Automatic Construction of User-Interface Displays*

Yigal Arens **Lawrence Miller** **Stuart C. Shapiro** **Norman K. Sondheimer**
USC/Information Sciences Institute
4676 Admiralty Way
Marina Del Rey, CA 90292
(213) 822-1511

Abstract

Construction of user interfaces for most computer applications remains time consuming and difficult. This is particularly true when the user interface system must dynamically create displays integrating the use of several interface modes. This paper shows how Artificial Intelligence knowledge base and rule technology can be used to address this problem.

NIKL is used to model the entities of the application domain and the facilities of the user interface. Rules are written connecting the two models. These rules range from application specific to general rules of presentation. The situation to be displayed is asserted into a PENNI database. A **Presentation Designer** interprets this data using the domain model, chooses the appropriate rules to use in creating the display, and creates a description of the desired display in terms of the interface model.

A system, **Integrated Interfaces**, using this design for an integrated multi-modal map graphics, natural language, menu, and form interface has been created and applied to a database reporting application.

1 Introduction

In spite of the development of user interface tool kits, construction and enhancement of user interfaces for most computer applications remains time consuming and difficult. Estimates of user interface code as a percentage of application code run as high as 60%. Among the most difficult interfaces to build are those that dynamically create displays. Such systems must automatically choose between multiple media (hardware), multiple modes (software systems), and multiple methods (choices with software systems).

Simply having several modes available is not enough — their use must be *integrated*. By this we mean that different items of information must be distributed to appropriate modes, the amount of redundancy should be limited to the amount needed to establish co-reference, and the different presentation modes must all work from a common meaning representation to assure accurate presentation. Further, the interface system integrating a set of modes must be capable of *dynamically* producing displays. Fixed multi-modal displays are not sufficient for rapidly changing environments. Finally, the techniques employed must support *generalization* and *enhancement* since the initial interface is certain to require enhancement over time. Existing systems do not and cannot achieve these objectives.

Artificial Intelligence knowledge base and rule technology can be used as a basis for automatic display construction. Information to be displayed can be recognized and classified, and display creation can then be performed based on the categories to which information to be presented belongs. Decisions can be made based on given rules. This approach to developing and operating a user interface allows the interfaces to be more quickly created and more easily modified. We call such a system a **model-driven presentation design system.**

In the **Integrated Interfaces** project at ISI we have begun to address the problem of constructing integrated user-interface displays. We have produced a design that supports integration of display modes, dynamically produces multi-modal displays, and supports generalization and enhancement. It does all this through a system of models and rules. The interface model brings together the different modes in a single uniform way. Another model describes the application, providing a uniform meaning representation. The rules explicitly state how information described in application terms relates to presentation modes. These rules take advantage of the model of interface capabilities to integrate the modes. Given information to display, a **Presentation Designer** applies these rules to dynamically produce display descriptions. Device drivers interpet such descriptions to create the actual displays.

Employing this design, our Integrated Interfaces system is able to present retrieved information using a combination of output modes — natural language text, maps, tables, menus, and forms. It can also handle input through several modes — menus, forms, and pointing. As a demonstration, we have implemented an interface to an existing Naval database reporting application. Our presentation designer creates displays similar to those being prepared manually for the Navy on a daily basis.

Section 2 of this paper discusses knowledge bases and rules in more detail. Section 3 describes the knowledge representation systems we are depending on. Section 4 gives examples. Section 5 compares Integrated Interfaces to the two systems most like ours. Section 6 summarizes our conclusions, and Section 7 discusses some of our plans for the future. The paper concludes with a description of our current status.

*This research is supported by the Defense Advanced Research Projects Agency under Contract No. N0014-87-K-0130. Views and conclusions contained in this report are the authors' and should not be interpreted as representing the official opinion or policy of DARPA, the U.S. Government, or any person or agency connected with them.

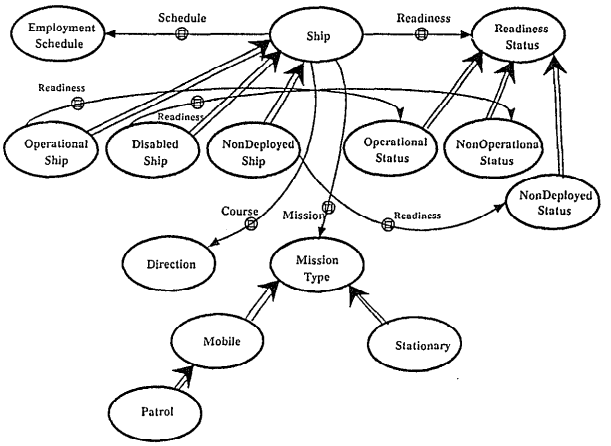

Figure 1: Fragment of Application Model

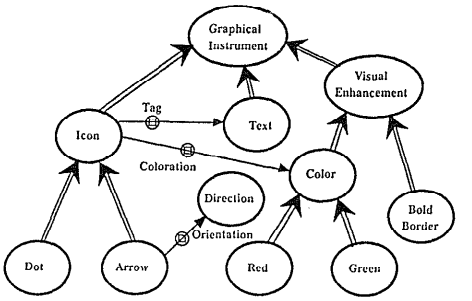

Figure 2: Fragment of Interface Model

2 Knowledge Bases & Rules For Presentation Design

Presentation design is achieved in our system by the application of a system of antecedent-consequent rules. The rules classify the information that needs to be presented and map types of information to appropriate types of presentations.

2.1 Models

Our models characterize or define the categories of *entities* our user interface can deal with.

The **application model** identifies the categories of objects and actions in the application's view of the world. We indicate subclass relations present among categories, as well as relationships between objects and actions. For the Naval database application, for example, we have a class of ships, which has subclasses categorized by operational status. (See Figure 1 for a small fragment of this model.)

The **interface model** describes the categories of objects and actions of the interface world. The objects here include windows, tables, maps, text strings, and icons. The actions include creation, deletion, movement, and structuring of displays. Describing all interface modes together in a single model is a necessary basis of integration. (See Figure 2 for a small fragment of this model.)

Only the application model needs to be created for each new application interface.

2.2 Rules

The presentation rules are simple in essence: they map objects from the application model into objects in the interface model. For example, a daily status report may be mapped into a map. A position report may be mapped onto a point on the map. A ship's planned future activities may be mapped onto a text string. The following is a paraphrase of part of a rule for the naval briefing application: "To display a *Ship* whose *Mission* is *Mobile*, use an *Arrow*, with its *Course* establishing the arrow's *Orientation*, and *Text* as a *Tag* presenting its *Schedule*." As can be seen, this rule takes its condition from the application model and the rest of its vocabulary from the application and interface models.

It is the rules in conjunction with the interface model, that allow integration. They can be used to distribute information among modes, minimize redundancy, and coordinate presentations. For example, the above rule creates an mixed graphic and natural language text display.

These rules are arranged according to the class subsumption hierarchy of the models. For example, the rules applicable to all ships are further up the hierarchy than those applying only to ships on exercises.

We allow both "low-level," application-specific rules, and "high-level," application-independent rules. The above rule is an example of the first type. The following is an example of the second: "To request a *Choice Among Alternatives* when the *Cardinality* is *Large*, use a *Fill-in-the-Blank From*; otherwise use a *Menu*."

2.3 Rule Application

Presentation design can now be described as the task of *realizing* the application domain categories within which a request for information presentation falls, *selecting* the appropriate rules that apply to those categories, and *redescribing* the application terms in the request into appropriate presentation terms.

Realization relates the facts about instances to the abstract categories of the model. For example, the concrete facts about *Sprite*, a ship with a malfunctioning radar, must lead to the realization that it is a *Disabled Ship*. Selection works by allowing for the appropriate mapping rules to be chosen, allowing for additivity. Selection also assures that all aspects of the demand for presentation are met by some rule. Redescription applies the rules, mapping each aspect of a common-sense view of a presentation into an equivalent presentation form.

The forms produced by rule application are not actually the commands to the output subsystems (i.e., the map graphics system, text generator, and the forms system). Instead, they are interpretable by *device drivers* that control these systems.

3 Knowledge Representation Tools

Our implementation of presentation design depends on two knowledge representation systems: NIKL and KL-TWO. NIKL holds our models. KL-TWO automatically carries out realization. KL-TWO also holds the demands for presentation and receives the forms read by the device drivers. This section provides a brief introduction to these tools.

NIKL [Kaczmarek *et al.*, 1986] is a network knowledge-base system descended from KL-ONE [Brachman and Schmolze, 1985]. This type of system supports description of the categories of entities that make up a domain. The central components of the notation are sets of concepts and roles, organized in IS-A hierarchies. These hierarchies identify when membership in one category entails membership in another. The roles are associated with concepts (as *role restrictions*), and identify the relationships that can hold between individuals that belong to the categories. The role restrictions can also hold number restrictions on the number of entities that can fill these roles.

We have been experimenting with a naval assets domain model for the naval database reporting application mentioned above. It has a concept *Disabled-Ship* that is meant to identify the ships that are unable to carry out their missions. *Disabled-Ship* IS-A type of *Ship* distinguished from *Ship* by having a role restriction *Readiness* that relates *Disabled-Ship* to *NonOperational-Status,* i.e., all ships with nonoperational status are disabled. All *Ships* can have exactly one filler of the *Readiness* role restriction. The concept of *NonOperational-Status* is partly defined through the IS-A relation to a concept *Readiness-Status.* This situation is shown graphically in Figure 1 in the typical network notation used for KL-ONE knowledge bases.

KL-TWO is a hybrid knowledge representation system that takes advantage of NIKL's formal semantics [Vilain, 1985]. KL-TWO links another reasoner, PENNI, to NIKL. For our purposes, PENNI can be viewed as managing a data base of propositions of the form $(P\ a)$ and $(Q\ a\ b)$ where the forms are variable free. The first item in each ordered pair is the name of a concept in an associated NIKL network and the first item in each ordered triple is the name of a role in that network. So the assertion of any form $(P\ a)$ states that the individual a is a kind of thing described by the concept P. The assertion $(Q\ a\ b)$ states that the individuals a and b are related by the abstract relation described by Q.

NIKL adds to PENNI the ability to do taxonomic reasoning. Assume the NIKL database contains the concepts just described in discussing NIKL. Assume that we assert just the following three facts: (*Ship Sprite*), (*Readiness Sprite C4*) and (*NonOperational-Status C4*); *C4* is a U.S. Navy readiness code. Using the knowledge base, PENNI is able to deduce that (*Disabled-Ship Sprite*) is true.

PENNI also provides a truth maintenance system that keeps track of the facts used to deduce others. When our rules are used to determine aspects of a presentation from facts about the world, the truth maintenance system records the dependencies between the application domain and the presentation. For example, (*Readiness Sprite C4*) triggers a rule which asserts (*Disabled-Ship Sprite*). If (*Readiness Sprite C4*) is retracted, PENNI's truth maintenance system will automatically retract the assertion that Sprite is disabled.

4 Examples

The power of a model-driven presentation design is in its flexibility. The designer of a system does not specify rigidly in advance in what form information will be requested from the user, and how data and results will be displayed. Instead, our models contain descriptions of the types of information the application programs deal with, and of the types of graphical tools and instruments available. The rules for presentation enable the system to generate on-demand displays appropriate for given needs. Here are some concrete examples.

4.1 Construction of a Visual Representation of an Object

Consider the knowledge about ships and about graphical instruments encoded in the NIKL models in Figure 1 and Figure 2. Let us assume that the user wishes to show ships engaged in a *Mobile* mission with a special *Icon,* and that the icon should be oriented in a direction identical to the ship's course. In addition, assume that *Disabled-Ships* are to be shown with *Red* icons and that the *Schedule* of a ship is to be shown in the natural language *Tag* of the *Icon* representing it. A version of the rules that we would use to achieve this is shown in Figure 3. The antecedent considers the categories of one or more individuals and their relationships, all in NIKL terms. The consequents provide assertions about the graphic representation of objects for the PENNI database. These rules are asserted into PENNI so that the truth maintenance system may keep track of the dependencies between antecedent facts and their resultant consequents, as explained in the previous section.

The functions *Image* and *Textual-Description* map the constants of the common sense world into constants of the visual and textual world, respectively. For example, Rule 5 states that if some individual, x, is a *Ship* and another individual, y, is its *Schedule,* then the *Tag* of the image of x is the textual-description of y. The textual-description of y will be created by the invocation of our text generator.

1. IF (Operational-Ship x) or (NonDeployed-Ship x)
   THEN (Coloration Image(x) Green)
2. IF (Disabled-Ship x)
   THEN (Coloration Image(x) Red)
3. IF (Ship x) and (Course x y)
   THEN (Orientation Image(x) y)
4. IF (Ship x) and (Mission x y) and (Mobile y)
   THEN (Icon-Type Image(x) Arrow)
5. IF (Ship x) and (Schedule x y)
   THEN (Tag Image(x) Textual-Description(y))

Figure 3. Sample Presentation Rules.

To complete the example, suppose that the following set of facts was asserted into the PENNI database: (*Ship Sprite*), (*Readiness Sprite C4*), (*NonOperational-Status C4*), (*Mission Sprite X37*), (*Patrol X37*), (*Schedule Sprite U46*), (*Course X37 220*), and (*Employment-*

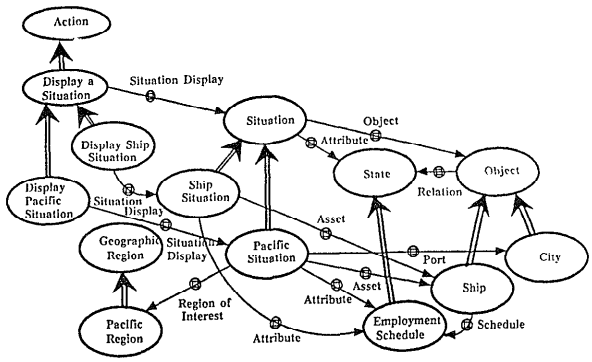

Figure 4: Model Fragment

Schedule U46). Suppose further that the NIKL model defined *Patrol* to be a subclass of *Mobile* missions. Realization would recognize the 'Sprite' as a *Disabled Ship* and one engaged in a *Mobile* mission on a course of 220 degrees. Selection would identify that Rules 2, 3, 4 and 5 apply. Redescription would result in the addition to the PENNI database of the description of the image of the 'Sprite' as a red arrow with an orientation of 220, and with a textual representation of its schedule as its label.

Due to the use of KL-TWO's truth maintenance system, if any of the facts pertaining to *Sprite* is retracted, an automatic change in the description of its graphic image will occur.

4.2 Classifying Collections of Data

For many requests for information encountered in our application domain the design of a presentation requires global considerations that rules of the kind listed above cannot provide for. It would therefore be hopeless, at this point, to try to write rules that would attempt to derive an elaborate presentation entirely from low-level information about the objects to be described. Our approach provides us with a partial solution to this problem.

The availability of models of the application and of displays to our Presentation Designer gives it the advantage of being able to recognize collections of data as representing information of a certain known type. The Presentation Designer can then make use of presentation techniques specialized for this type of data to provide the user with more appropriate displays.

For example, Figure 4 provides portions of our model that include the class *Pacific Situation*, a collection of data about ships and ports in the *Pacific Region*, which includes certain specific information from the ships' employment schedules.

When provided with data about ships in the Pacific region and their employments, the Presentation Designer would classify the data in its model of the application, recognizing that it has received a collection of data belonging to the class *Pacific Situation*. Then the Presentation Designer can use specific presentation rules appropriate for displaying the information. In the application domain we have considered there is a preferred way for presenting this information, to which we try to conform. This preferred presentation has developed in the Navy in the course of years of handcrafted situation briefing presentations.

The specific presentation rules appropriate only for *Display Pacific Situation* will combine the entities created by more general rules, of the kind described in the previous section, to produce the final presentation.

4.3 Generation of an Input Display

A presentation design system must also deal with the preparation of displays for the purpose of soliciting necessary information from the user. Here, again, the models of all aspects of the task and the application are valuable.

At some point the user may indicate a desire to view data concerning one or more ships in some region. In terms of our model (see Figure 4), that would mean indicating a preference for *Display a Situation*. As it turns out, the Presentation Designer does not have any rules that can be used to redescribe this general request into a presentation, but there exist ways of satisfying more specific requests. For example, there exist ways to satisfy a request for displaying a single ship's situation or the situation of all ships in a region.

In this case, the system collects all options the user can choose among to construct an executable request. A rule of the Presentation Designer is used to compose a display form that will present these options to the user. The result of this design is a set of assertions in PENNI that the device driver for a separate form management package (QFORMS) [Kaczmarek, 1984] will use to prepare the input form.

The form below, presented to the user, allows the user to make one of several specific choices:

Pacific Regions:
- Western Pacific ☐
- South China Sea ☐
- Indian Ocean ☐
- Eastern Pacific ☐
- Pacific Command Region ☐

Ship: _____

It is instructive to examine precisely how this form is created. The concept *Display a Situation* has two subclasses of actions, namely *Display Ship Situation* and *Display Pacific Situation*. Our system considers the possibility of generating an intermediate two item submenu, something like:

- Situation in Pacific Region ☐
- Situation of Ship ☐

Such a small menu is unsatisfactory from a human factors standpoint. We therefore formulated a general condition stated in the rule used here, saying that if the number of choices is less than N, and if the choices can be further subdivided, then the proposed menu should not be displayed. Instead, a more detailed form should be generated, one based on the subchoices. Our prototype uses the value 3 for N, so in this case the rule causes the Presentation Designer to immediately generate the more specific form. A user is free to change the value of N, thus modifying the design of forms the system generates in situations like the one above.

Note that the geographic regions available were specified by name in the form created, while ships were not. Rather, the user is allowed to specify the desired ship by typing it on the form[1]. This distinction is a result of information concerning the cardinality of the relevant collections of objects — information encoded in our models. Since the number of possible choices for region is small, they are enumerated. However, the number of ships is larger, so the user is provided with a way to specify a choice explicitly instead.

Finally, the result of an end user completing this form is also controlled by the model. QForms allows actions to be associated with menu choices and form fillings. In creating a menu, the Presentation Designer adds an action to each field conditioned on the field being selected with the mouse. This action will result in an assertion in PENNI, indicating that the user is requesting the action described by the model category from which the menu choice originated. Fill-in-the-blank forms work similarly.

5 Related Work

Perhaps the best known previous work dealing with the issue of Presentation Designer is that of Mackinlay [Mackinlay, 1986].

Much like part of our system, Mackinlay's *APT* uses information about characteristics of data provided to it, to produce a graphical representation of that data. The differences between the two systems become clear when we consider the variety of data each deals with and the variety of presentations they produce. *APT* produces graphs of various kinds, and much of its effort goes into deciding which axes to choose, and how to indicate the values along each axis. Data dealt with is limited to what can be presented using such graphs. Consequently, Mackinlay has succeeded in producing a system which can generate graphical presentations automatically using only "low-level" information about the objects and their attributes.

Our system is expected to generate a much wider variety of displays. Certain display layouts are often chosen simply to conform to pre-existing preferences of Navy personnel. Consequently, unlike Mackinlay, we must provide for the possibility of following pre-set sterectypical instructions in certain cases. We thus must devote considerable effort to recognizing which cases require these special displays.

A further significant difference between the systems is the complexity of the data we are required to present. We needed a sophisticated knowledge representation language, NIKL — a facility which Mackinlay found unnecessary. Both systems make use of sophisticated reasoning facilities.

The CUBRICON system [Neal and Shapiro, 1988] shares many of the same goals with our system, but differs in initial focus. Like our system, CUBRICON uses a sophisticated knowledge representation/reasoning system to manage an integrated, multi-modal interface, including maps, icons, tables, and natural language text. Whereas the CUBRICON project is trying to construct a unified communicating agent, with multi-modal input/output melded within a natural language understanding/generation system, our system highlights the rules that map between the application and interface models, and views natural language generator as a rather impermeable display agent. CUBRICON is more focused on producing the grammar and rules for a multi-modal language, we are more focused on producing an easily used, multi-modal user-interface management system.

[1]The actual form (Figure 5.) uses the title *Report* as opposed to *Ship*, since it allows other types of reports as well.

6 Conclusions

We have realized the Integrated Interfaces design in a system that utilizes natural language, graphics, menus, and forms. Specifically, the Integrated Interfaces system can create maps containing icons with string tags and natural language descriptions attached to them. It can further combine such maps with forms and tables presenting additional, related information. In addition, the system is capable of dynamically creating menus for choosing among alternative actions, and more complicated forms for specifying desired information.

We have constructed application models describing concepts in an important real-world domain — the naval situation briefing. We have implemented rules that enable the creation of different types of integrated multi-modal output displays based on the Navy's current manual practices. We have represented large enough portions of both the general and application specific domains to demonstrate that a model-driven presentation design approach is potentially useful in real-world situations.

In achieving this result, we have done more than produce a system for constructing and controlling multi-modal application interfaces. We have shown that what would otherwise appear to be distinct communication mechanisms, viz., graphics, natural language, tables, etc., can be treated as part of an integrated whole, all relating to a common level of meaning representation. We have further shown that the decisions on the use of the appropriate mode can be represented straightforwardly by explicit rules relating information to be presented to the method of presentation. This work can serve as the basis of a comprehensive theory of multi-modal communication.

7 Future Work

Despite the successes illustrated in the previous examples outstanding problems remain. Our future plans include adding the following structures to our system.

- A User Model — A user model will enhance the Presentation Designer by allowing it to tailor presentations to individual user preferences. For example, it would enable the system to label only ports and regions unfamiliar to a user, thereby reducing screen clutter.
- A Dialogue Model — A dialogue model will allow the presentations to be more closely tailored to specific users' requests. Currently, the Presentation Designer is simply provided with data to display. It is not aware of the purpose of the display.
- A Screen Model — A screen display is more than a bitmap; it is viewed by a user as containing icons and text which have real world denotations. The interface's internal description of the display must be

rich enough to allow a user to alternate between references to screen entities and their denotations. A screen model will make such relationships explicit.

8 Current Status

A demonstration version of the Integrated Interfaces system is now available at ISI. The current version models the domain of Navy ships in the Pacific Ocean. A user may use the system to access information about ships' locations, tasks, readiness status, and more. The resulting information is displayed using combinations of maps, menus, tables, and natural language output (Figure 5).

The system is written in Common Lisp and runs in the X windows environment under UNIX on HP 9000 Model 350 workstations. Displays are presented on a Renaissance color graphics monitor. The map graphic modality is supported by ISI's Graphics Display Agent. Menus and forms are created using QFORMS [Kaczmarek, 1984]. Natural language output is produced by ISI's Penman system [Sondheimer and Nebel, 1986].

9 Acknowledgements

We wish to acknowledge the crucial help provided by others working on the Integrated Interface project. Paul Raveling has developed the graphical interface and continues to maintain the GDA. Chin Chee has ported QFORMS and Penman to the HP workstation and is responsible for coordinating the various parts of the system. Jim Geller has contributed to the implementation of the spatial reasoner.

References

[Brachman and Schmolze, 1985] Ronald J. Brachman and James G. Schmolze. An Overview of the KL-ONE Knowledge Representation System. *Cognitive Science* 9(2), 1985, pp. 171-216.

[Kaczmarek, 1984] Tom Kaczmarek. CUE Forms Description. ISI Internal Report. USC/ISI, Marina del Rey, CA, 1984.

[Kaczmarek *et al.*, 1986] Tom Kaczmarek, Ray Bates, and Gabriel Robins. Recent Developments in NIKL. *Proceedings, AAAI-86.* Philadelphia, PA., August, 1986.

[Mackinlay, 1986] Jock D. Mackinlay. Automatic Design of Graphical Presentations. Ph.D. Thesis, Department of Computer Science, Stanford University. Stanford, CA, December 1986.

[McAllester, 1982] D. A. McAllester. Reasoning Utility Package User's Manual. Massachusetts Institute of Technology, AI Memo 667. Cambridge, MA., April, 1982.

[Neal and Shapiro, 1988] J. G. Neal and S. C. Shapiro. Intelligent Multi-Media Interface Technology. *Proceedings, Architectures for Intelligent Interfaces: Elements and Prototypes*, J. W. Sullivan & S. W. Tyler, Eds., Lockheed AI Center, 1988, pp. 69-91.

[Sondheimer and Nebel, 1986] Norman K. Sondheimer and Bernhard Nebel. A Logical-Form and Knowledge-Base Design For Natural Language Generation. *Proceedings, AAAI-86*, Philadelphia, PA., August, 1986, pp. 612-618.

[Vilain, 1985] Mark Vilain. The Restricted Language Architecture of a Hybrid Representation System. *IJCAI-85: Proceedings of the Ninth International Joint Conference on Artificial Intelligence.* Los Angeles, CA., August, 1985, pp. 547-551.

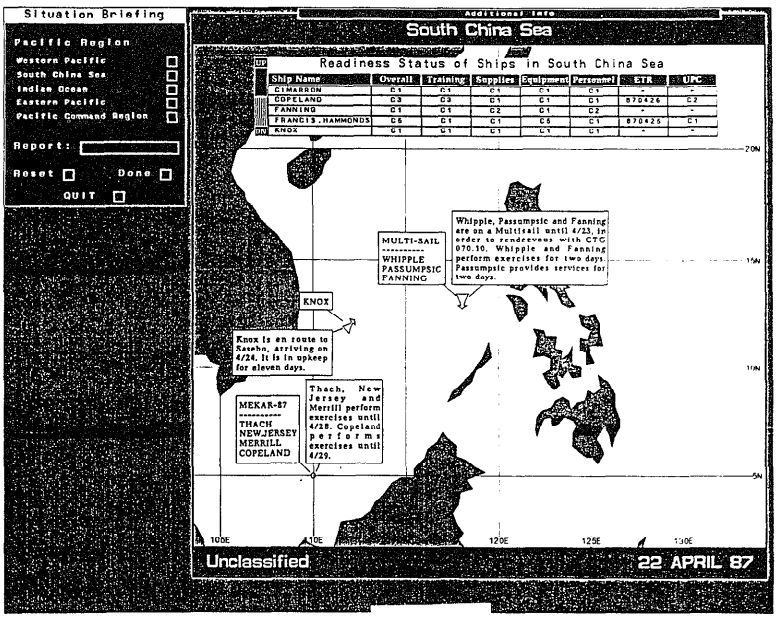

Figure 5.