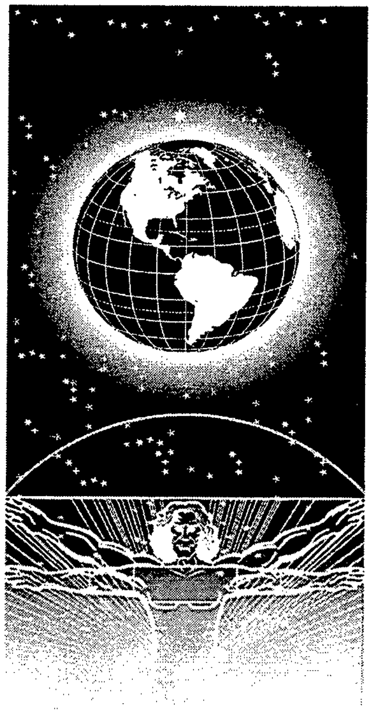
AFRL-HE-WP-TR-2002-0007



UNITED STATES AIR FORCE RESEARCH LABORATORY

Human Interaction with Software Agents (HISA)

Alice M. Mulvehill

BBNT Solutions LLC
10 Moulton Street
Cambridge, MA 02138

Randall Whitaker

Logicon Technical Services, Inc.
P.O. Box 317258
Dayton, OH 45431

November 2000

Final Report for the Period February 1999 to November 2000

20020409 080

Human Effectiveness Directorate
Deployment and Sustainment Division
Sustainment Logistics Branch
2698 G Street
Wright-Patterson AFB OH 45433-7604

TECHNICAL REVIEW AND APPROVAL

AFRL-HE-WP-TR-2002-0007

This report has been reviewed by the Office of Public Affairs (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER

Mark M. Hoffman

MARK M. HOFFMAN
Deputy Chief
Deployment and Sustainment Division
Air Force Research Laboratory

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** November 2000	**3. REPORT TYPE AND DATES COVERED** Final - February 1999 - November 2000

4. TITLE AND SUBTITLE Human Interaction with Software Agents (HISA)	**5. FUNDING NUMBERS** C: F33615-99-D-6001 DO: 03 PE: 63106F PR: 2745 TA: 00 WU: 31
6. AUTHOR(S) Alice M. Mulvehill, Randall Whitaker	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) BBNT Solutions LLC, 10 Moulton Street, Cambridge, MA 02138; Logicon Technical Services, Inc., P.O. Box 317258, Dayton, OH 45431	**8. PERFORMING ORGANIZATION REPORT NUMBER** BBN Report No. 8291
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory, Human Effectiveness Directorate, Deployment and Sustainment Division, Air Force Materiel Command, Sustainment Logistics Branch, Wright-Patterson AFB OH 45433-7604	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER** AFRL-HE-WP-TR-2002-0007
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	**12b. DISTRIBUTION CODE**

13. ABSTRACT *(Maximum 200 words)*

Agent based systems are rapidly emerging to support users as the knowledge and complexity of computer-based information systems increases. Software agents are currently being used to monitor data sources, bringing back only the information that is of relevance to a particular user. The primary object of the Human Interaction with Software Agents (HISA) project was to develop a graphical user interface (GUI) that allows users to task and manage software agents. Software agents were modeled and implemented using BBN's Distributed Operator Model Architecture (D-OMAR). D-OMAR was chosen since the D-OMAR environment supports discrete-event simulations of agents driven by goals and procedures. In addition, D-OMAR supports parallel activities, its agents can respond to asynchronous signals of events, and it supports semantic level communications between agents. The focus of the effort was placed on how Channel Plans are developed at the Tanker Airlift Control Center (TACC), specifically the early detection and communication of Maximum on the Ground (MOG) restrictions, and messages from Notice to Airmen (NOTAM) files regarding airfield restrictions. This report covers the development of a GUI where D-OMAR software agents monitoring NOTAMs could post notices to human planners. The intent was to alert the planner of problems as they occur so that the Channel Planner has more time to research problems caused by the change and to make the required mission adjustment.

14. SUBJECT TERMS Human Interaction with Software Agents (HISA); Graphical User Interface (GUI); Distributed Operator Model Architecture (D-OMAR); Software Agents	**15. NUMBER OF PAGES** 68
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

THIS PAGE INTENTIONALLY LEFT BLANK

PREFACE

This final report documents research performed under the Technology for Readiness and Sustainment (TRS) contract, F33615-99-D-6001, Delivery Order 03, for the Air Force Research Laboratory (AFRL), Sustainment Logistics Branch (HESS), Wright-Patterson AFB, OH. The Air Force Program Manager was Captain Sarah DeMers. The research covered the time period from February 1999 through November 2000. The primary objective of the Human Interaction with Software Agents (HISA) project was to develop user interface technology that allows users to task and manage software agents to improve the airlift mission planning process at Air Mobility Command. The contractors were BBNT Solutions LLC, TASC Inc., Logica (formerly known as the Carnegie Group Inc.) and Logicon.

Table of Contents

List of Appendices

List of Tables

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

Human Interaction with Software Agents (HISA)

Introduction

Agent based systems are rapidly emerging to support users as the knowledge and complexity of computer-based information systems increases. Software agents are currently being used to monitor data sources, bringing back only the information that is of relevance to a particular user. In this project the primary objective was to develop a graphical user interface (GUI) that allows users to task and manage software agents.

Software agents were modeled and implemented using BBN's Distributed Operator Model Architecture (D-OMAR). D-OMAR was chosen because its environment supports discrete-event simulations of agents driven by goals and procedures. In addition, D-OMAR supports parallel activities, its agents can respond to asynchronous signals of events, and it supports semantic level communications between agents.

Our goal was to build tools that allow the user to view, control, task, and create D-OMAR agents to do things. Our efforts should not be misconstrued as giving the end user the ability to programmatically develop new software agents. Rather, we proposed an interface widget to enable a user to modify or generate new agents from a class of existing agents (like building an instance) by changing the parameters associated with an agent or agent class.

We focused on how Channel Plans are developed at the Air Mobility Command's (AMC) Tanker Airlift Control Center (TACC). We were interested in all aspects of the channel planning position, from building an initial plan to finally getting it to execution. Knowledge acquisition, user modeling, human computer interface (HCI) design development, and software development were the primary tasks employed to achieve the vision. For example, early meetings with the subject matter experts (SMEs) indicated two areas of value to them: early detection and communication of Maximum on the Ground (MOG) restrictions, and messages from the Notice to Airmen (NOTAM) system regarding airfield restrictions. Through early knowledge acquisition, the BBN Technologies and Logica data collectors learned that Channel Planners work 90-120 days out in the future for passenger missions and 30-100 days out in the future for cargo missions. If a change occurs a week before the execution of the mission, it might not be discovered until the aircrew is in final flight mission planning and checking NOTAMs.

During the course of this project we developed a GUI where D-OMAR software agents monitoring NOTAMs could notify human planners about specific NOTAM changes. The intent was to alert the planner of problems as they occur so that the Channel Planner has more time to research problems caused by the change and to make the required mission adjustments.

Our studies also focused on identifying any Air Force systems that our software agents might eventually need to communicate with and we collected examples of the computer screens associated with a limited number of AF systems, e.g., GDSS, NOTAMs. We discovered other agent-based systems were being developed for use in the TACC. For example, the Lockheed Martin Cooperating Agents for Specific Tasks (CAST) agents were already being tailored to access data from Global Decision Support System (GDSS). Instead of duplicating this effort, we initiated a Technology Integration Experiment (TIE) to build communication links between our GUI and the CAST software agents, and to build communication links between the CAST software agents and the D-OMAR software agents via the "Grid." The Control of Agent Based Systems (CoABS) "Grid" is a developing Defense Advanced Research Projects Agency (DARPA) research infrastructure that is intended to facilitate agent communication and integration. The results of the TIE were demonstrated in the January 2000 Human Interaction with Software Agents (HISA) demonstration (the 1st Concept Demonstration).

In order to make better use of our efforts, our knowledge engineers developed several scenarios to serve as guides in the design and development of the HISA GUIs, D-OMAR agents, and concept demonstrations. Appendix A describes our efforts in detail.

Building the Concept Demonstrations

For the 1st Concept Demonstration, the goal was to focus on how a human user would use the GUI and the software agents to support some task or set of tasks. Instead of focusing on the MOG conflict, the system was directed to provide the user with alerts about the changes in a number of conditions, like restrictions in parking MOG (PMOG) or working MOG (WMOG) for which an alert notification would be useful to the Channel Planner in developing and managing his/her channel plans. The 1st Concept Demonstration was centered around a TIE between the Lockheed Martin CAST agents, the HISA D-OMAR agents and GUI. From the HISA perspective, the GUI would be used to view processing of data by multiple heterogeneous agents, e.g., D-OMAR and CAST agents. From a Lockheed Martin perspective, the HISA GUI would replace their GUI. From the DARPA CoABS perspective, the CoABS Grid would be demonstrated as an environment for facilitating heterogeneous agent communication.

A month's worth of operational GDSS data was procured for this effort, and software had to be written to connect the D-OMAR agents to the GDSS data and the CoABS Grid. The demonstration used the MOG conflict detection engine that was developed by Lockheed Martin, while the D-OMAR agents were used to collect data from the Lockheed Martin CAST agents and send the data to the GUI. Problems were encountered with providing valid time, aircraft width, and mission ownership information to the GUI and these problems facilitated discussions on the development of better methods for displaying temporal values, specifically sunrise and sunset values per port given the current time. Note: This particular problem was initially resolved with the use of a sunrise/sunset Web Server. This was finally replaced with a DOS utility that the domain operational users utilize to calculate sunrise and sunset for ports around the world.

In the 2nd Concept Demonstration, alerts are provided to the user about: WMOG conflicts, need for Prior Permission Requests (PPR), and problems associated with ports, as indicated by changes in the NOTAMs data. The idea was to provide alerts much like a user would get new mail. An icon flashes and/or beeps when new alerts arrive and are pushed onto a queue of alerts. The user can click on the alerts in the queue to view them, and click on buttons associated with the alerts to get further information, or to take some action, e.g., contact another planner to resolve a conflict. The user can then simply close, defer or disregard an alert. All actions taken by the user are stored with the alert in an alert history, freeing the alert queue to be available for displaying newly arrived alerts. One could view the Alert Queue and the Alert History windows as mailboxes in a mail system. Linking software agents to activities like receiving mail has been investigated.

A messaging application program interface (API) (Appendix B) between the front-end GUI and the back-end D-OMAR agents was developed and hardened to support additional functionality for the 2nd Concept Demonstration. Much of the interface from the 1st Concept Demonstration was reused, but new GUIs were also defined and implemented. Additions to the GUI D-OMAR API were required to support the enhanced functionality of the 2nd Concept Demonstration. In addition, a method for reading raw data directly into D-OMAR via a pseudo database was developed for use in the 2nd Concept Demonstration. This database allows for general structured query language (SQL) like queries and updates to be made.

Development of the HCI Based on the Direct Manipulation Metaphor

One of the goals of the project was to develop a HCI that is based on a direct manipulation metaphor. Direct manipulation is a concept that makes use of a bitmapped screen to present the user with direct visual representations of objects such as documents and printers. To print a document, for example, the user can point (using the mouse) to the icon for the document and the icon for the printer, while using a key on the keyboard to indicate a copy operation. Logicon produced several conceptual interface designs to support TACC Channel mission planning/monitoring. Table 1 lists all of the GUIs designed during the project divided into three categories: those that were delivered in the final demonstration and require minor changes, those that were delivered in the final demonstration and require redesign, and those that are still only in a conceptual design state. Each GUI is delimited with the terms "must", "nice", or "desirable", indicating the end user's disposition toward the requirement for the window.

Existing Window - Requires Minor Changes Only	Conceptual Design Exists - Requires Detailed Design & New Window	Requires New Design & Window
Alert Notification - MUST	More Mission Info – DESIRABLE	Mission Summary - NICE
Port Viewer - MUST	Contact Info – NICE	Airfield Restrictions-Conflict Detail - DESIRABLE
Conflict Summary - MUST	Multi-Port Viewer (for "what-if's") a. "Read Only (view, no what if) – DESIRABLE b. "What If" capable – NICE	Airfield Restrictions-Setting Detail - NICE
Alert Deferral - MUST	EnRoute Viewer – NICE	Rule Violations (for "what-if's") - NICE
NOTAM/DIP/PPR Info (for port) a. NOTAM – DESIRABLE b. PPR – DESIRABLE c. DIP - NICE	Package Viewer – NICE	User Setup – Alerts – MUST a. "Create a List" of my missions (copy from GDSS and paste into this list) b. b. "Add a Mission" (after main list created)
	Smart Lt. - Icon View – MUST	User Setup - User Profile – MUST • they want to call this "User Preferences", not "Setup"
	Smart Lt. - Opening Pallet – MUST	PPR Template - DESIRABLE
	Alert Queue – MUST	DIP Window(s)-conflict &/or setting - NICE
	Alert History – MUST	Weather Window(s)-conflict/setting - NICE
	Query Assistant-Predefined – NICE	NOTAM Window(s)-conflict/setting – MUST a. MUST get an alert about a NOTAM conflict/issue b. NICE to see the "setting"
	Query Assistant-User-defined – NICE	Agent Query Assistant - NICE
		Smart Lt. – Logon – MUST Must be able to logon and start the application.

Table 1. User Defined GUI Usefulness

Our vision of the interface evolved over the course of the project. From early on, the vision was to link displays like the Port Viewer (Figure 1) with other function-oriented displays to orchestrate an interface environment designed to support a particular TACC Duty Officer position, specifically the Channel Plans area of the TACC. Figure 1 displays the final implementation of the Port Viewer. Mission bars are displayed in the time they are at the Port, e.g., Licz-Sigonella. The width of the mission bar is symbolic of the wide body versus narrow body aircraft requirement. The vertical bar through the mission bars indicates the duration of the conflict.

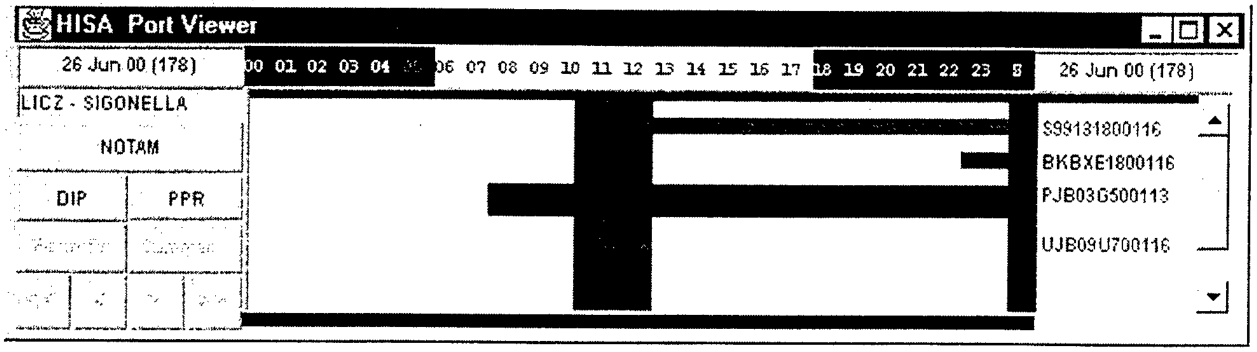


Figure 1. Port Viewer

Figure 2 illustrates two Early Port Planner display designs. These figures illustrate how airfield operating hours, sunrise/sunset hours, local time, multiple flights on ground during a 24-hour period, and resultant MOG problems might be displayed to the user. Function tabs or buttons allow access to additional airfield information such as NOTAM, Diplomatic Clearance (DIP) and PPR requirements.

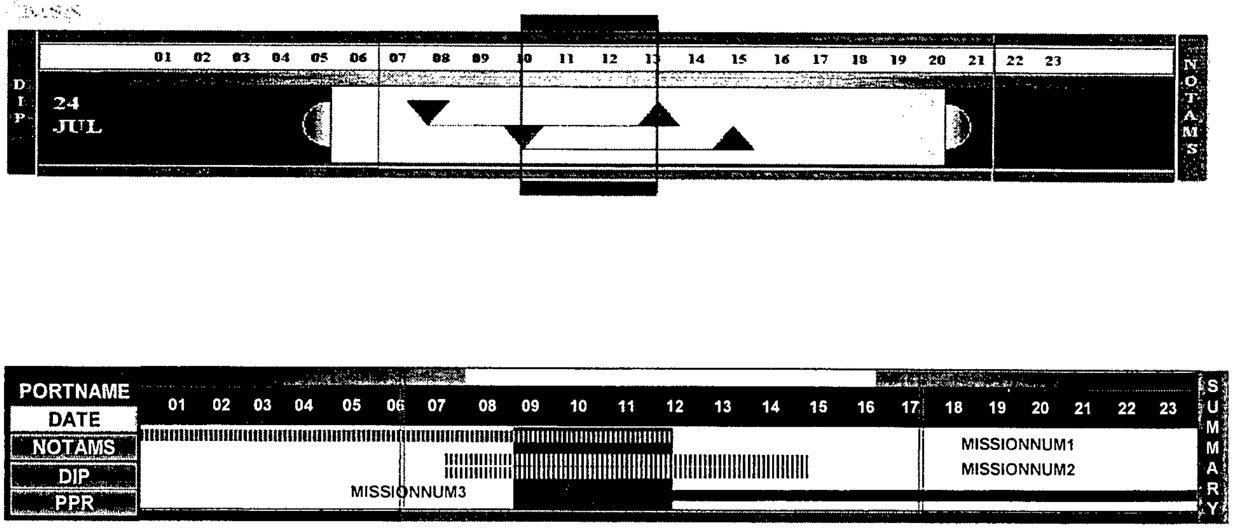


Figure 2. Early Port Planner

The Package Planner (Figure 3) and the Passage Planner (Figure 4) were designed to be used with the Port Planner to form a composite view like that shown in Figure 5, the Composite Mission Display. However, only the Port Planner was implemented during this project.

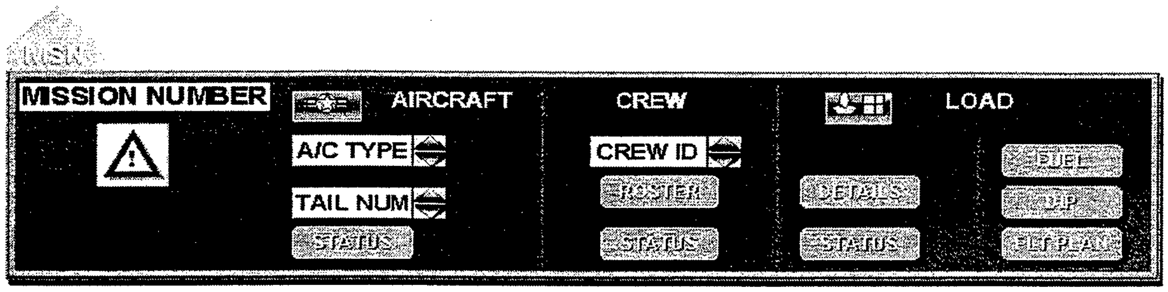


Figure 3. Package Planner

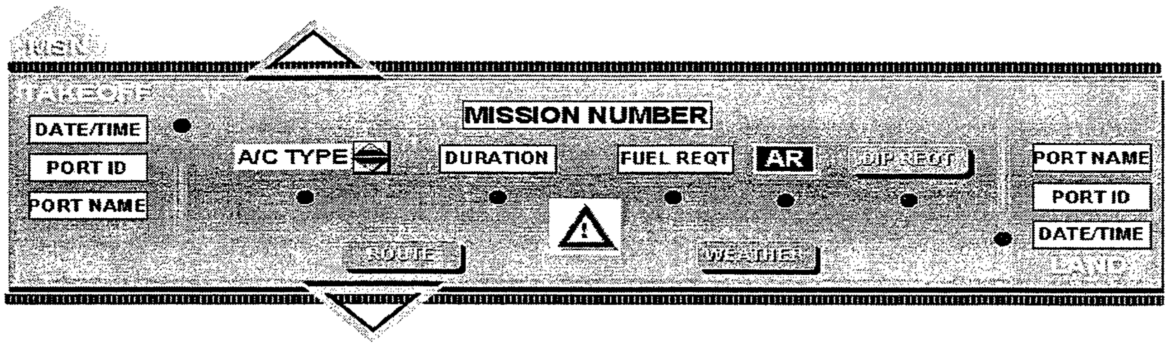


Figure 4. Passage Planner

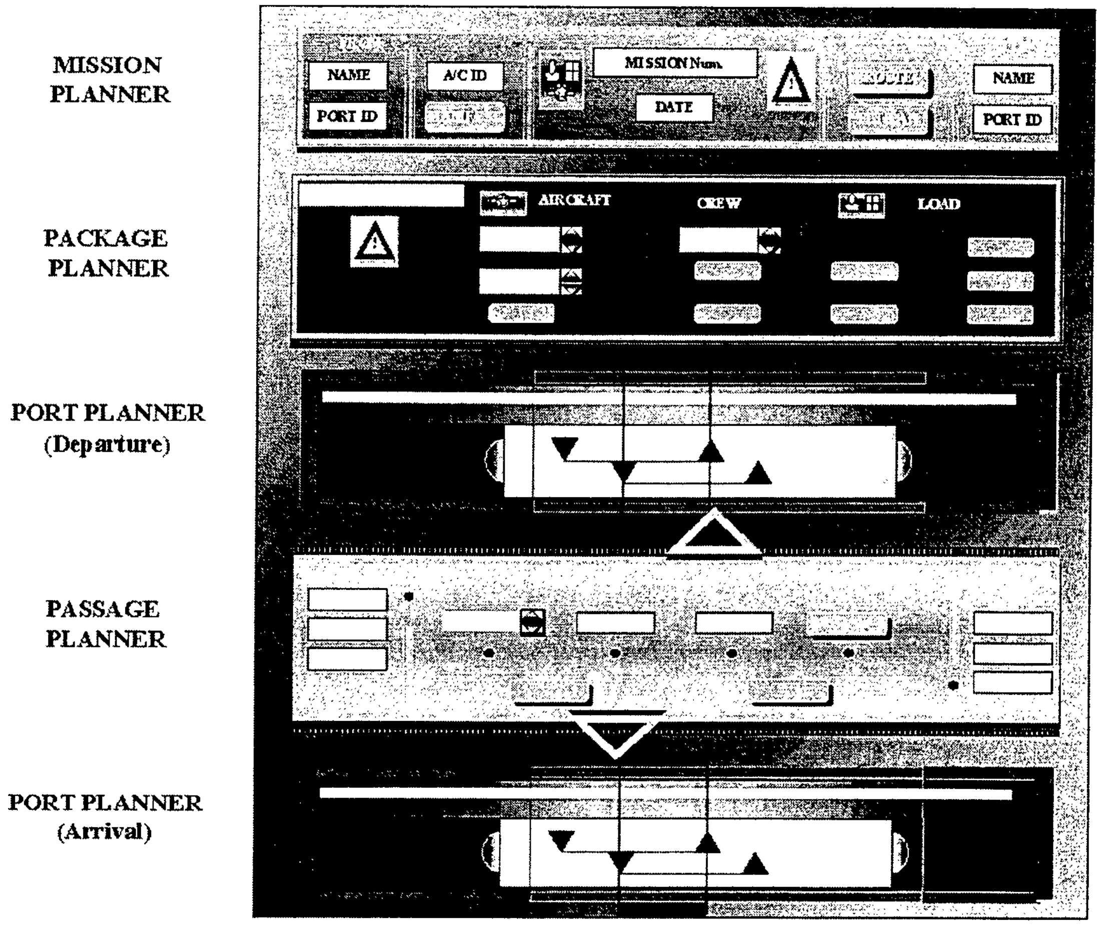


Figure 5. Composite Mission Display

Scenario-Driven Implementation

A scenario was developed to focus the concept demonstrations. The 1st Concept Demonstration Scenario (Figure 6) consisted of a single channel mission originating at Dover Delaware, conducting stops at several airfields and terminating at Bahrain. The 1st concept demonstration flow is depicted in Figure 7, indicating where the conflict is injected and when the resulting conflict alert is detected.

SCENARIO: CHANNEL FREQUENCY MISSION
(02F3 Outbound)

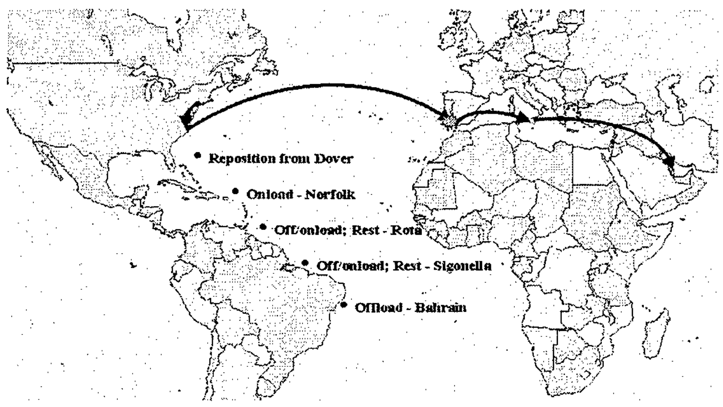
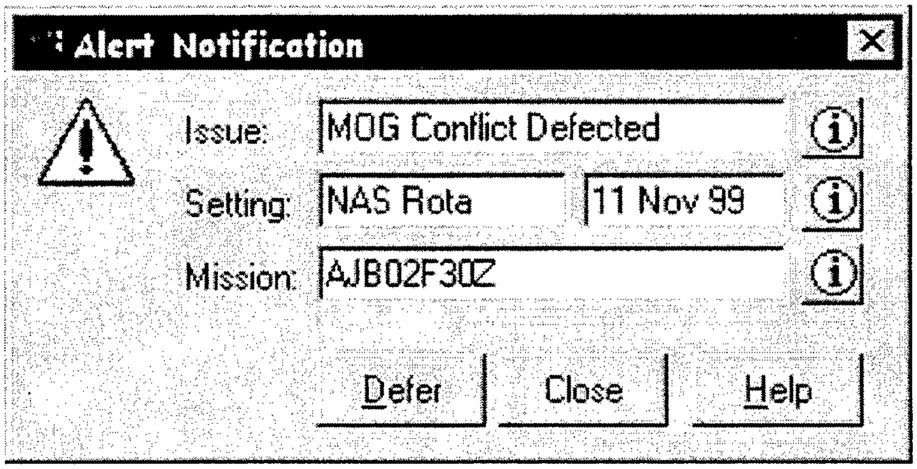


Figure 6. Scenario to First Concept Demonstration


Alert Notification

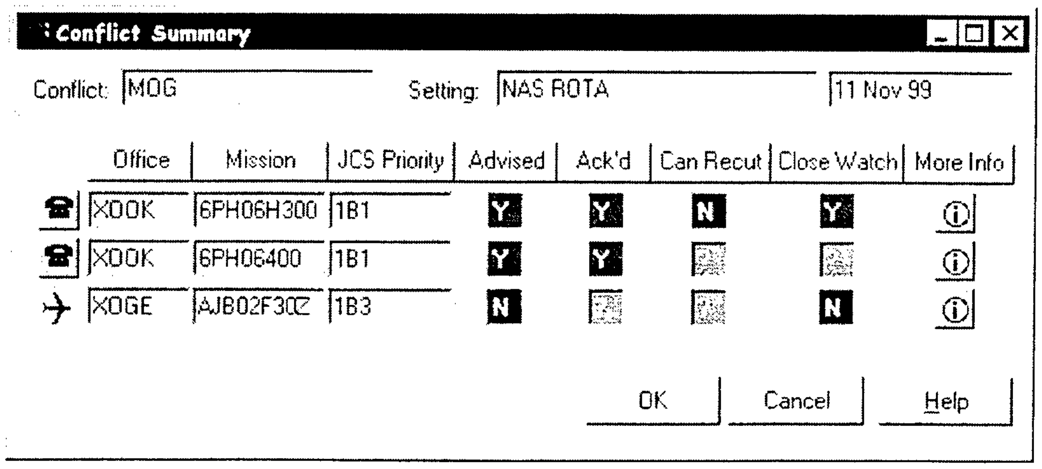
Issue: MOG Conflict Defected

Setting: NAS Rota 11 Nov 99

Mission: AJB02F30Z

Figure 7. Alert Notification

The Alert Notification pop-up window (Figure 7) is displayed when a conflict is detected. The user is offered three paths (indicated by the three "i" buttons) of relevant drill-down options to the alert recipient. For example, when the user clicks on the "i" button next to the Issue, the Conflict Summary information (Figure 8) associated with the Alert is displayed. The Conflict Summary presents the user with a listing of all published missions conflicting with his/her own. The Conflict Summary display allows a user to review the conflict situation with respect to other planners and their published plans. The user can also view port information via the Port Viewer (Figure 1) by selecting the "i" next to port "Setting". From the Port Viewer, the user can obtain other data, e.g., NOTAMs, DIP, and PPR that may be affecting the missions at the port.



Conflict: MOG Setting: NAS ROTA 11 Nov 99

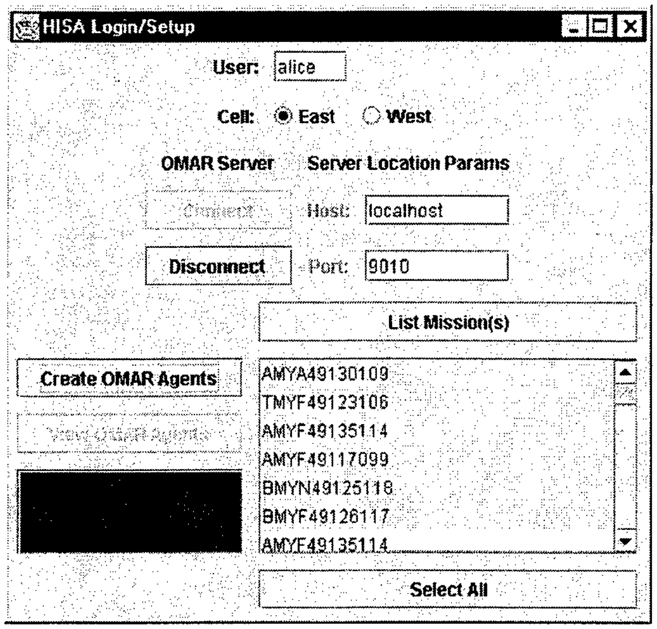
	Office	Mission	JCS Priority	Advised	Ack'd	Can Recut	Close Watch	More Info
☎	XOOK	6PH06H300	1B1	Y	Y	N	Y	ⓘ
☎	XOOK	6PH06400	1B1	Y	Y	▨	▨	ⓘ
✈	XOGE	AJB02F30Z	1B3	N	▨	▨	N	ⓘ

Figure 8. Conflict Summary

Adding Additional GUIs and Functionality

In building the 1st Concept Demonstration, it became apparent that some top-level control panel was needed in addition to some mechanism for storing the application and for managing the alerts (software agent messages) to the user. This section describes several GUIs that were designed and implemented to support these additional system needs. All of the GUIs displayed in this section were used in the 2nd Concept Demonstration and serve as the basis for the follow-on effort.

The HISA Login/Setup window (Figure 9) was used to start the application in the 1st Concept Demonstration. It provided a meager amount of functionality, simply allowing the user to log-on, create and view agents associated with a mission, and get information on when alerts were available. This window was replaced by the Smart LT window (Figure 10), which was designed and implemented for use in the 2nd Concept Demonstration.


HISA Login/Setup

User: alice

Cell: [✓ East] West

OMAR Server | Server Location Params

Connect | Host: localhost

Disconnect | Port: 9010

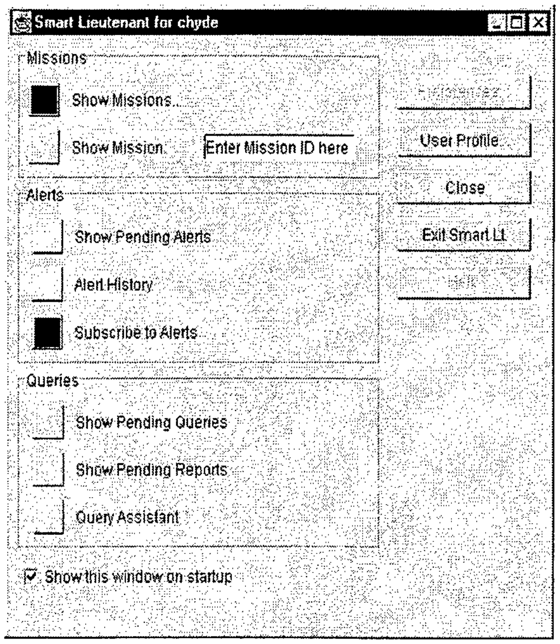
List Mission(s)

Create OMAR Agents

View OMAR Agents

AMYA49130109
TMYF49123106
AMYF49135114
AMYF49117099
BMYN49125118
BMYF49126117
AMYF49135114

Select All


Smart Lieutenant for chyde

Missions
- ■ Show Missions...
- Show Mission: Enter Mission ID here

Alerts
- Show Pending Alerts
- Alert History
- ■ Subscribe to Alerts

Queries
- Show Pending Queries
- Show Pending Reports
- Query Assistant

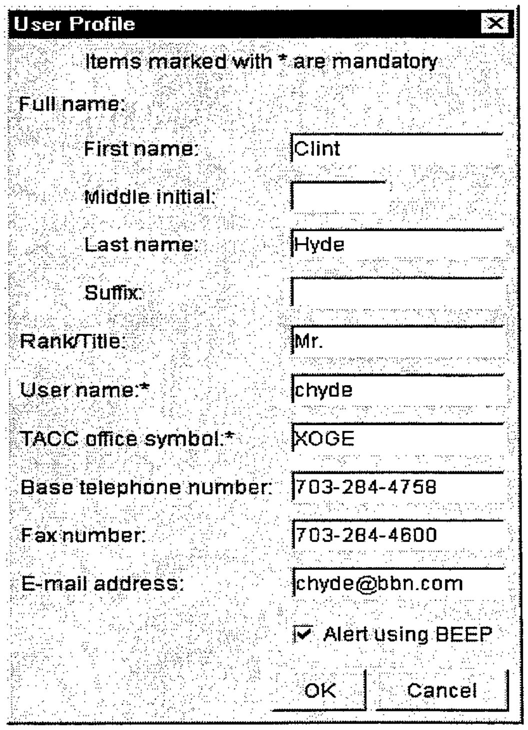
[✓] Show this window on startup

User Profile | Close | Exit Smart Lt

Figure 9. HISA Login/Setup

Figure 10. Smart LT

The Smart LT window provides the user with access to information about missions, alerts and other related information. Through this pallet, the user can set and modify a User Profile (Figure 11), view mission information, view alert information, link mission to alerting agents, and perform a variety of queries (not implemented at this time). When the user is busy using another application, the Smart LT Icon (Figure 12) will appear and stay on the desktop to provide a visual indication of new alerts as they are received. Clicking on the icon allows the user easy access (expansion) to the application windows.


User Profile

Items marked with * are mandatory

Full name:	
First name:	Clint
Middle initial:	
Last name:	Hyde
Suffix:	
Rank/Title:	Mr.
User name:*	chyde
TACC office symbol:*	XOGE
Base telephone number:	703-284-4758
Fax number:	703-284-4600
E-mail address:	chyde@bbn.com

[✓] Alert using BEEP

OK | Cancel

Figure 11. User Profile

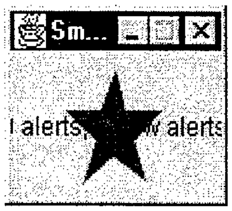
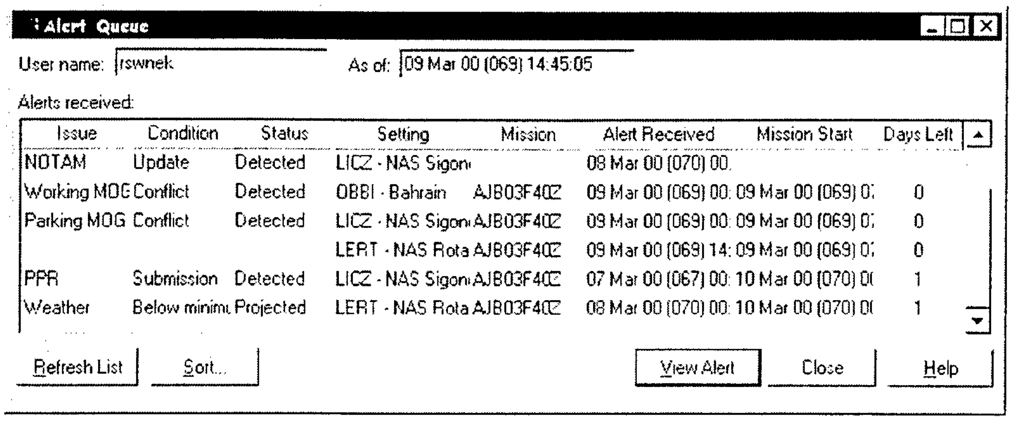


Figure 12. Smart LT Icon

For the 2nd Concept Demonstration, we implemented the Alert Queue window (Figure 13) to provide a listing of all alerts received but not previously opened. The list is sortable based on a menu of selectable options. The user can double click on any alert to open it. Opening the alert results in a display of the Alert Notification window (Figure 7.)

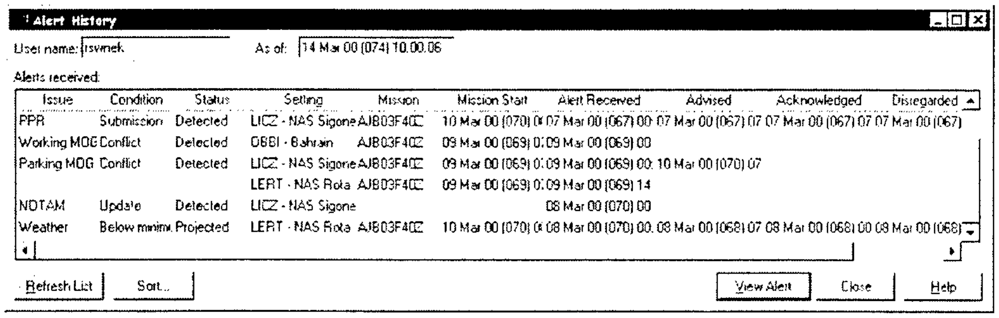


User name: rswnek   As of: 09 Mar 00 (069) 14:45:05

Alerts received:

Issue	Condition	Status	Setting	Mission	Alert Received	Mission Start	Days Left
NOTAM	Update	Detected	LICZ - NAS Sigon		08 Mar 00 (070) 00		
Working MOG	Conflict	Detected	OBBI - Bahrain	AJB03F40Z	09 Mar 00 (069) 00:	09 Mar 00 (069) 0:	0
Parking MOG	Conflict	Detected	LICZ - NAS Sigon	AJB03F40Z	09 Mar 00 (069) 00:	09 Mar 00 (069) 0:	0
			LERT - NAS Rota	AJB03F40Z	09 Mar 00 (069) 14:	09 Mar 00 (069) 0:	0
PPR	Submission	Detected	LICZ - NAS Sigon	AJB03F40Z	07 Mar 00 (067) 00:	10 Mar 00 (070) 0(	1
Weather	Below minim	Projected	LERT - NAS Rota	AJB03F40Z	08 Mar 00 (070) 00:	10 Mar 00 (070) 0(	1

Figure 13. Alert Queue



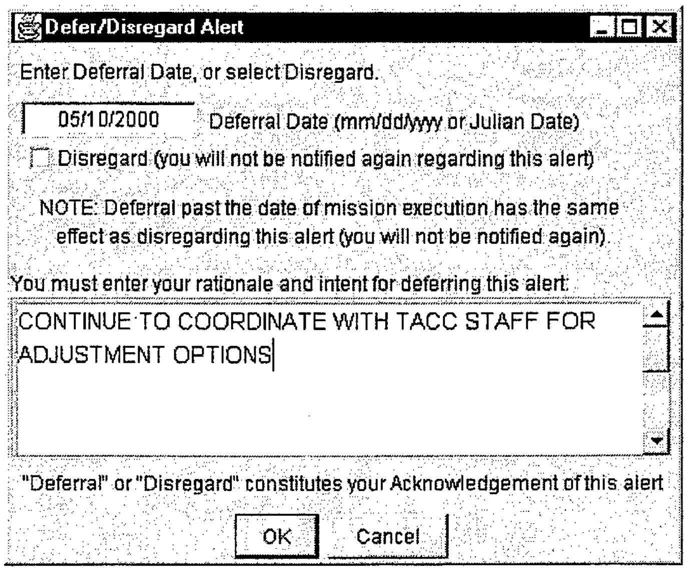
User name: rswnek   As of: 14 Mar 00 (074) 10.00.06

Alerts received:

Issue	Condition	Status	Setting	Mission	Mission Start	Alert Received	Advised	Acknowledged	Disregarded
PPR	Submission	Detected	LICZ - NAS Sigone	AJB03F40Z	10 Mar 00 (070) 0(	07 Mar 00 (067) 00:	07 Mar 00 (067) 07	07 Mar 00 (067) 07	07 Mar 00 (067)
Working MOG	Conflict	Detected	OBBI - Bahrain	AJB03F40Z	09 Mar 00 (069) 0:	09 Mar 00 (069) 00			
Parking MOG	Conflict	Detected	LICZ - NAS Sigone	AJB03F40Z	09 Mar 00 (069) 0:	09 Mar 00 (069) 00:	10 Mar 00 (070) 07		
			LERT - NAS Rota	AJB03F40Z	09 Mar 00 (069) 0:	09 Mar 00 (069) 14			
NOTAM	Update	Detected	LICZ - NAS Sigone			08 Mar 00 (070) 00			
Weather	Below minim	Projected	LERT - NAS Rota	AJB03F40Z	10 Mar 00 (070) 0(	08 Mar 00 (070) 00:	08 Mar 00 (068) 07	08 Mar 00 (068) 00	08 Mar 00 (068)

Figure 14. Alert History.

The Alert History window (Figure 14) keeps a log of all alerts (an "audit trail"). When an action is taken through the Defer/Disregard Alert window (Figure 15), which allows the user to either "defer" an action until a specified date, or "disregard" the alert altogether, the date of the action is recorded in the related field of the Alert History. The user can also double click on alerts in the Alert History window to review information, including any rationale or comments about an alert. (Figure 16).



Enter Deferral Date, or select Disregard.

05/10/2000 Deferral Date (mm/dd/yyyy or Julian Date)

☐ Disregard (you will not be notified again regarding this alert)

NOTE: Deferral past the date of mission execution has the same effect as disregarding this alert (you will not be notified again).

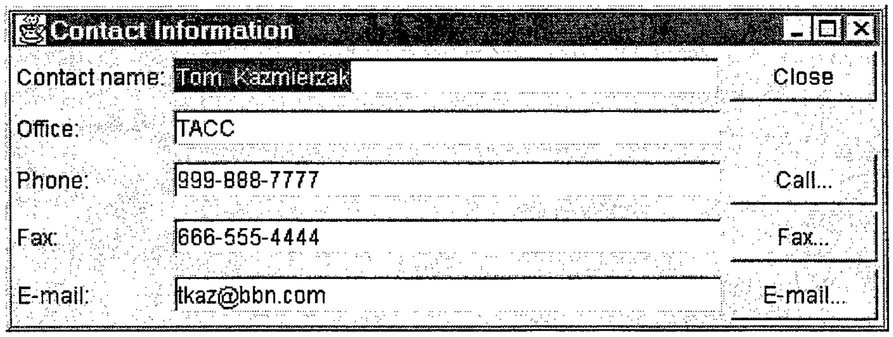
You must enter your rationale and intent for deferring this alert:

CONTINUE TO COORDINATE WITH TACC STAFF FOR ADJUSTMENT OPTIONS

"Deferral" or "Disregard" constitutes your Acknowledgement of this alert

Figure 15. Defer/Disregard Window

In addition to deferring or disregarding an alert, the user can use the Contact Information window (Figure 16) to provide basic information about how to contact the "owner/scheduler" of another mission who he is in conflict with. We envision the user being able to fax, email, or telephone another scheduler in future versions of this window. The Mission Additional Information window (Figure 17) provides more information necessary for re-planning considerations. It displays factors typically taken under consideration that may provide added "weight" or influence on the decision, as well as a "deferral rationale" entry to explain a user's "thinking" when deferring action.



Contact name: Tom Kazmierzak

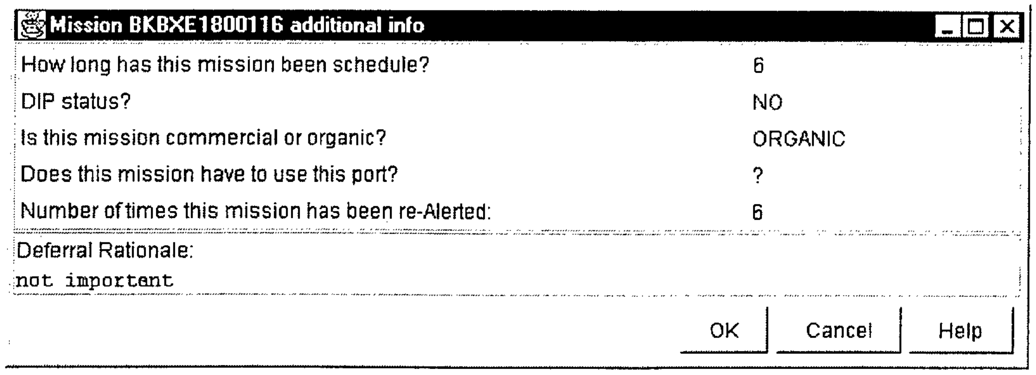
Office: TACC

Phone: 999-888-7777

Fax: 666-555-4444

E-mail: tkaz@bbn.com

Figure 16. Contact Information



Mission BKBXE1800116 additional info

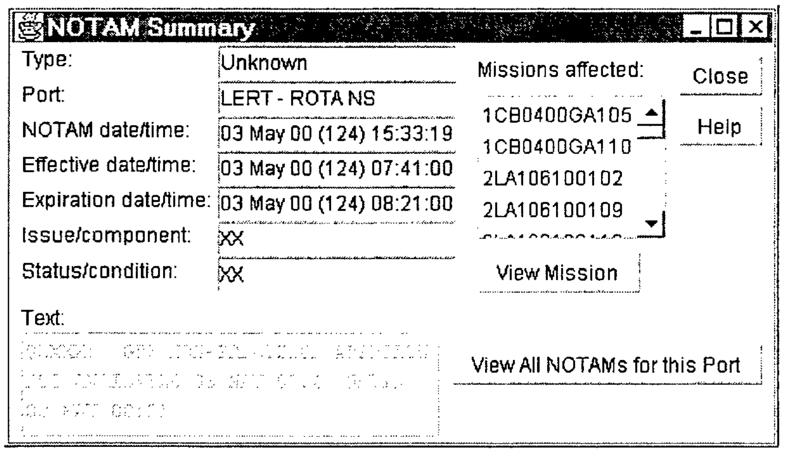
How long has this mission been schedule?	6
DIP status?	NO
Is this mission commercial or organic?	ORGANIC
Does this mission have to use this port?	?
Number of times this mission has been re-Alerted:	6

Deferral Rationale:

not important

Figure 17. Mission Additional Information

Finally, in order to enable a D-OMAR agent to obtain NOTAM data directly from the NOTAM web as part of the D-OMAR simulation, a D-OMAR agent was set up to directly receive selected mission identifying Q-code changes from the NOTAM system. A NOTAM Summary window (Figure 18) displays key information pertaining to the NOTAM alert including the airport and mission(s) involved, effective dates, and a drill-down capability to allow viewing of the full NOTAM.



NOTAM Summary

Type:	Unknown
Port:	LERT - ROTA NS
NOTAM date/time:	03 May 00 (124) 15:33:19
Effective date/time:	03 May 00 (124) 07:41:00
Expiration date/time:	03 May 00 (124) 08:21:00
Issue/component:	XX
Status/condition:	XX

Missions affected:

1CB0400GA105
1CB0400GA110
2LA106100102
2LA106100109

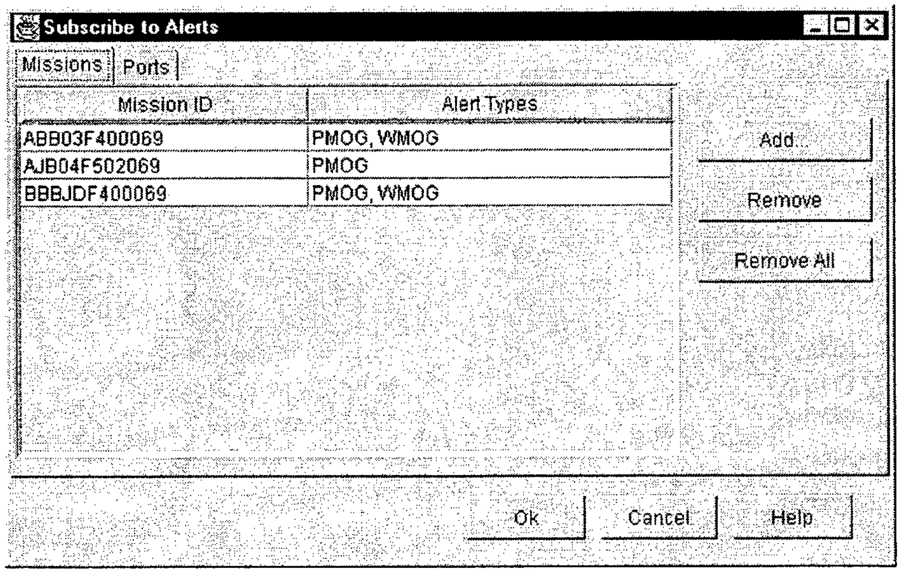
Text:

[illegible]

Figure 18. NOTAM Summary

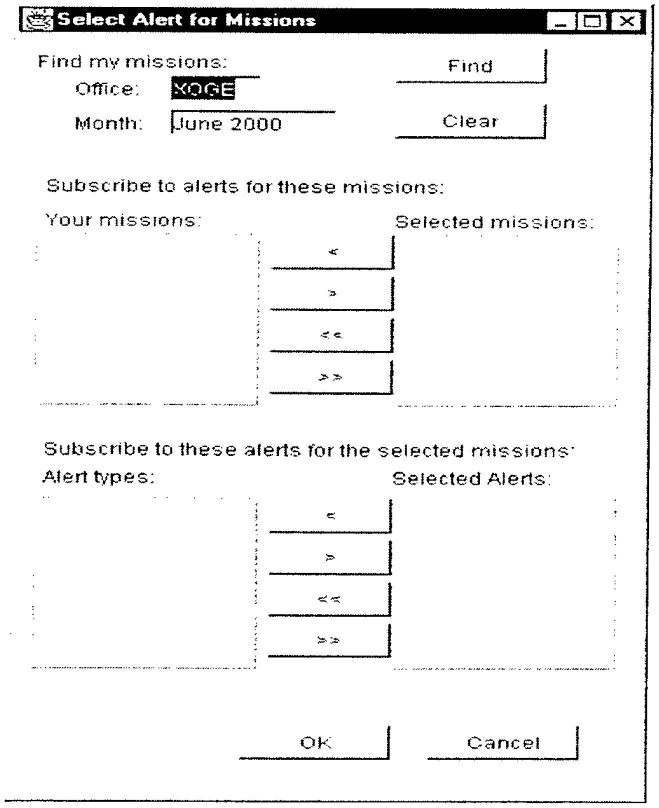
Development of Interface Mechanisms for Creating and Directing Intelligent Agents

In this project we developed three types of agents: GUI agents, database monitoring agents, and agents that can receive messages from other agents (via the TIE in the 1st Concept Demonstration). We also wanted to enable the user to task and monitor the agents. In order to support interaction of the human with the underlying software agents, Logicon developed several conceptual HCI modules: the Query Assistant, the Agent Query Assistant, and the Agent Viewer (a HCI display element intended to provide information on agents themselves). None of these designs were implemented under this project. However, two GUIs were developed to allow the user to tailor the types of alerts they want an agent to monitor with respect to a particular port or mission. Subscribe to Alerts (Figure 19), displays a tool intended for the user to set up alerts for a selected mission(s); a similar tool is available for setting up alerts at particular ports. The Select Alert for Missions window (Figure 20) presents a related tool that enables the user to specify how long they want the alert to be associated to the mission or port. These tools were built in response to the users' requests to build filters so they do not have to receive alerts for *all* ports. Instead, the users could specify certain types of alerts for certain ports for certain times. These GUIs are launched from the Smart LT window (Figure 10). These are expected to evolve in the follow-on activity.



Mission ID	Alert Types
ABB03F400069	PMOG, WMOG
AJB04F502069	PMOG
BBBJDF400069	PMOG, WMOG

Figure 19. Subscribe to Alerts


Select Alert for Missions

Find my missions:

Office: XOGE

Month: June 2000

Find

Clear

Subscribe to alerts for these missions:

Your missions:

Selected missions:

Subscribe to these alerts for the selected missions:

Alert types:

Selected Alerts:

Figure 20. Select Alert for Missions

D-OMAR provides the software developer (and interested user) several tools that can be used to examine the structure and behavior of the D-OMAR agents. For example, the D-OMAR Toolbar (Figure 21) provides links to the Concept Editor (Figure 22), the Procedure Browser (Figure 23), the Simulation Control/Run Time Trace window (Figure 24), and the Event and Task (agent) Timelines (Figures 25a and 25b).

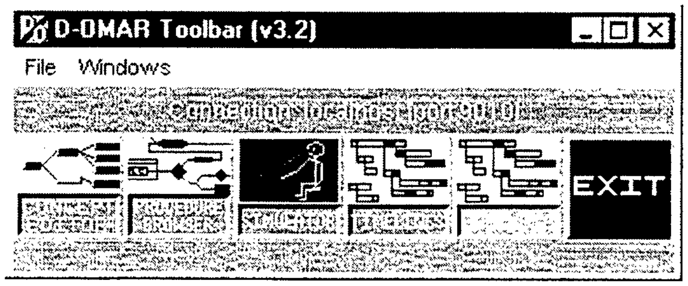
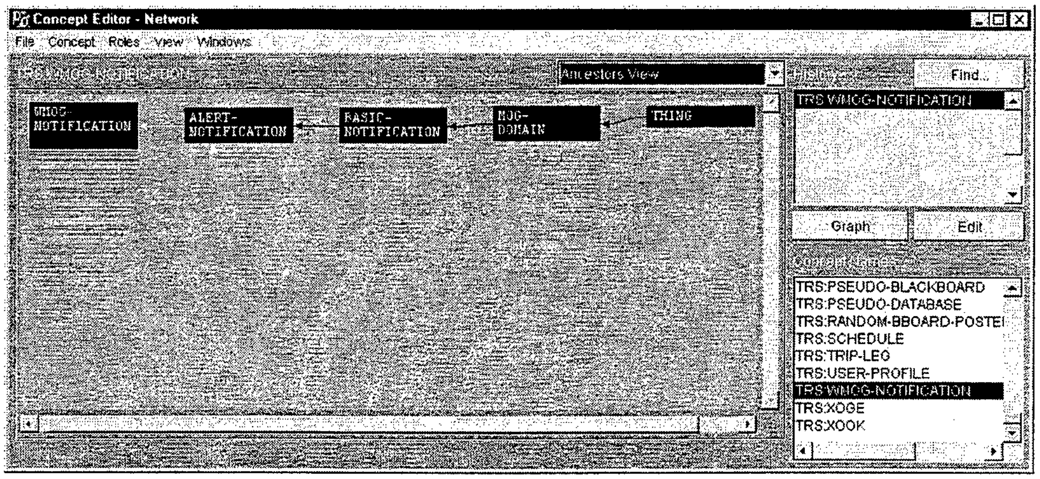


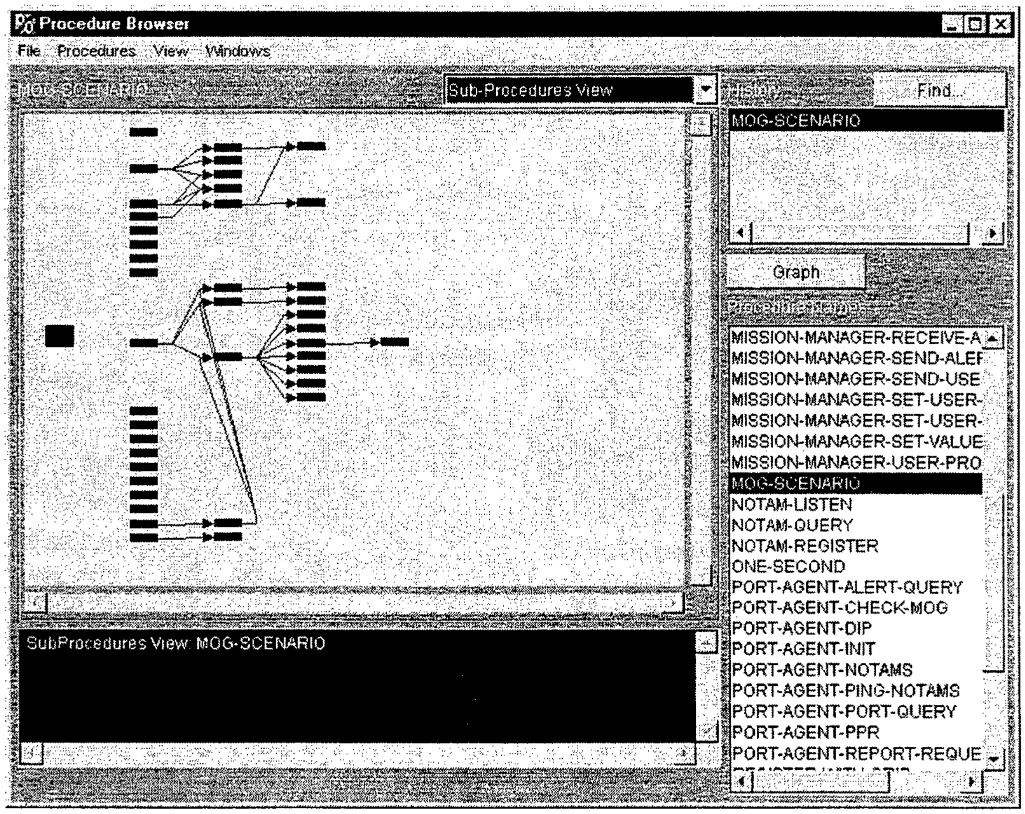
Figure 21. D-OMAR Toolbar



TRS:WMOG-NOTIFICATION

TRS:PSEUDO-BLACKBOARD
TRS:PSEUDO-DATABASE
TRS:RANDOM-BBOARD-POSTEI
TRS:SCHEDULE
TRS:TRIP-LEG
TRS:USER-PROFILE
TRS:WMOG-NOTIFICATION
TRS:XOGE
TRS:XOOK

Figure 22. Concept Editor

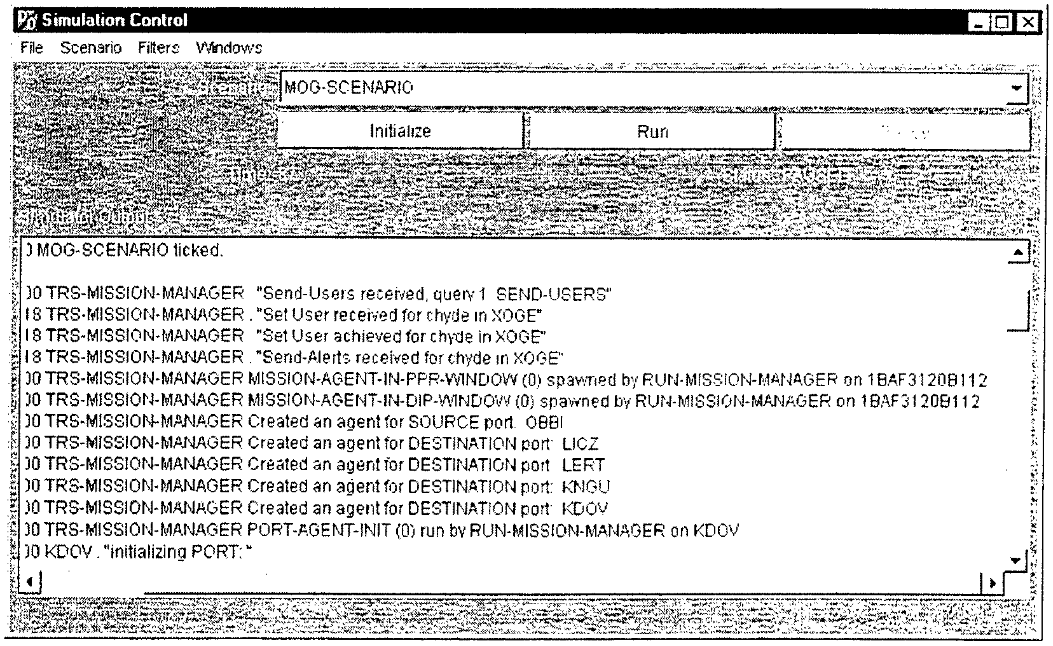


MOG-SCENARIO

MISSION-MANAGER-RECEIVE-A
MISSION-MANAGER-SEND-ALEF
MISSION-MANAGER-SEND-USE
MISSION-MANAGER-SET-USER-
MISSION-MANAGER-SET-USER-
MISSION-MANAGER-SET-VALUE
MISSION-MANAGER-USER-PRO
MOG-SCENARIO
NOTAM-LISTEN
NOTAM-QUERY
NOTAM-REGISTER
ONE-SECOND
PORT-AGENT-ALERT-QUERY
PORT-AGENT-CHECK-MOG
PORT-AGENT-DIP
PORT-AGENT-INIT
PORT-AGENT-NOTAMS
PORT-AGENT-PING-NOTAMS
PORT-AGENT-PORT-QUERY
PORT-AGENT-PPR
PORT-AGENT-REPORT-REQUE

SubProcedures View: MOG-SCENARIO

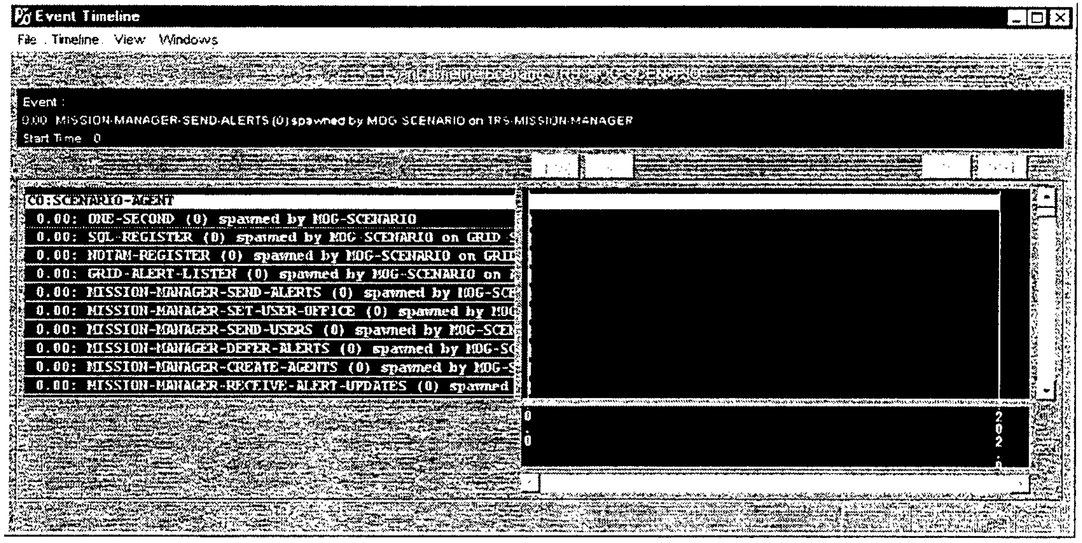
Figure 23. Procedure Browser



```
) MOG-SCENARIO ticked.

)0 TRS-MISSION-MANAGER  "Send-Users received, query 1  SEND-USERS"
18 TRS-MISSION-MANAGER . "Set User received for chyde in XOGE"
18 TRS-MISSION-MANAGER  "Set User achieved for chyde in XOGE"
18 TRS-MISSION-MANAGER . "Send-Alerts received for chyde in XOGE"
)0 TRS-MISSION-MANAGER MISSION-AGENT-IN-PPR-WINDOW (0) spawned by RUN-MISSION-MANAGER on 1BAF3120B112
)0 TRS-MISSION-MANAGER MISSION-AGENT-IN-DIP-WINDOW (0) spawned by RUN-MISSION-MANAGER on 1BAF3120B112
)0 TRS-MISSION-MANAGER Created an agent for SOURCE port  OBBI
)0 TRS-MISSION-MANAGER Created an agent for DESTINATION port  LICZ
)0 TRS-MISSION-MANAGER Created an agent for DESTINATION port  LERT
)0 TRS-MISSION-MANAGER Created an agent for DESTINATION port:  KNGU
)0 TRS-MISSION-MANAGER Created an agent for DESTINATION port  KDOV
)0 TRS-MISSION-MANAGER PORT-AGENT-INIT (0) run by RUN-MISSION-MANAGER on KDOV
)0 KDOV . "initializing PORT: "
```

Figure 24. Simulation Control / Run Time Trace



```
Event :
0.00  MISSION-MANAGER-SEND-ALERTS (0) spawned by MOG SCENARIO on TRS-MISSION-MANAGER
Start Time  0
```

```
C0:SCENARIO-AGENT
 0.00:  ONE-SECOND  (0)  spawned by MOG-SCENARIO
 0.00:  SQL-REGISTER  (0)  spawned by MOG SCENARIO on GRID
 0.00:  NOTAM-REGISTER  (0)  spawned by MOG-SCENARIO on GRI
 0.00:  GRID-ALERT-LISTEN  (0)  spawned by MOG-SCENARIO on
 0.00:  MISSION-MANAGER-SEND-ALERTS  (0)  spawned by MOG-SC
 0.00:  MISSION-MANAGER-SET-USER-OFFICE  (0)  spawned by MO
 0.00:  MISSION-MANAGER-SEND-USERS  (0)  spawned by MOG-SCE
 0.00:  MISSION-MANAGER-DEFER-ALERTS  (0)  spawned by MOG-S
 0.00:  MISSION-MANAGER-CREATE-AGENTS  (0)  spawned by MOG-
 0.00:  MISSION-MANAGER-RECEIVE-ALERT-UPDATES  (0)  spawned
```

Figure 25a. Event Timeline

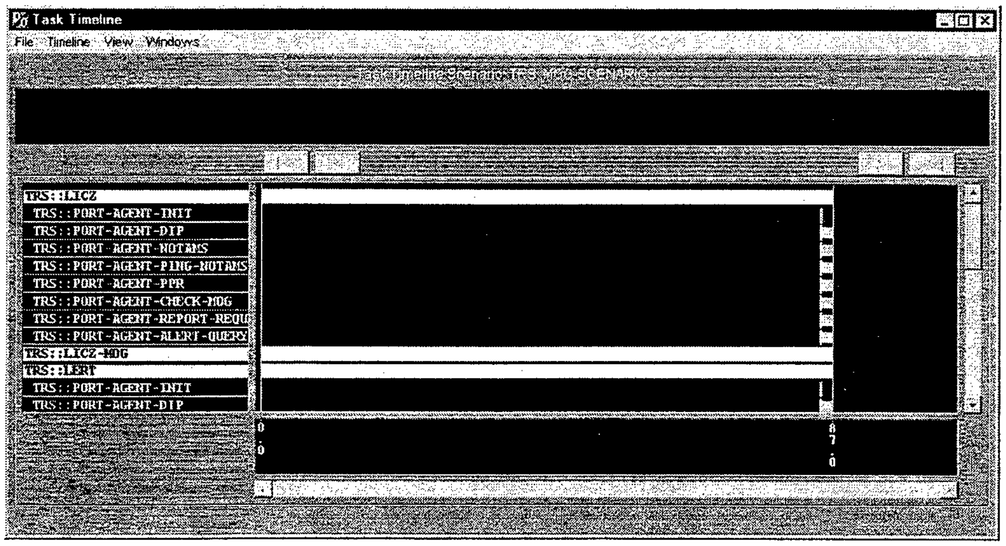


Figure 25b. Task Timeline

The Development of the Application Program Interface (API)

A set of data requirement tables were generated by Logicon to illustrate interrelationships among GUI elements "on the front end", agents "in the middle", and the AMC/TACC databases "on the back end". Two classes of such tables were produced: (a) a table organized with respect to the storyline for the scenario being written, and (b) a table for each of the GUI themselves.

A detailed API was also developed during the course of the project. In brief, it describes the interface between the HISA GUI process, which supports the various application windows and other elements of the user interface, and the D-OMAR engine, which accesses remote databases, generates and resolves alerts, stores user profiles, status, and history, and generally performs the data processing required by HISA. Appendix B contains a more complete version of the API.

A proposal for an alert taxonomy (Appendix C) was developed by Logicon in response to team members' questions about the type of alerts to be generated by the software agents. Over the course of the project, data requirements for each GUI were developed. Appendix D has a summary of the functional requirements.

A System Architecture

TASC Inc. and BBN Technologies drafted an architecture combining a JAVA GUI, D-OMAR agents, and CAST agents. The idea was to employ a JAVA cap to support the D-OMAR/JAVA interface. BBN also investigated engineering concerns associated with interacting with the Lockheed Martin CAST agents, external data sources, and the CoABS Grid. Figure 26 displays the System Architecture vision, which describes how agents could link the GUI to external data sources like GDSS, NOTAMS, and the Consolidated Air Mobility Planning System (CAMPS) or to other persistent storage areas such as the airfield restriction database (ShopTalk).

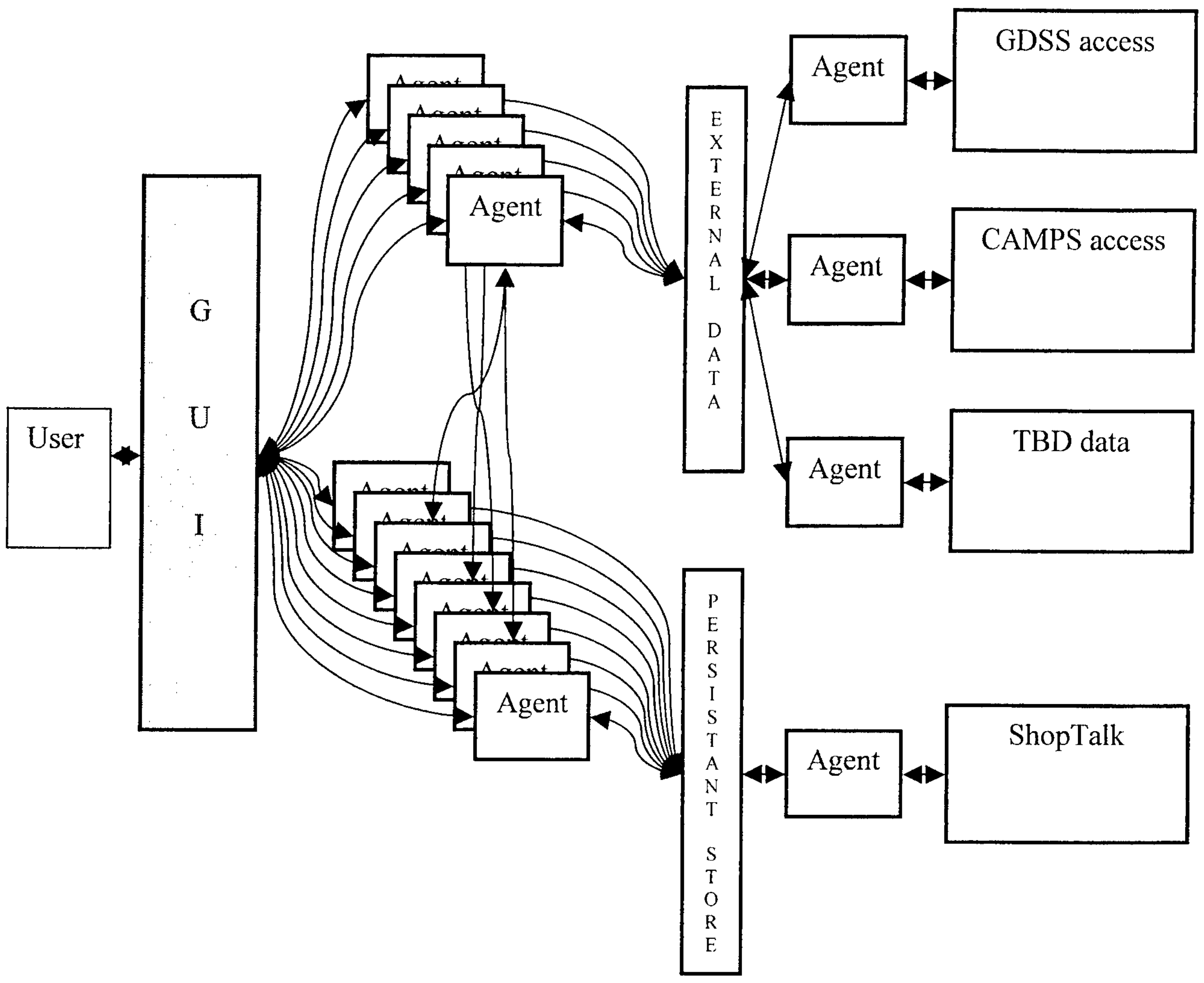


Figure 26. System Architecture

Other Issues - Some Problems

We encountered problems with computing and deploying temporal values in the Port Viewer. We also encountered problems in maintaining changes made in the GUI by the user. The biggest problems in the prototype were: (1) the delay from when you push a button to when you get a display (a function of JAVA); (2) the accurate computation of MOG (PMOG or WMOG) constraints, as the data we obtained was already constraint-free; and (3) the timing between simulated data exchanges and numeric real data exchanges. These problems will need to be addressed in the follow-on effort.

Other Issues - Value Added Metrics

Since the goal of the 2nd Concept Demonstration was to showcase some new functionality in a start-to-end working scenario during which a problem is identified and ultimately "resolved", a set of user-approved Value Added Metrics was applied to the 2nd demonstration (Table 2) to compare current practice to future capability.

ISSUE / CONFLICT	CURRENT PRACTICE	FUTURE PRACTICE
1. MOG	Spend 10 hrs/MO just checking; currency perishable. Run Station Workload Report only once.	Automatically check daily; data always current.
2. PPR	Spend 3 hrs/wk just tracking Sigonella PPRs; more hours to track other airports.	"Automatically Reminded" PPRs.
3. NOTAM	Do not check NOTAMs; blindsided by changes.	Automatically notified if pertinent NOTOAM issued.
4. DIP	Need to know if "problem mission" has a Diplomatic Clearance.	Automatically notified if DIP clearance involved.

Table 2. Value Added Metrics

Other Issues - Defense Information Infrastructure - Common Operating Environment (DII-COE) Level Compiance

The DII-COE is a set of software specifications that form the foundation for mission applications. It is a fundamentally new approach emphasizing both software reuse and interoperability. The DII-COE is a foundation for building an open system. The goals of the DII-COE are to: guarantee operation of any compliant application on any compliant platform; promote interoperability through access to data independent of its location; and provide a reliable, scalable operating environment that is thoroughly tested. Since the DII-COE standards were associated with interoperability, we chose to leave the issue of DII-COE compliance until the follow-on effort where integration with other systems will be more important. In addition, since the result of this effort is a concept demonstration, the team did not determine it necessary to adhere to the DII-COE interface guidelines, reserving the implementation of these guidelines to some follow-on, more operationally oriented effort.

References

1. Horvitz, E., “Principles of Mixed-Initiative User Interface”, pp. 159–166, of CHI99 Conference Proceedings, AMC Press, 1999.

2. Kaminsky, M., Dourish, P., Edwards, W.K., LaMarca, A., Salisbury, M., and Smith, I., “Sweetpea: Software Tools for Programmable Embodied Agents”, pp. 144-151 of CHI99 Conference Proceedings, ACM Press, 1999.

3. Mayhew, D., “The Usability Engineering Lifecycle”, Morgan Kaufmann Publishers, Inc., San Francisco CA, 1999.

4. Shneiderman, B., “Designing the User Interface: Strategies for Effective Human Computer Interaction", Addison Wesley, Reading MA, 1998.

5. Winograd, T., “Bringing Design to Software”, Addison Wesley, Reading MA, 1996.

List of Acronyms

AFRL	Air Force Research Laboratory
AMC	Air Mobility Command
API	Application Program Interface
CAMPS	Consolidated Air Mobility Planning System
CAST	Cooperating Agents for Specific Tasks
CoABS	Control of Agent Based Systems
DARPA	Defense Advanced Research Projects Agency
DII-COE	Defense Information Infrastructure-Common Operating Environment
DIP	Diplomatic Clearance
D-OMAR	Distributed Operator Model Architecture
GDSS	Global Decision Support System
GUI	Graphical User Interface
HCI	Human Computer Interface
HISA	Human Interaction with Software Agents
MOG	Maximum on the Ground
NOTAM	Notice to Airmen
PMOG	Parking Maximum on the Ground
PPR	Prior Permission Request
SME	Subject Matter Expert
SQL	Structured Query Language
TACC	Tanker Airlift Control Center
TIE	Technical Integration Experiment
WMOG	Working Maximum on the Ground

Appendix A

Knowledge Acquisition Efforts and Results

Collection of Task Analytic and Business Rules

Knowledge acquisition and task analysis methods were employed to collect process models for eliciting domain knowledge from domain experts. Use cases and storyboards were developed to focus development as a way to define the concept demonstration scenarios.

Process flow and rules were collected early in the project. Previous process model work from 1992 to 1993 with the TACC was explored. Scenarios replaced the process flows. Less structured data sheets and documents replaced previously hard rules. It became clear that the channel mission data had to be linked to geographical data, and more specifically, to airfield restriction data and to the NOTAM data. An early approach involved the transformation of the formatted airfield restriction data that we received from the users into Oracle database records. A web-based browser was created for airfield restriction data and mechanisms were built to link the airfield restrictions to NOTAM via the Ports' name. In doing this work, it became obvious that methods needed to be developed to parse the NOTAM data so that the user could be informed of specific data, e.g., airfield closure information, changes to airfield capabilities and services, etc. related to their missions.

Problem Area Defined

The initial problem defined was to focus on Special Assignment Airlift Missions (SAAMs). However, in meetings with the Air Force Research Laboratory (AFRL) and the TACC, the focus shifted to an agent effort that concentrated on tracking aircraft scheduled into Air Mobility Command (AMC) operating locations to ensure that channel, SAAM, Joint Chiefs of Staff (JCS) exercise, and contingency schedulers, planning independently, did not exceed the MOG capacity of any particular airfield. The users indicated that they would like to see an agent monitor the NOTAM system and warn planners when one of their missions was impacted by a change to operating hours or procedures. Currently, once a mission is scheduled, personnel must monitor flights in the system using a long, tedious pen/paper method. Frequently, no one is able to monitor the NOTAM system for such changes until the aircrew checks them at base operations just prior to filing a flight plan. By then it's too late to efficiently reschedule the affected mission.

With this area identified, knowledge acquisition efforts focused on obtaining expanded descriptions of the TACC XOG Global Channel Operations domain and several use cases were developed. Three use cases were developed during the project and each was coordinated with the XOG (Channel Planning) staff before being used by the HISA team.

The three use cases include:

Schedule Organic Mixed and Cargo Channel (Figure 1)
Establish New Channel (Figure 2)
Schedule Commercial Passenger Channel (Figure 3)

Screen snapshots of an actual Master Template File (MTF) and Route Set Template (RST) were obtained from the domain specialists and provided to the HISA team for use in developing the GUIs. A Channel Operations Domain slide with accompanying descriptive text was also developed and provided to XOG for critical review. Finally, an initial concept for a Storyboard was developed to support HCI and agent design as well as for developing the concept demonstrations.

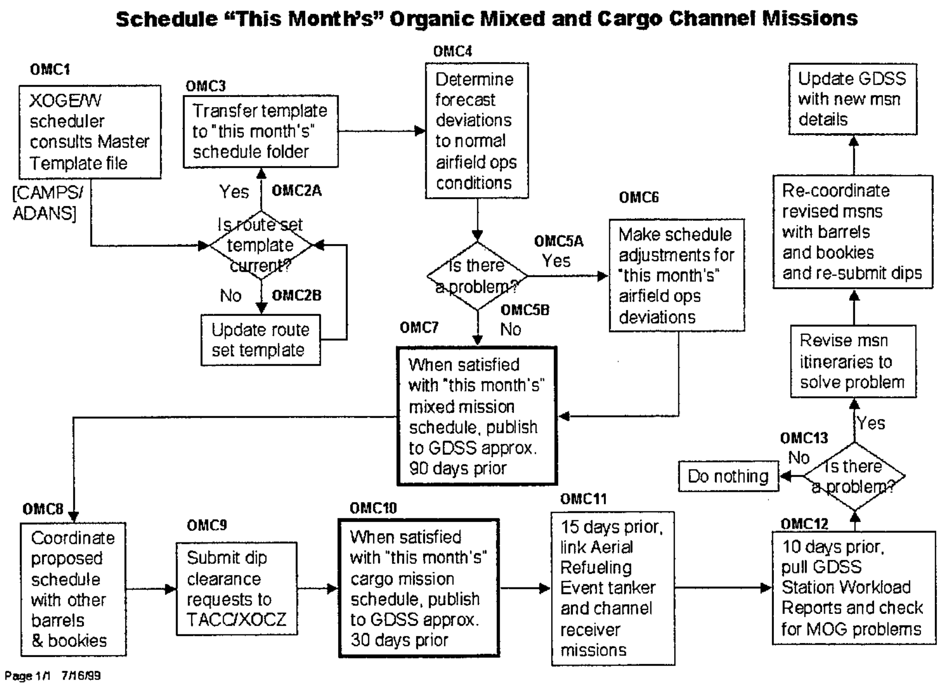

Schedule "This Month's" Organic Mixed and Cargo Channel Missions
OMC1
XOGE/W scheduler consults Master Template file
[CAMPS/ ADANS]
OMC3
Transfer template to "this month's" schedule folder
Yes
OMC2A
Is route set template current?
No
OMC2B
Update route set template
OMC4
Determine forecast deviations to normal airfield ops conditions
Is there a problem?
OMC5A
Yes
OMC5B
No
OMC6
Make schedule adjustments for "this month's" airfield ops deviations
OMC7
When satisfied with "this month's" mixed mission schedule, publish to GDSS approx. 90 days prior
Update GDSS with new msn details
Re-coordinate revised msns with barrels and bookies and re-submit dips
Revise msn itineraries to solve problem
Yes
OMC13
No
Do nothing
Is there a problem?
OMC8
Coordinate proposed schedule with other barrels & bookies
OMC9
Submit dip clearance requests to TACC/XOCZ
OMC10
When satisfied with "this month's" cargo mission schedule, publish to GDSS approx. 30 days prior
OMC11
15 days prior, link Aerial Refueling Event tanker and channel receiver missions
OMC12
10 days prior, pull GDSS Station Workload Reports and check for MOG problems
Page 1/1 7/16/99

Figure 1. Schedule Organic Mixed and Cargo Missions

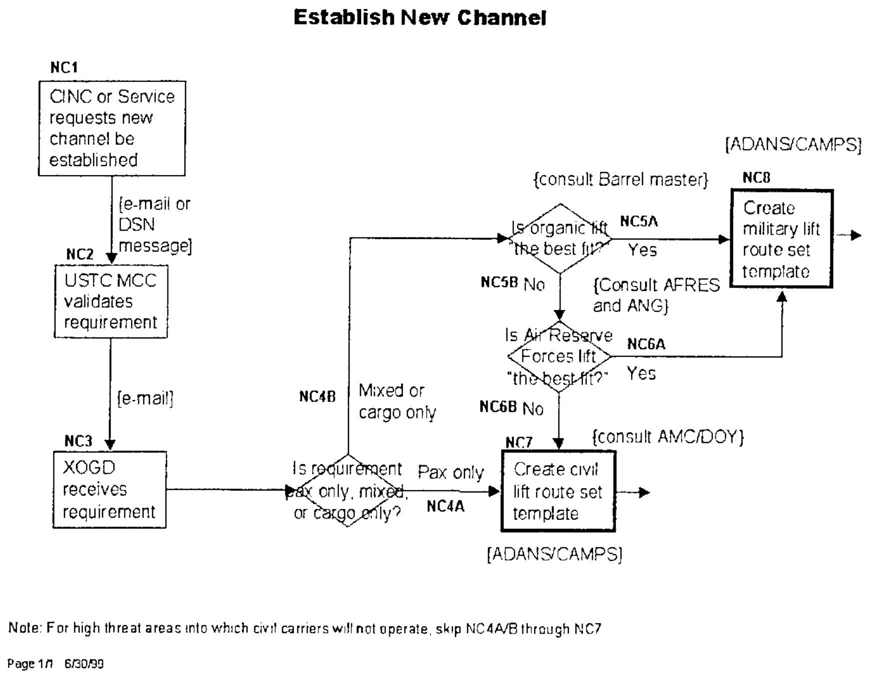

Establish New Channel
NC1
CINC or Service requests new channel be established
{e-mail or DSN message}
NC2
USTC MCC validates requirement
{e-mail}
NC3
XOGD receives requirement
Is requirement pax only, mixed, or cargo only?
Pax only
NC4A
NC4B
Mixed or cargo only
{consult Barrel master}
Is organic lift "the best fit?"
NC5A
Yes
NC5B No
{Consult AFRES and ANG}
Is Air Reserve Forces lift "the best fit?"
NC6A
Yes
NC6B No
{consult AMC/DOY}
NC7
Create civil lift route set template
[ADANS/CAMPS]
[ADANS/CAMPS]
NC8
Create military lift route set template
Note: For high threat areas into which civil carriers will not operate, skip NC4A/B through NC7
Page 1/1 6/30/99

Figure 2. Establish New Channel

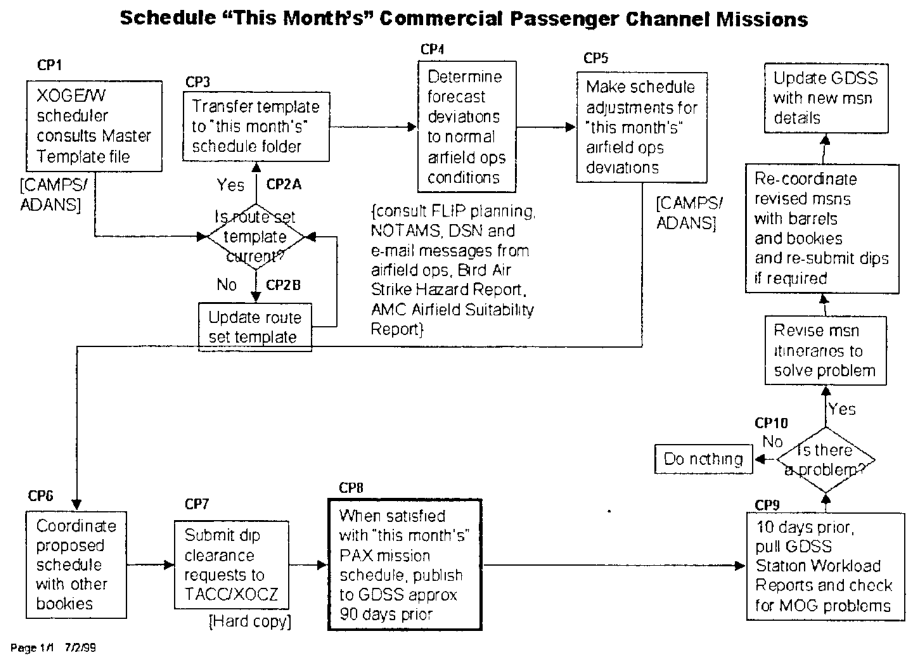

Schedule "This Month's" Commercial Passenger Channel Missions
CP1
XOGE/W scheduler consults Master Template file
[CAMPS/ ADANS]
CP3
Transfer template to "this month's" schedule folder
Yes
CP2A
Is route set template current?
No
CP2B
Update route set template
CP4
Determine forecast deviations to normal airfield ops conditions
{consult FLIP planning, NOTAMS, DSN and e-mail messages from airfield ops, Bird Air Strike Hazard Report, AMC Airfield Suitability Report}
CP5
Make schedule adjustments for "this month's" airfield ops deviations
[CAMPS/ ADANS]
Update GDSS with new msn details
Re-coordinate revised msns with barrels and bookies and re-submit dips if required
Revise msn itineraries to solve problem
Yes
CP10
No
Is there a problem?
Do nothing
CP6
Coordinate proposed schedule with other bookies
CP7
Submit dip clearance requests to TACC/XOCZ
[Hard copy]
CP8
When satisfied with "this month's" PAX mission schedule, publish to GDSS approx 90 days prior
CP9
10 days prior, pull GDSS Station Workload Reports and check for MOG problems
Page 1/1 7/2/99

Figure 3. Schedule Commercial Passenger Channel Missions

An initial draft Storyboard depicted tasks the Channel Operations staff perform and suggested HCI capabilities and software agents' interaction. The scenario was repeatedly updated and coordinated for accuracy with the TACC planners. A table that links the A task-HCI-system-data element thread was also developed (Table 1).

AR	Air Refueling mission
CAMPS	Database (Consolidated Air Mobility Planning System)
GDSS	Database (Global Decision Support System)
JCS	Joint Chiefs of Staff
MOG	Maximum # of aircraft allowed on ground (Maximum on the Ground)
NAS Rota	Naval Air Station Rota, Spain
SWR	Station Workload Report
XOG	Channel Planners
XOGE	East Cell Channel Planners
XOOK	Current Operations Air Refueling Division
XOPE	Global Readiness

TASK / EVENT	INTERACTION	DATA & SOURCE	NOTES
1. XOGE publishes channel mission to GDSS.	CAMPS to GDSS.	1B3 msn; XOG Route Set Template.	5 weeks ago.
2. XOOK publishes AR msn to GDSS in support of contingency ops.	CMARP-CAMPS to GDSS.	1B1 msn; XOOK.	(Today) 3 days before channel executes.
3. Identify MOG conflict.	NAS Rota notifies XOGE.	Verbal notification; Rota noticed conflict in GDSS.	Telephonic.
4. Validate report of MOG conflict.	XOGE runs Station Workload Report (SWR) for Rota.	SWR results from GDSS.	Confirms that new MOG conflict exists.
5. Initiate conflict coordination with XOPC.	XOGE walks over to XOPC to talk.	Verbal, based on GDSS report; USEUCOM late decision for msn requirement.	Unable to change contingency msn.
6. Consider options; XOGE reviews "what if".	XOGE Mentally, determines 10-hour delay and implications.	Channel msn in GDSS; ripple effect on other ports.	Aware that a change will affect other ports.
7. Determine impact on Bahrain. Determine	XOGE views Atlantic Theater	Airfield Restriction DB for station ops	Daylight hours only.

MOG at Bahrain.	Airfield Restrictions db. Pulls SWR in GDSS for Bahrain.	hours at Bahrain. Identifies in GDSS more missions (2B1 Exercise c/o 2 C-141s and 1 comm PAX flight).	Working MOG is 1, with one-hour separation between a/c. Determines option will disrupt exercise msns.
8. Review JCS priorities for conflicting missions.	XOGE reviews which msn "out-prioritizes" who.	JSC priority in GDSS per mission.	Channel (1B3) out-prioritizes exercise (2B1).
9. Initiate conflict coordination with XOPE (exercise planner).	XOGE walks over to XOPE to talk.	Verbal, based on data in GDSS; tells XOPE they will have to change ex msn.	Dismay; angst. Will affect 10-day exercise flow. Too late to change commercial carrier, or Large $ penalty.
10. Determine impact on Sigonella, Italy.	XOGE Bookie calls aerial port.	Verbal; Port will not support 10 hour delayed arrival, due to > cost for overtime local labor to offload a/c.	Italian national port workers.
11. Continued coordination with XOPE and USCENTCOM.	XOPE calls USCENTCOM. XOPE tells XOGE results.	Verbal; Centcom will not accept disruption to flow, threatens General Officer pressure.	More angst. Need more options.
12. Consider briefing TACC command on the problem.	Mentally, determines 2 days of time preparing and briefing.	Mentally dismisses this route, knowing that Sigonella won't agree to a delay anyway.	Need another option.
13. Check Rota for more information.	XOGE calls Rota base ops.	Verbal. Rota Ops officer offers one narrow body (NB) parking spot during the scheduled channel RON time.	A C5 won't fit in a narrow body slot. Need more information/options.
14. Check with Barrel master for a different type a/c.	XOGE walks over to Barrel; asks for help. Discuss option.	Barrel has a SAAM msn cancellation; a C17 w/crew is available for the channel msn. With a	C17 carries less pallets (18) than a C5 (36). Need port's agreement to < lift capacity.

		"wing walker" can fit a C17 in to NB slot.	
15. Check for High Priority cargo at the ports, and it's "age" to see if < lift capacity is OK.	Channel planner and his Bookie need to know if hi-pri cargo is waiting.	Check Global Channel Ops Briefing Summary (reports amount and age of cargo waiting at ports).	The Summary shows that < lift would be feasible to use.
16. Check with the ports to confirm what the Summary is reporting.	Bookie calls Norfolk, Rota, Sigonella and Bahrain.	Verbally, confirms amount of cargo awaiting lift.	A C17 would adequately support this channel msn. Are there any other considerations?
17. Check for any special designations for this mission.	XOGE checks in GDSS.	A "Z" code in 9th character of the mission ID indicates, "fenced channel" (not to be cancelled).	What impact would swapping type a/c have on this "fenced" msn?
18. Check with Barrel for impact of swapping type a/c for fenced msn.	XOGE walks over to Barrel.	Verbal. May be a reason cannot swap type (crew training, maintenance planning, customer reqm't).	Swap is OK. Mission can go via C17.

Table 1. Task-System-Data Source Threads

Acquisition of Domain Data

As part of our program, we used a CD of actual GDSS database elements from the April 1999 timeframe for demonstrations, as was airfield restriction data for both the Atlantic Region and the Southern Region. Additional missions were developed to introduce conflict with in support of the scenario.

We also collected information about the office code development (it is derived from the AMC mission number) and the "Channel Bulletin Part C", which provides detailed information on the AMC Mission Identifier Program Encode/Decode tables. The Channel bulletin information was converted into a set of rules that could be implemented in the D-OMAR simulation.

In dealing with NOTAM information, we used the Air Force Joint Manual 11-208, DoD NOTAM System.

Functional Requirements Documents

Detailed user feedback was obtained in follow-on knowledge acquisition sessions with the domain specialists after the 1st Concept Demonstration. The results were compiled into a detailed set of functional requirements. Gaps in the functional requirements drove further research into the repository of data previously acquired with questions to TACC domain specialists regarding the specifics of MOG calculations, JCS priority, and Airfield Restrictions to name a few.

Development of the Interface between the Demonstration system and Actual AF Systems

The original project was interested in building agents that could communicate with AF database systems, e.g., GDSS and CAMPS. We prepared the 1st Concept Demonstration to communicate with the GDSS database, but through the Lockheed Martin CAST software agents. The 2nd Concept Demonstration included D-OMAR software agents capable of listening to a limited number of NOTAM changes. More work needs to be done in this area for an operational system.

Appendix B

HISA User Interface Application Program Interface (API) to Distributed Operator Model Architecture (D-OMAR)

1. Purpose of this Document

This document describes the interface between the HISA GUI process, which supports the various application windows and other elements of the user interface, and the D-OMAR engine, which accesses remote databases, generates and resolves alerts, stores user profiles, status, and history, and generally performs the data processing required by HISA.

2. Overview of Communications between the GUI and D-OMAR

In this section we provide a general overview of the communication mechanism between the GUI and D-OMAR.

2.1 Events

Communication between the HISA GUI and D-OMAR occurs via the mechanism of D-OMAR's Java Cap. The Java Cap consists of classes that enable Java programs (such as the HISA GUI) to communicate with the LISP-based D-OMAR process, by sending or receiving events to or from D-OMAR.

In the Java Cap, an event is essentially an asynchronous one-way message to or from D-OMAR. D-OMAR will from time to time spontaneously generate an event to the GUI (e.g., because it has detected a conflict to which the user asked to be alerted); therefore the GUI implements a handler for these asynchronous events. Likewise the GUI will from time to time generate an event to D-OMAR due to user input; D-OMAR listens for these events.

HISA adopts a simplified format of D-OMAR events, in which the contents of every event are a data structure equivalent to a list of strings. The length of the list depends on the event. Numeric data is converted to a string format by the sender and parsed by the recipient.

2.2 Event Syntax

The first element of each event is the type of the event. (In the case of events sent by the GUI to D-OMAR, the Java Cap requires that the type of this element be of a special Java class representing a LISP symbol. This is the only element in the HISA events that is not actually of type String. Only the equivalent string value of the symbol is of interest to us, however, so throughout the rest of this document we will refer to this symbol by its string value, just as we refer to the other elements of events.)

Following the type are zero or more parameters. Each parameter consists of an identifier, followed by a value. The purpose of the identifier is to specify the semantics of the parameter. If we think of the event as a structure with several named fields, the identifiers are the names of the fields.

In order to avoid confusing identifiers with values, each identifier begins with a dash (-). When it is necessary to pass as a data value a string that begins with a dash, the sender shall prefix the string with the sequence dash-blank ("- ") and the recipient shall Strip this sequence from the start of the string. That is, in order to send the string "--none--", for example, the event shall contain the string "- --none--".

2.3 Query-Response Events

In addition to asynchronous one-way events, the HISA interface needs to support various user queries that require a query-response or remote procedure call semantics. In order to support this, a pair of events is employed:

* A query event from the GUI to D-OMAR.

* A response event from D-OMAR to the GUI.

The query event always includes an integer "-queryID" value, and D-OMAR's response to that query will always echo the same "-queryID" value that the GUI sent in the query event. Because the response includes this matching "-queryID" value, the GUI can always identify the query that the response answers, and so can complete the ``procedure call" even if other events have occurred between the generation of the query and the response.

3. Types of Events

In this section we describe the details of the interface between the GUI and D-OMAR, consisting of the various types of events that may pass between them, and the semantics of each event type including the interpretation of its fields.

3.1 Syntax

As described in the previous section, an event consists of an event type (a string or string equivalent) followed by zero or more additional strings. Any additional strings consist of alternating field names and values. The fields that occur in the event will depend upon the event type, as described in this section.

To describe the detailed syntax of events, we use the following notation:

Quoted text represents the literal text of a string, e.g., "xyz" represents a three-character string consisting of the lower case letter x followed by y and z.

A numeral represents the literal value of an integer, converted into a string (the decimal representation of the integer).

Text in angle brackets represents an element or elements that are to be assigned a value at run time (the entire text including the enclosing angle brackets will be replaced by some value). The text inside the brackets consists of descriptive text, followed by a colon, followed by the type of the value.

The syntax [x|y], where x and y may be literal or run-time values, indicates that either x or y (but not both) may appear in the indicated place in any single instance of the event.

Italicized text is a placeholder for some syntactical group within an event, for example a list of several name-value pairs. Whenever such a placeholder appears in an event syntax, the syntax of the represented group will be provided later in the description of the event.

3.2 Types of Values

We distinguish three types of values in events.

* string. The most generic value type is string. Any string value may be contained in this field.

* integer. The value of this field is an integer (that is, the recipient should parse this string as a decimal integer in order to recover the integer value of the field).

* time. The value of this field is an integer representing a date and time of day. The integer value is a Unix-style time value, that is, the number of seconds that will have elapsed from January 1, 1970 to the date and time in question.

3.3 Categories of Interactions

The events passed between the GUI and D-OMAR support interactions between the GUI and D-OMAR that fall into three categories:

* One-way D-OMAR to GUI. D-OMAR passes a single asynchronous event to the GUI.

* One-way GUI to D-OMAR. The GUI passes a single asynchronous event to D-OMAR.

* Query-response. The GUI sends a query event to D-OMAR, after which D-OMAR sends a matching response event to the GUI.

At this time, query-response interactions are never initiated by D-OMAR.

In the detailed list of event types, one-way interactions (in either of the first two categories) are described individually. If the recipient's handling of the event triggers any processing that generates additional events, those events are described separately.

For a query-response interaction, we first describe the syntax of the query event, followed by the syntax of the matching response event.

3.4 Detailed List of Event Types

The following list describes the interactions between the GUI and D-OMAR, and the event types that support these interactions.

Interaction: REPORT ALERT (one way D-OMAR to GUI)

When a conflict or other alertable condition is detected, D-OMAR passes an event to the GUI:

"Alert-notification" *header detail-list*

The *header* consists of the following fields in any order:

```
"-alertType" <alert type:string>
"-issue" <issue:string>
"-eventID" <unique ID generated by D-OMAR:integer>
"-status" <status:string>
"-alertLevel" <degree of severity of the alert, where
                1=low severity, 2=medium, 3=high severity:integer>
"-notification" <number of times this detection has been raised:integer>
"-datasource" <describe source of data:string>
"-timestamp" <instant at which D-OMAR (re)raised the alert:time>
```

The "-alertType" and "-issue" values depend on what caused the alert. If the alert is for a Working MOG conflict, these are:

```
"-alertType" "conflict"
"-issue" "WMOG"
```

For a Parking MOG conflict:

```
"-alertType" "conflict"
"-issue" "PMOG"
```

For a change in NOTAMs:

```
"-alertType" "change"
"-issue" "NOTAM"
```

For the beginning of the window for submitting a PPR request:

```
"-alertType" "window entry"
"-issue" "PPR"
```

When an alert is initially raised, the "Alert-notification" event is sent with the value "-status" "detected". The only other possible values at this time occur during the "Alert History Replay" protocol. (See the note on this below.)

The *detail-list* consists of the following fields in any order:

```
"-location" <ICAO code of airfield where MOG exceeded:string>
"-portName" <full English name of airfield:string>
"-beginDate" <instant at which condition is predicted to begin:time>
"-endDate" <instant at which condition is predicted to end:time>
"-mission" <mission number:string>
```

For the anticipated future, all fields in detail-list will occur exactly once, except that the "-mission" field does not occur at all in a NOTAM alert.

The "-eventID" value can be used as an identifier to help make queries about this alert (as opposed to any other alert).

The "-notification" value is 1 the first time an alert is detected. Each time the alert is deferred and raised again by D-OMAR, the "-notification" value increases by 1.

The "-timestamp" value is the time when D-OMAR first sent the "Alert-notification" event to the GUI, unless the alert was deferred and the deferral period has run out (see below). Note that after either the first report or after a deferral period expires and the alert is reported again, it is possible for D-OMAR to generate additional reports generated (see "Alert History Replay", below), but D-OMAR will continue to use the same timestamp value.

When an alert has been deferred, D-OMAR continues to track the alert. If the deferral period expires and the condition has not been resolved, then D-OMAR will resend the "Alert-notification" event using the same "-eventID" value as in the first time that alert was reported, but with a new timestamp.

Alert History Replay

In the current implementation, when the user logs out and logs back in again, the GUI reloads the Alert History from D-OMAR. To do this:

First, the GUI sends the "Alert-request" event (see RESEND ALERTS, below).

Second, D-OMAR proceeds to send an "Alert-notification" event for every alert in the alert history (i.e., for everything that happened since the user's very first login). These messages are similar to the "Alert-notification" messages that are sent when alerts are first detected, with the following exceptions:

1. The "-status" value depends on the current status of the alert.
   If the alert has been deferred by the user, then this field is

   "-status" "deferred"

   If the user has disregarded the alert, then this field is

   "-status" "disregarded"

   If the alert has been resolved, then this field is

"-status" "resolved"

Otherwise the field is

"-status" "detected"

just as it was in the original Alert-notification. Only alert with "-status" "detected" belong in the Alert Queue.

2. The following optional fields may appear in the *details* section of the message:

   If the user has been advised of the alert:

   "-advised" <date when user was advised:time>

   If the user has acknowledged the alert:

   "-acknowledged" <date when alert was acknowledged:time>

   If the user has deferred the alert:

   "-deferred" <date when alert was deferred (to?):time>
   "-deferText" <reason for deferral:string>

   If the user has disregarded the alert:

   "-disregard" "Y"

Interaction: RESEND ALERTS (one way GUI to D-OMAR)

In order to replay the history of alerts, the GUI sends the following to D-OMAR:

"Alert-request" "-userID" <ID of the user:string>

Following this, D-OMAR will proceed to send "Alert-notification" events for every alert in the alert history. The GUI should handle these as described under "Alert-notification", above.

This event is useful when a user logs in. It will cause D-OMAR to resend the alerts that the user received during the previous login, plus any alerts that occurred while the user was logged out, so that the GUI can reconstruct up-to-date copies of the alert queue and the alert history. If the GUI has performed GET ALERT QUEUE TIMESTAMP, it can now set the "New alerts" indicator in the opening palette as needed (cf. the discussion of the timestamp under REPORT ALERT).

Interaction: GET CONFLICT SUMMARY (query-response)

In order to get details of the conflicting missions in a MOG conflict, the GUI sends the following to D-OMAR:

```
"Alert-query" "-eventID" <unique ID from Alert-notification event:integer>
  "-queryID" <ID generated by GUI: integer>
  "-detail" "conflict"
```

D-OMAR responds:

"Alert-reply" "-queryID" <queryID from Alert-query event:integer>
  "-alertType" "conflict"
  "-issue" "MOG"
  "-detail" "conflict"
  "-location" <ICAO code of airfield where MOG exceeded:string>
  "-portName" <full English name of airfield:string>
  "-date" <when conflict occurs:time>
  "-numMissions" <number of missions:integer> *mission-list*

where the *mission-list* consists of number-of-missions repetitions of

  "-mission" <mission number:string> *mission-details*

where *mission-details* consists of the following fields in any order:

```
  "-own" ["Y"|"N"]
  "-office" <office name:string>
  "-priority" <JCS priority of mission:string>
  "-advised" ["Y"|"N"|"?"]
  "-acknowledged" ["Y"|"N"|"?"]
  "-canRedo" ["Y"|"N"|"?"]
  "-closeWatch" ["Y"|"N"|"?"]
  "-disregard" ["Y"|"N"]
```

"-own" indicates whether the office making the query schedules that particular mission. (For missions scheduled by other offices, the value here is "N".)

Interaction: GET ALERT SETTING (query-response)

In order to get details of the setting of a MOG conflict, the GUI can send to D-OMAR the following:

```
"Alert-query" "-eventID" <unique ID from Alert-notification event:integer>
  "-queryID" <ID generated by GUI:integer>
  "-detail" "setting"
```

D-OMAR responds:

```
"Alert-reply" "-queryID" <queryID from Alert-query event:integer>
  "-alertType" "conflict"
  "-issue" ["WMOG"|"PMOG"]
  "-detail" "setting"
  "-location" <ICAO code of airfield where MOG exceeded:string>
  "-portName" <Full common name of airfield:string>
  "-sunrise" <time of sunrise:time>
  "-sunset" <time of sunset:time>
  "-opsBegin" <time airfield ops begin:integer>
  "-opsEnd" <time ops end:integer>
  "-conflictBegin" <date and time conflict begins:time>
  "-conflictEnd" <date and time conflict ends:time>
  "-mog" <MOG limit value that was exceeded:integer>
  "-numAC" <number of aircraft in the conflict:integer>
  "-numMissions" <number of missions:integer> mission-list
```

where the *mission-list* consists of number-of-missions repetitions of

```
  "-mission" <mission number:string> mission-details
```

where *mission-details* consists of the following fields in any order:

```
  "-own" ["Y"|"N"]
  "-office" <office name:string>
  "-priority" <JCS priority of mission:string>
  "-actype" <type of aircraft:string>
  "-width" <1 for narrow body, 2 for wide:integer>
  "-land" <date and time when mission lands:time>
  "-takeoff" <date and time when mission takes off again:time>
```

"-own" indicates whether the office making the query schedules that particular mission. (For missions scheduled by other offices, the value here is "N".)

Interaction: DEFER ALERT (one way GUI to D-OMAR)

In order to inform D-OMAR that the user has deferred an alert (e.g., via the Defer Rationale window), the GUI sends the following to D-OMAR:

"Alert-deferral" *header-fields*

The *header-fields* consist of the following fields in any order:

```
  "-eventID" <unique ID generated by D-OMAR:integer>
  "-reason" <explanatory text supplied by user:string>
  "-wakeup" <when the alert should be raised again:time>
  "-disregard" ["Y"|"N"]
  "-userID" <user ID:string>
```

The "-wakeup" field always contains an absolute date/time value. If "-disregard" "N" is given, D-OMAR will raise the alert again at the time specified by "-wakeup", if it has not already been resolved. If the user requested to defer the event for a relative time, e.g. "until 3 days from now," the GUI should read the current system time and compute the absolute future time at which the user's deferral period would expire.

If "-disregard" "Y" is given, then a wakeup time should still be specified, but D-OMAR will ignore the wakeup time, and will never raise this alert again.

D-OMAR then defers the alert, i.e. removes it from its saved copy of the user's alert queue and adds a deferral event to the audit trail in the user's alert history.

Interaction: SET ALERT QUEUE TIMESTAMP (one way GUI to D-OMAR)

In order to record the latest alert that the user has been informed of, the GUI sends the following to D-OMAR:

Syntax:

```
    "Set-value" "-userID" <user ID:string> <latest timestamp in Alert Queue:time>
```

This helps the GUI save and recover the status of the Alert Queue and Smart LT via D-OMAR, to support the "New alerts since queue last viewed" feature. The "-userID" field identifies the user whose latest timestamp this is, since other users typically will have different timestamp values.

The exact use of this message depends on the precise interpretation of when the user last "viewed" the Alert Queue. But, for example, if the viewer opens the Alert Queue, then it seems safe to conclude that the user has viewed the alerts there, and the GUI might SET

ALERT QUEUE TIMESTAMP using the latest timestamp that is then in the queue. Then, if the user logs out and in again, the GUI can GET ALERT QUEUE TIMESTAMP to find out what the latest timestamp was that the user saw, and indicate "new alerts" only if there are alerts newer than that timestamp.

Interaction: GET ALERT QUEUE TIMESTAMP (query-response)

When the user logs in, the GUI sends the following to D-OMAR:

```
"Value-query" "-queryID" <arbitrarily assigned by the GUI:integer>
  "-userID" <ID of the user:string>
  "-variable" "alertQueueTimestamp"
```

D-OMAR responds:

```
"Value-reply" "-queryID" <queryID from the Value-query event:integer>
  "-variable" "alertQueueTimestamp" "-value" <time now:time>
```

In this way the GUI can determine when the user last looked at the Alert Queue, even though it was during the last login session.

The purpose of the Set-value and Value-query interactions to save and restore the alertQueueTimestamp variable is to display the correct state of the "New alerts" indicator on the Smart LT when a user has logged in after an absence. If there are alerts in the alert queue that are newer than the saved timestamp, then the "New alerts since last look at queue" indicator should be ON.

Interaction: SAVE CONFLICT DATA (one way GUI to D-OMAR)

The GUI uses the event type described here to inform D-OMAR of actions the user has taken regarding a MOG conflict alert. When the user changes the "advised", "acknowledged", or "can redo" status of a mission in the conflict, the GUI sends to D-OMAR:

"Alert-save-data" *header-fields conflict-data*

where the *header-fields* consist of the following fields in any order:

```
  "-eventID" <unique ID from Alert-notification event:integer>
  "-userID" <user ID:string>
  "-timestamp" <time at which the user made this request:time>
```

and where *conflict-data* is:

"-mission" <mission number:string> *mission-data*

where *mission-data* consists of one of the following fields:

"-advised" ["Y"|"N"]
"-acknowledged" ["Y"|"N"]
"-canRedo" ["Y"|"N"]

The *mission-data* field indicates to D-OMAR that the corresponding field of the mission in question (whose mission ID was given in the preceding "-mission" field) is set to the indicated "Y" or "N" value. D-OMAR will set its stored value to match this value. D-OMAR will not alter the stored values of any options that were not specified explicitly in the "Alert-save-data" message.

The purpose of this message is so that the next time the conflict window is displayed (e.g. with the help of "Alert-reply" ... "-detail" "conflict") the settings shown will match those that were shown the last time the user clicked the OK button. The GUI should send data for all Y/N options that the user changed, it may send data for options that had Y/N values but were not changed by the user, and it must not send data for any options that still have the "?" value. If the user has taken multiple actions regarding the conflict, that is, has generated new values for two or more of the *mission-data* fields, the GUI must send multiple messages, one for each field that is to be updated.

Interaction: GET MISSION CONTACT (query-response)

In order to get the mission contact data when the user pushes the contact button in the Conflict Summary window, the GUI sends:

"Mission-contact-query" "-missionID" <mission ID:string>

D-OMAR responds:

"Mission-contact-reply" *contact-data*

where *contact-data* is the following in any order:

"-office" <office name:string>
"-phone" <phone number:string>
"-fax" <FAX number:string>
"-email" <email address:string>

Since there is space for only one telephone number in the Contact Data window, D-OMAR should send just one telephone number, etc.

Interaction: GET NOTAM DETAILS (query-response)

When the user clicks the info button next to "issue" in the Alert Notification window for a NOTAM change alert, the GUI sends D-OMAR the following:

```
"Alert-query" "-eventID" <unique ID from Alert-notification event:integer>
 "-queryID" <arbitrary value generated by the GUI:integer>
 "-detail" "NOTAM"
```

D-OMAR responds:

"Alert-reply" "-queryID" <queryID from Alert-query event:integer>
 detail-fields mission-list

where *detail-fields* consists of the following in any order:

```
 "-NOTAMType" ["New"|"Replacement"|"Cancellation"]
 "-NOTAMIssuedDate" <when the NOTAM was entered into the NOTAM system:time>
 "-qcode" <Q code from the NOTAM:string>
 "-component" <component:string>
 "-status" <status:string>
 "-text" <NOTAM text:string>
```

and where *mission-list* is zero or more repetitions of

```
 "-mission" <mission ID:string>
```

Interaction: GET REPORT (query-response)

When the user clicks on the NOTAMs, DIP, and PPR buttons in a Port Viewer, when no mission is selected, the following query-reply pair retrieves the relevant data for this port:

The GUI sends the query:

```
"Report-request" "-queryID" <unique GUI-selected ID:integer>
 "-report" ["DIP"|"PPR"|"NOTAM"]
 "-location" <ICAO code of airfield:string>
 "-date" <relevant date:time>
```

The "-queryID" is whatever the GUI needs to route the reply event back to the correct object when a reply is received.

D-OMAR sends an event to the GUI:

```
"Report-reply" "-queryID" <the queryID from the Report-request:integer>
 "-report" ["DIP"|"PPR"|"NOTAM"]
 "-location" <ICAO code:string>
 "-text" <text of the report:string>
```

This assumes that for now, each of these reports will be gotten from an agent that simply returns a (possibly long) text describing the DIP requirements, PPR requirements, or NOTAMs applicable to this airfield.

Interaction: GET USER LIST (query-response)

This interaction enables the GUI to retrieve a list of all user Ids known to D-OMAR.

The GUI sends an event to D-OMAR:

```
"Send-users" "-queryID" <unique GUI-selected ID:integer>
```

The "-queryID" is whatever the GUI needs to route the reply event back to the correct object when a reply is received.

D-OMAR responds:

```
"UsersList" "-queryID" <the queryID from the Get-users request:integer>
 user-list
```

where user-list is zero or repetitions of

```
 "-userID" <user ID of a user:string> user-profile
```

one repetition for each user known to D-OMAR, giving that user's ID, where user-profile is the following in any order:

```
 "-name" <user's name:string>
 "-office" <user's office:string>
 "-branch" <user's branch:string>
 "-phone" <user's phone number:string>
 "-fax" <user's fax number:string>
 "-email" <user's email address:string>
```

Interaction: GET USER PROFILE (query-response)

This interaction lets the GUI retrieve the profile of a single user if it knows the user ID. To get information on multiple users, call the query several times.

The GUI sends to D-OMAR:

```
"User-profile-request" "-queryID" <unique GUI-selected ID:integer>
  "-userID" <user ID:string>
```

D-OMAR responds:

```
"User-profile-reply" "-queryID" <the queryID from User-profile-request:integer>
  "-userID" <user ID:string>
  user-profile
```

where the user-profile consists of the following fields in any order:

```
  "-name" <user's name:string>
  "-office" <user's office:string>
  "-branch" <user's branch:string>
  "-phone" <user's phone number:string>
  "-fax" <user's fax number:string>
  "-email" <user's email address:string>
```

Interaction: SET USER PROFILE (one way GUI to D-OMAR)

The GUI can use this to enter a new user in the database, and to update information for an existing user.

The GUI sends to D-OMAR:

"Set-user-profile"
  "-userID" <user ID:string>
  user-profile

where u*ser-profile* has the same format as in GET USER PROFILE.

If the user ID already exists in the database, D-OMAR should treat this as an update and replace the old profile of that user with the new data.

Interaction: CLEAR ALL (one way D-OMAR to GUI)

D-OMAR sends:

"Clear-all"

This tells the GUI to reset/reinitialize all data. This is currently used only for purposes of resetting a (simulated) Smart Lt session to the beginning.

Appendix C

Alert Condition Taxonomy

Randall Whitaker/Logicon Technical Services, Inc.

This document is an *initial sketch* for a taxonomic breakdown of alert conditions and alert notification parameters in the AMC/TACC HISA product(s).

The intention behind the design sketches done to date has been that there would be an ordered set of alert conditions and alert notification types, which could progressively be instantiated as the work continued.

This document is intended to address the following issues:

- provide a draft sketch / specification for alert conditions / parameters
- provide some clarification of the alert condition / notification presumptions embedded in the HCI design to date.
- provide some basis for discussing emergent issues (e.g., 'alert severity' assessment)
- provide some basis for sorting out issues involving HISA-defined data elements (e.g., 'Alert Type'), which are already evidencing problematical levels of ambiguity.

The thrust of this document is to try and shed some light on how we might define those Alert Types upon which the severity metric(s) would be applied, not the severity metric(s) per se. Phrased another way, this document doesn't affect the advisability or feasibility of Alert Severity – it only attempts to map out the building blocks for implementing it.

I. Nomenclature

I.A. Nomenclature for Alertable Conditions

This section presents a tentative set of labels to be used for the conditions, which motivate alerts to the planner(s). The conditions relate to the plans and missions, which are the foci of the planners' work activities. They are therefore general/high-level categories for "states of the plan/mission which the planner needs to know about."

Availability

Availability denotes one of the only 'positive' alertable conditions – something (e.g., Aircraft) is now known to be available for usage. This might prove useful eventually if we provide planners with the ability to post requests for assets, and then let agents notify them of new options.

Cancellation

Cancellation denotes that a given plan or mission has been cancelled (prior to execution). This is distinct from *termination*, which denotes an end to a mission under way. NOTE: A case might be made for having 'cancellation' refer to missions alone, and having a distinct label (e.g., 'withdrawn') refer to plans.

Clash

Clash refers exclusively to disagreements/discords involving one or more missions under way. A clash is presumably of more immediate concern than a *conflict* because it relates to actual transport services 'in the air.' *Clash* therefore refers to discord among plans and/or missions, where at least one of the discordant items is a mission under way.

Closure

Closure would denote the specific condition of a port being closed and hence unavailable for usage. The necessity of this condition is questionable, because a closed port could just as well be treated as constituting a 'deficiency'.

Conflict

Conflict refers to any discord or disagreement among plans *not involving missions under way.* In effect, *conflict* pertains to future/predicted conditions inferred from abstract plans.

It is important to note that 'conflict' (as the term is used herein) is something that occurs *among* plans, and not within a plan or its subject matter. The condition of MOG is not a conflict, whereas the problem arising when multiple plans would induce excess MOG at Port X constitutes a conflict among those plans. This is consistent with our illustrations of the HCI concept.

Constraint

Constraint would denote the specific condition of a plan being such that its execution violates one or more procedural limitations or proscriptions (e.g., an itinerary violating regulations on crew rest periods). The necessity of this condition type is subject to debate, because such a condition could be subsumed under a liberal interpretation of (e.g.) 'deficiency' (vis a vis a rested crew) or 'inconsistency'.

Deficiency

Deficiency denotes that some particular element or requirement is lacking in a plan or mission. This could be minor (e.g., DIP clearances not applied for) or major (e.g., a mission's aircraft has broken down and is unavailable for the remaining mission legs).

It's important to note that 'deficiency' means one or more particular elements of a plan or mission are lacking, whereas 'inconsistency' means all elements are accounted for, but 'don't add up.'

Delay

Delay denotes the specific condition of a mission under way being delayed relative to its scheduled itinerary.

Divert (Diversion)

Divert denotes the specific condition of a mission under way being diverted from a scheduled port of arrival. Such a diversion would presumably set off a number of alertable conditions.

Inconsistency

Inconsistency denotes a system-inferred discord among elements of a plan. For example, a published plan, which (initially or subsequently) puts the mission's aircraft at Port X, while some component of the cargo is supposed to be at Port Y, would be 'inconsistent'.

It is important to note that this alertable condition has to do with (e.g.) *intra-* (and not) inter-plan/inter-mission factors. It is also important to note that 'inconsistency' means, "all the pieces are there but don't add up," whereas 'deficiency' means, "one or more pieces are not in place."

Suspension

Suspension denotes that a plan or mission has been 'set aside' but remains pending. This is distinct from both *cancellation* and *termination* - both of which denote an actual end to the plan/mission involved.

Termination

Termination denotes that a given mission has been ended once under way (e.g., stopped after X of Y legs are accomplished). This is distinct from *cancellation,* which denotes ending or dropping a mission (or plan) *prior* to execution. Note that 'termination' (as defined here) refers exclusively to missions, and not plans.

I.B. Nomenclature for Alert Condition Qualifications / Characterizations

This section addresses not the alert conditions themselves, but the manner in which those conditions are addressed in the contexts of (a) identification/recognition (e.g., by the HISA agents) and (b) reporting (e.g., to the owner of the mission(s) affected).

Detected

Detected denotes that a given alert condition has been identified for either a plan or a mission. It only means that the system has inferred the existence/potential of the given alert condition. In our HCI presentations, we have illustrated the alert "Conflict Detected."

Confirmed

Confirmed denotes that a given alert condition has been 'upgraded' from a detectable prospect to a probable/definite problem. It means that the system has inferred the existence/potential of the given alert condition has reached a threshold 'level' or 'certainty.' This would be utilized in characterizing levels of alert 'certainty' or 'severity' (at least with regard to 'actionability'), as well as for re-notifying a planner that some previously 'suggested' or 'hypothesized' alertable condition is now something he/she must deal with.

Critical

Critical denotes that a pending alert condition is of such a nature that it demands the planner's immediate attention (e.g., when a MOG conflict 'deferred' repeatedly simply *must* be resolved).

Note that this is *not* an indicator of 'alert severity' (in terms of a quantification of problematicity in general), but rather an indicator of 'immediacy of actionability'. If 'blown off' too long, even a condition of relatively low 'alert severity' may become 'critical' (in terms of immediate action required.)

De-Escalated

De-Escalated denotes that a *pending* issue has (a) become less acute (e.g., an excess MOG of 4 has now become 3) and/or (b) decreased in a quantifiable degree of problematicity (e.g., the 'alert severity' value of a conflict has gone down.)

NOTE: This is a finer-grained thing than simply issuing an 'Updated' alert (see entry for 'Updated'), and specifically pertains to issues already pending resolution.

Escalated

Escalated denotes that a *pending* issue has (a) become more acute (e.g., an excess MOG of 5 has now become 6) and/or (b) increased in level of problematicity (e.g., the 'alert severity' value of a conflict has gone up.)

NOTE: This is a finer-grained thing than simply issuing an 'Updated' alert (see entry for 'Updated'), and specifically pertains to issues already pending resolution.

Resolved

Resolved denotes that a pending alert condition has ceased to be an alertable issue or problem.

Updated

Updated denotes that a given alert condition previously identified for either a plan or a mission persists, but that the details have changed. It means the system has inferred that the given alert condition is still operant, but that there's new info on that condition.

II. Alert Type(s)

The issue of 'Alert Type' frequently emerged in the design specifications/discussions. In particular, this has become a recurrent point of reference with respect to the Alert Notification module and, by implication, the process of alert notification.

The point is that 'Alert_Type' is the classificational 'anchor' for inferring, for example,

- Alert severity
- The setting/context relevant to a particular Issue (e.g., MOG) which is motivating the condition (and, hence, the alert)
- The drill-down options offered with Alert Notification

We therefore defined 'Alert Type' to provide a set or class of referents at a suitable level of granularity such the above-cited inferences can be done.

Having outlined the condition/characterization elements above, the following is a definition of 'Alert_Type.'

Alert_Type (as a data element) is comprised of (or mappable to) the set *{Condition, Characterization, Issue}* where:

- *Condition* is a data element corresponding to one of the condition labels defined above (Section I.A.)
- *Characterization* is a data element corresponding to one of the qualification/characterization labels defined above (Section I.B.)
- *Issue* is a data element corresponding to the mission state or problem motivating the alert.

In our illustrations to date, the Alert Notification window would be an example of Alert_Type = {Conflict, Detected, MOG}.

In our canonical MOG example:

- *Condition* (Conflict) mandates that inter-plan review be a drill-down option (conflict being a discordance among plans)
- *Condition* (Conflict) mandates that 'own plan' review be a drill-down option (though I suspect this is the 'universal drill-down option').
- *Issue* (MOG) mandates that an appropriate Port Viewer be a drill-down option (MOG being something manifest only at a given port on a given date.)

Now, to illustrate how this modularity makes a difference, consider a situation in which (e.g.) a XOOK staff publishes a Plan involving a Tail_Num (AC) already supposedly allocated to a previously published XOGE Plan. Assuming the XOOK proceeds after notification of the issue, the XOGE would get an Alert Notification window of Alert_Type {Conflict, Detected, AC_Availability). This situation would be the same as outlined above, except that in this case the *Issue* (AC_Availability) would mandate that a *Package* (not a Port…) Viewer be a drill-down option (AC being a component of the 'Package' as defined to date).

Furthermore, if the business rules permitted the XOOK above to simply 'take' the Tail_Num (no discussion to the matter), our poor XOGE might well receive an Alert Notification of Alert_Type {Deficiency, Detected, AC_Availability}, where the combination of *Condition* (Deficiency) and *Issue* (AC_Availability) would mandate different / additional drill-down options to allow going off and searching for another AC.

III. What About 'Alert Severity'?

The proposed taxonomic/data-architectural points in this document do not in any way rule out or constrain the notion of our having a scalar 'Alert_Severity' metric. The notion of a severity metric is something that must be applied with respect to different Alert_Types. As mentioned earlier, this document is intended to try to shed some light on how we might define those Alert_Types upon which the severity metric(s) would be applied, not the severity metric(s) per se. Phrased another way, this document doesn't affect the advisability or feasibility of Alert_Severity – it only attempts to map out the building blocks for implementing it.

SUMMARY

This document was not a final specification – it was only a 'serving suggestion' of terms and ideas. Many of the ideas were incorporated in the final designs. However, we wanted to show the reader what types of issues we were addressing when deciding upon the final design for demonstration. This document is intended only to (a) clarify some of the broad background presumptions underlying our narrow illustrations to date; and (b) offer a proposed organization for this area.

Appendix D

Summary of Functional Requirements

TASC Inc.

This project is developing a prototype GUI for the AMC TACC Global Channel Operations Directorate (XOG). The GUI prototype will provide channel mission schedulers the ability to task and manage software agents that will assist them in their duties. The scheduler must have visibility on planning factors that change and new information that will or may affect his/her scheduled channel mission. Visibility of both physical object attributes (ports/airfields, aircraft, weather) and information (mission details and relevant changes to planning factors) and their relationship in a dynamic environment is required. The scheduler must be quickly alerted when conflicts to missions have developed and be provided planning information to coordinate and resolve the conflict or issue.

The XOG channel mission consists of a scheduled aircraft to fly between two or more ports (the Port) to transport specified cargo and/or passengers (the Package) at a predetermined date and time. Missions are planned in the CAMPS and executed in the GDSS. The GUI will access the GDSS system for required information on scheduled missions, the DoD NOTAM system for real time information for notices to airmen, and locally developed reference tables for XOG quick-lookup port planning information. Additional data sources will be utilized when appropriate and as they become available.

The primary functional problems or "issues" this prototype will address are: identifying MOG parking conflicts, NOTAM changes with emphasis on airfield closure, and changes to selected planning factors at ports the AMC channel flights are scheduled to transit through. The underlying principle is to provide the channel planner the most timely and advance notification about the problems as is possible. Additionally, the prototype will aid in coordinating the resolution to the problem by extending timely conflict awareness to other TACC staff.

The prototype will allow the channel scheduler to graphically view key information pertaining to his mission: the port or series of ports to readily see the parking conflict, key port details, the mission details, and the software agent notifications or alerts pertaining to the problems. The prototype will provide additional aids to the planner including a PPR template useful to create the PPR request document, and a "submission notifier" when a request is within the time window for submission. A detailed description of the functional requirements is provided in the following table.

Detailed Functional Requirements

#	Functional Requirement				
1	Missions				
1.1	Summary				
1.1.1	The system shall provide a summary information about a mission or mission leg: a. Mission ID b. Operator c. Load d. Owner e. Launch date f. JCS priority g. Publication date (to GDSS) h. Aircraft type i. Start port j. End port k. Can recut (yes or no) l. Close watch (yes or no, but not editable here!)				
1.2	Port				
1.2.1	Single port				
1.2.1.1	The system shall display the following port information for a given period of time: a. Begin date b. End date c. Port name d. Port ICAO code e. Operating hours (both Z and L)[1] f. Sunrise time (both Z and L) g. Sunset time (both Z and L) h. NOTAM for the port i. DIP clearance requirements for the port j. PPR request requirements for the port k. List of all missions at the port during the time period sorted by JCS priority				

[1] Z and L indicates that time must be displayed Zulu and local time zones.

#	Functional Requirement				
1.2.1.2	The system shall display the following information for each mission at a port: a. Arrival time b. Departure time c. Mission identifier d. Aircraft type e. Aircraft category (Narrow / Wide body) f. Time on ground g. Previous port h. Next port i. Type of mission j. Mission scheduler k. DIP clearance information for each mission l. PPR information for each mission m. Scheduler's telephone number n. Scheduler's fax number o. Scheduler's e-mail address p. Office symbol of Scheduler q. JCS DoD priority r. Mission Scheduler advised s. Mission Scheduler acknowledged t. Can recut mission *[Y/N]* u. Designated Close watch *[Y/N]* v. Duration mission scheduled in GDSS w. Operator (Organic or Commercial) x. Port required y. Number of mission re-plans				
1.2.1.3	The system shall display following issues at a port: a. MOG conflict b. Airfield closure conflict[2]				
1.2.2	Multi-Port				
1.2.2.1	The system shall display all legs of a mission.				
1.2.2.2	The system shall display the interrelationship of time between legs in the sequence the legs are flown.				
1.2.2.3	The system shall allow the user to analyze mission options by modifying takeoff and departure times.				
1.2.2.4	The system shall utilize planning "rules" to explore options to include: a. Minimum ground time for type aircraft[3] b. Minimum on-ground time for crew rest[4]				
1.2.2.5	The system shall notify the user when adjustments violate planning rules.				
1.2.2.6	The system shall allow the user to "over ride" a rule violation.				
1.3	En Route[5]				
1.4	Package[6]				

[2] Airfield closure is the most significant form of limitation to the airfield status
[3] Refer to Standard Ground Times table in Attachment A (use Normal Peacetime planning times)
[4] Refer to Standard Ground Times table in Attachment A.
[5] No requirements have been defined for this functionality yet. We concentrated on the functionality that we knew was important to XOG. Requirements in this section will be based on the design work of our team. Future knowledge acquisition will be necessary to gather these requirements.
[6] Requirements in this section are based on the design work of the team. No user has specifically stated any of these requirements. Data fields (the details) are meant to represent what could be there. Future knowledge acquisition will be necessary to gather these requirements.

-

#	Functional Requirement				
1.4.1	The system shall provide summary information about the package of a mission or mission leg: a. Mission ID b. Operator c. Aircraft type d. Tail number e. Load description (pax, cargo, mixed) f. Crew ID				
1.4.2	The system shall provide details about the aircraft's status: a. Aircraft type b. Tail number c. Aircraft status d. Last maintenance				
1.4.3	Load				
1.4.3.1	The system shall provide details about the load: a. Load type b. Load number				
1.4.3.2	The system shall provided details about the status of the load: a. Load type b. Load number c. Load status				
1.4.4	Crew				
1.4.4.1	The system shall provide details about the crew roster: a. Crew ID *b. Crew details*				
1.4.4.2	The system shall provide details about the status of the crew: a. Crew ID b. Number of hours flown in current mission c. Number of hours rest in current mission d. Number of hours since last rest				
1.4.5	The system shall provide details about the fuel and its status: a. Fuel requirement b. Fuel loaded c. Air refueling required				
1.4.6	The system shall provide a list of documents required for this mission or mission leg, including: a. Air refueling plan b. Communication Specifications c. Contract d. DIP clearance designator e. PPR designator f. Flight plan g. Manifest				
2	Alerts				
2.1	The system shall allow each user to determine the types of alerts presented: a. Parking MOG conflict detected b. Airfield closure conflict detected c. PPR request window open d. DIP clearance request window open e. Weather below minimum detected				

#	Functional Requirement				
2.2	The system shall allow each user to determine how frequently he is alerted about an alert type, limited to the following choices: ▪ Daily ▪ Every two days ▪ Once per week ▪ Once per month				
2.3	The system shall notify the user when a new alert is received.				
2.3.1	This immediate alert notification shall be unobtrusive.				
2.3.2.1	This immediate alert notification shall be viewable by the user while other non-HISA windows are open.				
2.3.2.2	This immediate alert notification shall not be obscured while other non-HISA windows are open.				
2.3.3	This immediate alert notification shall present a visual and an audio cue when a new alert is received.				
2.3.4	Alert notification should identify or describe the originating data source (? How does this help the users solve a problem?)				
2.4	The system shall notify users of issues pertaining to published missions. Published missions are those that have been "pushed to GDSS."				
2.4.1	The system shall allow users to request alerts by: a. Mission type b. Alert type c. Mission number d. Port				
2.4.2	The system shall present the following information to the user about each issue: a. Issue identification (common name) b. Issue setting (location or port name) c. Mission identifier				
2.4.3	Parking MOG Conflict Detected				
2.4.3.1	The system shall notify users when a mission is involved in a Parking MOG conflict.				
2.4.3.2.1	The system shall display the Parking MOG value exceeded in the conflict.				
2.4.3.2.2	The system shall display the number of aircraft involved in the conflict.				
2.4.3.3	The system shall display the name of the port at which the Parking MOG conflict occurs.				
2.4.3.4	The system shall display the date on which the Parking MOG conflict occurs.				

#	Functional Requirement				
2.4.3.5	The system shall display the following details about each mission involved in the Parking MOG conflict: a. Mission scheduler b. Scheduler's telephone number c. Scheduler's fax number d. Scheduler's e-mail address e. Office symbol of scheduler f. Mission identifier g. JCS priority h. Mission scheduler advised i. Mission scheduler acknowledged j. Alert disregarded k. Can recut mission l. Designated close watch m. Duration mission scheduled in GDSS n. DIP clearance status o. Operator (Organic or Commercial) p. Port required q. Number of mission re-plans r. Previous port s. Next port				
2.4.4	Airfield Restriction Detected				
2.4.4.1	The system shall notify users when a mission is scheduled to arrive or depart from a port while the port is restricted.				
2.4.4.2	The system shall display the following details about the mission involved in the airfield restriction conflict: a. Issue name (Airfield restriction detected) b. Port ICAO c. Port name d. Date and time conflict begins e. Q-code[7] f. Issue/component of conflict[8] g. Status/condition of conflict[9]				
2.4.4.3	The system shall notify users when a mission is scheduled to arrive or depart from a port while the port is closed.				
2.4.5	PPR Request				
2.4.5.1	The system shall notify users when a Prior Permission Required (PPR) request may be submitted for a mission.				

[7] Documented in Air Force Joint Manual 11-208

[8] Examples are: Aerodrome, Aerodrome Control Tower, Runway (specify runway), Fuel availability, All landing area lighting facilities, Air Display, All radio navigation facilities.

[9] Examples are: Not available, Prior permission required, Closed, Changed, Deactivated, Closed to all night operations, Closed to IFR operations, Closed to VFR operations, Unmonitored, Limited to, Prohibited to, Usable length…and width of…., Operations cancelled.

#	Functional Requirement				
2.4.5.2	The system shall ignore weekend days when determining PPR notification date and time. For example, if a mission starts on Monday or Tuesday and the PPR window is 48-hours prior to mission start, then they would like to know about the PPR on Thursday or Friday, rather than Saturday or Sunday.				
2.4.5.3	The system shall display the following details about the PPR request: a. Squadron[*] b. US aircraft c. Military aircraft d. Weapons on aircraft? e. Reason for stop f. Type of cargo g. Number of aircraft h. Aircraft type[*] i. Mission number[*] j. Tail number[*] k. Routing from[*] l. Routing to[*] m. Estimated time of arrival[*] n. Date of arrival[*] o. Estimated time of departure[*] p. Date of departure[*] q. Call sign r. Dip clearance designator s. Any DVs on board? t. If yes, what code? u. Total personnel on board v. Point of contact w. POC phone number x. Special information or questions				
2.4.5.4	The system shall allow users to enter the following details about a PPR request: a. Reason for stop b. Type of cargo c. Any DVs on board? d. If yes, what code? e. Total personnel on board f. Point of contact g. POC phone number h. Special information or questions				
2.4.5.5	The system shall allow users to print PPR request details.				
2.4.5.6	The system shall allow users to export PPR request details to a word processing document.				
2.4.6	DIP Clearance Request				
2.4.6.1	The system shall notify users when a diplomatic clearance request may be submitted for a mission.				

[*] These fields can be found in GDSS.

#	Functional Requirement				
2.4.6.2	The system shall display the following details about the dip clearance a. Mission ID b. Port name c. Port ICAO d. Date mission will be at the port e. Dip clearance requirements for port f. Port contact name g. Port contact phone number h. Port contact fax number i. Port contact e-mail address				
2.4.6.3	The system shall notify users when a mission has a conflict and a DIP clearance is associated with it.				
2.4.7	Weather Below Minimums				
2.4.7.1	The system shall notify users when weather conditions are forecast to be below the minimum thresholds for landing the type of aircraft used for a mission.				
2.4.7.2	The system shall display the following details about the weather required: a. Mission ID b. Forecast area c. Date and time of forecast d. Planned date and time mission will be in forecast area e. Published IFR approach minimum ceiling f. Published IFR approach minimum visibility				
2.4.7.3	The system shall display the following details about the forecast weather at estimated flight arrival: a. Mission ID b. Port name c. Port ICAO d. Date and time of forecast e. Date and time of arrival f. Ceiling g. Visibility h. Below VFR (y/n)				
2.4.7.4	The system shall display forecast severe weather at a port to include hurricane and typhoon.				
2.5	The system shall notify users of issues pertaining to ports.				
2.5.1	NOTAM Change				
2.5.1.1	The system shall notify users when there is a new NOTAM for a port or a NOTAM for a port has changed.				
2.5.1.2	The system shall display the following details about the NOTAM: a. Port name b. Port ICAO c. Date of NOTAM d. NOTAM text				
2.6	The system shall allow users to defer taking action on an alert.				
2.6.1	The system shall allow the user options to set the length of time the alert will be deferred in hours, days, and weeks.				
2.6.2	The system shall re-alert the user after the designated defer period has expired.				
2.7	Alert Queue				
2.7.1	The system shall display all alerts the user has not taken action on.				

#	Functional Requirement				
2.7.2	The system shall display the following alert information: a. Object b. Condition c. Event d. Setting e. Mission f. Date of departure g. Number of days to mission start (calculated) h. Issue date i. Issue time				
2.8	Alert History				
2.8.1	The system shall display a history of all alerts received.				
2.8.2	The system shall display the following alert history information: a. Object (example: MOG, NOTAM) b. Condition (example: Conflict, Deficiency) c. Event (example Detected, Projected, Updated) d. Setting e. Mission f. Date of departure g. Number of days to mission start (calculated) h. Issue date i. Issue time j. Advised date k. Advised time l. Acknowledged date m. Acknowledged time n. Alert identification number				
2.9	The system shall allow a user to ignore a specific alert for a specific mission.				
3	Queries				
3.1	Pre-defined Queries				
3.1.1	The system shall present a series of pre-defined queries to the user to include the following: a. MOG at a specified Port b. Runway closure at a specified port c. Airfield closure at a specified port d. Air shows at a specified port				
3.1.2	The system shall allow the user to stipulate the lifetime of the query and the number of repetitions (query multiple times, or quit after one alert).				
3.2	User-defined Queries				
3.2.1	The system shall allow the user to create a new query.				
3.2.2	The system shall allow the user to save a new query as a pre-defined query.				
3.2.3	The system shall allow the user to stipulate the lifetime of the query and the number of repetitions (query multiple times, or quit after one alert).				
4	Agents				
4.1	Agent management				
4.2	Agent viewer				
4.3	Agent editor?				
5	Security				
5.1	The system shall provide secure access to the application via user name and password.				

#	Functional Requirement				
5.2	The system shall maintain a detailed profile about each user that includes the user's: a. Full name b. User ID c. TACC Department office symbol d. Cell or Branch e. Base telephone number f. E-mail address g. Fax number				
5.2.1	The system shall allow initial user setup				
5.2.2	The system shall allow changes to user profile.				
6	Other				
6.1	Date and time				
6.1.1	The system shall present all dates in the following format: "dd mmm yy (jjj)", where "jjj" is the 3-digit Julian day. For example, 11 Nov 99 (315).				
6.1.2	All dates shall be Zulu dates.				
6.1.3	All times shall be Zulu times, except where otherwise noted.				
6.3	The system shall save user ID and timestamp for all user entry items.				
6.3.1	User cannot "acknowledge" an alert until user has been "advised."				
6.3.2	User cannot modify "recut" or "ignored" until user has "acknowledged."				
6.3.3	User cannot modify another user's "advised", "acknowledged, or "recut" status.				

Attachment A: Standard Ground Times for AMC Operations

Aircraft Type	Ground Times, Pax and Cargo Operations (hours + minutes) (Normal peacetime planning times on first line by aircraft type) (Wartime planning times on second line by aircraft type) Onload	Refuel only	Offload	Expedited (1)	Minimum Crew Rest Times
C-130	2+15	2+15	2+15		15+15
	1+30	1+30	1+30	0+45	15+15
C-141	3+15	2+15	3+15		16+00
	2+15	2+15	2+15	1+15	16+00
C-17	3+15	2+15	3+15		16+30
	2+15	2+15	2+15	1+45	16+00
C-5A/B	4+15	3+15	4+15		17+00
	4+15	3+15	4+15	2+00	17+00
KC-10	4+15	3+15	4+15		18+15
	4+15	3+15	4+15	3+15	17+00
KC-135	4+15	3+15	4+15		17+00
	3+30	2+30	3+30	2+30	17+00
B-747					
(2)	3+30/5+00	1+30	2+00/3+00		
B-757					
	2+00	1+30	2+00		
B-767					
	2+00	1+30	2+00		
DC-8					
(2)	2+30/3+30	1+30	2+00/1+15		
DC-10					
(2)	2+30/5+00	1+30	3+00		
L-1011					
(2)	2+30/5+00	1+30	2+00/3+00		
MD-11					
(2)	3+30/5+00	1+30	3+00		

Note (1): Expedited means no refueling or reconfiguration accomplished. The aircraft just lands, loads and/or offloads and departs.
Note (2): Times for passengers/times for cargo for commercial aircraft.
Note (3): Vast majority of channel missions will use Normal Peacetime planning times.