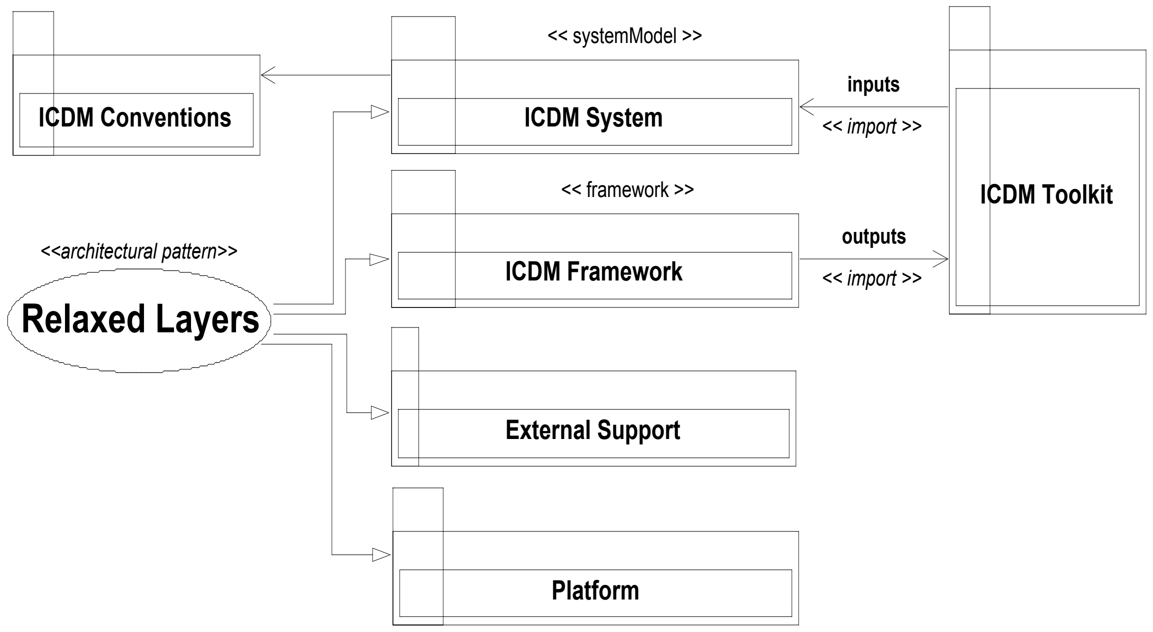


Figure 6: System layers

The Relaxed Layers pattern (Buschmann 1996) indicates that the call-level dependencies between the ICDM System layer and the ICDM Framework layer should be only in the system to framework direction. The framework contains many high level subsystems that are indirectly dependent on the system layer to provide domain specific context. The subsystems work with these elements at the meta-level and therefore do not violate the call-level dependencies. These elements are often specified in a high-level form, such as UML, that is abstracted from any particular implementation.

The External Support and Platform layers group the externally developed elements of the system. It is important to identify external design elements at the architectural level. The external design elements are relatively static and may limit the flexibility of the system to evolve over time. They may also have associated runtime issues such as licensing fees and runtime validation problems.

The Platform layer is distinguishable from the External Support layer in that it groups the relevant external elements provided by the computing infrastructure of the client enterprise. These elements need to be distinguished from other external elements because their configurations and upgrades are outside the control of the system developers.

5.2 System Tiers

The system specific design is further structured into three distinct tiers as described by the Information System pattern (Fowler 1997) as depicted in Figure 7. The Domain Tier provides a direct executable model of the system domain that is independent of any particular application or source data model. It represents the active core of the system and provides the central focus for the development effort. The Application Tier provides local applications to support the domain interactions of specific user groups. The Data Tier provides for the persistent storage of the data that underlies the information represented by the domain model.

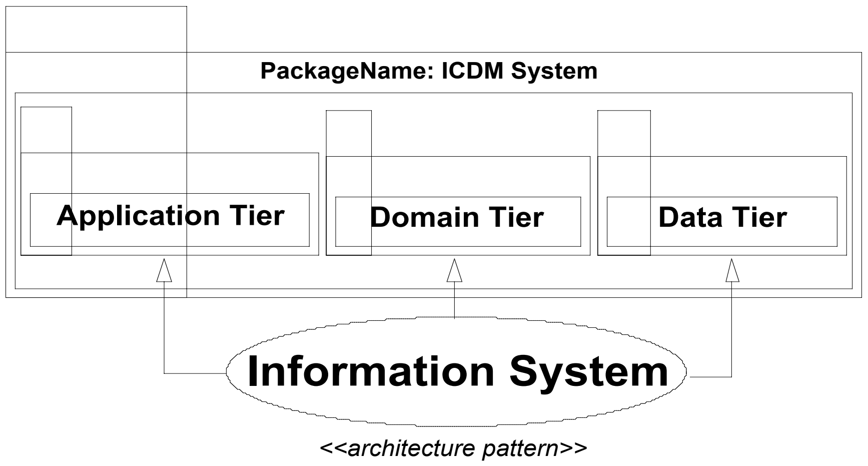


Figure 7: System tiers

The Information System pattern was selected to address the fundamental decision-support system requirements for providing concurrent collaborative support to multiple users, a high-level objectified model of the domain that constitutes the necessary context in support of agent-based reasoning, and interoperability with existing data-centric systems within the system domain.

5.2.1 Domain Tier

The core of the SILS system architecture is the Domain Tier. The Information System pattern assigns the responsibility for saving and restoring the associated domain model to and from the Data Tier to the Domain Tier. This responsibility is typically addressed by providing the individual domain model objects with the capability to save and restore themselves, which is reasonable for simple stand-alone systems that have the complete freedom to specify the storage format of their persisted elements. Unfortunately, real-world systems are rarely this simple, especially those geared toward decision assistance. Decision-support systems must interact with existing systems, taking feeds as necessary, and dealing with the fact that many systems with varying representations (i.e., relational, hierarchical, flat files, etc.) may have to be accessed to maintain the integrated picture required to provide the desired level of assistance.

The code required to implement this type of external system interaction is substantial. It will pollute the purity of the domain model by masking its initial intent and limiting its utility in other contexts. This dilemma is addressed by partitioning the responsibilities of the Domain Tier between a Data Interface Tier and a Representation Tier as shown in Figure 8. The Domain Tier representation provides the executable model of the domain, while the Domain Tier data interface assumes the responsibility for moving information between the Representation Tier and Data Tier. This level of indirection also provides the system with the additional flexibility required to more easily adapt to Data Tier changes over time, or to adapt due to local variations at different deployment locations. The 'Data Broker' pattern (Fowler 1997) describes the internal structure and dynamic behavior of the Data Interface.

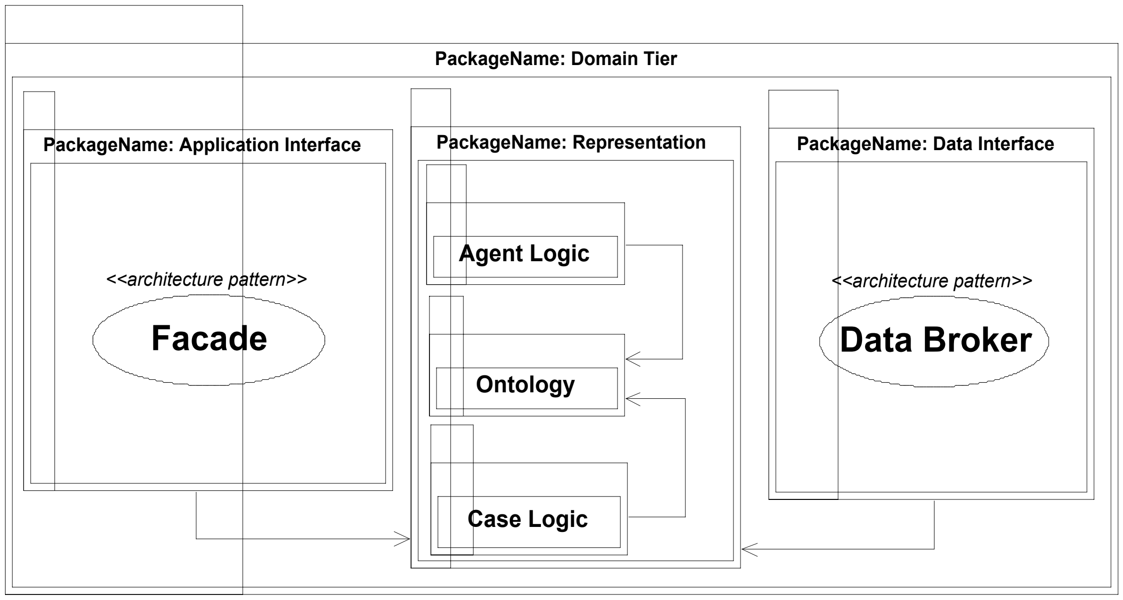


Figure 8: Domain Tier

The Domain Tier must also provide an interface for the various applications within the domain. The pure representational model is not ideally suited for this purpose due to the complex interrelationships and high-level domain-specific specifications. It also does not address the transactional nature of the interactions between the user applications and the domain. In order to address these deficiencies an additional façade-based (Fowler 1997) sub-tier tailored to the needs of the system's applications is inserted into the Domain Tier. This application interface is responsible for all accesses to the domain representation and does no processing other than that specifically required for the user interface presentations of the Application Tier. The addition of this layer also favored the development process since it allowed the design of the user interface and the domain model to occur in parallel more or less independently of each other.

Within the context of this architecture, an information system utilizes a class-based object model to represent the domain. Classes represent the types of entities (i.e., objects) within the domain, and may be generalized or specialized to relate similar types. They serve as templates for the creation of objects that specify individual characteristics in terms of attributes, behavior in terms of operations, and context in terms of associations. A decision-support system can be thought of as a value added extension to a traditional information system. Within the context of such a system, the internal information model (i.e., object model) is referred to as the ontology. The latter provides the domain vocabulary upon which the agent logic is specified in the form of expert system rules. This logic is used to express the business rules of the domain, maintain high-level derived information, and generate alerts and statements of implication.

From the perspective of the developer, the rule-based representation of the agent logic is very flexible in dealing with dynamic changes. However, since the rules are compile-time entities they do not provide this same flexibility to the users of the system during execution. This is where the case logic is particularly useful, because it uses a fixed compile-time model composed of problems, questions, actions, and their interrelationships. The domain specific nature of the case logic is therefore represented in the form of object instances rather than model classes. This allows the case logic to be dynamically extended or modified during execution, either directly or indirectly (e.g., through embedded system learning processes) by system users. The case logic is also expressed in a form that serves as an appropriate basis for an English language form of interactive dialog between the system and the system users to formulate appropriate courses of action. The current version of the SILS proof-of-concept system does not incorporate case logic. It is included in the design because it is recognized at this time as being necessary for the long-term success of SILS.

5.3 Component View

The component view exists at a level of abstraction above the logical view. It defines versioned chunks of software compiled from the source level artifacts represented in the logical view. Components are runtime entities that are hosted on the client hardware of the deployed system. They are loaded by the system as needed to execute specific functionality. The internal details of the individual components within the component model are beyond the intended scope of this document, however, brief descriptions of the high-level components provided directly by ICDM are listed Table 1.

Table 1: Component descriptions

Name	**Description**
CAD Viewer	A full-featured application-oriented subsystem designed to view and manipulate CAD drawings in three-dimensions.
GIS Viewer	A full-featured application-oriented subsystem designed to view and manipulate maps and geographic information.
Speech Interface	An application-oriented plug-in component that allows users to control application functionality by voice and provides a means for applications to respond to users with sound.
Reporting Interface	A full-featured application-oriented subsystem for displaying, printing, and manipulating information displayed in tables or forms.
System Interface	An application-oriented subsystem for performing basic system functions such as account management and login.
Embeddable Web Browser	An application-oriented subsystem that provides a lightweight constrainable Web browser within an application process.

OML	A programmatic interface to the object-serving communication facility employed by ICDM-based systems.

5.4 Process View

The process view exists at a level of abstraction above the component view. At the top-level, it defines the directly executable processes that collaborate in a distributed fashion to implement the system requirements. The top-level processes of the SILS proof of concept system fall into four functional groupings: client applications; domain servers; domain clients; and, data servers. Processes are typically more system specific than the components existing in the underlying abstraction layer. They usually embed and configure selected higher-level components to provide the system specific capabilities for which they are responsible. The core processes are depicted in Figure 9 and are discussed in more detail in Section 7.

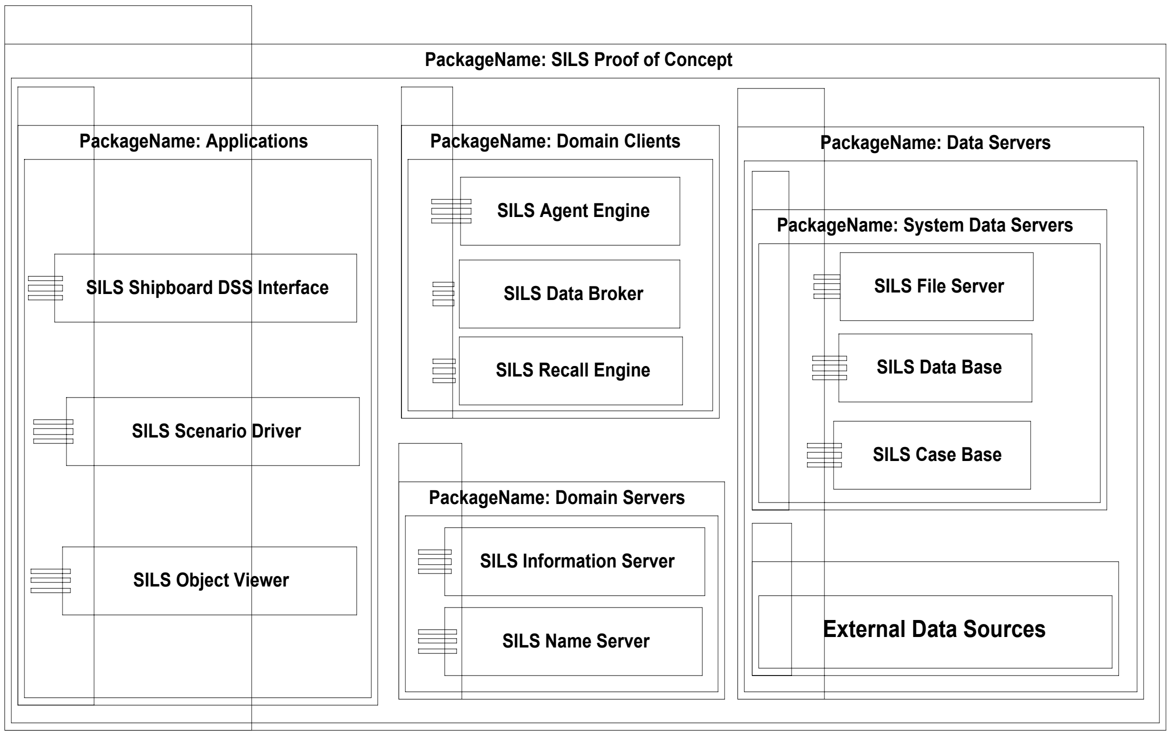

PackageName: SILS Proof of Concept
PackageName: Applications
SILS Shipboard DSS Interface
SILS Scenario Driver
SILS Object Viewer
PackageName: Domain Clients
SILS Agent Engine
SILS Data Broker
SILS Recall Engine
PackageName: Domain Servers
SILS Information Server
SILS Name Server
PackageName: Data Servers
PackageName: System Data Servers
SILS File Server
SILS Data Base
SILS Case Base
External Data Sources

Figure 9: SILS process model

5.5 Interaction View

The interaction view describes how the pieces of the system collaborate to perform the capabilities that the system implements. At the top-level, it describes the overall pattern of interaction among the primary system processes. This interaction is best described by the Blackboard Pattern. This classic architecture pattern has been employed successfully by the Artificial Intelligence (AI) community since the early 1970s as an approach to problems for which no deterministic solution strategies are known. The name blackboard was chosen because the approach parallels the situation in which human experts sit in front of a classroom blackboard and work together to solve a problem (Buschmann 1996).

The blackboard architecture employs a collection of independent programs (i.e., knowledge sources) that work cooperatively on a common data structure (i.e., blackboard). Each program is specialized for solving a particular part of the overall task, and all programs work together on the solution. The specialized programs are completely independent of each other. They do not call each other and there is no predetermined sequence for their activation. The direction taken by the system is primarily determined by the current state of the solution. This type of data directed control facilitates experimentation with different types of algorithms and allows experimentally derived heuristics to control processing.

Within the context of the SILS design, the Information Server plays the role of the blackboard while the domain clients play the role of knowledge sources as depicted in Figure 10. The manager within the agent engine instance provides control over the application of knowledge to the solution being developed by the associated agent federation. The human users, through their client applications, provide an additional source of knowledge and control.

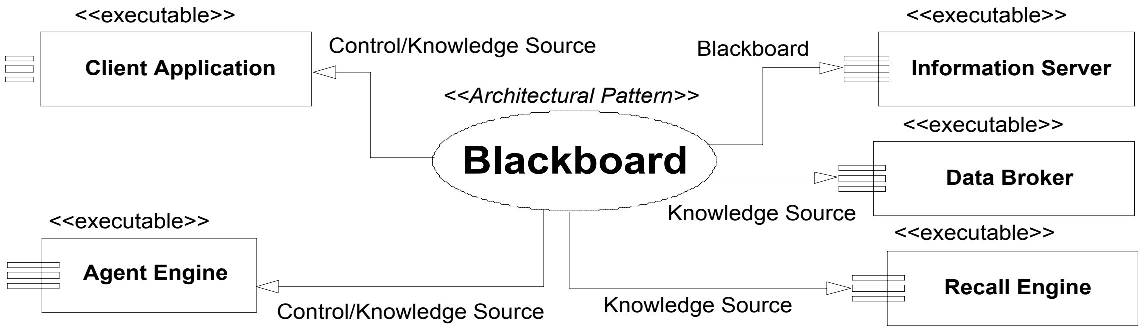


Figure 10: SILS process interaction

The incorporation of one or more human users distinguishes this architecture from traditional blackboard implementations that were designed to solve problems for users, rather than collaboratively with them. The partnership between human users and the software agents (i.e., knowledge sources) is employed to eliminate the control problems often associated with blackboard architectures. Humans can keep the developing solution on track and provide the stimulus to resolve conflicts when stalled. The same data-driven features that provide for the interaction of diverse independent software agents may also be employed to simultaneously link spatially distributed human users into a collaborative environment; - thereby realizing an information age version of the conceptual stimulus for which the blackboard pattern is named.

5.6 Deployment View

The deployment view is an abstraction layer built on top of the abstraction provided by the process view. At the top level, it defines the different hardware configurations that will be supported by the system and the ways in which the system processes will be distributed on hardware. At lower levels of the deployment model the versioned identity and installed locations of the underlying components that support the resident processes on each computer within a given configuration are defined. In the context of the proof-of-concept system, four primary configurations need to be defined for the development, integration, and demonstration of the proof-of-concept system.

5.6.1 Interface Engine Development and Test Configuration

The Interface Engine Development and Test Configuration defines the installation of components necessary to support the SILS system processes that are required for the development and testing of the SILS Interface Engine. These include: the SILS Object Viewer; the SILS Information Server with the SILS Interface Domain; the SILS Name Server; the SILS Interface Engine; and, any simulated or actual external system data servers and their associated scripted drivers.

5.6.2 Decision-Support Development and Test Configuration

The Decision Support Development and Test Configuration defines the installation of components necessary to support the SILS system processes that are required for the development and testing of the agent-based decision-support aspects of the system, independently of the existing system interfaces. These include: the SILS Shipboard DSS Interface; the SILS Scenario Driver; the SILS Object Viewer; the SILS Information Server with all domains; the SILS Name Server; the SILS Agent Engine; and, the SILS Data Servers.

5.6.3 System Integration Configuration

The System Integration Configuration targets the joint verification and error correction of the two separately developed subsystems of the proof-of-concept system. This configuration defines the installation of components necessary to support the integrated system partitioned across a minimum of two sets of networked machines. The first set includes the following processes: the SILS Interface Engine; and, any simulated or actual external system data servers and their associated scripted drivers. The second set includes all of the processes within the Decision Support Development and Test Configuration.

5.6.4 System Demonstration Configuration

The System Demonstration Configuration targets the final demonstration of the proof-of-concept system. This configuration defines the installation of all components necessary to support the planned demonstration scenarios for the integrated system. The System Demonstration Configuration differs from the System Integration Configuration in that the components are distributed across multiple machines in a manner more suitable for demonstration and more similar to the installation configuration that might be used aboard ship. For this configuration, one or more client machines are used with components necessary to execute the client applications, one or more domain machines are used with components necessary to execute the domain server and domain client processes, and one or more data machines are used with components necessary to execute the data server processes.

6. System Agents

The agents employed by ICDM are software processes, components, or modules that have the ability to perceive the external environment and autonomously act on it in collaboration with other agents. Agents act in a manner conducive to achieving the individual and collective goals of the system and its users. The external environment in which an agent is situated is both bounded and defined by the ontology the agent employs to interact with it. The ontology provides a vocabulary to describe the external environment that is constrained in accordance with the underlying principles operating in the environment, such as the physical laws that constrain our own real world environment. In this respect, the ontology allows agents to express their specific interests in an environment, communicate their thoughts about it, specify actions to be executed in it, and record knowledge about it.

The level of perceived agent intelligence is coupled to and bounded by the specialized depth (i.e., the number of concepts) and richness (i.e., the number of associations) of the ontology. Effective agent collaboration requires unambiguous communication, which is related to the level of orthogonality provided by the ontology. The ICDM framework employs the standardized Unified Modeling Language (UML) to formally specify the types of physical and conceptual entities within the target domain, and the types of relationships between them. The ICDM Toolkit code generation utilities operate on this specification to implement a distributed ontological framework for the specific architectural configuration(s) and platform(s) targeted for deployment. In this manner, the generated ontological framework provides an enabling foundation upon which the rest of the application is implemented.

ICDM-based software employs two major categories of agents: subscription-based agents; and, rule-based agents. Subscription-based agents are individual processes or components that operate at the architectural level of the system. Rule-based agents correspond to modules within an expert system shell environment each of which contains a rule-set targeted to encode a particular area of expertise within the application domain. Both categories of agents are proactive in that they automatically act in response to changes in the virtual representation of the external environment and therefore do not have to be explicitly told to act, as is typically the case in traditional procedural paradigms.

Subscription-based agents use the standardized Common Object Request Broker Architecture (CORBA) services that ICDM provides along with a proprietary subscription service to register their individual interests within the domain. The ICDM Subscription Service alerts individual subscribers to changes in the collection of distributed objects used to represent the domain, which satisfy their registered interests by pushing the changes to the corresponding subscriber. This capability allows the individual subscribers (i.e., agents) to collaboratively interact in an efficient and scalable fashion, without any prior knowledge of each other.

Subscription-based agents may in turn contain rule-based agents that operate in modules within the parent process or component. Rule-based agents utilize specialized declarative languages to precisely specify a state in the external environment and the action that should be performed when the specified state is observed. They work at a much lower level of granularity than that employed by subscription-based agents. This level of granularity requires specialized data structures and algorithms to efficiently match on the states of the external environment. To date the required level of efficiency is only found in expert system shells that are based on the RETE algorithm (Forgy 1982).

The traditional problem with expert system shells is that they are stand-alone development environments that do not interoperate with the general-purpose environments required for graphical user interface (GUI) development or for relational database interaction. Interoperability with expert system shell environments is a core technological feature of ICDM-based applications. This interoperability is provided through proprietary adaptations to existing expert system shell environments that enable them to seamlessly operate as plug-in clients to the distributed ICDM Object Serving Communication Facility that houses the virtual representation of the external environment. Additional extensions have also been created to better manage the distribution of processor time between the resident agent rule sets.

All processes within the system that are above the service level (i.e., distributed CORBA services that support the SILS MRAT Information Server) are implemented as subscription-based agents. This allows them to proactively react to changes in the state of shared objects and thereby ensure a more efficient use of network bandwidth, since only the changes in the state of interest of the individual processes need to be communicated across the network. The current subscription-based agents include: all of the core application tools and a majority of the support tools; the Agent Engine; the Recall Engine; and, the Interface Engine.

6.1 Rule-Based Agents

As is typical with systems of this nature, the development of the system architecture and the domain representation comprise the bulk of development at the beginning of the project. The specialized expertise embodied within a particular agent is most efficiently implemented on top of a stable core ontology and information system. To date, the primary focus within the agent area has been to support a range of test scenarios with the objective of improving the agent infrastructure and reasoning facilities. As the project transitions from proof-of-concept to prototype development, the focus will shift first to the implementation of agent expertise that operates on the high-level generalized portions of the ontology, and subsequently to the implementation of specialized expertise that operates on those portions of the ontology that are specific to SILS MRAT. The current and planned capabilities of the SILS MRAT rule-based agents are depicted in Figure 11 and described in the following subsections.

6.1.1 Mission Capability Agent

The Mission Capability Agent is responsible for continuously maintaining an assessment of the mission readiness of the host ship. A hierarchical assessment tree is maintained for each individual mission type associated with the ship, for each of the individual departments onboard the ship. This agent is the key to being able to provide dynamically responsive graphical views of readiness to different users with different biases. Since this agent is continuously working in the background to calculate the readiness picture across all bias combinations the information is already available when requested by users (i.e., it does not have to be generated at the time the request is registered.

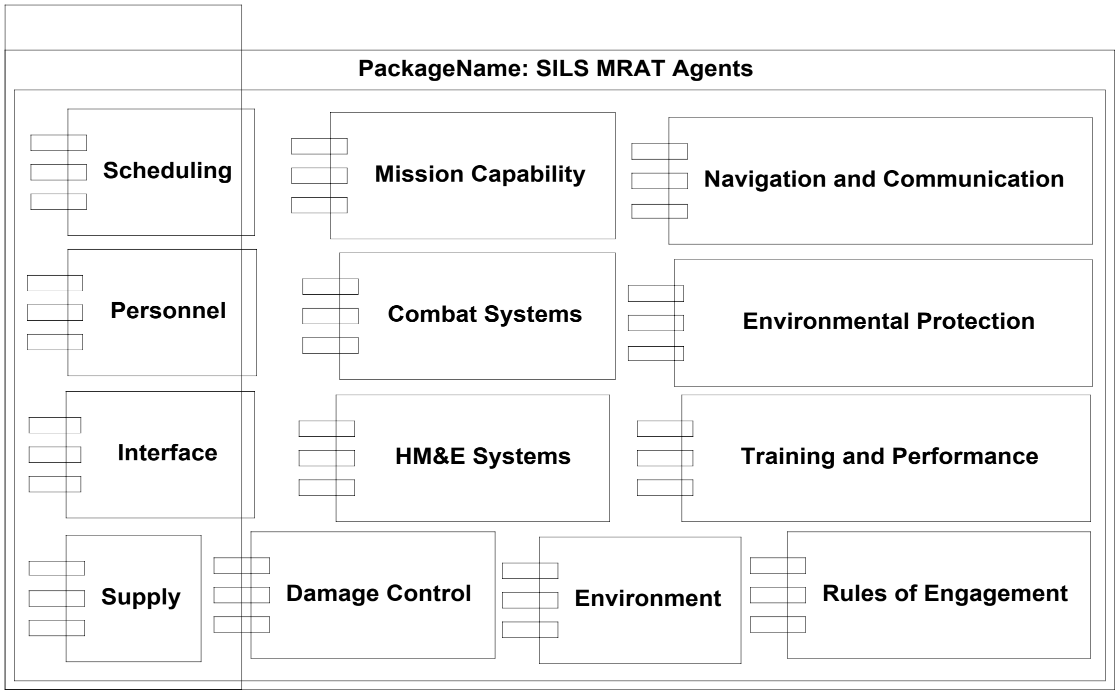


Figure 11: SILS MRAT agents

6.1.2 Scheduling Agent

The Scheduling Agent is responsible for identifying resource issues with scheduled activities and resource conflicts between scheduled activities. This agent utilizes a few simple rules that operate on the ontology at a generic level and provide for functionality that predicts possible conflicts by comparing current schedules with proposed activities. For example, the Scheduling Agent would generate an alert if a scheduled event with associated tasks and resource requirements is planned to occur when there are inadequate resources for the event. The value of this agent is associated with large complex schedules created by many different personnel, with competing interests and limited resources, rather than simple schedules created by a single person.

6.1.3 Personnel Agent

The Personnel Agent is responsible for monitoring the manning status of the crew, identifying levels of shortfalls, and providing assistance in locating personnel for tasks requiring individuals not specified in the ships manning document. It also assists in identifying and locating appropriate human resources to perform or support specific activities. In addition, the Personnel Agent notifies appropriate departments of the arrival and departure status of personnel.

6.1.4 Supply Agent

The Supply Agent is responsible for monitoring the supply status of the ship, alerting appropriate departments of their order status, aggregating order requests, and identifying and locating appropriate supply items for a given repair task.

6.1.5 Interface Agent

The Interface Agent is responsible for monitoring the status of the external system interfaces required to provide up-to-date accurate information to the users and for alerting users to potential problems or changes of state. It also manages the swap space employed for collaboration with the SILS Interface Engine. It is the purpose of the swap space to isolate the core object model that supports agent reasoning from the demands of the existing system interfaces that often impede the natural evolutionary growth of the model. The types of objects contained within the swap space are defined by the interface domain model that is composed of simple non-overlapping model fragments, which correspond to the conceptual blocks of information sourced within external systems. Swap space management consists of mapping the information fragments to and from the core problem domain, which is a task ideally suited to a declarative rule-based implementation.

6.1.6 The Combat Systems Group of Agents

This group of agents includes the HM&E Agent, the Damage Control Agent, the Navigation Agent, and the Communication Agent. They are responsible for monitoring the status of the systems within the corresponding system category in order to alert users to potential problems or changes of state. In particular, these Agents are intended to estimate the resultant degradation of affected systems and provide these results for use by other system level agents such as the Mission Capability Agent. Agents in this group are critically dependent on links to automatic monitoring systems and access rules that enable the high-level analysis of the implications of data received. At the time of writing this report, the data feeds and subject matter expertise necessary to enable these agents have yet to be identified.

6.1.7 Environmental Protection Agent

The initial proof-of-concept implementation of this agent was little more than a concept of an agent that advises on environmental issues. This was primarily due to the difficulty in identifying and obtaining applicable automated data feeds without which the agent could be little more than a browser for text-based instructions. The US Navy Pollution Discharge Restrictions (OPNAVINST 5090.1) has recently been brought to the attention of the development team. This document deals with regulations regarding the dumping of materials at sea and has been identified as a useful area for agent-based support within the SILS MRAT application.

Implementation of a subset of the instructions contained in the OPNAVINST 5090.1 document, to determine whether a given category of material can be dumped at sea given the current distance from shore, is currently under investigation. These explorations focus on the possible objectification of information within Digital Nautical Chart (DNC) maps to automatically determine the distance from shore given the current geographic coordinates of the ship. DNC maps embed data, distinguishing land from sea, which can be used to precisely determine the current distance to land. This is accomplished by an algorithm that systematically searches for the nearest map data element that the objectification scheme indicates as land and converting this element to the appropriate distance units.

6.1.8 Environment Agent

The Environment Agent will be responsible for monitoring the external physical environment for conditions that may affect the ship in regard to the types of missions it is designed to perform. While this agent has not yet been implemented, it is expected that its rules will require access to weather and tide information and other external environmental influences. It will then be necessary to quantify impact of these external influences on mission, mission tasks, ship equipment, and crew.

6.1.9 Rules of Engagement (ROE) Agent

The Rules of Engagement Agent (ROE) is responsible for monitoring the operations of the ship and the associated shipboard activities for compliance with the published rules of engagement. In addition to Navy-wide standards, the agent should provide a customization capability that allows it to support applicable portions of the commanding Officer’s standing orders.

6.1.10 Training and Performance Agent

The Training and Performance Agent is responsible for identifying training deficiencies in regard to published standards and measured performance. Knowledge acquisition interviews for the SILS MRAT project indicate a perceived need by training officers aboard ship for support in this area.

7. Component Functionality

The focus of this section is to provide an overview of the functionality provided by each of the top level SILS MRAT components. These components are divided into two categories within the SILS MRAT system: Graphical User Interface (GUI) components that provide functionality directly to human users; and, Application Interface components that provide functionality to other components through an application interface (API).

The top-level SILS MRAT components consist of directly executable processes that collaborate in a distributed fashion to implement the system requirements. These processes are typically more system specific than the components existing in the underlying abstraction layer. They embed, configure, and supplement selected ICDM components to provide the system-specific capabilities for which they are responsible.

The top-level components fall into three functional groupings: Application; Domain; and, Data. The Domain components provide an executable model of the system domain that is independent of any particular application or source data model. This model also includes active objects known as agents and the domain business rules that control their activities. It represents the active core of the system and provides the central focus of the development effort. The Application components provide local tools to support the domain interactions of specific user communities, and the Data components provide for the persistent storage of the data that underlie the information represented by the domain model. These component categories are depicted in Figure 12.

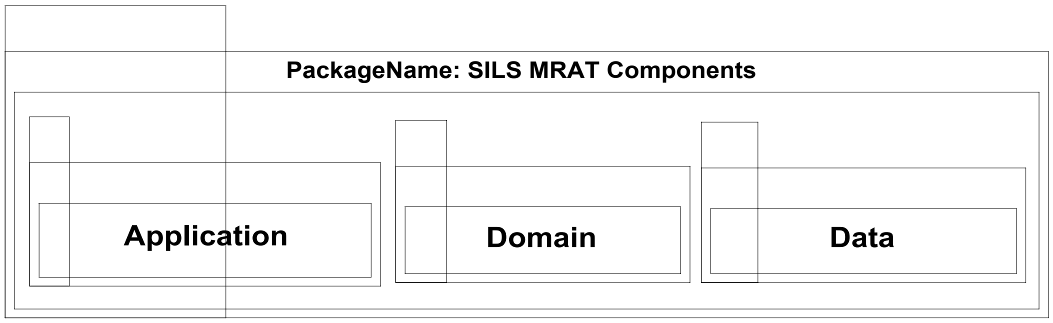


Figure 12: SILS MRAT component categories

7.1 SILS Application Components

The Application Components provide user-specific interfaces to the shared domain representation, software agents, and services provided by the Domain Components. The Application Components are grouped into two categories: Application Tools; and, Support Tools. Application Tools are the client-side applications through which the end-users, such as the ship's captain and his department heads, interact with the SILS system. Support Tools, on the other hand, are geared towards system developers, testers, or advanced users. They provide functionality useful for testing, demonstrating, and diagnosing the system. The application tool categories are depicted in Figure 13.

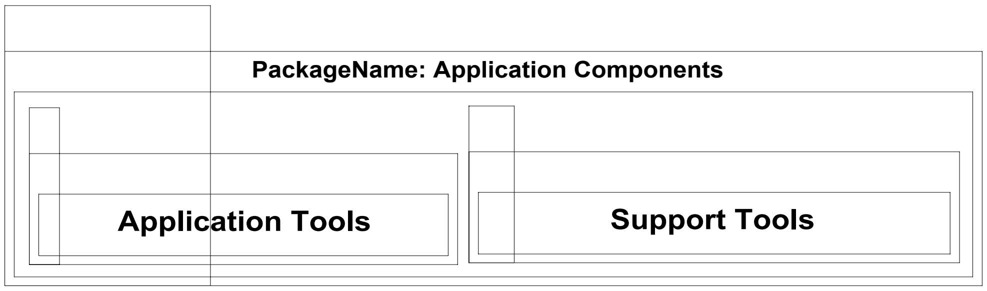


Figure 13: SILS MRAT Application Component Categories

7.1.1 Application Tools

The SILS MRAT client application takes the form of a suite of application tools that allow users to collaboratively interact with each other and the domain-specific software agents via the shared ontology-based object model. The tools within the collaborative suite are completely decoupled from each other, but may work in close collaboration using the same mechanisms that allow remotely distributed users to collaborate. The application tools may be used independently or with the suite controller that allows them to work in close conjunction with one another, and allows users to seamlessly switch from on application tool to another. The Application Tools are depicted in Figure 14.

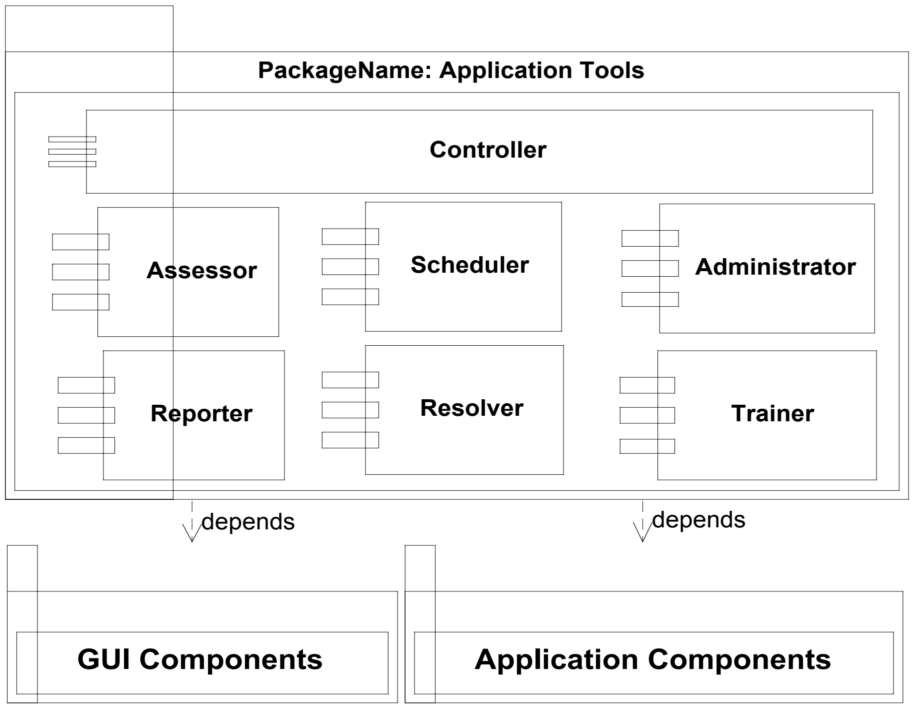


Figure 14: SILS MRAT Application Tools

The Application Tools share an application framework and embed many of the same ICDM components. These components either provide common GUI elements, underlying code-level APIs, or frameworks for structural similarity and compatibility.

The Common GUI Components are used by all of the Application Tools. The component capabilities are in turn exposed to the Application Tool users through the provided GUIs. This approach leverages the functional overlap between the individual Application Tools and helps provide a common look-and-feel across the suite of tools. The common GUI components are depicted in Figure 15.

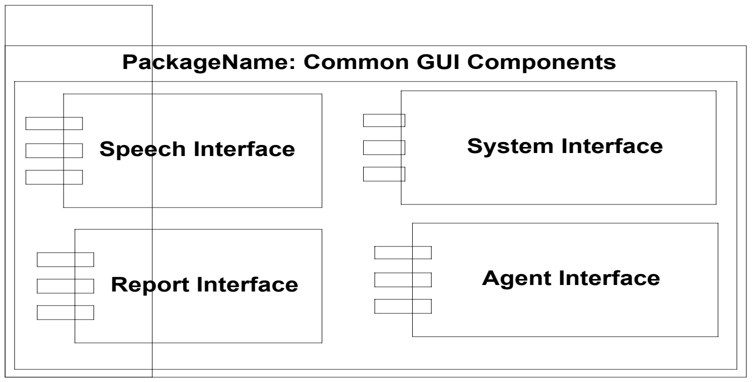


Figure 15: The Common GUI Components

The Speech Interface works in conjunction with the Application Framework to allow the GUI elements of an Application Tool to be controlled by voice and to allow for spoken audio prompts.

Current Capabilities

1. Displays the element currently in focus.
2. Displays all possible commands for the GUI element currently in focus.
3. Allows GUI element focus to be controlled by voice.
4. Allows commands to be selected by voice
5. Supports input of free text by voice.
6. Provides an internal API that accepts text strings to be spoken to the user.

Proposed Future Capabilities

1. Settings to tailor voices for specific types of spoken output.
2. Settings to turn spoken output on and off.

The System Interface provides basic password encrypted login security and system reporting features to the Application Tools.

Current Capabilities

1. Prompts for and processes user login and password.
2. Provides system information such as version and build numbers.

Proposed Future Capabilities

1. Support for changing passwords.
2. Tool and component level information, versioning, and build number.
3. End-user level system status reporting.

The Report Interface provides common generic functionality for all graphical displays of forms, tables, and trees within the system.

Current Capabilities

1. Preference selection of displayed attributes, by individual report, and by individual user.
2. Preference selection of the order of displayed attributes, by individual report, and by individual user.
3. Preference selection of table column widths by individual report and by individual user.
4. Preference selection of attribute value formats. Where applicable, by individual report and by individual user.
5. Preference selection of attribute value justification (right, center, left) by individual report and by individual user.
6. Preference selection of multiple filter constraints on a per displayed attribute basis per individual user.
7. Remembering and utilizing last used preferences for each report per individual user.
8. Saving report preferences by logical name on a per user basis.
9. Restoring report preferences by logical name on a per user basis.
10. Restoring pre-configured system preferences by individual report on a per user community basis.
11. Support for preset system configurations.
12. HTML generation.
13. Printing support for all reports with print preview.
14. Support for aggregate reports that allow preset combinations of form, table, and/or tree reports to be configured, displayed, and printed.

Proposed Future Capabilities

1. Unit support with user selection, runtime conversion, and preference support.
2. Enhanced attribute value (i.e., field) representation invariant (i.e., form) constraints.
3. Intelligent handling of view, add, edit, and delete permissions at the generic report level.

The Agent Interface is the primary means by which users directly interface with the software agents associated with a particular application tool.

Current Capabilities

1. Generic common display for all direct agent output.

2. Graphical display of violation, activity, and acknowledgement state for each agent.
3. Detailed view dialog that provides running chronological list of alerts for all agents associated with a given application tool.
4. Individual alert dialogs for each agent that in turn provides detailed information about a specific alert.
5. Roll-up of alerts by primary alert target (e.g., typically a mechanical or human asset) for each agent.

Proposed Future Capabilities

1. Settings to toggle individual agents on or off.
2. Settings to toggle individual logical agent rules on or off.
3. Settings to adjust the severity of individual alerts.
4. Individual user acknowledgement of specific alerts.

The Common Application Components are used by all the Application Tools. As for the Common GUI Components, the Application Component capabilities are exposed to the Application Tool implementations through the provided APIs (Figure 16).

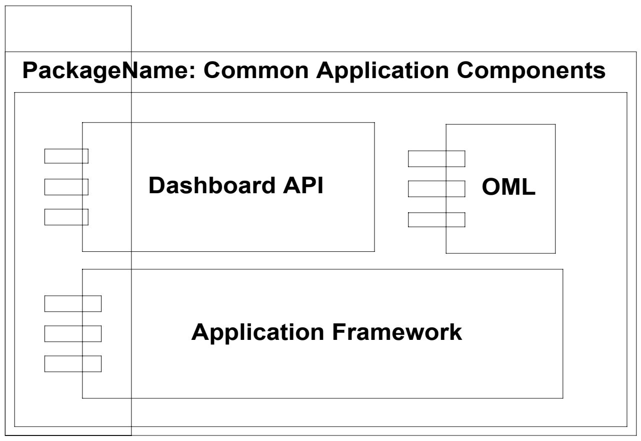


Figure 16: The Common Application Components

The Dashboard API provides the Application Tools with a standard interface, primarily intended for use by the Dashboard Controller to query the installed object code-base of an individual Application Tools for the properties and capabilities it supports.

Current Capabilities

1. Ability to query for available indicators.
2. Ability to query for available instruments.
3. Ability to bring up a particular instrument independent of the parent Application Tool

The Application Framework constrains application elements to a specific architectural paradigm and consolidates common functionality useful for most Application Tool implementations.

Current Capabilities

1. Maintains a menu of open dialogs or windows for easy access.
2. Remembers the size and location (i.e., last used) for each dialog on a per user basis, and brings it up the same way next time it is accessed.
3. Cleanly separates the application logic from the corresponding GUI, thereby providing for architectural flexibility, sharing, and reuse of application logic components.
4. Supports standardized GUI spacing constants for a consistent look and feel across GUI elements for all Application Tools.
5. Provides intelligent subscription management.

Proposed Future Capabilities

1. A consistent generic framework for dealing with exceptional conditions and user dialogs.

The OML (Object Management Layer) provides code-level APIs for accessing and manipulating the collaboratively shared distributed objects that represent the system domain.

Current Capabilities

1. Object level add, edit, and delete capabilities.
2. Object level constrained queries.
3. Constraint-based subscriptions.
4. Local object caching and cache management.
5. Support for object servers of different type through an abstract server API that allows applications to plug into new types of servers.
6. Simultaneous support for multiple object servers.

The Controller Application Tool implements a digital dashboard paradigm that allows the other tools in the suite to be controlled and monitored from a single unobtrusive interface.

Current Capabilities

1. Display of icons for available Application Tools.
2. Launching of a specific Application Tool from a displayed icon.
3. Display of selected Application Tool status indicators.
4. Display of selected Application Tool instruments.
5. Launching of the Application Tool functionality corresponding to a particular instrument.

6. Provision of control by voice.

Proposed Future Capabilities

1. Creation of custom named tabs with which to organize controller displays.
2. Ability to query installed Application Tools for available indicators and instruments.
3. Allow selected indicators and instruments to be added to a named tab.
4. Allow the layout of an instrument or indicator to be customized based on user preferences.
5. Automatically bring up the controller as configured the last time it was used.
6. Allow for name controller configurations to be saved and loaded on demand.
7. Support context sensitive help facilities.

The Assessor Application Tool provides an explanation of the current readiness assessment of the ship.

Current Capabilities

1. Displays mission-biased and department-biased readiness status.
2. Traces combat readiness assessments to their constituent components.
3. Infers readiness issues from supporting observations.
4. Display the causal inference chain for each readiness issue
5. Displays details of the individual observations within an inference chain.
6. Displays the impact of a readiness issue on readiness assessment.
7. Displays information associated with the context and resolution of a particular readiness issue.
8. Provides a vectored graphical view of the ship that supports pan, zoom, tilt, rotate, and other similar functions.
9. Provides direct interaction with the graphical entities in the objectified ship view.
10. Allows reports to be brought up on ship areas directly from the graphical view.
11. Allows objectified ship areas to be colored, based on the violation state.
12. Provides a vectored graphical view of the map for the area of operations (supports pan and zoom).

Proposed Future Capabilities

1. Provides for typed hierarchical commentary.
2. Allows incidents to be defined that relate issues and actions.

3. Allows overlays on ship and map views that can be collaboratively marked-up.

The Reporter Application Tool is the primary means by which users may directly enter information into the system. It is also intended to allow users to create custom reports for printing, publishing, or posting on a Web server.

Current Capabilities

1. User customizable content, layout, and format for formalized reports.
2. HTML translation capability for Web-based distribution.
3. Automated objectification of user-supplied content for the purpose of agent reasoning.
4. Published report repository with browsing support.

Proposed Future Capabilities

1. Provides typed hierarchical commentary with which to annotate published reports.
2. Supports the automated incorporation of designated external system information.
3. Provides ability to build and publish custom reports and to subscribe to features of interest within the periodic publications of others.
4. Supports the automatic notification of changes within periodic publications based on registered interests.

The Scheduler Application Tool provides a comprehensive environment for developing, maintaining, and presenting the operational schedule of a ship.

Current Capabilities

1. Calendar and Gantt Chart presentations.
2. Generates plans for day, week, and month time periods, at arbitrary levels in the hierarchy.
3. Supports hierarchical displays, based on organizational structure.
4. Provides intelligent scheduling assistance based on accrued knowledge of standard operating procedures and resource dependencies.
5. Supports the automatic identification of scheduling conflicts.

Proposed Future Capabilities

1. Generates task lists for specific people or organizations.
2. Supports task rollup and inter-task dependencies.
3. Provides the ability to display operational schedules in terms of primary mission activities and supporting preparatory activities.

4. Allows the design of proposed missions or activities and incorporates them into the existing operational schedule.
5. Provides for a typed hierarchical commentary.
6. Supports the automatic notification of scheduled events based on registered interests.
7. Provides automated assistance to coordinate conflict resolutions.

The Resolver Application Tool provides assistance in the development of options and impacts to address the readiness issues indicated by the assessment component.

Current Capabilities

1. Provides fixed model-based options to resolve a particular issue.
2. Provides fixed model-based opportunity costs for each particular option.

Proposed Future Capabilities

1. Provides a 'lessons learned' repository for use in future situations.
2. Provides a conversational resolution dialog for human operators.
3. Supports the automated extraction, collection, and collation of cases (i.e., problems and their resolution) from the current operational context.
4. Applies the case-based reasoning approaches developed in the COACH application (CDM 2000).
5. Utilizes a GOTS case-based reasoning library developed in conjunction with the Naval Research Laboratory, Washington (Aha et al. 2002).

The Administrator Application Tool will provide the capability to administer the server-side components of the installed application.

Proposed Future Capabilities

1. Ability to reset the entire distributed software system to the original installed state.
2. Ability to gracefully shutdown the entire distributed software system.
3. Ability to startup the entire distributed software system.
4. Ability to backup the current state of the entire distributed software system.
5. Ability to restore the state of the entire distributed software system from a backup.

The Trainer Application Tool provides the means to develop, initiate, and control mock scenarios for the purpose of training exercises, gaming strategies, and software testing and evaluation. Scenarios are controlled by scripts based on a simple text-based command language in conjunction with the domain specific vocabulary provided by the object model in order to manipulate the state of the domain at playback. Scripts may be hand written, generated by recording specific interactions performed by other clients to the domain, or built from existing pieces using the script development capabilities of the tool.

Current Capabilities

1. Stores scenarios and supporting data in a persistent repository.
2. Constructs new scenarios from existing scenario fragments.
3. Loads specific scenarios and associated data sets for execution.
4. Initiates and dynamically controls loaded scenarios.
5. Records scenario results and compares them to previous runs of the same scenario.

Proposed Future Capabilities

1. Ability to graphically display and manipulate the timing of scenario elements.

7.1.2 Support Tools

The Support Tools differ from the Application Tools in that they work outside the client application framework and are therefore not supported by the Dashboard Controller.

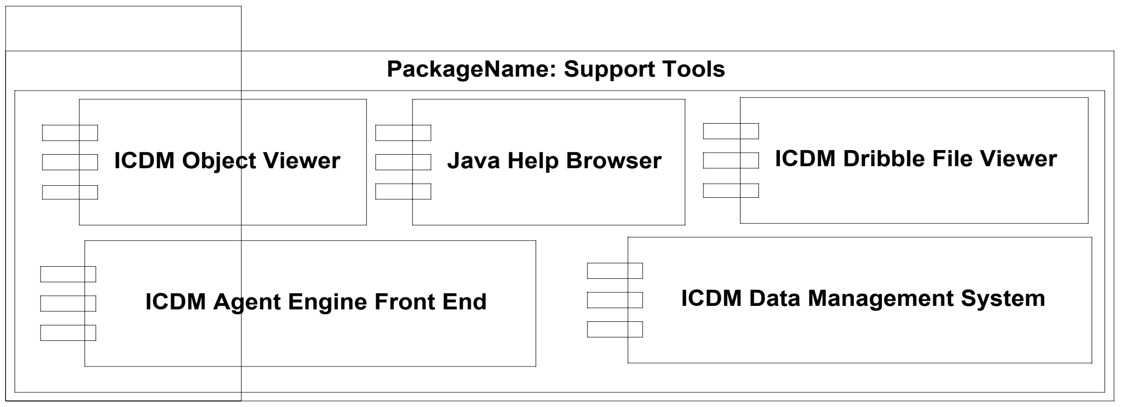

PackageName: Support Tools
ICDM Object Viewer
Java Help Browser
ICDM Dribble File Viewer
ICDM Agent Engine Front End
ICDM Data Management System

Figure 17: SILS MRAT Support Tools

The ICDM Object Viewer is a development and administration application that provides users with a graphical interface to perform basic object management services. Examples include the creation, destruction, modification, or association of objects. Query by example (template) functionality is also provided. This tool primarily targets system developers, but may also prove useful to administrators of fielded ICDM systems. Key high-level capabilities of the ICDM Object Viewer include:

1. Domain ontology self-discovery capabilities.
2. Ontology browsing capabilities.
3. Object view, add, edit, and delete capabilities.
4. Object association view, add, and remove capabilities.
5. Object query by example capabilities.

The Help Browser employed in SILS is the standard Java help browser configured to work with the SILS application toolset. This COTS component provides the following functional support features to the SILS application toolset:

1. Hyperlinked table of contents.
2. Hyperlinked index.
3. Consolidation of installed components into integrated help presentations.
4. Support for context sensitive help facilities.
5. Search support.
6. XML-based help content.

The ICDM Agent Engine Front End is a comprehensive run-time agent engine diagnostic tool that is used to look inside of an executing agent session to tune performance characteristics, and diagnose problems. This tool primarily targets system developers, but may also prove useful to administrators of fielded ICDM systems. Key high-level capabilities of the ICDM Agent Engine Front End are as follows:

1. View facts.
2. View rules.
3. Browse the ontology.
4. View the activation list.
5. View partial matches.
6. Watch rule firings.

The ICDM Dribble File Viewer is an agent engine diagnostics tool that complements the Agent Engine Front End. While the Agent Engine Front End is designed to look inside an ICDM Agent Engine during execution, the Dribble File Viewer is designed to assist users in examining Agent Engine log files and productively use the large volume of data that the Agent Engine is capable of logging. These capabilities assist developers in attempting to retroactively determine the cause of an ‘exception’ condition within an ICDM Agent Engine. The Agent Engine Front End can be used only to detect ‘exceptions’ when they occur, while the Dribble File Viewer must be used after-the-fact. This tool primarily targets system developers, but may also prove useful to administrators of fielded ICDM systems. Key high-level capabilities of the ICDM Dribble File Viewer include the following:

1. Support for JESS, CLIPS, COOL, and Eclipse expert system shell syntax.
2. Ability to selectively turn the various categories of information found in expert system log files on or off.
3. Ability to collect and collate facts that match a particular rule pattern.

The ICDM Data Management System (DMS) may be categorized as an ontology configuration, versioning, and instance management tool. The generic base ontology employed by ICDM is extended in SILS to more specifically apply to US Navy ships, and their operation and readiness conditions. The precompiled ontology is still generic in regard to ship type (class and series) and ship, but may be configured during runtime to address specific ship types and ships with a knowledge instance model. The size of knowledge instance models in the current version of SILS MRAT range from 3,000 to 8,000 objects, and will likely double or triple in size for a fully deployed system.

The primary use of the DMS in SILS is to manage the SILS knowledge instance models that tailor the generic ICDM ontologies to the SILS domain, and in turn tailor the SILS domain model first to a particular Navy ship type and then to a specific Navy ship. There can be much overlap between knowledge instance models for different ships. The DMS manages persisted data from which knowledge instance models for particular ships, scenarios, or development purposes can be generated. These data are not static and will require regular update to address changes in Navy policies, procedures, and equipment. This tool primarily targets system developers. Key high-level capabilities of the ICDM Data Management System include:

1. Logging of changes in the ontology model.
2. Generation of table schemas from XMI representations of the ontology.
3. Named logical tagging of data records.
4. Semantic Net Object Restore (SNOR) file generation for a set of logical tag names.
5. Semi-automated updating of existing data to new ontology versions utilizing the ontological model change log.

7.2 SILS Domain Components

The SILS Domain Components constitute the shared core of the system that provides users (as accessed through their Application Tools) a common operational picture of the domain, expert assistance in the form of software agents, and the means for effective collaboration with other system users.

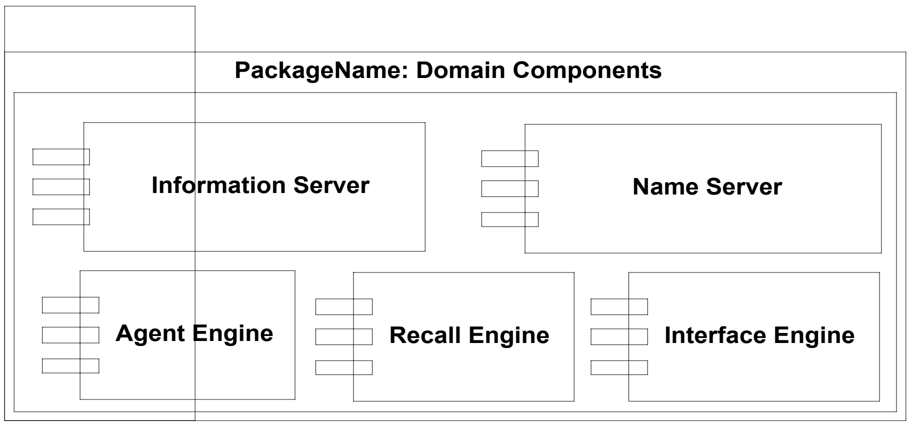


Figure 18: SILS MRAT Domain Components

The domain components are implemented with proprietary generic subsystems that are tailored by means of generated code, interpreted text files (such as agent rules and ontology translation maps), and properties. Together these components house the data, information, and knowledge that essentially model the physical world within the confines of the targeted system domain. This virtual model in turn may grow to service (in whole or part) the needs of many user communities at many levels of the military hierarchy, both afloat and ashore. The SILS MRAT Domain Components are depicted in Figure 18.

7.2.1 Information Server

The Information Server is responsible for managing the executable information model that represents the system domain for the SILS MRAT users and software agents. This model consists of shared distributed objects that may physically reside on many different machines across the network. In this respect, the Information Server is a conceptual entity that is physically implemented by a collection of semi-autonomous distributed services that may also reside on many different machines across the network. These services work together in a collaborative fashion to manage the distributed domain objects, and provide client applications, such as the Application Tools, with convenient means to access and manipulate them. Access to the services underlying the Information Server is encapsulated by the interfaces provided by the OML component. The services provided by the Information Server include the following:

1. Persistence Service
2. Query Service
3. Event Notification Service
4. Constraint Based Subscription Service
5. Factory Service
6. Association Service

7.2.2 Name Server

The Name Server provides the system with two key features. First, it provides the means to uniquely identify objects by working in conjunction with the object factories spawned by the individual information servers to generate a unique key for each newly created object. Second, it provides location transparency. Clients need only know where the Name Server is and need not be concerned with the physical location of the object being requested. By separating this functionality from the ICDM Information Server functionality (i.e., the generic core of the SILS Information Server) it can be used in conjunction with a data transport facility and network infrastructure to link multiple Information Servers together into a single object-serving communication facility as conceptually described in a previous SILS Framework paper (CDM 2001b). The SILS MRAT Name Server implementation is provided by a COTS software product with the following key capabilities:

1. Location transparency
2. Object identity
3. Standardized interface

7.2.3 Recall Engine

The Recall Engine provides the SILS MRAT system with a lower level reasoning facility to house the case-based logic of the system and complement the rule-based reasoning facility provided by the ICDM Agent Engine. The Recall Engine concept was developed and successfully demonstrated in the context of the Collaborative Agent Based Control and Help system (COACH). COACH was the result of ONR sponsored research into the application of collaborative agent-based decision-support systems to the diagnosis and repair of naval systems and equipment (CDM 2000).

A collaborative effort between CDM Technologies and the Navy Center for Applied Research in Artificial Intelligence at the Naval Research Laboratory (NRL) has been underway for over a year. This effort has adapted two NRL developed case-based reasoning engines, namely the Naval Conversational Decision Aids Environment (NaCoDAE) and the Taxonomic Case Reasoning System (TCRS), for use as generic components that may be embedded within the processes of a larger more general-purpose system. It is currently planned that this work will serve as the core of the next generation of the Recall Engine concept that was pioneered in the COACH project. This effort is intended to provide a facility to software agents and human users (i.e., through the Resolver Application Tool) for the development of courses of action to resolve current issues based on the collective experience captured and collated over time in the casebases of the Recall Engine. The objective of this work is to provide the following key capabilities:

1. Storage provisions for cases that capture problems, situational context, and courses of action.
2. Computational mechanisms for determining the similarity between current incidences and stored cases.
3. A mechanism for retrieving stored cases, based on similarity.
4. Mechanisms to probabilistically associate observational phenomena to individual cases that can be used to strengthen or weaken the correlation of individual cases to the existing situation.
5. Mechanisms to adjust the values of probabilistic associations over time, based on newly accrued information and user input.
6. Mechanisms to merge accrued casebases from ships of similar type.
7. Mechanisms to associate multiple courses of action with stored cases.
8. Mechanisms to probabilistically associate the observational phenomenon characteristic of the observed results of a particular course of action.

7.2.4 Interface Engine

The Interface Engine provides the SILS MRAT system with the capability to interface with the existing systems aboard ship. The Interface Engine also serves as the conduit for information passed between SILS MRAT installations at the various levels of the military hierarchy. This component is being designed and developed by ManTech Advanced Systems International. Some of the key capabilities proposed for the Interface Engine are as follows:

1. Ability to translate information from external system ontologies to a system-neutral internal ontology.
2. Ability to translate information to external system ontologies from a system-neutral internal ontology.

3. Ability to access external information regardless of the storage format or interfacing standard provided by the external system.
4. Ability to synchronize logical information fragments across multiple interfacing systems.
5. Support of constrained queries for logical information.

7.2.5 Agent Engine

The ICDM Agent Engine provides a proactive data-driven reasoning facility for use by the agents operating in the system. Incorporating an adaptation of the Jess expert system shell (Sandia National Laboratories 1997) the Agent Engine has been extended to allow it to seamlessly operate as a plug-in client to the SILS Information Server. Extensions have also been created to better manage the distribution of processor time between the resident agent rule-sets. The resulting agent operating environment provides an inference engine based on the efficient RETE algorithm (Forgy 1982) and a high-level language for rule specifications. Some of the key capabilities of the Agent Engine are listed below.

1. Efficient pattern matching for rule-based inferencing.
2. Support for truth maintenance.
3. Support for focus management.
4. Ability to directly access Java implemented functional capabilities from the action portions of an agent rule.
5. Easily embedded within any Java process.

The generic ICDM Agent Engine manages the rule-based implementations of the SILS MRAT specific agents. These agents are depicted in Figure 11 and discussed in the following subsection.

7.3 SILS Data Components

The SILS MRAT Data Components are explicitly designed to support the persistent storage needs of the system. Access to these data is typically provided indirectly through the SILS Information Server. Direct access to the data is not desirable, as it will encumber changes to the structure of the data if optimizations need to be made to enhance the performance of the persistence mechanisms of the system. The Data Components are depicted in Figure 19 and described in the following subsections.

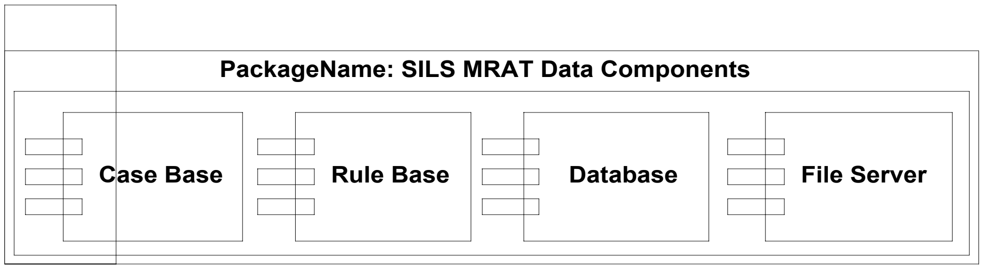


Figure 19: SILS MRAT Data Components

7.3.1 SILS Database

The SILS Database is designed to provide persistent storage for the objects resident within the individual Information Server Domains. A COTS relational database management system (RDBMS) that supports standard SQL interfaces is employed for this purpose. A uniform object to relational mapping is provided for by the ICDM Persistence Service (see Section 7.2.1). The ICDM Toolkit supports database schema generation directly from the XMI representation of the system ontology.

7.3.2 SILS Casebase

The SILS Casebase provides for the persistent storage of the cases and the associative memory indexes used by the Recall Engine (see Section 7.2.3). Each casebase is conveniently stored in a standardized XML-based file format.

7.3.3 SILS Rulebase

The SILS Rulebase provides for the persistent storage of the agent rules managed at runtime by the Agent Engine (see Section 7.2.5). Each agent has a rulebase associated with it, and each rulebase consists of an ASCII file containing individual rule specifications in the format required by Jess.

7.3.4 SILS File Server

The SILS File Server is implemented with a standard commercial Web server, to provide the SILS system and other external systems with remote access to files utilizing the Universal Resource Locator (URL). The files available through the File Server can be partitioned into two distinct groups implemented as root nodes by the server. The User Reports Group will contain reports generated by SILS for publication to applicable non-SILS users. The System Group will contain files necessary to support the systems that are not normally directly accessible by the users of SILS MRAT (e.g., ship-drawing files, geographical map files, and files used to support access to the implementation of the Name Server).

8. External Interfaces

8.1 Interface Description

The SILS software development project is set in the broader context of a vision for a system of systems, with the objective of preparing and supporting ship readiness through enhanced decision-support in logistic, supply, and budgetary matters. The key to the realization of this overarching SILS concept lies in those mechanisms that link new and existing shipboard information systems and decision-support systems into a suite of seamlessly integrated systems. In the approach taken by this initiative, these mechanisms will be encapsulated within the SILS Interface Engine (IE), designed and developed by ManTech Advanced Systems International, Inc. (Fairmont, West Virginia).

When considering the myriad of existing ship systems that may not all be amendable to code changes, the mechanisms employed by the SILS IE must assume the primary responsibilities for integration and thus incur most if not all of the intersystem dependencies as well. The systems that are required to be integrated, their installed versions, and their configurations may vary greatly across the range of ships and ship types that may employ a SILS system. It is expected that participating systems will typically have distinct and independently developed representations (i.e., largely data schemas as opposed to information models) of their individual, potentially overlapping, areas of concern. They are also likely to have differing interfacing mechanisms based on a wide variety of implementation technologies.

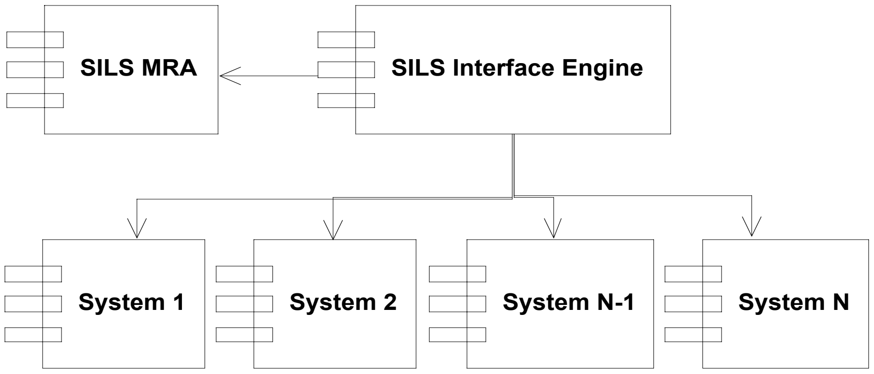


Figure 20: System dependencies

The extensive set of architectural and functional requirements associated with the SILS IE are likely to rival or exceed those of the majority of the systems it is intended to integrate. In this regard, the SILS IE should be thought of as a core system in its own right, with the context of its utility spanning beyond the SILS project concept as it directly targets the system integration problem that has become a central issue for the military services in recent years. This document reflects the view of the SILS Interface Engine as seen from SILS MRA (Mission Readiness Analysis). The SILS MRA view of the SILS IE as a simple 'black box' component can be misleading. It is likely that

MOXIE (the generalized base for SILS IE) will develop into a distributed multi-user system framework that employs a variety of intelligent agents, end user applications, and development tools.

This report is concerned only with the interface between SILS IE and SILS MRA. For the most part the relationship between SILS IE and SILS MRA is the same as that between SILS IE and the existing systems with which it intends to interface. The primary difference will be in the typical direction of information flow. The majority of the existing shipboard systems, depicted as System 1 through System N in Figure 20, targeted for inclusion within the SILS confederation of systems may be classified as stand-alone information systems or information system families. By definition, these types of systems should primarily be considered as information providers designed to manage, store, and share information for a multi-user community. By definition, collaborative decision-support systems such as SILS MRA should primarily be considered as information consumers designed to provide an integrated picture of the domain from multiple information sources, and to provide mechanisms for experimenting with and evaluating different courses of action in a collaborative fashion with other decision makers and supporting software agents. In this regard, the initial focus of SILS IE is to target the flow of information from existing information systems to the SILS MRA decision-support system.

From the perspective of SILS MRA, the SILS IE component is just another client to the Object Serving Communication Facility; however, the nature of the interactions with the server varies between the three clients under discussion. The nature of interaction for the three clients is discussed in the following paragraphs and summarized in Table 2.

User interfaces typically have a varying but low volume of subscription traffic, primarily associated with open dialogs or views in the graphical user interface (GUI) presentation. They also typically have a relatively small rate of object creations, deletions, and modifications. This is due to both the nature of decision-support clients that primarily display information to the user, and the relatively slow speeds with which human users interact with the interface as compared to automated clients.

The Agent Engine client has a large but relatively constant subscription profile, as the internal domain information model it maintains is in essence a synchronized microcosm of that maintained within the object serving communication facility. The primary responsibility of the Agent Engine is to infer new information from that provided by the other information sources and to remove inferences as they become invalid due to changing source data. This is commonly referred to as truth maintenance. The number of inferences created and maintained by the Agent Engine is relatively small when compared to the number of objects from which the inferences are derived.

The IE must maintain a small constant subscription profile of object instances used to represent data driven requests for service. In addition, a varying profile must be maintained that is based on the volume, variance, and type of client requests. The results returned from information requests may be categorized as static or dynamic. A static result provides a snapshot of the requested information as represented by the source provider(s) at the time the result was assembled. A dynamic result is maintained by the Interface Engine to reflect changes that may occur to the source information after the information request is processed and the results returned. Static results will not add to the subscription profile but may result in a large number of object creations, and indirectly in deletions, depending on the size of the result set. Dynamic results will not typically involve large numbers of creations and deletions after the initial request is processed, but may result in a large number of object modifications if the requested information is subject to a great deal of change in the information provider(s). Dynamic results will only incur additional subscriptions if the information can be modified directly in SILS and is to remain synchronized with the source data (i.e., changes must be propagated back to the source providers).

Table 2: Client Object interaction profiles

	Object Subscriptions	**Object Creations**	**Object Deletions**	**Object Modifications**
User Interface	Low	Low	Low	Low
Interface Engine	Low to Medium	High	High	Medium
Agent Engine	High	Low	Low	Low

8.2 Interaction Types

SILS MRAT is able to post standing requests for logical information sets to SILS IE, which is able to pull the requested information from one or more external systems and pass it to SILS MRAT. SILS IE is able to keep this information synchronized by monitoring the information for changes in both SILS MRAT and in the corresponding external systems, then propagate the noted changes to or from SILS MRAT. The following subsections describe three levels at which the prototype version of SILS MRAT may interact with external systems.

8.2.1 Physical Level

At the physical level of interaction, SILS MRAT interfaces to an actual physical deployment of an existing system through SILS IE through the interface provided by the external system developer. This level of interaction requires some sort of formal agreement between the SILS MRA program management office and the program management office of the external system. It also requires in-house installation of the existing system for development, testing, and demonstration purposes at both CDM Technologies and ManTech Advanced Systems International. Likely, some level of support from the existing system developer will be required to interpret the data dictionary, solidify the ontology mappings, and understand any proprietary interfaces.

8.2.2 Information Level

At the information level of interaction, SILS MRAT interfaces to typical data sets resident in the existing systems via the ontological model employed by the system but not directly to the system itself. The data is instead resident in a database controlled by the SILS IE component. Interaction at this level requires existing system data dictionaries along with data dumps of typical data sets for the specific vessel platforms targeted by SILS MRAT. This level of interaction validates the capability of the SILS Interface

Engine to translate from the actual external system ontologies to that required for SILS MRAT and is a step along the way toward the physical level of interaction. It also allows the development of SILS MRAT to continue while the political, contractual, and implementation details of interaction at the physical level are being worked out with the program management office of a particular existing external system. Some level of support from the external system contractor is expected to be provided to interpret the data dictionary and to solidify the ontology mappings.

8.2.3 Data Level

At the data level of interaction, SILS MRAT interfaces to fragmented data sets available in one or more existing external systems by the most convenient format for generation and use, via SILS IE as determined by the external system developer. This level of interaction applies to data sets deemed useful for SILS MRAT users and agents. Such data sets would most likely be sourced by an external system that may not yet have been identified or is not yet available for access at the physical level. Data level interactions allow data and information targeted to be sourced by external systems to be completely partitioned from data to be sourced within SILS MRAT. The loose coupling provided by SILS IE allows this sort of interaction to easily migrate toward the information level and then to the physical level, given the initial partitioning.

8.3 System Level Interface

The system level interface addresses the architectural level mechanisms by which client applications may interact with the ICDM Object Serving Communication Facility that provides the generic core of the SILS MRA subsystem. Given that SILS IE must incur the majority of dependencies in regard to the systems it integrates, the ICDM System Level Interface also defines the interface between SILS MRA and SILS IE.

From the perspective of physical clients, the ICDM Object Serving Communication Facility provides a number of domain ontologies and the Object Management Layer (OML) interface library as a convenient way of interacting with them. Clients incur a compile time call dependency on OML as OML methods are directly invoked by client code to interact with the objects resident within the Information Server. They also incur logical dependencies on the domain ontologies of interest as the ontologies provide the domain terminology that identifies object classes and attributes. OML works with domain specific ontologies at the meta-object level in order to obviate the need for compile time dependencies on system specific ontology definitions. This allows client applications to be free from system specific compile time dependencies.

As a means of eliminating these dependencies, the arguments of many OML methods include the textual representation (i.e., name) of entities from the terminology provided by the ontology. This introduces a logical dependency on the ontologies with which a client application interacts. The logical dependency manifests itself at runtime in two ways. The Information Server variant targeted to support the SILS system is based on the Common Object Request Broker Architecture (CORBA). CORBA provides client applications transparent access to remote objects through the corresponding client side proxy objects. Whenever a client manipulates a particular object through the OML functionality running within its process, OML instantiates and accesses the appropriate client side proxy. The direct and transitive client dependencies under discussion are depicted in Figure 21.

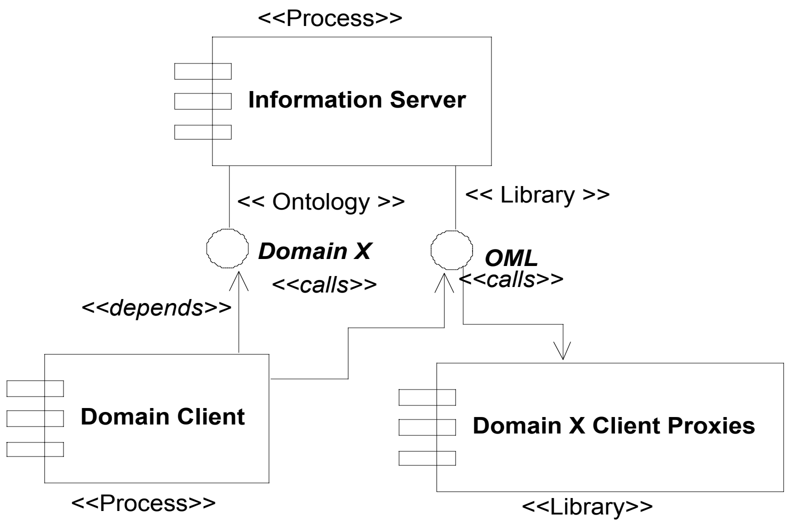


Figure 21: Client dependencies

The call-level and logical dependencies associated with the SILS MRA interface by SILS IE are described in more detail in the following sections. The specific top level interface dependencies incurred by the primary clients in SILS are depicted in Figure 22.

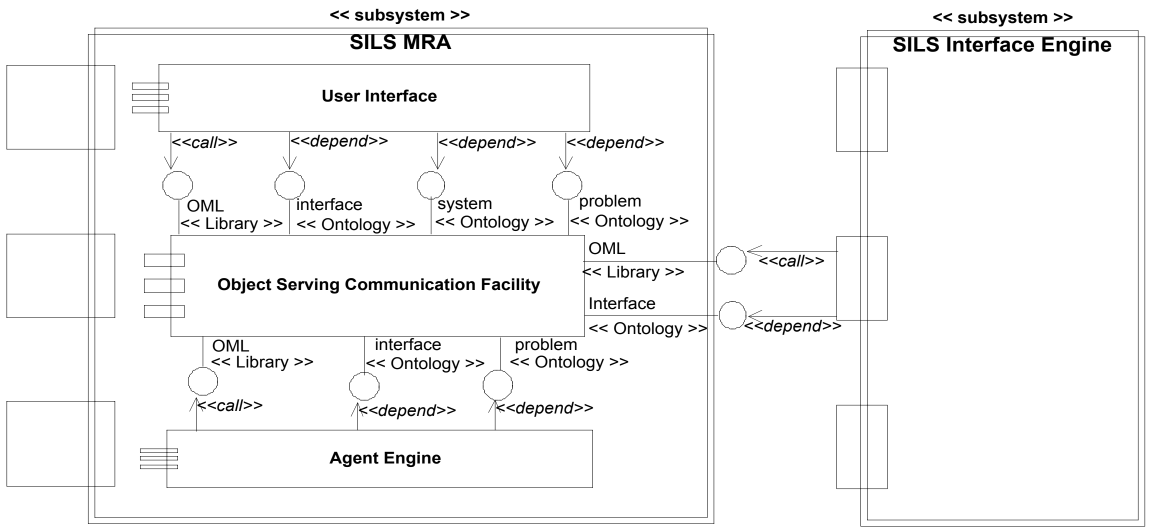


Figure 22: Interface dependencies

8.4 Call-Level Interface

The call-level interface to the SILS MRA Object Serving Communication Facility is provided by the OML class library. OML provides general functionality for the complete life-cycle management of objects, their attributes (characteristics) and associations (relationships). Interaction with object instances is simplified using simple strings with attribute value constraints handled internally. Association management is also provided internally alleviating the requirement (and complexity) for ensuring referential integrity by the using application. Management of interests are also provided and implemented internally. They are exposed to using applications through the standard Java event model. Interests may also be constrained using conditions. Condition satisfaction checking is performed utilizing an inference engine. Additionally, support is provided for accessing multiple servers simultaneously and transparently. The primary application of this library is envisioned to be for use by applications requiring little to no prior knowledge of the object domain model. Internally, the required management and information is provided through runtime reflection and properties. Good examples of such applications are user interfaces where a hard coded notion of the domain is expensive to both develop and manage.

An object-oriented representation of information necessarily incurs a requirement for managing objects and their associations. OML was designed to simplify client application object management functionality. The design of OML centers on the Template, POW (proxy object wrapper), and Attribute classes. Figure 23 presents a class diagram showing these classes and their relationships. The POW class adds generic functionality to the object model classes to aid in object manipulation and, in particular, association management.

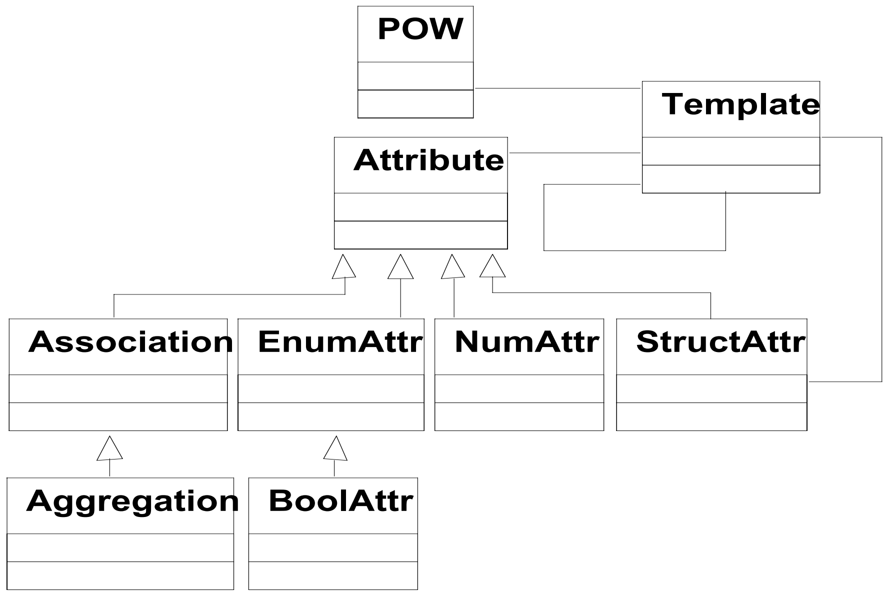


Figure 23: OML classes

Association management is encapsulated in the *set, add*, *remove* and *delete* methods of the POW class. For example, when a call is made to add an object reference to an association (referred to by the role name defined in the object class) the POW class adds the appropriate object reference to the other end of the association. In the process, the POW class verifies the existence (and therefore validity) of the associated object. Additionally, if a call is made to remove an association and the associated object is an aggregate part of the object then the associated object is also deleted. It should be noted that object creation, deletion and attribute modification transactions are queued locally and will not be reflected in the object server instance store until a call is made to the POW *post* method.

The Template class implements functionality to support attribute constraints and validation. Additionally, it contains support for class constructor and access method determination through runtime class reflection and properties. A Template instance is created for each class, as required, with each class represented through the defined hierarchy. The associated Attribute class and its subclasses provide constraints on attribute values. One of the benefits incurred with the POW class is the fact that all attribute values are entered and obtained as strings. The constraints on attribute values are handled internally to the Attribute classes. The benefit from a user interface point of view is that specialized attribute value management becomes unnecessary, or is at least greatly simplified since only strings need be accommodated.

The POW and Template classes also include methods for managing both instance and class-based interests. The implementation of these methods follows the design pattern specified by the Java event model. Specifically, instances of the POW and Template classes are event producers and contain methods defined for registering listeners (i.e., instances of classes that implement an appropriate listener interface). When a POW or Template instance fires an event, methods defined by the listener interface are invoked and passed in the event as an argument. Subscriptions registered with the object server are managed internally through calls to these listener registration methods.

Additionally, conditions may be defined which provide for complex constraints on interest satisfaction. By providing runtime definable conditions, within the OML framework, client applications are not required to filter incoming events to enable specialized interests. This capability allows general client application use in very specialized environments without specialized code support. OML supports the definition of conditions for interests on classes of objects and on individual object instances. Condition satisfaction checking is performed using a rule-based inference engine. The inference engine provides high-level mechanisms for specifying complex conditional patterns, a complete full-featured environment for managing the associated rules, and efficient, scalable mechanisms for identifying satisfied conditions and triggering the appropriate action.

A generic interface is provided in the OML framework to support client interaction with object servers. Each implementation of an object server interface may provide access to servers based on different architectures. There are only three requirements, as follows: (1) the object interaction must take place through client-side instance methods; (2) the client-side classes must adhere to a prescribed pattern; and, (3) the interest notification must be event-based. Object server interfaces are tied to unique domains (i.e., class namespaces). Objects that are remotely serviced by an object server provide for a distributed/collaborative framework, however, the use of purely local objects (i.e. objects that are not maintained outside of the local client application environment) provides additional flexibility. Examples include objects whose characteristics are all derived (e.g., facades/views), objects that implement behavior alone (e.g., private agents), or client-side user-interface objects (i.e., objects that interact directly with client-side functionality).

By providing an object server interface to local objects, interaction with these objects may take place through the same client interface (i.e., OML). Both the POW and

Template classes make use of the Object Factory class that provides the central interface to the object server interfaces. Since each domain is associated with a single server interface, the Object Factory can determine which server interface to use through class identification within a domain. Therefore, interaction with objects and classes (through POW and Template instances) is transparently handled without any direct domain specification by the client application.

Startup and shutdown methods are also provided to allow for specific initialization and cleanup of the object server interfaces. The Attribute class and its subclasses (i.e., Association, Aggregation, etc.) provide specialized management functionality for various attribute types. Additional management classes may be added by extending the appropriate Attribute subclass. These additional classes may be used to replace or add to existing management classes.

OML does not provide direct access to general methods defined for objects. However, indirect access to the methods that provide for instantiation and the retrieval/setting of attribute values is provided through the constructor and get/set methods. Therefore, for example, implementing derived attributes and accessing them is simply a matter of implementing the appropriate access method (i.e., a method whose name reflects the attribute name and has no parameters). The body of these methods may contain calls to other methods that are directly accessible from the calling environment. Hence, if these methods are implemented in client-side classes (i.e., local classes) calls to any available client-side functionality are possible (including calls to OML methods, the graphical user interface environment, etc). Likewise, if these methods are implemented in server-side classes, calls to any server-side functionality are possible (i.e., direct database access, centralized common services, etc).

8.5 Logical Domain Interface

The generic call-level interface that OML provides does not directly address the system specific needs of particular client applications. These needs are instead indirectly addressed through system specific ontology definitions that are referenced by arguments in the applicable OML methods that have been discussed in previous sections. In regard to the interface between SILS MRA and SILS IE, the supporting definitions can be characterized as belonging to one of three possible groups, namely: export definitions; import definitions; or, service requests. From the perspective of the SILS MRA subsystem, export definitions are those ontological elements that are designed to present information to external clients as opposed to import definitions, which are designed to receive information. These groups may have overlapping membership. In many respects, the amount of overlap is representative of the degree of coupling between the subsystems. However, it should be noted that this abstract concept is not so easily defined as the number of definitions in total. Service requests are different in that they incur aspects of both of the other two groupings.

In order to support the desired one-way dependency between SILS IE and the systems it connects the Interface Engine must know the specific interface services desired by each individual system that it is targeted to support. This knowledge does not have to be hard coded. It can be implemented by configurable properties that are loaded at startup to configure the engine to the targeted domain. While this approach can support the continuous synchronization (i.e., either real-time or periodic) of data elements across differing system ontologies, it cannot address demand driven services such as a query for external information whose result set is temporary and will be discarded after use. For this type of interaction the SILS MRA subsystem must assume the role of a client to SILS IE, thereby incurring a dependency on the Interface Engine. Note that by definition those systems requiring information from the Interface Engine (i.e., information resident in one or more external systems) assume the role of a client while those that provide information assume the role of a server.

In this regard, the existing systems initially targeted for interaction with SILS IE are classified as information providers since they were designed to operate in a stand-alone mode (i.e., as individual systems or system families) without the benefits that are associated with the Interface Engine concept. It is also acceptable for the clients of the Interface Engine to incur a minor dependency on a narrow interface for service requests, because only those that are designed after the implementation of SILS IE would be designed in a manner that require its services.

Within the context of the blackboard-like interaction model employed by the SILS MRA subsystem (CDM 2001c), the service request interface takes the form of an ontology-based logical dependency rather than the call-level style of interface typically employed to support interactions between systems. Since the underlying core of SILS IE (i.e., MOXIE) has a broad range of applicability independent of the SILS MRA subsystem, it can be argued from a longer-term point of view that it should provide a call-level interface in addition to or in lieu of the logical interface currently being discussed. This consideration is particularly pertinent since the typical systems currently being produced lack the collaborative mechanisms necessary to support a blackboard-like interaction model. However, at this stage of the SILS project a blackboard style of interaction is appropriate since it facilitates the kind of experimentation likely to be required in this ambitious endeavor. After the experimental phase of the project has focused in on a suitable implementation design, the logical ontology-based representation of system dependent interactions utilized by the blackboard paradigm is easily converted to a call-level interface.

Given the blackboard style of interaction currently provided by the ICDM core of SILS MRA, the ontological models involved in the interface to the Interface Engine must be both acceptable to and designed in conjunction with the developers of the SILS Interface Engine due to the required dependency. This is a design decision with issues for both of the top-level subsystems being discussed.

Without providing a detailed ontological model for interaction, the ontological elements of the model can still be broadly characterized. Export elements may be either directly exposed from the core problem domain and/or the system domain partitions to the SILS MRA object model (discussed in the SILS Architectural Design Report (CDM 2001c)), or exposed indirectly through interface facades. Interface facades simplify the model according to the needs of the clients and shield the clients from the inevitable changes to the core model in much the same way as a 'view' does in a relational system. Currently there is no requirement for export elements in the context of the initial proof-of-concept system. However, it may be beneficial to devise a minimal set of export elements in order to flush out the issues and test the machinery of the Interface Engine.

Import elements have the issue of external players directly modifying the instances of the system's core model, which is not desired if the model implementation cannot ensure the integrity of its state. In the context of the current SILS MRA design these model elements reside in the interface domain partition as well as those used to represent service requests. The specific types and nature of the import elements and service requests are dictated as much by the capabilities of SILS IE and the information available in the external systems it can interact with, as by the needs of the SILS MRA subsystem. Service requests are related to import elements in that the result sets returned and maintained by the corresponding Interface Engine service are captured in the SILS MRA process space as import element. Service request must also have a relatively small export element or set of export elements to provide parameters to the request specifying its configurable features.

8.6 Interface Domain Ontology

The SILS Interface Domain is a SILS specific implementation of the generalized concept of an interface domain. An interface domain serves as a 'swap' space for the exchange of information with external systems. It defines simple and easy to understand information sets without the complexities found in the core problem domain model. It also provides a layer of isolation between the problem domain and the interfaces with external systems, thereby ensuring the flexibility to evolve the core model over time. The domain is implemented as a Java based CORBA package.

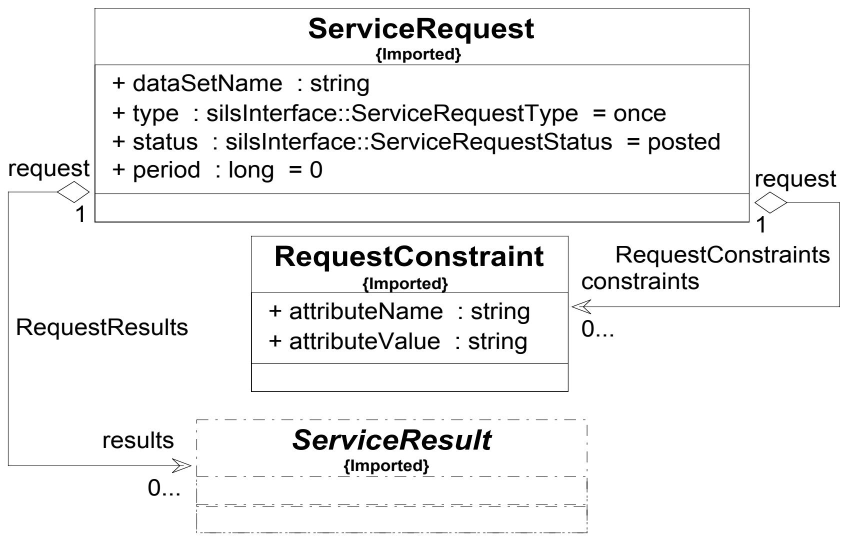


Figure 24: Service Request framework

The two primary classes within the package are the Service Request class and the Service Result class. These two classes provide a framework for the interface domain, as depicted in Figure 24. Service Request objects are typically posted by the system agents. They are picked up by SILS IE, which then queries the appropriate external information systems to post the corresponding Service Result objects. The Interface agent picks up additions, deletions, or modifications to Service Result objects and then translates them into the core problem domain for use by the rest of the system. The results that correspond to a Service Request object can be constrained by associating Request Constraint objects. If more than one constraint is specified, then the constraints are combined by a logical ***and*** construct.

The data set name attribute of the Service Request class specifies the type of information being requested. The permitted values correspond directly to the class names of concrete classes deriving from the Service Result class. The currently supported data set names are as follows:

1. ***Machinery Problem -*** correlates a uniquely identifiable domain asset to a specific phenomenon.
2. ***Part Availability -*** returns information about available parts such as cost and location.
3. ***Part Order -*** returns current information about a specific part order as identified by the requisition number.
4. ***Maintenance Activity -*** returns results on the individual maintenance activities scheduled to be performed onboard ship.
5. ***External System Down -*** returns results of external systems that are currently down.
6. ***Personnel Training*** – returns information about the training status of individual crewmembers.
7. ***Personnel Availability*** – returns information about the availability of personnel by NEC.
8. ***Personnel Gain and Loss*** – returns information about arriving and departing crewmembers of the ship.
9. ***Temporary Personnel Assignment*** – returns information about temporary duty assignments of individual crewmembers.
10. ***ANOR*** – returns information about equipment that is anticipated to be not operational ready.
11. ***CASREP*** – returns information about equipment casualties.
12. ***Degraded Equipment*** – returns information about degraded equipment.

The type attribute indicates the manner in which the request should be updated. The supported types are as follows:

1. ***Once -*** fulfill the request but do not continue to update.
2. ***Refresh -*** update the results of an existing request.
3. ***Periodic -*** update the request every x milliseconds (where x is the value of the period attribute).
4. ***Continuous -*** update the request whenever the corresponding external information changes.

The status attribute defines the values that may be used to indicate the current state of the request results. Status values are as follows:

1. ***Posted -*** result processing has not begun and any results currently associated to the request should be considered invalid.
2. ***Processing -*** the Interface Engine is currently responding to the service request.
3. ***Completed -*** the request was processed successfully.
4. ***Failed -*** the request could not processed due to an error.

The following subsections consist of a list of digital information systems with which interfaces to SILS MRAT may provide benefits to: the users of SILS MRAT; the users of the external system; or, to the US Navy as a whole. This list was created on the basis of knowledge acquisition interviews aboard USS Fletcher (ManTech 2001a), USS Comstock (ManTech 2001b), and USS Peleliu (ManTech 2001c). A few additional candidate systems that were identified at briefings or discussions with client representatives have also been placed on the list. The systems have been preliminarily grouped into three categories (i.e., Primary, Secondary, and Other) based on the relative value to be obtained in interfacing with them (based on the judgment of SILS MRAT development team).

8.7 Candidate System Interfaces

8.7.1 Primary Candidates

The primary candidates are those digital information systems that are currently being considered as having the highest relative value. These systems are initially targeted for a physical level interface (see Section 8.2.1) with SILS MRAT.

NTCSS: The Naval Tactical Command Support System (NTCSS) is an integrated suite of systems (i.e., OMMS-NG, R-Supply, and R-Admin) that together perform ship and aviation maintenance management, material and financial management, as well as medical and other related administrative management.

OMMS-NG: The Organizational Maintenance Management System – Next Generation (OMMS-NG) targets ship and aviation maintenance management. It tracks all maintenance and repair jobs aboard ship by Job Serial Number (JSN), and maintains records of the configuration and location of every piece of equipment on the ship along with corresponding equipment reference information such as: blueprints; drawings; flow charts; and, diagrams. The OMMS-NG system interfaces with R-Supply to provide such data as: part numbers; part order status; and, part availability aboard ship. It provides on-line automated management of: CSMP, APL, and COSAL as described in the following subsections. When a part has been ordered through OMMS-NG it is automatically moved to R-Supply.

CSMP: The Current Ships Maintenance Project (CSMP) provides a list of maintenance actions and other outstanding work to be performed. Jobs must be created in this system before a part can be ordered. The system lists all parts required for purchase and schedules when maintenance work will be completed by assigning work completion dates. It provides shipboard maintenance managers with a consolidated listing of deferred corrective maintenance so that they can manage and control its eventual accomplishment of all required maintenance work.

APL: The Allowance Parts List (APL) provides schedules of parts required for maintaining shipboard equipment and a set of easy-to-use applications that facilitate the processing of parts and supply research needs. These products are offered as part of an annual subscription service, include monthly or quarterly updates on CD-ROM, and are continuously updated when accessed on-line. The subscription services include Navy specific spare parts, equipment and repairable items associated with Naval shipboard and aviation equipment, and also provide access to detailed information on more than 30 million NSN/NICN items within the Navy Supply Catalog system.

COSAL: The Coordinated Shipboard Allowance List (COSAL) is both a technical and a supply document prepared for an individual ship. It is the basic source of information on repair parts and materials needed for a job. The COSAL lists the equipment and components that a ship requires to perform its operational mission, and the material required for support of all installed and portable equipment aboard ship such as: the repair parts; the special tools required; the overhaul and repair equipment; and, the miscellaneous portable items necessary for the care and upkeep of the ship. The type, number, and ordering data are provided for the equipment and supplies that should be aboard ship. It provides information on such items as the name of a system (e.g., engine, pump, ejector, etc.), the manufacturer's name and the identification number (e.g., General Motors Corporation #3255), the technical manual number for the system, the manufacturer's drawing numbers, and the Allowance Parts List (APL) numbers for related systems (e.g., governors, starters, transmissions, etc.). It also provides specific information about the National Stock Number (NSN), unit of issue, cost, and the number of items needed. It may also include lists of part numbers and the Federal Stock Number (FSN) for cross-over checks.

R-Supply: The Relational Supply System (R-Supply) is the part of the NTCSS suite that targets the ordering, receiving, tracking, and issuing of supplies and material. It also maintains the financial records associated with the operational target (OPTAR) for the ship. It has the requisite interfaces required to communicate information throughout the ship and to other activities afloat or ashore. After a part has been ordered through OMMS-NG, it is moved to R-Supply.

R-Admin: The Relational Administration System (R-Admin) targets personnel and manpower management. The functions of R-Admin include: absences; addresses; advancements; awards; berthing/stateroom assignment, career information management, command training, Division Officer's Notebook; lifeboat assignment; Personnel Qualification Standards (PQS); Quarterdeck Management; Security Clearance; Visitor Control; and Watch Bills. It provides ratings of enlisted personnel in the form of Personnel Qualification Standards. Personnel Management functions include: information about prospective gains and losses; entitlements; classifications; and, the capability to quickly find a person's records. Manpower management functions include records of the billets required and authorized for the command in the Activity Manpower Document. R-Admin records and monitors deployments and other personnel absences, special qualifications and certifications listed, produces training muster sheets, and a record of training for the service record. It manages Navy training courses and can assign personnel to them. It also identifies and assigns qualified personnel to watches and produces watch bills associated with identified conditions, evolutions, watch stations, duty sections, and watch teams. R-Admin is also available for use on a PDA for recording class and muster attendance, to be uploaded to R-Admin.

TRMS: The Type Commander (TYCOM) Readiness Management System (TRMS) is used to document and plan the crew training for a ship. TRMS ensures that the standardized basic, intermediate, and advanced evolutions are completed once during each 24-month work-up and deployment cycle. The system tracks M-Ratings in each mission area and reports on the completion of each evolution (e.g., exercises, inspections, and training). The combined ratings of pertinent evolutions provide a single rating in each mission area. When workup starts, every area has a M-Rating of M-5 with a C-Rating of C-5. The goal is to have a rating of M-1 in each mission area by the deployment phase. An M-Rating can be overwritten by Training Readiness Capping, which identifies specific areas that alone can decrease the M-Rating in a mission area if not complete. Data from TRMS is submitted to SORTS upon significant change or once a month.

MRDB: The Material Readiness Data Base (MRDB) is a Data Warehouse managed by the Corona division of the Naval Surface Warfare Center. It is used along with other data collected by Corona to create reference data sets such as the Equipment Breakdown Model, the Equipment Operational Capability (EOC), and the Material Condition Model. These models would be very useful to SILS MRAT in support of its readiness assessment capability.

8.7.2 Secondary Candidates

The secondary candidates are those digital information systems that are currently considered as not having the highest relative value, but as potentially useful across a wide variety of ships and individual departments. These systems are initially targeted for information level interfaces with SILS MRAT.

ICAS: The Integrated Condition Assessment System (ICAS) automatically monitors selected mechanical systems aboard ship through remote sensors or automated feeds from PDAs. It provides: a better understanding of a given system's operation; the simultaneous use of multiple predictive technologies; the ability to detect minute failures; the ability to predict overall process performance; and, a paperless engineering log.

SORTS: The Status of Resource and Training System (SORTS) provides general status information for a ship such as: major equipment; special capabilities; and, the mission readiness of the crew. It contains the Mission Essential Personnel List by NEC, and tracks the M-Ratings and C-Ratings for the ship.

Logistics Toolbox: The Logistics Toolbox provides supply officers with access to the Defense Logistics System that is connected to the shore-side supply systems of each of the DoD services. It allows users to order supplies then track the shipment and delivery of the order. The Logistics Toolbox links to logistic information based on the type of information desired through four individual applications that assist in the identification, location, acquisition, and tracking of the supply item under consideration. It also provides training and desk guides to several other logistics systems.

PMS: The Planned Maintenance System (PMS) strives to reduce complex maintenance to simplified procedures that are easily identified and managed at all levels. It defines the minimum planned maintenance required to schedule and control PMS performances, and the methods and tools to be used. Specifically, PMS: provides for the detection and prevention of impending casualties; allows for the forecasting and planning of manpower and material requirements; supports the planning and scheduling of maintenance tasks; facilitates the estimation and evaluation of material readiness; and, assists in detecting areas requiring additional personnel training and improved maintenance techniques to ensure the readiness of the ship.

SKED: The SKED system is designed to track and schedule equipment maintenance on a weekly and quarterly basis, and to assign maintenance responsibilities. It provides maintenance schedules marked to show what has been completed, rescheduled, or not accomplished.

GCCS-M: The Global Command and Control System-Maritime (GCCS-M) aids the war fighting capability and decision making of operational commanders by receiving, retrieving, and displaying information relative to the current tactical situation. It receives, processes, displays, and manages data on the readiness of neutral, friendly, and hostile forces in order to execute the full range of Navy missions in near real-time.

8.7.3 Other Candidates

The other candidates are those digital information systems that are currently considered as being useful only for select ship types or departments or as having only a limited amount of information that would be useful to SILS MRAT. These systems are being initially targeted for data level interfaces with SILS MRAT.

SAMS: The Automated Medical System (SAMS) is used by the Medical Department aboard ship. It is used to track: medical supplies; crew immunizations; hearing examinations; and, physical examinations. The basic functions are: appointments; treatment; immunizations; and, general administration. It has interfaces available for use on a PDA to track environmental conditions and inventory management to directly input into the SAMS system. R-Supply and SAMS are required to be updated manually to synchronize shared information.

FSM: The Food Service Management (FSM) system supports food services with: automated menu production: receipt of inventory: and, issue and accounting processing. It provides Web-based data including: subsistence prime vendor catalogs; standard menus; HACCP guidelines; nutrition; and, cost analysis. It is available for use on a PDA for creating breakout lists (i.e., shopping lists).

CSCS: The Combat Systems Casualty Control Computer System (CSCS) integrates combat system procedures, operational states, and alarm conditions.

IBFT: The Integrated Battle Force Training (IBFT) database identifies training requirements in the areas of communications, information systems, and networking. It enables training officers aboard ship to assign and track personnel training requirements and their fulfillment. Targeted users include: C4I-SR Training Officers and their staff, CO, XO, OPS Officers, and TYCOM/CINC training representatives. IBFT shows which personnel are required to go to C4ISR training based on assigned job and training requirements for all C4ISR systems.

NTFS: The Navy Training Feedback System (NTFS) allows Navy activities and personnel to identify, report, and validate training related deficiencies, such as: an individual has not been trained in specific required skills; an individual has been trained in required skills but cannot perform them; the required training is not available; the training provided is outdated; and, the training or the discrepancy involves other broader issues.

NAVFIT98: The NAVFIT98 system provides users with the capability to create, store, organize, and print fitness, counseling, and evaluation reports.

TRIMS: The Technical Risk Identification and Mitigation System (TRIMS) supports technical risk management. It is designed to provide early indication of potential problems, identify areas of risk, and track program goals and responsibilities.

WECAN: The Web Centric Anti-Submarine Warfare Net (WECAN) provides fleet-wide enhanced situational awareness for detecting and prosecuting enemy submarines. It has a near real-time capability to disseminate and collaborate information on a 24-hour basis, and allows users to watch unfolding events regardless of their current location.

8.7.4 Existing Support Candidates

The existing support candidates are systems, processes, or manuals that may have information of value to SILS MRAT.

PERA: Planning and Engineering for Repairs and Alterations (PERA) is a program for improving the advance planning, integration, and control procedures associated with ship availabilities. The primary objective of the PERA program is to provide intensive management for the accomplishment of effective, efficient, orderly, and timely ship availabilities. The PERA program develops a complete and integrated ship availability planning work package that is usable by an overhaul activity with minimum additional planning.

BUPERS: The Bureau of Naval Personnel System (BUPERS) is designed to provide accurate, reliable and readily accessible personnel information to fleet personnel.

IBS: The Integrated Bar Code System (IBS) provides the capability to quickly scan received items.

CMP: The Continuous Monitoring Program (CMP) ensures that supply departments are consistently meeting force standards in critical readiness areas by tracking pulse points such as stock validity (i.e., how often requests can be found on-hand). Pulse points are graded as green, yellow, or red based on supply system or force standards.

CART: Command Assessment of Readiness (CART) is a process to ensure mission readiness. It consists of a review of personnel qualifications and inspection of areas in two phases (CARTI and CARTII).

EDVR: The Enlisted Distribution Verification Record (EDVR) is used for manning and assignment decisions. Distributed on a monthly basis by EPMAC (Enlisted Personnel Management Center), it is organized into eight sections as follows: Sections 1 to 3 list members who are expected to report, are detached, or are in a duty or temporary duty status; Section 4 contains the total personnel account of the activity; Sections 5 to 8 contain statistical and authorized billet information, such as on board count ratings, NEC codes, distribution NECs, and projected losses and gains.

9. Future Work

Commencing with the 2004 fiscal year the SILS project will transition from a research program under the Office of Naval Research to an operational program under the Distance Support Program of NAVSEA, where it will be known as the Mission Readiness Assessment System Next Generation (MRAS-NG). The objective of the distance support effort is to develop tools that assist in reducing shipboard workloads, increasing readiness, and improving feedback and reach-back capabilities. The general focus of MRAS-NG under the Distance Support Program is to field a fully operational and sustainable decision-support system for individual ships.

The SILS proof-of-concept system with its integrated SILS IE and SILS MRAT subsystems demonstrated the viability of applying an agent technology to the problem of mission readiness assessment. However, within the current state-of-the-art of software science technology, and especially software agent technology, significant risks still exist in ensuring that this type of application can be designed to operate without degradations within the current constraints of shipboard network bandwidth, information security, and information assurance. In addition, significant risks also exist in the anticipated high costs associated with both the initial roll-out of the MRAS-NG software application to the Fleet and the life cycle costs to sustain it in the Fleet. The primary two areas of risk are in the design and recurrent maintenance of the system and context unique decision processes and data resources for each of the hundreds of systems that comprise the nine Ship Work Breakdown Structures (SWBS) for each of the 30 currently operational ship classes that constitute the Fleet.

In order to mitigate these risks, the SILS prototype system architecture must be re-designed to function within the shipboard network processing capacity, information security and assurance requirements, database systems interfaces, available systems mission capability knowledge, and mission decision processes that currently exist. Furthermore, this re-design of the prototype system architecture must also minimize the cost and lead-time associated with adding unsupported or new systems, database interfaces, mission capability knowledge, and mission decision processes as they are introduced into the Fleet. The fundamental new design requirement is that MRAS-NG must be designed as a ***software shell*** application package.

In the near term, the MRAS-NG development team will focus on the redesign of the architecture for compatibility with the new environment, the development of a set of Web browser-based user development and application tools, and fielding of an initial MRAS-NG version for sea trials. The MRAS-NG team will then incorporate additional data sources and assessment tools to progressively extend the information source domain, the analysis domain, and the automatic inferencing services. Government acceptance tests of the redesigned prototype are anticipated to begin in June 2004 with sea trials scheduled for September 2004. Complete implementation of MRAS-NG is planned in four phases spread over 60 months.

In the first phase, using feedback from NSWCCD and Distance Support (DS) program personnel, the MRAS-NG team will redefine and modify the existing MRAS architecture and implementation of a domain specific (rich client) application to that of a generalized software shell (thin client) application. The software shell concept is borrowed from the expert system shell concept, and refers to the ability to delete the specific application domain content from an expert system and utilize the remaining framework and inference engine for another application domain. This phase will culminate with a prototype demonstration and deliverable at a hosted design review.

During the second phase, the MRAS-NG team will continue to develop MRAS-NG using an iterative build and test process with frequent design and demonstration reviews. This effort will culminate in an MRAS-NG system that is limited to Government selected missions and equipment. The MRAS-NG software will demonstrate the maturity level required prior to deployment to the operational environment. It will go through Factory Acceptance Testing at ManTech facilities, Acceptance Testing at NSWC Crane facilities, and then through Certification Testing for verification and validation that the software is "ready" to install on a ship. Once the maturity level has been demonstrated, the system or incremental version will be base-lined, and a methodical and synchronized deployment plan will be implemented for the applicable locations.

The goal of this development phase will be to locate the appropriate data within the data available from the shipboard Distance Support Database and watch for changes in the SLQ32 radar system and the Allison 501-K34 Electrical Power Generation System. Changes to these systems will be collected by the Interface Engine (IE) and passed to the intelligent agents of the Mission Readiness Analysis Toolkit (MRAT). The agents will examine the data and decide if the equipment is operational or not. The results of this operation will be displayed to the user. This simple starting point will allow all parties to validate the results of the system as it expands. While the MRAS-NG system will be designed to help with managing the status of the ship, the results of the system will only be as good as the input data. One critical task that will have to be immediately undertaken is to gain a complete understanding of the available data sources. While the currently collected SORTS and Casualty Report (CASREP) data will be made available by the Distance Support program, it is not immediately clear that all of the required data are currently collected by the available external systems.

In the third phase, MRAS-NG will be installed and tested on a DDG-51 class ship. Appropriate ship personnel will be trained and given mechanisms for providing feedback on its operation and use. Development and refinement of MRAS-NG will continue, as will installations on additional ships.

In the fourth and succeeding phases, the Navy will have full responsibility for the acquisition and support of the MRAS-NG application. Additionally, enhancements to the functionality and capabilities of MRAS will continue.

10. Appendix A: Agent Rule Specifications

10.1 Framework Support Rules

10.1.1 Alert

Purpose: Contains rules and functions regarding alert creation and modification.

Alert Rule Set: ***Event Dependencies and Subsequent Actions***

Condition 1: An observation.GeneralObservation exists whose type is the target concept of a ruleBasedAgent.AlertType.
Action 1: A ruleBasedAgent.RuleBasedAlert is created and associated to the observation.GeneralObservation.

Condition 2: A ruleBasedAgent.AlertType exists without its default attribute types set.
Action 2: The ruleBasedAgent.AlertType's default attribute types are set.

Condition 3: A ruleBasedAgent.RuleBasedAlert exists whose targetObservation has a different endTime.
Action 3: The ruleBasedAgent.RuleBasedAlert's endTime is set to the targetObservation's endTime.

Condition 4: A ruleBasedAgent.RuleBasedAlert exists whose targetObservation is nil.
Action 4: The ruleBasedAgent.RuleBasedAlert is deleted.

Condition 5: A ruleBasedAgent.AlertType exists without its interestedAgentType association set.
Action 5: The ruleBasedAgent.AlertType's interestedAgentType association is set to its ruleBasedAgent.RuleSetType's agentType

Resulting Alerts: None

Alert Attribute Rule Set: ***Event Dependencies and Subsequent Actions***

Condition 1: An alert's message attribute does not have the correct value
Action 1: The message attribute is correctly set.

Condition 2: An alert exists with no alert attribute for a given alert attribute type
Action 2: An alert attribute object is created from the appropriate attribute of or association to the alert

Condition 3:	A ruleBasedAgent.AlertAttributeType exists without its interestedAgentType association set.
Action 3:	The ruleBasedAgent.AlertAttributeType's interestedAgentType association is set to its ruleBasedAgent.RuleSetType's agentType.

Resulting Alerts: None

10.1.2 Initialization

Purpose: Contains general rules and functions necessary for agent initialization.

Agent Status Rule Set: ***Event Dependencies and Subsequent Actions***

Condition 1:	A ruleBasedAgent.AgentStatus object does not exist for a framework.View object
Action 1:	A ruleBasedAgent.AgentStatus object is created for the given framework.View

.

Condition 2:	A ruleBasedAgent.PlanningTime object does not exist for a framework.View object.
Action 2:	A ruleBasedAgent.PlanningTime is created and associated to the given framework.View.
Condition 3:	ruleBasedAgent.AgentStatus exists with its requiredAgentsInstantiated attribute set to FALSE and agent initialization is complete.
Action 3:	The ruleBasedAgent.AgentStatus requiredAgentsInstantiated field is set to TRUE.

Resulting Alerts: None

Operating Entity Rule Set: ***Event Dependencies and Subsequent Actions***

Condition 1:	A system.GeneralSystem exists and there is no associated system.Domain for a framework.View.
Action 1:	A system.Domain object is created for the given system.GeneralSystem and framework.View.
Condition 2:	A ruleBasedAgent.RuleBasedAgentType exists in which its behaviorType attribute is set to required and there is no ruleBasedAgent.RuleBasedAgent instance for a framework.View.
Action 2:	A ruleBasedAgent.RuleBasedAgent instance is created for the given ruleBasedAgent.RuleBasedAgentType and framework.View.
Condition 3:	A ruleBasedAgent.RuleSetType exists in which its behaviorType attribute is set to required and there is

no ruleBasedAgent.RuleSet instance for a given framework.View.

Action 3: A ruleBasedAgent.RuleSet instance is created for the given ruleBasedAgent.RuleSetType and framework.View.

Resulting Alerts: None

10.1.3 Observation

Purpose: Contains rules regarding observation creation and modification.

Observation Rule Set: ***Event Dependencies and Subsequent Actions***

Condition 1: A general observation exists with %<type>% flags in its label

Action 1: Replaces each %<type>% flag with applicable primary of supporting subject label. Also sets the general observation's objectName to the new label.

Condition 2: A projected task exists whose task protocol is associated to a phenomenon with superType task protocol planned concept category

Action 2: An observation of phenomenon with superType task protocol planned concept category is created

Condition 3: An observation of phenomenon with superType task protocol planned concept category exists but corresponding task has been implemented

Action 3: The end time of the observation is set to the current planning time

Resulting Alerts: None

Collection Observation Rule Set: ***Event Dependencies and Subsequent Actions***

Condition 1: Observation that corresponds to a collection concept exists and no AndCollection ObservationFact has been created yet

Action 1: Creates a new AndCollection ObservationFact containing all of the subjects held within the observation

Condition 2: Observation that belongs in an AndCollectionObservationFact but has not yet been included exists

Action 2: Adds the newly found observation and its corresponding information to AndCollection ObservationFact

Condition 3:	An observation which is listed as lacked by a CollectionConcept exists and corresponds to an already created AndCollectionObservationFact
Action 3:	Adds the newly found lacked observation and its corresponding information to an AndCollectionObservationFact, and creates a new LackedFact
Condition 4:	A lacked observation is discovered to have been altered due to it's times being different than its corresponding LackedFact and it is contained within an AndCollectionObservationFact
Action 4:	This rule updates the lacked times contained in the AndCollectionObservationFact
Condition 5:	Collection observation found which is not associated to an AndCollectionObservationFact but should be
Action 5:	Collection observation is added to And CollectionObservationFact's collections at the correct position corresponding to it's start and end time
Condition 6:	AndCollectionObservationFact with correct number of observations contained exists
Action 6:	Creates collection observations around lacked observations or modifies ones that need to have their times changed
Condition 7:	Observation exists which relates to an or type collection concept
Action 7:	Creates a collection observation and associates it to it's observations
Condition 8:	Collection observation exists which is a supporting collection for an observation concept
Action 8:	Creates an observation and associates it to the collection observation
Condition 9:	A collection observation exists whose end time is greater than the earliest end time of its supporting observations
Action 9:	Sets the collection observation's end time to the earliest end time of its observations

Condition 10:	Observation exists whose start time does not equal its supporting collection start time
Action 10:	Sets the start time of the observation to the start time of its supporting collection
Condition 11:	Observation exists whose end time does not equal its supporting collection end time
Action 11:	Sets the end time of the observation to the end time of its supporting collection

Resulting Alerts: None

10.2 SILS MRAT Agent Rules

10.2.1 Combat Systems Agent

Purpose: Contains rules that allow SILS MRAT to monitor the health of the ship's combat systems.

Combat Systems Task Requirement: ***Event Dependencies and Subsequent Actions***

Condition 1:	task.Task to fix existing Combat Systems problem has all task.ResourceRequirements allocated and received
Action 1:	observation.Observation of task has all required assets allocated and received is created
Condition 2	task.Task to fix existing Combat Systems problem has status set to implemented or completed and observation.Observation of task has all required assets allocated and received is current.
Action 2:	observation.Observation of task has all required assets allocated and received has endTime set to the current planning time.

Resulting Alerts: None

Combat Systems Machinery Problem: ***Event Dependencies and Subsequent Actions:*** None

Resulting Alerts: <u>Alert 1:</u> **Machinery Problem – Warning**

Severity: Warning

Attributes:

1. Context Start Time
2. Context End Time
3. Message
4. Severity
5. Material Asset
6. Problem

<u>Alert 2:</u> **Machinery Problem – Violation**

Severity: Violation

Attributes:

1. Context Start Time
2. Context End Time
3. Message
4. Severity
5. Material Asset
6. Problem

10.2.2 HM&E Systems Agent

Purpose: Contains rules that allow SILS MRAT to monitor the health of the ship's hull, mechanical and electrical systems.

HM&E Task Requirement: ***Event Dependencies and Subsequent Actions***

Condition 1: task.Task to fix existing H,M & E problem has all required assets allocated and received.
Action 1: observation.Observation of task has all required assets allocated and received is created

Condition 2 task.Task to fix existing H, M & E problem has status set to implemented or completed and observation.Observation of task has all required assets allocated and received is current.
Action 2: observation.Observation of task has all required assets allocated and received has endTime set to the current planning time

Condition 3: A silsInfo.CASREP for an existing H, M & E problem has status set to isComplete.
Action 3: observation.Observation of CASREP repaired CASCOR should be issued is created.

Resulting Alerts: **<u>Alert 1:</u> Task Has All Required Assets Allocated and Received**

Severity: Information

Attributes:

1. Context Start Time
2. Context End Time
3. Message
4. Severity
5. Task

<u>Alert 2:</u> CAREP Repaired CASCOR should be issued

Severity: Information

Attributes:

1. Context Start Time
2. Context End Time
3. Message
4. Severity
5. CASREP

HM&E Machinery Problem: ***Event Dependencies and Subsequent Actions:*** None

Resulting Alerts: <u>Alert 1:</u> **Machinery Problem – Warning**

Severity: Warning

Attributes:

1. Context Start Time
2. Context End Time
3. Message
4. Severity
5. Material Asset
6. Problem

<u>Alert 2:</u> **Machinery Problem – Violation**

Severity: Violation

Attributes:

1. Context Start Time
2. Context End Time
3. Message
4. Material Asset
5. Problem

10.2.3 Interface Agent

Purpose: Contains rules that allow SILS MRAT to monitor the health of interfacing decision support and information systems and process incoming feeds from those systems.

Process Machinery Problem: ***Event Dependencies and Subsequent Actions***

Condition 1:	An external system and/or translator posts a silsInterface.MachineryProblem.
Action 1:	An observation.Observation with concept and primarySubject specified by the interface object is created.
Condition 2:	An external system and/or translator removes an existing silsInterface.MachineryProblem.
Action 2:	observation.Observation correlating to silsInterface.MachineryProblem has its endTime set to the current planning time.

Resulting Alerts: None

Process Asset Problem: ***Event Dependencies and Subsequent Actions***

Condition 1:	An external system and/or translator posts a silsInterface.ANOR.
Action 1:	A silsInfo.ANOR that mimics the silsInterface.ANOR is created.

Condition 2:	An external system and/or translator posts a silsInterface.CASREP
Action 2:	A silsInfo.CASREP that mimics the silsInterface.CASREP is created.
Condition 3:	An external system and/or translator posts a silsInterface.DegradedEquipment object.
Action 3:	A silsInfo.DegradedEquipment object that mimics the silsInterface.DegradedEquipment is created.
Condition 4:	Existing silsInterface.AssetProblem's endTime is modified.
Action 4:	Correlating silsInfo.AssetProblem has endTime set to the new value.
Condition 5:	Existing silsInterface.AssetProblem's repair status is modified.
Action 5:	Correlating silsInfo.AssetProblem has repair status set to the new value.

Resulting Alerts: None

Process Maintenance Activity: ***Event Dependencies and Subsequent Actions***

Condition 1:	An external system and/or translator posts a silsInterface.MaintenanceActivity.
Action 1:	A task.Task with protocol and target specified by silsInterface.MaintenanceActivity is created.
Condition 2:	An external system sets existing silsInterface.MaintenanceActivity's status to complete.
Action 2:	Correlating task.Task has status set to completed.

Resulting Alerts: None

Process Part Order: ***Event Dependencies and Subsequent Actions***

Condition 1:	An external system and/or translator posts a silsInterface.PartOrder.
Action 1:	A task.AllocatedAsset for the task.ResourceRequirement correlating to the silsInterface.PartOrder is created. An observation.Observation of required part scheduled to arrive is created.
Condition 2:	An external system and/or translator sets silsInterface.PartOrder's received flag to TRUE
Action 2:	Correlating task.AllocatedAsset's received flag is set to TRUE.

Resulting Alerts: None

Process Temporary Personnel Assignment: ***Event Dependencies and Subsequent Actions***

Condition 1: An external system and/or translator posts a silsInterface.TemporaryPersonnelAssignment object with an NEC correlating to a task.ResourceRequirement.

Action 1: A task.AllocatedAsset correlating to the silsInterface.TemporaryPersonnelAssignment is created. An observation.Observation of required person scheduled to arrive is created.

Condition 2: Existing silsInterface.TemporaryPersonnelAssignment has arrived attribute set to TRUE

Action 2: task.AllocatedAsset correlating to the silsInterface.TemporaryPersonnelAssignment has its received flag set to TRUE

Resulting Alerts: None

Process Personnel Gain And Loss: ***Event Dependencies and Subsequent Actions***

Condition 1: An external system and/or translator posts a silsInterface.PersonnelGainAndLoss object in which a person is lost.

Action 1: A silsInfo.PersonChange object is created with incumbentPerson set to the person who is leaving.

Condition 2: An external system and/or translator posts a silsInterface.PersonnelGainAndLoss object in which a person is gained.

Action 2: The silsInfo.PersonChange object has its replacementPerson association set to the sils.Person who is being gained.

Condition 3: An external system and/or translator posts a silsInterface.PersonnelGainAndLoss object in which a person is gained and is associated to a NEC.

Action 3: A humanAsset.PersonAbilityPeriod is created which ties the gained sils.Person to the NEC.

Condition 4: A silsInterface.PersonnelGainAndLoss object that specifies a loss has arrivalTime attribute modified.

Action 4: Correlating silsInfo.PersonChange object has its incumbentPerson's departureDate set to the new arrivalTime.

Condition 5:	A silsInterface.PersonnelGainAndLoss object that specifies a gain has arrivalTime attribute modified.
Action 5:	Correlating silsInfo.PersonChange object has its replacementPerson's arrivalTime set to the new arrivalTime.

Resulting Alerts: None

Process Personnel Training Request: ***Event Dependencies and Subsequent Actions***

Condition 1:	An external system and/or translator posts a silsInterface.PersonnelTrainingRequest object.
Action 1:	An observation.Observation of training scheduled is created.

Resulting Alerts: None

Process External System Down: ***Event Dependencies and Subsequent Actions***

Condition 1:	A system.ExternalSystemDown object exists.
Action 1:	An observation.Observation with concept of external system down's description and with primary subject of the external system down's identifier of the downed system is created.

Resulting Alerts: **Alert 1: External System Down**

Severity: Violation

Attributes:

1. Context Start Time
2. Context End Time
3. Message
4. Severity
5. External System

10.2.4 Mission Capability Agent

Purpose: Contains rules that allow SILS MRAT to identify high-level problems that affect the ship's overall ability to perform a mission

Readiness: ***Event Dependencies and Subsequent Actions***

Condition 1:	A readiness.ReadinessAreaType exists without any correlating readiness.ReadinessArea for a given framework.View..
Action 1:	A readiness.ReadinessArea is created and associated to the readiness.ReadinessAreaType and framework.View.
Condition 2:	A readiness.ReadinessArea exists without a readiness.ReadinessLevel for an existing readiness.ReadinessClass.

Action 2: The readiness.ReadinessArea's readiness.ReadinessLevel is set for the readiness.ReadinessClass to the readiness.ReadinessLevel with the highest rank.

Condition 3: A readiness.ReadinessObservation exists which has not been processed.
Action 3: The readiness.ReadinessArea the readiness.ReadinessObservation is on and all parent readiness.ReadinessAreas are updated to the lowest readiness.ReadinessLevel between the readiness.ReadinessArea's current level and the level specified by the readiness.ReadinessObservation.

Condition 4: A readiness.ReadinessArea exists whose readiness.ReadinessLevel is not correct based on its sub readiness.ReadinessArea readiness.ReadinessLevels and any readiness.ReadinessObservations currently on it.
Action 4: The readiness.ReadinessArea and all its parent readiness.ReadinessAreas are associated to their correct readiness.ReadinessLevels.

Condition 5: A readiness.ReadinessObservation's endTime has been set to a time less than or equal to the current planning time.
Action 5: The readiness.ReadinessArea the readiness.ReadinessObservation is on and all its parent readiness.ReadinessAreas have their readiness.ReadinessLevels set to the correct value not influenced by the out-of-date readiness.ReadinessObservation.

Condition 6: An observation.CollectionObservation with supported concept corresponding to a readiness.ReadinessConcept exists.
Action 6: A readiness.ReadinessObservation associated to the observation.CollectionObservation is created.

Condition 7: A readiness.ReadinessRequirement for an NEC exists but no person with that NEC is on the ship.
Action 7: A readiness.ReadinessObservation with type readiness.ReadinessRequirement is created.

Condition 8: A humanAsset.PersonAbilityPeriod for required NEC exists.
Action 8: readiness.ReadinessObservation endTime is set to the humanAsset.PersonAbilityPeriod startTime.

Condition 9: A readiness.ReadinessRequirement for a materialAsset.MaterialAssetType exists but no materialAsset.MaterialAsset of that type is on the ship.
Action 9: A readiness.ReadinessObservation with type readiness.ReadinessRequirement is created.

Condition 10: Required materialAsset.MaterialAsset is on the ship.
Action 10: readiness.ReadinessObservation endTime is set to the materialAsset.MaterialAsset activationDate.

Resulting Alerts: None

Air Operations: ***Event Dependencies and Subsequent Actions***

Condition 1: An observation.CollectionObservation has been created which infers an observation.Observation of recommended air operations suspension.
Action 1: An observation.Observation of recommended air operations suspension is created.

Condition 2: A silsInfo.Message object exists whose type is suspend air operations message type.
Action 2: An observation.Observation of air operations suspended is created.

Condition 3: A silsInfo.Message object whose type is suspend air operations message type has its endTime attribute changed.
Action 3: The observation.Observation correlating to the silsInfo.Message object has its endTime and applicableEndTime attributes set to the endTime of the silsInfo.Message.

Resulting Alerts: **Alert 1: Recommended Air Operations Suspension**

Severity: Warning
Attributes:

1. Context Start Time
2. Context End Time
3. Message
4. Severity
5. Problem

Alert 2: Air Operations Suspended

Severity: Violation
Attributes:
1. Context Start Time
2. Context End Time
3. Message
4. Severity

10.2.5 Personnel Agent

Purpose: Contains rules that allow SILS MRAT to monitor status of the manning status of the crew.

Personnel Task Requirement: ***Event Dependencies and Subsequent Actions***

Condition 1: A task.ResourceRequirement for a humanAsset.HumanAsset with no task.AssetRequest exists.
Action 1: A task.AssetRequest object corresponding to the task.ResourceRequirement is created.

Condition 2: A task.AssetRequest for a humanAsset.HumanAsset with no task.AllocatedAsset exists.
Action 2: An observation.Observation of resource requirement has no allocated asset is created.

Condition 3: A task.AllocatedAsset exists for a task.AssetRequest associated to a current observation.Observation of resource requirement has no allocated asset.
Action 3: observation.Observation's endTime and applicableEndTime are set to the current planning time.

Condition 4: A task.AssetRequest in which the available complete flag is set to FALSE exists.
Action 4: task.AvailableAsset objects are processed and associated to the task.AssetRequest if applicable. task.AssetRequest availableAssetsComplete flag is set to TRUE. If no task.AvailableAsset objects are located on ship then an observation.Observation of required person not on ship is created.

Condition 5: task.AllocatedAsset corresponding to required person in observation.Observation required person scheduled to arrive has received flag set to TRUE.
Action 5: observation.Observation of required person scheduled to arrive has endTime set to the current

planning time and an observation.Observation of required person arrived is created.

Condition 6: task.Task with task.ResourceRequirement of a humanAsset.PersonType has status set to implemented or completed.
Action 6: observation.Observation of required person arrived's endTime is set to current planning time.

Resulting Alerts: Alert 1: **Required Person Not On Ship**

Severity: Violation
Attributes:

1. Context Start Time
2. Context End Time
3. Message
4. Severity
5. NEC Type
6. Task

Alert 2: **Required Person Schedule To Arrive**

Severity: Warning
Attributes:

1. Context Start Time
2. Context End Time
3. Message
4. Severity
5. Person
6. Task
7. Expected Arrival Date

Alert 3: **Required Person Arrived**

Severity: Information
Attributes:

1. Context Start Time
2. Context End Time
3. Message
4. Severity
5. Person
6. Task

Alert 4: **Resource Requirement Has No Allocated Asset**

Severity: Violation
Attributes:

1. Context Start Time
2. Context End Time
3. Message
4. Severity
5. NEC Type
6. Quantity

7. Task

Personnel Loss: ***Event Dependencies and Subsequent Actions***

Condition 1:	A silsInfo.PersonChange object exists in which the incumbentPerson is the last person with a required NEC.
Action 1:	observation.Observation of losing all personnel with required NEC is created.
Condition 2:	A humanAsset.Person with the required NEC will be on the ship past the departureDate of the transferring person.
Action 2:	The observation.Observation of losing all personnel with required NEC has its endTime set to current planning time.

Resulting Alerts: **<u>Alert 1</u>: Losing All Certified Personnel**

Severity: Warning

Attributes:

1. Context Start Time
2. Context End Time
3. Message
4. Severity
5. NEC
6. Transfer Date

<u>Alert 2</u>: Crewman is Transferring

Severity: Information

Attributes:

1. Context Start Time
2. Context End Time
3. Message
4. Crewman
5. Transfer Date

Personnel Transport Scheduling: ***Event Dependencies and Subsequent Actions:*** None

Resulting Alerts: **<u>Alert 1</u>: Inefficient Use of Transport Resources**

Severity: Warning

Attributes:

1. Context Start Time
2. Context End Time
3. Message
4. Severity
5. Task 1
6. Task 1 Start Time
7. Task 1 End Time
8. Task 2
9. Task 2 Start Time

10. Task 2 End Time

10.2.6 Supply Agent

Purpose: Contains rules that allow SILS MRAT to monitor the supply status of the ship

Supply Task Requirement: ***Event Dependencies and Subsequent Actions***

Condition 1:	A task.ResourceRequirement for a materialAsset.MaterialAsset with no task.AssetRequest exists.
Action 1:	A task.AssetRequest object corresponding to the task.ResourceRequirement is created.
Condition 2:	A task.AssetRequest for a materialAsset.MaterialAsset with no task.AllocatedAsset exists.
Action 2:	An observation.Observation of resource requirement has no allocated asset is created.
Condition 3:	A task.AllocatedAsset exists for a task.AssetRequest associated to a current observation.Observation of resource requirement has no allocated asset.
Action 3:	observation.Observation's endTime and applicableEndTime are set to current planning time.
Condition 4	A task.AssetRequest in which the availableAssetsComplete flag is set to FALSE exists.
Action 4:	task.AvailableAsset objects are processed and associated to the task.AssetRequest if applicable. task.AssetRequest availableAssetsComplete flag is set to TRUE. If no task.AvailableAsset objects are located on the ship then observation.Observation of required person not on ship is created.
Condition 5:	task.AllocatedAsset corresponding to required material asset has received flag set to TRUE.
Action 5:	observation.Observation of Required part scheduled to arrive has endTime set to the current planning time and an observation.Observation of required asset arrived is created.
Condition 6:	task.Task with required material asset has status set to implemented or completed.
Action 6:	observation.Observation of required part arrived's endTime is set to the current planning time.

Resulting Alerts: <u>Alert 1</u>**: Required Part Not On Ship**

Severity: Violation

Attributes:

1. Context Start Time
2. Context End Time
3. Message
4. Severity
5. Part Type
6. Task

<u>Alert 2</u>**: Required Part Schedule To Arrive**

Severity: Warning

Attributes:

1. Context Start Time
2. Context End Time
3. Message
4. Severity
5. Part
6. Task
7. Expected Arrival Date

<u>Alert 3</u>**: Required Part Arrived**

Severity: Information

Attributes:

1. Context Start Time
2. Context End Time
3. Message
4. Severity
5. Part
6. Task

<u>Alert 4</u>**: Resource Requirement Has No Allocated Asset**

Severity: Violation

Attributes:

1. Context Start Time
2. Context End Time
3. Message
4. Severity
5. Material Asset Type
6. Quantity
7. Task

Supply Transport Scheduling: ***Event Dependencies and Subsequent Actions:*** None

Resulting Alerts: <u>Alert 1</u>**: Inefficient Use of Transport Resources**

Severity: Warning

Attributes:

1. Context Start Time
2. Context End Time

3. Message
4. Severity
5. Task 1
6. Task 1 Start Time
7. Task 1 End Time
8. Task 2
9. Task 2 Start Time
10. Task 2 End Time

10.2.7 Training and Performance Agent

Purpose: Contains rules that allow SILS MRAT to identify training and performance deficiencies.

Mandatory Officer Training: ***Event Dependencies and Subsequent Actions***

Condition 1: humanAsset.Person exists whose arrivalDate is after the startTime of a task.TrainingTask that has that humanAsset.Person as a trainee..

Action 1: An observation.Observation of officer will miss mandatory training is created.

Condition 2: observation.Observation exists of officer will miss mandatory training however the specified sils.Person has an expected arrivalDate before the task.TrainingTask startTime

Action 2: The endTime of the observation.Observation is set to the current planning time.

Resulting Alerts: <u>Alert 1</u>: **Officer Will Miss Mandatory Training**

Severity: Violation

Attributes:

1. Context Start Time
2. Context End Time
3. Message
4. Severity
5. Officer
6. Mandatory Training

<u>Alert 2</u>: **Training Scheduled**

Severity: Information

Attributes:

1. Context Start Time
2. Context End Time
3. Message
4. Severity
5. Officer
6. Start Time
7. End Time

11. Appendix B: Façade Specifications

The façades provide a very convenient way of providing an application with an application-specific view of the model. Façades interact with the logic layer and the model. They isolate the logic layer from changes in the model and essentially keep the knowledge of how to acquire data from complicating the logic. Following the specification of the façade API, the logic can be implemented. This improves parallel development.

The façades are separated into eight packages. The ***Agent Interface Facade*** package contains façades that provide interaction with the SILS MRAT agents. The ***Comment Façade*** package contains facades that provide information on comments made by users. The ***Department Facade*** package contains façades that provide functionality for viewing information pertaining to a specific ship department. The ***Launch Façade*** package contains the facades used during application startup. The ***Location Facade*** package contains façades that provide information about locations within the ship. The ***Message Façade*** package contains the facades pertaining to issues entered within SILS MRAT. The ***Readiness Facade*** package contains façades that provide the user with information about readiness, and the ***Scheduling Facade*** package contains façades that provide interaction with the ship's scheduled tasks.

11.1 Agent Facades

11.1.1 Agent Façade - ***Provides information about an agent.***

agentId Field
Provides the name of the agent.

animate Field
Provides the length of time that the agent should animate after this animate field is set.

isActive Field
Provides whether the agent is active and operating.

status Field
Provides the numeric state value of the worst alert currently registered by the agent.

name Field
Provides the displayable full name of the agent.

targets Association
Provides a link to facades representing all objects that are targeted by an alert from this agent. The facades are guaranteed to have a displayable name field, and an alerts association.

alerts Association
Provides Alert façades for all alerts that were created by this agent.

comments Association
Provides Comment façade for all comments targeting this agent.

issues Association
Provides an Issue façade for all issues targeting this agent.

11.1.2 Agents Façade - *Provides access to all agents in the system.*

agents Association
Provides all agents in the system as Agent façade objects.

11.1.3 Alert Façade - *Provides access to alerts posted by an agent on one or more objects.*

ack Field
Provides whether the alert has been acknowledged, and allows a user to acknowledge an alert by setting this field to TRUE.
alertType Field
Provides the name of the rule that found this alert.
attributeLabels and attributeValues Fields
Provides two parallel tab-delimited lists containing alert attributes and their corresponding values. These are all the alert attributes for this alert.
message Field
Provides the message created by an agent describing this alert.
ruleSet Field
Provides the rule set for this alert.
severity Field
Provides the severity of this alert as a number (0-6, 6 being the worst).
time Field
Provides the time that the alert was created.
name Field
Provides the name of the alert.
sourceObjectKey Field
Entered by the creator of this object, this is the objectKey of the source object used to derive this façade's fields.
alerts Association
Provides a link to all alerts that have been assigned to the source object.
comments Association
Provides a link to all comments that have been made on the source object.
issues Association
Provides a link to all issues targeting the source object.

11.1.4 Observation Façade - *Provides information concerning an observation.*

name Field
Provides the name of the observation.
sourceObjectKey Field
Entered by the creator of this object, this is the objectKey of the source object used to derive this façade's fields.
alerts Association
Provides a link to all alerts that have been assigned to the source object.
comments Association
Provides a link to all comments that have been made on the source object.

issues Association

Provides a link to all issues targeting the source object.

11.1.5 TargetFacade Façade - *Provides a representation of an alert's target not already represented by another façade.*

name Field

Provides a displayable name representing the source object.

sourceObjectKey Field

Entered by the creator of this object, this is the objectKey of the source object used to derive this façade's fields.

alerts Association

Provides a link to all alerts that have been assigned to the source object.

comments Association

Provides a link to all comments that have been made on the source object.

issues Association

Provides a link to all issues targeting the source object.

11.2 Comment Facades

11.2.1 Comment Façade - *Provides a comment created by a user upon an object represented by one or more façades.*

comments Association

Provides comments upon this comment.

message Field

Provides the contents of the message contained within this comment.

timePosted Field

Provides the time when the user created this comment.

user Field

Provides the full name of the user that posted this comment. An example of this field would be "Zachary Speck".

userName Field

Provides the login name of the user that posted this comment. An example of this field would be "zspeck".

11.2.2 NewComment Façade - *Allows a new comment to be created on the object whose objectKey is targetObjectKey.*

message Field

Entered by the user, this is the message of the comment.

targetObjectKey Field

Entered by the user, this is the objectKey of the target of this comment.

createFacade Field

Set by the user to create the new comment.

createFacadeFailed Field
Returned by the façade following createFacade being set to true and being posted. If the createFacadeFailed is set to true then the problemFacadeFields field should be checked.

checkFacadeFields Field
Set by the user before createFacade to check if there are problems with any of the façade fields.

problemFacadeFields Field
Returned by the façade following createFacade or checkFacadeFields being set to true. This will be empty if there were no problem fields in the façade.

11.3 Department Facades

11.3.1 AnchoringLog Façade - *Provides information about current anchoring log.*

anchor Field
Provides information on which anchor was dropped.

bottomType Field
Provides information on the ocean floor where the anchor was dropped.

depth Field
Provides information on the distance to the ocean floor.

scope Field
Provides information on the number of shots required.

time Field
Provides information on the time at which the anchor was dropped.

sourceObjectKey Field
Entered by the creator of this object, this is the objectKey of the source object used to derive this façade's fields.

alerts Association
Provides a link to all alerts that have been assigned to the source object.

comments Association
Provides a link to all comments that have been made on the source object.

issues Association
Provides a link to all issues created by users with this session.

11.3.2 ANOR Façade - *Provides information on a specific ANOR.*

daysToFailure Field
Provides information on the number of days expected until the item fails.

nomenclature Field
Provides information on the nature of the problem.

reported Field
Provides the time at which this ANOR was reported

requisitionNumber Field
Provides the requisition number for this ANOR.

status Field
Provides the current status of the ANOR
sourceObjectKey Field
Entered by the creator of this object, this is the objectKey of the source object used to derive this façade's fields.
alerts Association
Provides a link to all alerts that have been assigned to the source object.
comments Association
Provides a link to all comments that have been made on the source object.
issues Association
Provides a link to all issues created by users with this session.

11.3.3 CASREP Façade - *Provides information about a specific CASREP.*

casrepNumber Field
Provides the casrep's number.
daysToFailure Field
Provides information on the number of days expected until the item fails.
nomenclature Field
Provides information on the nature of the problem.
reported Field
Provides the time at which this CASREP was reported
requisitionNumber Field
Provides the requisition number for this CASREP.
status Field
Provides the current status of the CASREP
sourceObjectKey Field
Entered by the creator of this object, this is the objectKey of the source object used to derive this façade's fields.
alerts Association
Provides a link to all alerts that have been assigned to the source object.
comments Association
Provides a link to all comments that have been made on the source object.
issues Association
Provides a link to all issues created by users with this session.

11.3.4 DegradedEquipment Façade - *Provides information about a specific Degraded Equipment.*

estimatedTimeUntilRepair Field
Provides the estimated amount of time until the item is repaired.
jsn Field
Provides the job serial number of the degraded equipment.
nomenclature Field
Provides information on the nature of the problem.
reported Field
Provides the time at which this CASREP was reported

requisitionNumber Field
Provides the requisition number for this CASREP.
status Field
Provides the current status of the CASREP
sourceObjectKey Field
Entered by the creator of this object, this is the objectKey of the source object used to derive this façade's fields.
alerts Association
Provides a link to all alerts that have been assigned to the source object.
comments Association
Provides a link to all comments that have been made on the source object.
issues Association
Provides a link to all issues created by users with this session.

Department Façade

Provides information about a specific Department in the system.
anchoringLog Association
Provides access to all current anchoring log reports for this department.
anors Association
Provides access to all ANORs in this department.
casreps Association
Provides access to all CASREPs in this department.
degradedEquipment Association
Provides access to all the degraded equipment reports in this department.
messages Association
Provides access to all messages affecting this department.
personnel Association
Provides access to all the personnel staffing this department.
sourceObjectKey Field
Entered by the creator of this object, this is the objectKey of the source object used to derive this façade's fields.
alerts Association
Provides a link to all alerts that have been assigned to the source object.
comments Association
Provides a link to all comments that have been made on the source object.
issues Association
Provides a link to all issues created by users with this session.

11.3.5 Departments Façade - *Provides access to all the departments in the system.*

members Association
Provides access department facades, each representing a specific department.
currentUsersDepartment Field
Provides information on the department the currently logged in user belongs to.
currentShip Field
Provides information on the current ship.

11.3.6 Message Façade - *Provides information on a message posted into a department.*

description Field
Provides a description of the message type.

dtg Field
Provides information on the time at which the message was posted.

message Field
Provides the text of the message.

nextDue Field
Provides the time at which the next message is due.

serialNumber Field
Provides the message's unique serial number identification.

sourceObjectKey Field
Entered by the creator of this object, this is the objectKey of the source object used to derive this façade's fields.

alerts Association
Provides a link to all alerts that have been assigned to the source object.

comments Association
Provides a link to all comments that have been made on the source object.

issues Association
Provides a link to all issues created by users with this session.

Personnel Façade
Provides information on a specific person in a department.

comingOnBoard Field
Provides information on the expected arrival date of the person.

incumbant
Provides information on the incumbant person this person is replacing.

nec Field
Provides a tab-delimited string of this person's nec values.

rate Field
Provides information on the rate of this person.

required Field
Provides information on whether or not this person is required for deployment.

sourceObjectKey Field
Entered by the creator of this object, this is the objectKey of the source object used to derive this façade's fields.

alerts Association
Provides a link to all alerts that have been assigned to the source object.

comments Association
Provides a link to all comments that have been made on the source object.

issues Association
Provides a link to all issues created by users with this session.

11.4 Launch Facades

11.4.1 About Façade - *Provides general information about the application.*

applicationDescription Field
Provides a general description of the application, generally the full name behind the acronym.
applicationName Field
Provides the displayable name of the application, typically an acronym.
companyName Field
Provides the name of the company that implemented the application.
contract Field
Provides the displayable name of the contract for which the application was developed.
contributor and contributorDescription Fields
Provides two parallel tab-delimited fields that contain the name of each contributor to the project and their position.
copyright Field
Provides the copyright information protecting the application.
sponsor Field
Provides the sponsor of the application's development.
version Field
Provides the version of the application.

11.4.2 Domain Façade - *Represents a system.Domain object and is used to constrain its members to only those that apply to the this domain.*

agents Association
Association of agentfacade.Agent facades constrained to only those agents which apply to this domain.
ExistingCycles Façade
Provides all existing cycles.
members Association
Provides all existing sessions as existingCycle façades.
ExistingCycle Façade
Provides information about an existing cycle.
endTime Field
Provides the time in milliseconds since the epoch (January 1, 1970, 00:00:00 GMT) when the cycle will end.
name Field
Provides the name of this cycle.
ship Field
Provides the name of the ship used in this cycle.
startTime Field
Provides the time when the cycle began (in milliseconds since the epoch).

sourceObjectKey Field

Entered by the creator of this object, this is the objectKey of the source object used to derive this façade's fields.

alerts Association

Provides a link to all alerts that have been assigned to the source object.

comments Association

Provides a link to all comments that have been made on the source object.

issues Association

Provides a link to all issues created by users with this session.

11.4.3 ExistingCycleValidate Façade - *Provides a mechanism for the user to select a cycle to use for this session.*

cycleObjectKey Field

Entered by the user, this is the sourceObjectKey of the existingCycle façade to be used by the user for this session.

valid Field

Returned by the façade, indicates whether the cycleObjectKey is a valid cycle to be used for this session.

Login Façade

Provides a mechanism for a user to login to SILS MRAT.

loginName Field

Entered by the user, this is the login name of the user, such as "zspeck".

password Field

Entered by the user, this is the user's encrypted password.

valid Field

Returned by the façade, this indicates whether the login name is valid and the password is valid for the login name.

11.4.4 Logout Façade - *Provides a mechanism for a user to logout of SILS MRAT.*

logout Field

Set to true when the user wishes to logout of SILS MRAT.

11.4.5 NewCycle Façade - *Allows a new cycle to be created by a user.*

create Field

Set by the user to create a new cycle with this façade's values.

endTime Field

Entered by the user, this is the end time for the cycle in milliseconds since the epoch.

existingCycleNames Field

Provides a tab-delimited list of all currently existing cycle names.

name Field

Entered by the user, this is the name of the new cycle.

newCycleObjectKey Field

Provides the objectKey of the new cycle after the cycle has been created.

ship Field
Entered by the user, this is the name of the selected ship as provided in the ships field.
ships Field
Provides the names of ships this cycle can be based on.
startTime Field
Entered by the user, this is the start time for the cycle in milliseconds since the epoch.
type Field
Entered by the user, this is the selected cycle type name.
types Field
Provides a tab-delimited list of names for all cycle types. This list corresponds to types.
typeDescriptions Field
Provides a tab-delimited list of descriptions for all cycle types. This list corresponds to types.

11.4.6 System Façade - *Maintains the state of the façades and allows the façades to be shutdown.*

shutdown Field
Set by the user, this will shutdown the façade system services. This should be called before closing the application.
user Field
Provides the login name of the user currently logged into the system.
session Association
Provides the source object of the currently selected session.

11.5 Location Facades

11.5.1 CompleteLocation Façade - *Provides information regarding a location.*

alertLevel Field
Provides the numeric severity level of the worst alert on any objects located at this location.
alertLevelName Field
Provides the displayable name of the severity level of the worst alert on any objects located at this location.
hasCasreps Field
Provides whether this location has CASREPs on any objects located at this location.
locationId Field
Provides the location identifier of this location as found in CDMDFJ3D drawings of the ship.
name Field
Provides the displayable name of this location.

casreps Association
Provides CASREP façades for all the CASREPs on any objects located at this location.
sourceObjectKey Field
Entered by the creator of this object, this is the objectKey of the source object used to derive this façade's fields.
alerts Association
Provides a link to all alerts that have been assigned to the source object.
comments Association
Provides a link to all comments that have been made on the source object.
issues Association
Provides a link to all issues targeting the source object.

11.5.2 CompleteLocations Façade - *Provides a link to all locations represented by CompleteLocation façades.*

completeLocations Association
Provides a link to CompleteLocation façades representing all locations.

11.6 Message Facades

11.6.1 Issue Façade - *Provides information concerning the issue with the objectKey entered in sourceObjectKey.*

acknowledged Field
Provides TRUE if the issue has been acknowledged and FALSE if the issue has not been acknowledged.
affectedUsers Field
Provides a list of users affected by this issue.
message Field
Provides the message of this issue as entered by this issue's creator.
resolutionMessage Field
Provides the message entered by the party that resolved this issue, or N/A if this issue has not been resolved.
resolutionStatus Field
Provides the status of this message, whether or not it has been resolved. Values are "Resolved" and "Unresolved".
resolutionTime Field
Provides the time when the issue was resolved.
resolutionUser Field
Provides the login name of the user who resolved this issue, such as "jdoe".
resolutionUserName Field
Provides the full name of the user who resolved this issue, such as "John Doe".
timePosted Field
Provides the time when the issue was created.
user Field
Provides the login name of the user who created this issue.

userName Field
Provides the full name of the user who created this issue.
sourceObjectKey Field
Entered by the creator of this object, this is the objectKey of the source object used to derive this façade's fields.
alerts Association
Provides a link to all alerts that have been assigned to the source object.
comments Association
Provides a link to all comments that have been made on the source object.
issues Association
Provides a link to all issues targeting the source object.

11.6.2 Issues Façade - *Provides all the issues for the selected cycle that are created or targeted to the current user.*

createdIssues Association
Provides all issues created by the user.
targetedIssues Association
Provides all issues targeted to the user.
issues Association
Provides all issues created or targeted to the user.

11.6.3 NewIssue Façade - *Allows a new issue to be created.*

availableRespondents Field
Provides a tab-delimited list of names for all users available to respond to a new issue.
startTime Field
Not used.
endTime Field
Not used.
message Field
Entered by the user, this is the message of the issue.
primarySubject Field
Entered by the user, this is the primarySubject of the issue. This field is optional.
selectedRespondents Field
Entered by the user, this is a tab delimited list of users who can respond to the issue.
status Field
Entered by the user, this is the status of the new Issue.
createFacade Field
Set by the user, this creates the new issue when set to true.
createFacadeFailed Field
Returned by the façade following createFacade being set to true and being posted. If the createFacadeFailed is set to true then the problemFacadeFields field should be checked.

checkFacadeFields Field
Set by the user before createFacade to check if there are problems with any of the façade fields.
problemFacadeFields Field
Returned by the façade following createFacade or checkFacadeFields being set to true. This will be empty if there were no problem fields in the façade.

11.7 Readiness Facades

11.7.1 AffectedAreas Façade - *Groups all areas of the ship that affect the selected readiness element.*

name Field
Not used.
hme Association
Link to HMEArea façades representing all HME areas affected in the source object or it's sub-areas or observations.
personnel Association
Link to JobPersonnel façades representing all personnel allocations for the source object or it's sub-areas or observations.
supplies Association
Link to Supply façades representing all supply allocations for the source object or it's sub-areas or observations.
training Association
Link to Training façades representing all training tasks being performed for the source object or it's sub-areas or observations.
sourceObjectKey Field
Entered by the creator of this object, this is the objectKey of the source object used to derive this façade's fields.
alerts Association
Provides a link to all alerts that have been assigned to the source object.
comments Association
Provides a link to all comments that have been made on the source object.
issues Association
Provides a link to all issues targeting the source object.

11.7.2 HMEArea Façade - *Represents an HME system.*

location Field
Provides the physical location of the HME Area.
name Field
Provides a displayable name for this HME Area.
dependentSystems Field
Provides a link to other HME Area facades representing systems that depend upon this HME Area.
sourceObjectKey Field
Entered by the creator of this object, this is the objectKey of the source object used to derive this façade's fields.

alerts Association
Provides a link to all alerts that have been assigned to the source object.
comments Association
Provides a link to all comments that have been made on the source object.
issues Association
Provides a link to all issues targeting the source object.

11.7.3 InactionOption Façade - *Represents the option of not taking action to resolve a readiness problem.*

description Field
Provides a description of the option of inaction.
rating Field
Provides a rating of how favorable the inaction option is.
name Field
Provides a label for the inaction option.
impacts Association
Provides a link to ResolutionImpact facades representing all the impacts associated with this option.
sourceObjectKey Field
Entered by the creator of this object, this is the objectKey of the source object used to derive this façade's fields.
alerts Association
Provides a link to all alerts that have been assigned to the source object.
comments Association
Provides a link to all comments that have been made on the source object.
issues Association
Provides a link to all issues targeting the source object.

11.7.4 JobPersonnel Façade - *Represents a Job or NEC that needs to be filled.*

department Field
Provides the department of the needed JobPersonnel if applicable.
division Field
Provides the division of the needed JobPersonnel if applicable.
job Field
Provides the position of the needed JobPersonnel if applicable.
personnelOnHand Field
Provides the number of personnel on the ship that meet the criteria.
personnelPresent Field
Provides the number of personnel on the ship that meet the criteria.
personnelRequired Field
Provides the number of personnel required for the task or requirement.
personnelSpecified Field
Provides the number of personnel specified for the task or requirement.
presentRequiredSpecified Field
Provides the number personnel present, required, and specified for the task or requirement.

workcenter Field
Provides the workcenter of the needed JobPersonnel if applicable.
name Field
Provides the name of the position or NEC needed.
specificPersonnel Association
Provides a link to Person facades that fill this personnel need.
sourceObjectKey Field
Entered by the creator of this object, this is the objectKey of the source object used to derive this façade's fields.
alerts Association
Provides a link to all alerts that have been assigned to the source object.
comments Association
Provides a link to all comments that have been made on the source object.
issues Association
Provides a link to all issues targeting the source object.

11.7.5 Person Façade - *Represents a specific person filling an NEC or a Job.*

certificationStatus Field
Provides the status of the certifications held by this person, with regards to whether those certifications are current. Possible values are: "all", "none", "N/A", and "Some - ". "All" implies that all certifications are current. If only some certifications are current, certifications will be listed as "Some - " followed by a comma delimited list of the persons current certifications. None implies the person has certifications but none are current. N/A implies that the person has no certifications.
department Field
Provides the name of this person's department.
division Field
Provides the name of this person's division.
dueInOutStatus Field
Provides the status of whether this person is due in or out. "In" implies they are not on the ship. "Out" implies they are on the ship.
dueInOutTime Field
Provides the time that the person is due in or out. In implies they are not on the ship. Out implies they are on the ship.
dueInOutTimeAndStatus Field
Provides a single field combining the information in the dueInOutTime field and the dueInOutStatus field.
job Field
Provides the name of the person's position.
missionCritical Field
A comma delimited list of the missions for which this person is mission critical. If this person is not aboard, these missions cannot occur.
nec Field
Provides a tab delimited list of the NECs held by this person.

rate Field
Provides the person’s rate.
workcenter Field
Provides the workcenter to which the person belongs.
name Field
Provides the person’s name.
sourceObjectKey Field
Entered by the creator of this object, this is the objectKey of the source object used to derive this façade’s fields.
alerts Association
Provides a link to all alerts that have been assigned to the source object.
comments Association
Provides a link to all comments that have been made on the source object.
issues Association
Provides a link to all issues targeting the source object.

11.7.6 ReadinessArea Façade - *Represents a readiness view of a ship system.*

selected Field
Set by the user to select this ReadinessArea for use as the source object of the affected areas façade.
status Field
Displayable readiness status of this ReadinessArea.
statusRank Field
Numeric readiness status of this ReadinessArea.
name Field
Full name of this ReadinessArea.
allReadinessObservations Association
All ReadinessObservations that are sub-areas of this ReadinessArea.
problemReadinessObservations Association
All ReadinessObservations that are sub-areas of this ReadinessArea that have a readiness problem.
allSubReadinessAreas Association
All ReadinessAreas that are sub-areas of this ReadinessArea.
problemSubReadinessAreas Association
All ReadinessAreas that are sub-areas of this ReadinessArea that have a readiness problem.
sourceObjectKey Field
Entered by the creator of this object, this is the objectKey of the source object used to derive this façade’s fields.
alerts Association
Provides a link to all alerts that have been assigned to the source object.
comments Association
Provides a link to all comments that have been made on the source object.
issues Association
Provides a link to all issues targeting the source object.

11.7.7 ReadinessBias Façade - *Provides a representation of a mission biased readiness. Provides information about that mission bias and its readiness.*

abbreviation Field
Provides an abbreviation to represent this ReadinessBias.
exclusive Field
Provides whether this bias can only be exclusively selected.
status Field
Provides a displayable readiness status of this ReadinessBias.
statusRank Field
Provides a numeric readiness status of this ReadinessBias.
name Field
Provides a name for this ReadinessBias.
sourceObjectKey Field
Entered by the creator of this object, this is the objectKey of the source object used to derive this façade's fields.
alerts Association
Provides a link to all alerts that have been assigned to the source object.
comments Association
Provides a link to all comments that have been made on the source object.
issues Association
Provides a link to all issues targeting the source object.

11.7.8 ReadinessObservation Façade - *Represents an observation of a readiness problem.*

selected Field
Set by the user to select this ReadinessObservation for use as the source object of the affected areas façade.
status Field
Displayable readiness status of this ReadinessObservation.
statusRank Field
Numeric readiness status of this ReadinessObservation.
name Field
Full name of this ReadinessObservation.
resolutionOptions Association
ResolutionOption facades for all options to resolve this readiness observation.
sourceObjectKey Field
Entered by the creator of this object, this is the objectKey of the source object used to derive this façade's fields.
alerts Association
Provides a link to all alerts that have been assigned to the source object.
comments Association
Provides a link to all comments that have been made on the source object.
issues Association
Provides a link to all issues targeting the source object.

11.7.9 ReadinessSystem Façade - *Represents a readiness view of a system of the ship.*

abbreviation Field
Provides an abbreviation to represent this ReadinessSystem.
selected Field
Set by the user to select this ReadinessSystem for use as the source object of the affected areas façade.
status Field
Provides a displayable readiness status of this ReadinessSystem.
statusRank Field
Provides a numeric readiness status of this ReadinessSystem.
name Field
Full name of this ReadinessSystem.
allAreas Association
All ReadinessAreas that are sub-areas of this ReadinessSystem.
problemAreas Association
All ReadinessAreas that are sub-areas of this ReadinessSystem that have a readiness problem.
sourceObjectKey Field
Entered by the creator of this object, this is the objectKey of the source object used to derive this façade's fields.
alerts Association
Provides a link to all alerts that have been assigned to the source object.
comments Association
Provides a link to all comments that have been made on the source object.
issues Association
Provides a link to all issues targeting the source object.

11.7.10 ResolutionImpact Façade - *Represents the impact of implementing a resolution option or the impact of inaction.*

description Field
Provides a description of this impact.
rating Field
Provides a rating of how favorable this impact is.
name Field
Provides a label for this impact.
sourceObjectKey Field
Entered by the creator of this object, this is the objectKey of the source object used to derive this façade's fields.
alerts Association
Provides a link to all alerts that have been assigned to the source object.
comments Association
Provides a link to all comments that have been made on the source object.
issues Association
Provides a link to all issues targeting the source object.

11.7.11 ResolutionOption Façade - *Represents an option to resolve a readiness problem.*

description Field
Provides a description of the option.
rating Field
Provides a rating of how favorable this option is.
name Field
Provides a label for the resolution option.
impacts Association
Provides a link to ResolutionImpact facades representing all the impacts associated with this option.
sourceObjectKey Field
Entered by the creator of this object, this is the objectKey of the source object used to derive this façade's fields.
alerts Association
Provides a link to all alerts that have been assigned to the source object.
comments Association
Provides a link to all comments that have been made on the source object.
issues Association
Provides a link to all issues targeting the source object.

11.7.12 SilsOverview Façade - *Provides a representation of the top-most level of readiness.*

selectAllReadinessAreas Field
Entered as true by the user to select all readinessAreas.
availableBiases Association
Provides ReadinessBias facades for all available mission biases.
availableSystems Association
Provides ReadinessSystem facades for all available readiness systems.
selectedBiases Association
Associated by the user to select biases to use in biasing the readiness levels of the ReadinessSystems, the ReadinessAreas, and the ReadinessObservations.
selectedSystems Association
Associated by the user to select ReadinessSystems for viewing.

Supply Façade

Represents a needed supply.
daysToFailure Field
Provides the days until the asset fails.
estimatedCost Field
Provides the estimated cost of purchasing the repair item.
estimatedRepairTime Field
Provides the estimated repair time of the item, including the estimated time required to receive the part from its current location.
jsn Field
Provides the job serial number of the supply.

neededFor Field
Provides the name of the task or requirement for which the supply is needed.

ordered Field
Provides whether the supply has been ordered.

quantityAvailable Field
Provides the number of items that are currently available for use.

quantityNeeded Field
Provides the quantity of this supply that is needed.

received Field
Provides whether the supply has been received.

repairCapability Field
Provides the ability of the ship to repair the item. The possible values are: “Unknown”,"Internal", "BattleGroup", and "External"

replacementLocation Field
Provides the location of the supply or equipment item in question. If the item is aboard ship, then the text "OB-" plus the location on the ship is displayed. If the item is in the Battle Group, then the text "BG-" plus the name of the ship on which the item is located is displayed. If the item is on shore, then the name of the warehouse or distributing location which has the item is displayed (i.e. DLA Tracy).

requisitionNumber Field
Provides the requisition number of the ordered item.

name Field
Provides the name of the supply.

sourceObjectKey Field
Entered by the creator of this object, this is the objectKey of the source object used to derive this façade’s fields.

alerts Association
Provides a link to all alerts that have been assigned to the source object.

comments Association
Provides a link to all comments that have been made on the source object.

issues Association
Provides a link to all issues targeting the source object.

11.7.13 Training Façade - *Represents training needs.*

hoursRemaining Field
Provides the hours remaining to complete this training.

hoursRequired Field
Provides the hours required to complete this training.

hoursSpent Field
Provides the hours spent to complete this training.

hoursSpentRemainingRequired Field
Provides the hours spent, remaining, and required to complete this training.

phase Field
Provides the phase during which the training is performed.

name Field
Provides the name of this training.
sourceObjectKey Field
Entered by the creator of this object, this is the objectKey of the source object used to derive this façade's fields.
alerts Association
Provides a link to all alerts that have been assigned to the source object.
comments Association
Provides a link to all comments that have been made on the source object.
issues Association
Provides a link to all issues targeting the source object.

11.8 Scheduling Facades

11.8.1 NewSchedulingTask Façade - *Allows a user to create a new scheduling task.*

creator Field
The creator of this task.
dependeeObjectKeys Field
These are the tasks that this task depends upon.
dependentObjectKeys Field
These are the tasks that depend upon this task to be completed before they begin.
endTime Field
The time at which this task ends.
identifier Field
Unique identifier used to identify this task.
label Field
Identifier used for display on the screen.
missionImportance Field
A tab-delimited String where the first 'element' is the mission name, and the second 'element' is the rank, repeating for each mission and rank.
name Field
The name of the task.
parentTaskObjectKey Field
The objectKey for the parent task of this newly created task.
percentComplete Field
The percentage of this task that has been completed.
responsibleParty Field
Not used.
startTime Field
The time at which this task is scheduled to begin.
subTaskObjectKeys Field
Tab-delimited String of objectKeys for tasks which will be sub-tasks of this new task to be created.

targetObjectKey Field
objectKey of the target of this task.
taskTypeObjectKey Field
objectKey of the type of task that this new task will belong to.

11.8.2 OpportunityCost Façade - *Provides information on an opportunity cost associated with performing a task.*

description Field
Provides a description of the opportunity cost.
sourceObjectKey Field
Entered by the creator of this object, this is the objectKey of the source object used to derive this façade's fields.
alerts Association
Provides a link to all alerts that have been assigned to the source object.
comments Association
Provides a link to all comments that have been made on the source object.
issues Association
Provides a link to all issues created by users with this session.

11.8.3 ResourceRequirement Façade - *Provides information on a resource required to perform a task.*

description Field
Provides a description of the resource requirement.
quantity Field
Provides information on the quantity of the resource that is required.
quantityUnit Field
Provides information on the unit type for the quantity that is required.
resource Field
Provides information on the resource that is required by the task.
resourceArea Field
Provides information on the resource area of the resource requirement. An example would be 'personnel'.
sourceObjectKey Field
Entered by the creator of this object, this is the objectKey of the source object used to derive this façade's fields.
alerts Association
Provides a link to all alerts that have been assigned to the source object.
comments Association
Provides a link to all comments that have been made on the source object.
issues Association
Provides a link to all issues created by users with this session.

11.8.4 Schedule Façade - *Provides access to all the scheduled tasks, effectively the ship's schedule.*

tasks Association
Provides access to the tasks assigned to this schedule.

SchedulingTask Façade

Provides information on a specific task assigned to a schedule.

department Field

Provides information on which department this task is assigned to.

dependees Association

Provides access to the tasks upon which this task depends.

dependents Association

Provides access to the tasks which depend on this task for completion

endTime Field

Provides the time at which this task will end.

equipmentRequirements Field

Provides information on the requirements for the equipment aspect of this task.

equipmentToComplete Field

Provides information about whether the equipment needed to complete this task exists.

extEquipmentToComplete Field

Provides information about whether the external equipment needed to complete this task exists.

extMoneyToComplete Field

Provides information about whether the necessary external money exists to complete the task.

extPersonnelToComplete Field

Provides information about whether the necessary external personnel exist to complete this task.

extSupplyToComplete Field

Provides information about whether the necessary external supplies exist to complete this task.

moneyRequirements Field

Provides information on the money requirements necessary for this task.

moneyToComplete Field

Provides information about whether the money necessary to complete this task exists.

parentTask Association

Provides access to the parent task of this task.

percentComplete Field

Provides information about the current percentage of completion of this task.

personnelRequirements Field

Provides information on the requirements for the personnel aspect of this task.

personnelToComplete Field

Provides information about whether the personnel required to complete this task exist.

startTime Field

Provides information on the time at which this task will begin.

subTasks Association

Provides access to the sub-tasks of this task.

supplyRequirements Field
Provides information on the requirements needed for the supply aspect of this task.
supplyToComplete Field
Provides information about whether the supplies necessary to complete this task exist.
missionImportance Field
Provides information on the mission importance of this task. (Primary, Secondary, Support).
missionImportanceRank Field
Provides a numeric representation of the mission importance of this task. (1, 2, 3).
sourceObjectKey Field
Entered by the creator of this object, this is the objectKey of the source object used to derive this façade's fields.
alerts Association
Provides a link to all alerts that have been assigned to the source object.
comments Association
Provides a link to all comments that have been made on the source object.
issues Association
Provides a link to all issues created by users with this session.

11.8.5 TaskType Façade - ***Provides information on the type of a specific task.***

description Field
Provides a description of the task type.
opportunityCosts Association
Provides access to the opportunity costs associated with performing this type of task.
resourceRequirements Association
Provides access to the resource requirements associated with performing this type of task.
subTaskTypes Association
Provides access to the sub-task types that make up this task type.
missionImportance Field
Provides information on the mission importance of this task. (Primary, Secondary, Support).
missionImportanceRank Field
Provides a numeric representation of the mission importance of this task. (1, 2, 3).
sourceObjectKey Field
Entered by the creator of this object, this is the objectKey of the source object used to derive this façade's fields.
alerts Association
Provides a link to all alerts that have been assigned to the source object.
comments Association
Provides a link to all comments that have been made on the source object.
issues Association
Provides a link to all issues created by users with this session.

11.8.6 TaskTypes Façade - *Provides access to all the types of task that can exist, and access to all of the currently active tasks in the system.*

taskTypes Association

Provides access to all the task types that can exist.

tasks Association

Provides access to all the currently active tasks in the system.

12. Appendix C: Object Model Package Specifications

PACKAGE DEFINITIONS

action
Package Name: action

Description:
The focus of the classes in the action package is to record the actions carried out in the domain and to specify the types of actions that may be performed. Instances of the Operational Level classes are used to record specific actions within the domain, while those from the Knowledge Level are used to record the common prototypical actions that are commonly referred to as an standard operating procedures of an Organization. An action may use links Protocol, via the inherited type association, to precisely define what was or is to be done in terms of the standard operating procedures. The Activity class extends the Action class to provide support for independently recording both the planning and execution phases of an Action

Members:

1. GeneralAction - Class
2. ProtocolDependency - Class
3. ActivityStatusType - Class
4. Activity - Class
5. Protocol - Class
6. ActivityDependency - Class
7. AbstractProtocol - Class

asset
Package Name: asset

Description:
The focus of the classes in the asset package is to provide an abstract framework to record and represent the individual physical entities within the domain and their types. Two subclasses or asset are provided: discrete item and inventory for representing either specific, uniquely identifiable assets, or a quantity of indistinguishable assets of a particular type. The logical type of thing represented by an asset object is specified by association to the appropriate Asset Type object in the Knowledge Level. Asset Types for a specific domain are specified as instances. This allows the same model to be configured for different domains and at runtime. It allows provides for the implementation of

dynamic classification and can be modified to support multiple classification schemes.

Members:

1. GeneralAsset - Class
2. AssetType - Class
3. Inventory - Class
4. DiscreteItem - Class

configuration
Package Name: configuration

Description:
The purpose of the configuration package is to allow persistence of customizable attributes within a configurable report. Each ConfigurableReport object must contain a DefaultConfiguration and may contain any number of SystemConfigurations. The User object corresponds to a UI user and may be associated to any number of UserConfigurations. A Configuration object contains a number of BaseFields, which correspond to Columns in a table type report and Fields in any other type of report. The attributes within Configuration, Column and Field objects specify how the report has been customized within that configuration.

Members:

1. Field - Class
2. FilterInfo - Class
3. Column - Class
4. ReportConfiguration - Class
5. User - Class
6. BaseField - Class
7. DefaultConfiguration - Class
8. ConfigurableReport - Class
9. SystemConfiguration - Class
10. eSortType - Class
11. eFilterType - Class
12. eColumnFormatType - Class
13. QuantityTypeColumn - Class
14. GeneralConfiguration - Class
15. eFilterRelationshipType - Class
16. StringTypeField - Class
17. IntegerTypeField - Class
18. DateTypeField - Class
19. AssociationTypeField - Class

20. BooleanTypeField - Class
21. EnumerationTypeField - Class
22. FloatTypeField - Class
23. UserConfiguration - Class

---

framework
Package Name: framework

Description:
The focus of the classes in the Framework package are to provide on overarching structure for the object model as a whole. All the classes in all other packages are extensions from a class in this package with some minor exceptions. The framework primarily provides containment and classification for classes in derived packages. The containment is provided by the View and Session classes and is primarily targeted to support systems with the dual roles of operational support and training/gaming support. The containment provided by the Agent View class supports tailored views for individual agent engines and their associated community of agents. The classification provided by the other classes in the package further support the dual operational gaming roles of the system, provide rudimentary support for temporal systems, and provide for a formalized partitioning of the domain into operational and knowledge levels. Objects in the Operational Level record the day-to-day events of the domain while Knowledge Level record the general rules that govern the configuration of objects in the Operational Level, and type knowledge (encyclopedic knowledge) shared by multiple Operational Objects.

Members:

1. AgentView - Class
2. Session - Class
3. View - Class
4. OperationalObject - Class
5. PlanningObject - Class
6. ResourceObject - Class
7. DomainObject - Class
8. Comment - Class
9. Type - Class
10. AbstractType - Class
11. KnowledgeObject - Class
12. SessionType - Class
13. CommentType - Class

---

humanAsset
Package Name: humanAsset

Description:
The Human Asset package defines an abstract Human Asset class from which the concrete classes Organization, Position, and Person derive. The Operational Level hierarchy is duplicated in the Knowledge Level to allow dynamic runtime typing in addition to the static compile time typing that can be added by derived classes. By defining a common base class for Organization, Person, and Position, this simple hierarchy provides a very powerful abstraction as surprisingly many things relate to Human Asset rather than Person, Organization, or Position. All may have phone numbers and addresses, bills and debts, responsibilities and actions, and so on. The package provides specific mechanisms to define temporal relationships between Person objects, Organization objects, and Position objects. Additional types of relationships may be defined as required with Accountabilities, which provide an extensible and generic mechanism for specifying domain specific relationships between Human Assets.

Members:

1. Position - Class
2. Person - Class
3. Organization - Class
4. PersonType - Class
5. PositionType - Class
6. OrganizationType - Class
7. OrganizationBase - Class
8. PositionTypeAllocation - Class
9. Ability - Class
10. AbilityType - Class
11. HumanAssetType - Class
12. AbstractOrganizationType - Class
13. OrganizationPositionPeriod - Class
14. PositionPersonPeriod - Class
15. PersonAbilityPeriod - Class
16. GeneralHumanAsset - Class

---

location
Package Name: location

Description:
The focus of the classes in the location package is to record and represent locations.

Members:

1. StructureElement - Class

2. StructuredLocation - Class
3. Place - Class
4. PlaceType - Class
5. LocationType - Class
6. StructuredLocationType - Class
7. GeneralLocation - Class

---

materialAsset
Package Name: materialAsset

Description:
The Material Asset package extends the Discrete Item and Asset Type classes of the abstract asset framework defined in the Asset package to specifically represent physical, non-human related assets. The package defines the Material Asset class that represents a specific physical item that can be uniquely identified. It also allows assets to be hierarchically decomposed into subcomponents that are themselves Material Assets. The Knowledge Level parallels the structure of the Operational Level by defining types of assets and their hierarchical decomposition into possible subcomponent types. It additionally defines the Material Asset Class, which can be used classify Material Asset Types into higher level groups such as vehicles, bombs, or missiles.

Members:

1. MaterialAssetType - Class
2. ShutdownType - Class
3. Shutdown - Class
4. MaterialAssetTypeStructureElement - Class
5. ConveyanceType - Class
6. Conveyance - Class
7. AbstractMaterialAsset - Class
8. GeneralMaterialAsset - Class

---

observation
Package Name: observation

Description:
The focus of the classes in the Observation package is to record and represent both qualitative and quantitative observations within the domain. Observations extend action and therefore have all the characteristics of a standard action. An Observation may be designated a hypothesis, projection, or active observation. Three subtypes of observation are defined to represent Measurement Observations, Rejected Observations and Category Observations. Measurement Observations record a quantity of some Phenomenon Type. Category Observations record the absence or presence of an Observation Concept and

capture the evidentiary observations that lead to it. In the Knowledge Level, Observations concepts are sub-typed into a hierarchy in a manner that allows Presence Observations to propagate up the tree while Absence Observations propagate down the tree. The Knowledge Level also indicates those Observation Concepts that may be used as evidence for a particular Observation Concept.

Members:

1. ObservationType - Class
2. RejectionObservation - Class
3. Measurement - Class
4. CategoryObservation - Class
5. Phenomenon - Class
6. GeneralObservation - Class
7. ObservationProtocol - Class
8. CollectionObservation - Class
9. CalculatedMeasurement - Class
10. MeasurementProtocol - Class
11. CalculationProtocol - Class
12. SourceProtocol - Class
13. Method - Class
14. Query - Class
15. AbstractConcept - Class
16. CollectionConcept - Class
17. eConceptCollectionType - Class
18. PhenomenonType - Class
19. CollectionSupport - Class
20. ObservationConcept - Class

---

quantity
Package Name: quantity

Description:
The focus of the classes in the Quantity package is to record and represent abstract quantities. A quantity groups a numerical value with a unit of measure. The package defines standard quantities such as length, weight, time, pressure, velocity, acceleration, and temperature. The capability to specify a range for a given Quantity is also provided. The framework for defining specific Unit systems, and their associated units of measure including both atomic and compound units are captured in the Knowledge Level. The Knowledge Level also captures the information necessary to convert between compatible units. The notion of complex and simple units presented here is similar to ideas presented in Fowler's Analysis Patterns, except that this uses a simple unit that can be composed of other simple units, whereas Fowler uses an atomic unit that cannot be broken down, and

he does not allow compound units to be expressed in terms of other complex units. This system allows a joule to be expressed as a newton-meter. Because this system differs, different names than Fowler uses have been selected. These complex units can be dynamically created as necessary. Also differently than Fowler, this moves the knowledge of unit relationships to the unit's type. This decreases the necessary objects that would be required to relate units of the same type. It also allows generalized relationships between types of units to be made, such as Force = Mass * Acceleration.

Members:

1. SimpleUnit - Class
2. UnitType - Class
3. GeneralQuantity - Class
4. QuantityRange - Class
5. UnitSystem - Class
6. UnitTypeRelation - Class
7. Prefix - Class
8. ComplexUnit - Class
9. UnitRelation - Class
10. Unit - Class
11. UnitTypeReference - Class
12. NumericQuantity - Class
13. StatisticalQuantity - Class

---

readiness
Package Name: readiness

Description:
The readiness package contains classes relating to the mission readiness. The ReadinessClass represents the missions, while ReadinessArea and ReadinessAreaType represents the areas affecting readiness. Other classes related to readiness are included in this package.

Members:

1. Option - Class
2. ReadinessWeight - Class
3. ReadinessState - Class
4. ReadinessLevel - Class
5. ReadinessConcept - Class
6. ReadinessObservation - Class
7. ReadinessClass - Class
8. ReadinessEffect - Class

9. ReadinessArea - Class
10. AbstractReadinessAreaType - Class
11. ReadinessAreaType - Class
12. AbstractReadinessArea - Class
13. ReadinessRequirement - Class
14. Solution - Class
15. ResponsibleParty - Class
16. ReadinessBias - Class
17. ReadinessBiasEntity - Class

---

ruleBasedAgent
Package Name: ruleBasedAgent

Description:
The focus of the classes in the Rule Based Agent package is to provide a common framework for the representation of the rule based agents within the system. The package provides a Rule Based Agent class from which individual agents may be created or classes representing the individual types of agents should derive. Agents associated with a set of classes to represent the private information within its domain should be represented in their own package. The package also provides the means to represent the logical rules of an agent, and the capability to turn off individual agents and/or rules. An alert class is also provided that allows agent alerts to be posted within the context of a Session, and allows for the posting of user comments on alerts.

Members:

1. AlertType - Class
2. RuleSetType - Class
3. ResponseType - Class
4. AlertAttributeType - Class
5. PlanningTime - Class
6. eAlertSubjectRole - Class
7. eAlertAttributeType - Class
8. eAgentBehaviorType - Class
9. Response - Class
10. RuleBasedAccountabilityAlert - Class
11. RuleBasedAlert - Class
12. AlertAttribute - Class
13. RuleSet - Class
14. GeneralRuleBasedAgent - Class
15. RuleBasedAgentType - Class
16. AlertSeverity - Class

17. AgentStatus - Class

---

sils
Package Name: sils

Description:
This package provides classes specific to SILS.

Members:

1. DeploymentCycle - Class
2. Ship - Class
3. ShipType - Class
4. Person - Class
5. NEC - Class
6. Phase - Class
7. PhaseType - Class
8. SILSView - Class
9. Issue - Class
10. Resolution - Class
11. IssueConcept - Class
12. ResolutionConcept - Class
13. Acknowledgement - Class
14. AcknowledgementConcept - Class
15. NECType - Class
16. ExternalSystem - Class
17. ExternalSystemType - Class

---

silsInfo
Package Name: silsInfo

Description:
Used to fill the department summary, these are a collection of the observations pertinent to each department within SILS. These are all observations and are specific to SILS.

Members:

1. AnchoringLog - Class
2. Message - Class
3. PersonChange - Class
4. GeneralSILSInfo - Class
5. DegradedEquipment - Class

6. ANOR - Class
7. CASREP - Class
8. AssetProblem - Class
9. UpdatedAssetProblem - Class
10. RepairStatus - Class
11. PersonChangeType - Class
12. SILSInfoType - Class

---

silsInterface
Package Name: silsInterface

Description:
The SILS Interface package is a SILS specific implementation of the generalized concept of an interface domain. An interface domain serves as a swap space for the exchange of information with external systems. It defines simple easy to understand information sets without the complexities found in the core problem domain model. It also provides a layer of isolation between the problem domain and the interfaces with external systems; thereby, ensuring the flexibility to evolve the core model over time. The two primary classes within the package are the Service Request class and the Service Result class. Service Request objects will typically be posted by the system agents. Service Requests will be picked up by an external interface broker, which will then query the appropriate external information systems to post the corresponding Service Result objects. The Interface Agent picks up additions, deletions, or modifications to Service Result objects then translates them into the core problem domain for use by the rest of the system.

Members:

1. ServiceRequest - Class
2. ServiceResult - Class
3. MachineryProblem - Class
4. PartAvailability - Class
5. PartOrder - Class
6. PersonnelAvailability - Class
7. TemporaryPersonnelAssignment - Class
8. PersonnelGainAndLoss - Class
9. ScheduleStatus - Class
10. MaintenanceActivity - Class
11. PersonnelStatus - Class
12. Personnel - Class
13. NEC - Class
14. ErrorResult - Class
15. PositionType - Class
16. ServiceRequestType - Class

17. ServiceRequestStatus - Class
18. ANOR - Class
19. CASREP - Class
20. AssetProblem - Class
21. DegradedEquipment - Class
22. AssetRepairStatus - Class
23. PersonnelTraining - Class
24. ExternalSystemDown - Class
25. RequestConstraint - Class

---

system
Package Name: system
Description:
The purpose of the classes in system are to represent information specific to the encapsulating system. The system package provides support for user accounts, access permissions, and basic system properties.

Members:

1. Contributor - Class
2. UserAccountType - Class
3. GeneralSystem - Class
4. UnionGroup - Class
5. UserGroup - Class
6. Group - Class
7. DifferenceGroup - Class
8. IntersectionGroup - Class
9. OperatingEntity - Class
10. Domain - Class
11. User - Class

---

task
Package Name: task

Description:
The task package provides classes providing information about a task. The operational level of the task package is mirrored by the Knowledge level of the task package. This is evident from the two primary classes of the task package: Task and TaskProtocol. They are defined by their subTypes and also by their ResourceRequirement/RequestedAsset and TaskMissionImportance. The knowledge level of this package provides extra knowledge information not found on the operational level.

Members:

1. TransportTask - Class
2. TransportTaskProtocol - Class
3. TrainingTask - Class
4. TrainingTaskProtocol - Class

---

time
Package Name: time

Description:
This package contains classes related solely to time. The purpose of this package is to objectify time so that it can be modeled so that two events share a deadline.

Members:

1. Deadline - Class
2. DeadlineType - Class
3. Period - Class
4. PeriodType - Class
5. AdjustablePeriod – Class
6. FixedPeriod – Class

13. References and Bibliography

Aha, David W. and Gupta, Kalyan Moy (2002), Causal Query Elaboration in Conversational Case-Based Reasoning; Proceedings of FLAIRS'02.

Bancilhon F, C. Delobel and P Kanellakis (1992); Building an Object-Oriented Database System; Morgan Kaufman.

Booch, Grady and Ivar Jacobson and James Rumbaugh (1999); The Unified Modeling Language User Guide; Addison Wesley.

Buschmann Frank, Douglas Schmidt, Hans Rohnert, and Michael Stal (1996); Pattern-Oriented Software Architecture: A System of Patterns; Vol. 1, John Wiley and Sons, New York, New York, 1996.

Buschmann, Frank, Hans Rohnert, Douglas Schmidt and Michael Stal (2000); Pattern-Oriented Software Architecture Volume 2, Patterns for Concurrent and Networked Objects; John Wiley and Sons.

CDM Technologies (2000); Collaborative Agent Based Control and Help (COACH) Project Report, ARES Team, CDM Technologies, Inc., San Luis Obispo, California.

CDM Technologies (2001a); SILS (Shipboard Integrated Logistics System) Concept Paper; SILS Project, ARES Team, CDM Technologies, Inc., San Luis Obispo, California.

CDM Technologies (2001b); SILS (Shipboard Integrated Logistics System) Framework Report; SILS Project, ARES Team, 5 July 2001, CDM Technologies, Inc., San Luis Obispo, California.

CDM Technologies (2001c); SILS (Shipboard Integrated Logistics System) Architectural Design Report; SILS Project, ARES Team, 27 Sept 2001, CDM Technologies, Inc., San Luis Obispo, California.

CDM Technologies (2001d); SILS (Shipboard Integrated Logistics System) Integration Report; SILS Project, ARES Team, 28 Nov 2001, CDM Technologies, Inc., San Luis Obispo, California.

CDM Technologies (2001e); SILS MRAT Data Source Integration Report; SILS Project, ARES Team, 20 Aug 2002, CDM Technologies, Inc., San Luis Obispo, California.

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

Fowler, Martin (1997); Analysis Patterns, Reusable Object Models; Addison Wesley Longman.

Fowler, Martin and Kendall Scott (1997); UML Distilled, Applying the Standard Object Modeling Language; Addison Wesley Longman.

Gamma, Erich and Richard Helm and Ralph Johnson and John Vlissides (1994); Design Patterns, Elements of Reusable Object-Oriented Software; Addison Wesley.

Hay, David C. (1996); Data Model Patterns, Conventions of Thought; Dorset House.

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

Kleppe, Anneke and Jos Warmer (1999); The Object Constraint Language, Precise Modeling With UML; Addison Wesley.

ManTech Advanced Systems International (2001a); Trip Report - Visit to Headquarters CINCPACFLT and USS FLETCHER; CINCPACFLT Headquarters and USS FLETCHER, DD-992, Pearl Harbor Hawaii; March 29-March 31, 2001, ManTech Advanced Systems International, Inc., Fairmont, West Virginia.

ManTech Advanced Systems International (2001b); Trip Report - Visit to USS COMSTOCK; USS COMSTOCK, LSD-45, San Diego, Ca; May 22-May 23, 2001, ManTech Advanced Systems International, Inc., Fairmont, West Virginia.

ManTech Advanced Systems International (2001c); Trip Report - Visit to USS PELELIU; USS PELELIU, LHA-5, San Diego, CA; July 10, 2001, ManTech Advanced Systems International, Inc., Fairmont, West Virginia.

Mowbray T, and R Zahavi (1995); The Essential CORBA: Systems Integration Using Distributed Objects; John Wiley.

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

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

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; InterSymp-97, International Conference on Systems Research, Baden-Baden, Germany.

Sandia National Laboratories (1997); The Java Expert System Shell; http://herzberg.ca.sandia.gov/jess; Ernest J. Friedman-Hill Distributed Computing Systems.

14. Glossary of Acronyms

AI	Artificial Intelligence
AN/SPN	(marshalling air traffic control radar used on Navy ships)
API	Application program Interface
APL	Allowance Parts List
ARG	Amphibious Ready Group
ASCII	American Standard Code for Information Interchange
BUPERS	Bureau of Naval Personnel System
C4I-SR	Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance
CAD	Computer-Aided Design
CART	Command Assessment of Readiness
CASREP	Casualty Report
CD-ROM	Compact Disk – Read Only Memory
CINCPACFLT	Commander in Chief US Pacific Fleet
CIWS	Close-In Weapons System
CLIPS	C Language Integrated Production System
CMP	Continuous Monitoring Program
CO	Commanding Officer
COACH	Collaborative Agent Based Control and Help system
COOL	CLIPS Object Oriented Language
CORBA	Common Object Request Broker Architecture
COSAL	Coordinated Shipboard Allowance List
COTS	Commercial Off The Shelf
CSCS	Combat Systems Casualty Control Computer System
CSMP	Current Ships Maintenance Project
DMS	Data Management System
DNC	Digital Nautical Chart
DS	Distance Support
EDVR	Enlisted Distribution Verification Record
EOC	Equipment Operational Capability
EPMAC	Enlisted Personnel Management Center

FSM	Food Service Management
GCCS-M	Global Command and Control System-Maritime
GIS	Geographic Information System
GUI	Graphical User Interface
HACCP	Hazard Analysis and Critical Control Point
HM&E	Hull, Mechanical, and Electrical
HTML	Hypertext Markup Language
IBFT	Integrated Battle Force Training
IBS	Integrated Bar Code System
ICAS	Integrated Condition Assessment System
ICDM	Integrated Cooperative Decision Making framework
ICODES	Integrated Computerized Deployment System
IE	Interface Engine
IMMACCS	Integrated marine Multi-Agent Command and Control System
JESS	Java Expert System Shell
JSN	Job Serial Number
MOXIE	ManTech Object eXchange Interface Engine
MRA	Mission Readiness Analysis
MRAS-NG	Mission Readiness Assessment System - Next Generation
MRAT	Mission Readiness Analysis Toolkit
MRDB	Material Readiness Data Base
NaCoDAE	Naval Conversational Decision Aids Environment
NAVSEA	Naval Sea Systems Command
NEC	Navy Enlisted Classification
NICN	Navy Item Control Number
NRL	Naval Research Laboratory
NSN	National Stock Number
NSWCCD	Naval Surface Warfare Center, Carderock Division
NTCSS	Naval Tactical Command Support System
NTFS	Navy Training Feedback System
OML	Object Management Layer
OMMS-NG	Organizational Maintenance Management System – Next Generation
ONR	Office of Naval Research

OPNAVINST	Office of Chief of Naval Operations Instruction
OPS	Operations
OPTAR	Operational Target
PDA	Personal Digital Assistant
PERA	Planning and Engineering for Repairs and Alterations
PMS	Planned Maintenance System
POW	Proxy Object Wrapper
PQS	Personnel Qualification Standards
R-Admin	Relational Administration System
R-Supply	Relational Supply System
RDBMS	Relational Database Management System
RETE	(fast pattern matching algorithm devised by Charles Forgy in 1979)
ROE	Rules of Engagement
SAMS	Automated Medical System
SILS	Shipboard Integration of Logistics Systems
SNOR	Semantic Net Object Restore
SORTS	Status of Resource and Training System
SQL	Standard Query language
SUPPO	Supply Officer
SWBS	Ship Work Breakdown Structures
TCRS	Taxonomic Case Reasoning System
TRIMS	Technical Risk Identification and Mitigation System
TRMS	TYCOM Readiness Management System
TYCOM	Type Commander
UML	Unified Modeling Language
URL	Universal Resource Locator
WECAN	Web Centric Anti-Submarine Warfare Net
XMI	XML Metadata Interchange
XML	eXtensible Markup Language
XO	Executive Officer

15. Keyword Index