ARI TECHNICAL REPORT
TR-78-A25

MIQSTURE: An Experimental Online Language for Army Tactical Intelligence Information Processing

by

OPERATING SYSTEMS, INC.
21031 Ventura Boulevard
Woodland Hills, California 91364

JULY 1978

Contract DAHC 19-77-C-0023

Monitored technically by
Stanley M. Halpin and Lawrence M. Potash
Battlefield Information Systems Technical Area, ARI

Prepared for

U.S. ARMY RESEARCH INSTITUTE
for the BEHAVIORAL and SOCIAL SCIENCES
5001 Eisenhower Avenue
Alexandria, Virginia 22333

U. S. ARMY RESEARCH INSTITUTE
FOR THE BEHAVIORAL AND SOCIAL SCIENCES

A Field Operating Agency under the Jurisdiction of the Deputy Chief of Staff for Personnel

JOSEPH ZEIDNER
Technical Director

WILLIAM L. HAUSER
Colonel, US Army
Commander

Research accomplished
under contract to the Department of the Army

Operating Systems, Inc.

NOTICES

DISTRIBUTION: Primary distribution of this report has been made by ARI. Please address correspondence concerning distribution of reports to: U. S. Army Research Institute for the Behavioral and Social Sciences, ATTN: PERI-P, 5001 Eisenhower Avenue, Alexandria, Virginia 22333.

FINAL DISPOSITION: This report may be destroyed when it is no longer needed. Please do not return it to the U. S. Army Research Institute for the Behavioral and Social Sciences.

NOTE: The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

(18) ARI

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(19) **REPORT DOCUMENTATION PAGE**

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: TR-78-A25	2. GOVT ACCESSION NO. (9) Final Rept. 15 Mar 77-14 apr 78.
4. TITLE (and Subtitle): MIQSTURE: An Experimental Online Language for Army Tactical Intelligence Information Processing. (6)	5. TYPE OF REPORT & PERIOD COVERED: Final 3/15/77 - 4/14/78
	6. PERFORMING ORG. REPORT NUMBER: OSI-76-034
7. AUTHOR(s): Operating Systems, Inc.	(14) 8. CONTRACT OR GRANT NUMBER(s): (15) DAHC19-77-C-0023
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Operating Systems, Inc. ✓ 21031 Ventura Boulevard Woodland Hills, CA 91364	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: (16) 2Q762717A765
11. CONTROLLING OFFICE NAME AND ADDRESS: US Army Research Institute for the Behavioral and Social Sciences (PERI-OS) 5001 Eisenhower Avenue, Alexandria, VA 22333	(11) 12. REPORT DATE: July 1978
	13. NUMBER OF PAGES: 203
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): (12) 252p.	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Project supervised by Stanley M. Halpin and Larry Potash, Battlefield Information Systems Technical Area of the US Army Research Institute for the Behavioral and Social Sciences.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Mixed-initiative system	machine initiated contribution	task representation
interactive language	message retrieval	function keys
tactical intelligence	tabular data base	alerting query
man computer dialogue	layered language	input-driven query
man computer interaction	online system	cross referencing
query language	information proc.	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

An analysis and specifications for the application of a mixed-initiative query system to intelligence data processing are presented. The project had four principal steps: 1) Development of a statement of requirements for interactive data processing in tactical intelligence analysis; 2) Extension and application to the requirements identified in the first step of a concept of a mixed-initiative system, focused on task requirements and user needs; 3) Development of detailed design specifications for six selected subsets of MIQSTURE (the Mixed Initiative Query System with Task and User Related Elements); and

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

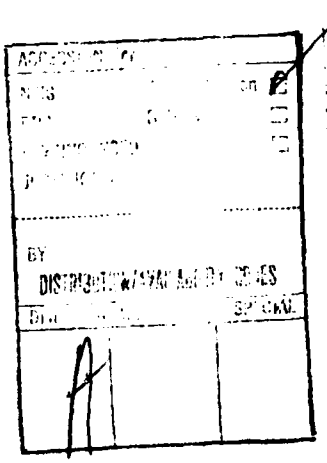
Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

391994

Block 20 (Continued)

4) Preliminary evaluation of MIQSTURE. The six subsets developed in some detail were a) normal querying of message records; b) automatic alerting and input-driven querying; c) normal tabular data base querying and calculation; d) aids for representing the tasks which form the context of MIQSTURE; e) procedures for cross-referencing graphic symbols and data-base elements; and f) defining and maintaining data file structures. An analytic evaluation of MIQSTURE indicated high potential for application to tactical intelligence processing and other data processing situations involving a relatively restricted range of well-defined tasks.



The Battlefield Information Systems Technical Area of the Army Research Institute is concerned with the demands of the future battlefield for increased man-machine complexity to acquire, transmit, process, disseminate, and utilize information. The research is focused on the interface problems and interactions within command and control centers and is concerned with such areas as topographic products and procedures, tactical symbology, information management, user oriented systems, staff operations and procedures, and sensor systems integration and utilization.

One area of special research interest is the design of effective and efficient on-line interaction between the operator/user and the computer. The frequent inability of users to interact effectively with developing tactical data systems has resulted in high error rates, low input rates and inappropriately structured outputs which have severely compromised the hoped-for benefits of battlefield automation.

One promising approach to reducing these problems is the development of mixed-initiative man-computer interaction. Instead of the man-computer dialogue being entirely initiated by the user, the computer has an active role. The computer may prompt or step the user through complex tasks, provide task specific error feedback, as well as alert the user to "other" information available in the data base. The research reported here provides specifications for a mixed-initiative query system for tactical intelligence analysis. However, the design concepts and lessons learned are relevant to other task areas. This effort is part of the exploration of improving ways for the user and computer to communicate and provides a necessary technological base for effective design of the man-computer interface.

Research in the area of concepts for man-computer synergism is conducted as an in-house effort augmented by contracts with organizations selected for their specialized capabilities and facilities. These efforts are responsive to general requirements of Army Project 2Q162722A765 and HRN 78-149, and to special requirements of the U.S. Army Intelligence Center and School, Ft. Huachuca, AZ and of the U.S. Army Combined Arms Combat Development Activity, Ft. Leavenworth, KS. The present research was conducted by personnel from Operating Systems Inc. under contract DAHC 19-77-C-0023, Army Project 2Q7627i7A765, with technical monitoring by Dr. Stanley M. Halpin and Dr. Larry Potash.

The research was made possible by the excellent cooperation of personnel in the Combat Developments Directorate of the U.S. Army Intelligence Center and School, Ft. Huachuca, AZ.

TABLE OF CONTENTS

TABLE OF CONTENTS (cont'd)

EXECUTIVE SUMMARY

The concept of mixed initiative emphasizes computer initiated contributions to the man computer dialogue. Instead of the dialogue being entirely initiated by the user, both machine system and the user analyst play active roles in initiating contributions to their interactive dialogue. The system actively contributes to the dialogue by volunteering information appropriate to the inferred intent of the query, prompting or "stepping" the user through complex tasks, diagnosing errors using contextual information to make feedback more specific and automatically monitoring input streams for conditions satisfying "templates" depicting situations to which the user must be alerted. Two features which keep the user firmly in control of the dialogue and make the system more flexible are: (a) human-initiated override for machine initiated elements, and (b) generative capability to build and modify machine-initiated elements more or less "on the spot."

The effects of machine-initiated contributions are to free the user-analyst of those task elements that are within the machine's capabilities, so that he can concentrate his energies on accomplishment of task elements beyond the machine's capabilities. The machine contributions should augment the data processing rate capacity of the user.

The advantage of a mixed initiative system are particularly important for the area of tactical intelligence analysis, since the pace of modern warfare radically increases the quantities of information to be processed, while at the same time curtailing the time available for processing activities. Although this project is oriented toward the tactical intelligence analysis area, many of the design concepts and lessons learned are relevant to other on-line systems.

The project progressed through four sequential steps. In the first, requirements for an on-line interactive language for intelligence information processing were analyzed, using as sources Army documentation and interviews with personnel in the Army Intelligence School, Ft. Huachuca, AZ. Results are summarized in Section 1.0.

In the second step, a concept of mixed-initiative operations for tactical intelligence processing (MIQSTURE) was developed and applied to the pattern of requirements identified in the first step. The results are contained in Section 2.0 of the present report.

The third step consisted of producing detailed design specifications for six selected subsets of the MIQSTURE language, to serve as concrete examples of its operating characteristics and potential:

- Normal querying of message records
- Automatic alerting and input-driven querying
- Tabular data base querying/calculation
- Interactive task representation/control aids
- Terrain symbology supporting data reference
- Define/maintain data structures

The detailed specifications of these subsets are contained in Appendix B and scenarios illustrating their use in Section 3.0.

Techniques to enhance the MIQSTURE language's adaptivity to change include (1) the use of specially designed formats for defining new capabilities and for modifying existing ones, (2) the additional use of a keyboardable substrate language to provide flexibility for new and emergent circumstances.

The language was designed to be sensitive to the elements and stages of interactive tasks. Interaction status and history are shown in separate display windows, and transaction and query numbering counters are incremented separately. A multistream dialog arrangement allows interleaving of tasks at a terminal, thus producing flexibility and better throughput capacity. A capability is provided for loading "knowledge" of detailed task structure into the system in order to allow the system to cooperate in the mixed-initiative manner described earlier.

The MIQSTURE language's learnability and retainability aspects are enhanced through arrangements for presenting materials to be learned by the user. Associative pairing of long and abbreviated forms of system message and command names is provided to increase the learnability of the short forms. The overall structures of the language distinguishes task-general and task-specific elements and a layered approach is taken to the query formulation. Many of the commands and logical operations used in handling simple message retrieval also underlie the language used for automated querying functions and for handling tabular stored data and hierarchical structures.

In the final step, the MIQSTURE language was evaluated. Discussion of the evaluation considerations, techniques, and results are contained in Section 4.0. Five of the six MIQSTURE subsets were evaluated in terms of the extent to which specific design goals had been reached. The sixth subset, for querying and manipulating tabular data in data bases, was compared formally with the data base language SEQUEL. Using a rating technique, it was found that MIQSTURE attained its design goals in each of the first five subsets, and that it exceeded SEQUEL capabilities on six out of seven evaluative attributes assessed for the sixth subset. It is concluded that a mixed-initiative language may have considerable potential applicability for Army tactical information processing.

MIQSTURE: An Experimental Online Language for Army Tactical Intelligence Information Processing

1.0 OVERVIEW

1.1 Background and Purpose of the Project

The design and evaluation of MIQSTURE (Mixed Initiative Query Structure with Task and User Related Elements) is a project undertaken by Operating Systems, Inc. (OSI), with evaluative consulting by Perceptronics, Inc., under Contract DAHC 19-77-C-0023 (3/15/77), U.S. Army Research Institute for the Behavioral and Social Sciences. The contracted tasks are to design a subset of a MIQSTURE interaction language oriented to Army tactical intelligence information processing and to evaluate the result.

The Need for a 'Mixed Initiative' System. The critical time factor of the tactical intelligence mission requires an information system that provides for rapid and effective information access and processing capabilities. However, a serious deficiency of current information systems is that the model on which they are based is essentially passive, that is. it is up to the user to recognize an information need and to seek out the required information via user-initiated communication with some information system. Thus, the user is the active element; the processes of information analysis and synthesis are user dependent and completely external to the system. The burden of information flow and control is on the user, who is forced to define a bottom-up query strategy involving many low-level questions to answer a high-level question such as assessment of enemy threat.

With the volume and complexity of available information on the increase, this burden is rapidly becoming intolerable to the user, whose abilities to assimilate and integrate information are essentially saturated.

What is required is the development of an active information model, that is, one in which the system, rather than only the user, is also an active agent capable of assuming part of the burden of information flow and control according to prestored information goals and algorithms.

The most critical component of an active information processing model for Army tactical intelligence is the user interface, which must be based on a user- and task-oriented mixed initiative interaction language as shown in Figure 1-1.

In contrast to the bottom-up, multiple low-level query approach, a mixed initiative interaction language provides a top-down, task-oriented framework that assists the user by supplying information in anticipation of the user's needs and by indicating appropriate alternative decision paths. The system underlying the mixed initiative interaction language is capable of generating hypotheses and making inferences based on new inputs (either user-initiated or system-acquired) and existing data.

The Importance of Balanced Initiative. Ideally, the mixed initiative query structure should be well balanced; for example, it should serve as a tutor to train the novice while at the same time it should be adaptive, utilizing input from the tactical intelligence officer who can add to the knowledge structures and procedures in the system's repertoire. Intelligence analysis is a skill rather than science; the experienced user should not be constrained by a rigid task-oriented structure, but must be allowed to manipulate the

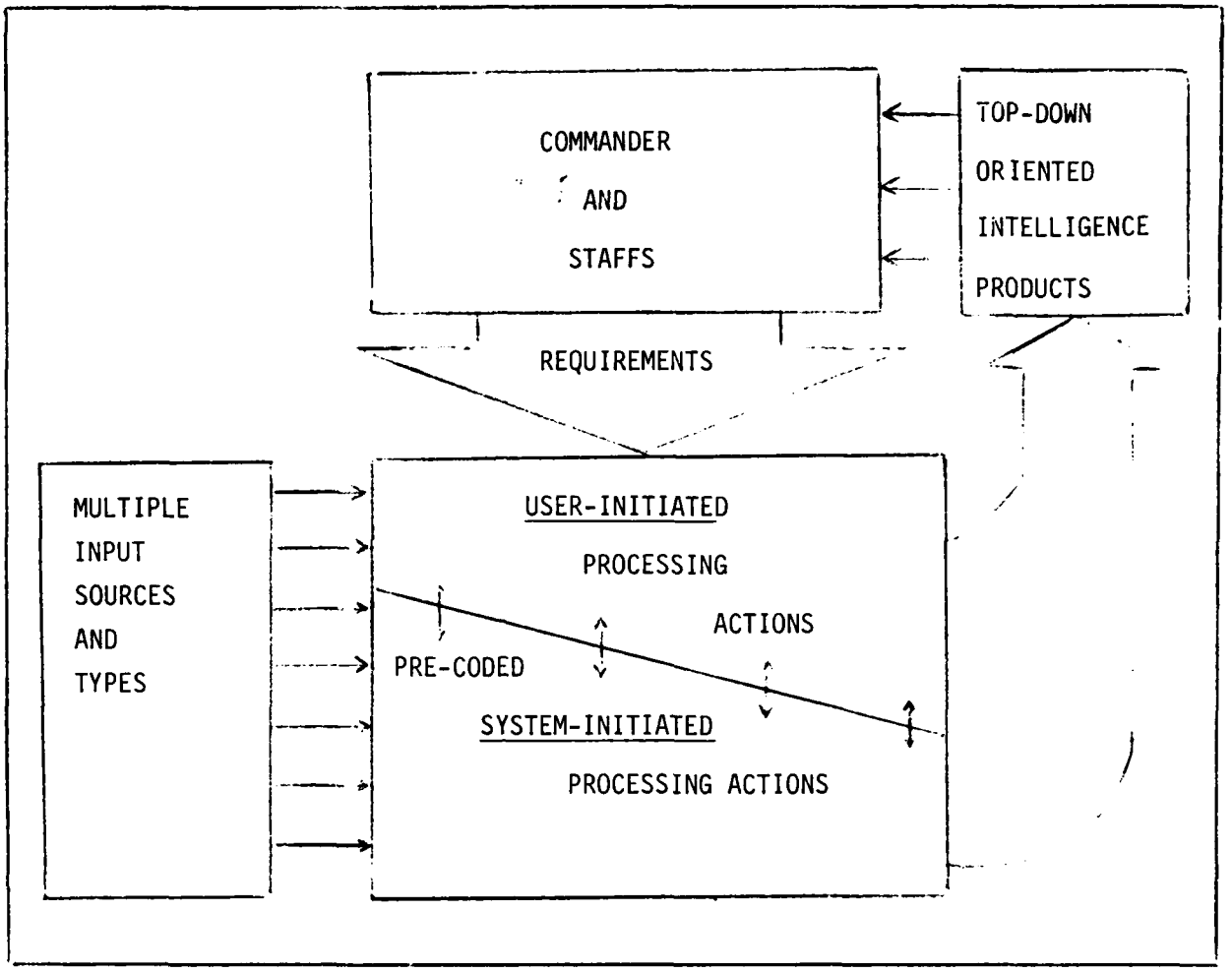


Figure 1-1. Mixed Initiative Processing System Concept[a]

[a]In the above figure, a major proportion of the early processing of raw inputs is based on automatic system-initiated capabilities, with user-initiated override features. Later stages of the processing, in which processed and stored inputs are further filtered, summarized, and assembled into finished intelligence products employ a heavier proportion of user-initiated interactions with the system. The system is immediately responsive to the commander's requirements for special products.

system as he desires, using the prestored algorithms when they serve his purpose and defining his own procedures as he wishes.

The first use of the phrase "mixed initiative" was by Carbonell (1970)* in describing the approach employed in "Scholar," the first CAI model to allow a student to interrupt his automated teacher. This represented a significant departure from previous CAI developments in which the student was not permitted to assume the initiative. The previous CAI developments represented an unbalanced approach to design of the user interface, since the initiative remained with the system.

As contrasted to a system in which the initiative is with the system, the typical information system depends entirely upon user-initiated information requests; it is unbalanced in having minimum system participation in the information seeking task. In a design study aimed at correcting this imbalance in current systems, OSI (1976)** defined some preliminary notions of an active information model centered around a data base that can introspect about its contents and procedures, using this self-awareness as a basis for dynamic derivation of information structures and goals to guide the user in his information-seeking task. This work provides the framework for OSI's approach to the design of a Mixed Initiative Query Structure with Task and User Related Elements (MIQSTURE).

Online Systems Technology. In the past decade, the number and variety of computer uses involving online interactive (conversational) techniques have expanded

---

*Carbonell, Jaime, "Mixed-Initiative Man-Computer Instructional Dialogues," PhD. Dissertation, M.I.T., June 1970

**Operating Systems, Inc. "An Introspective Data Base for an Active Information Model." OSI Technical Note N76-017, 17 November 1976

beyond optimistic expectations and have revolutionized information handling and processing practices in many fields. The popularity of the online mode of computer usage has many origins, including the immediacy of machine response to the user, the ability to make incremental adjustments to instructions given the machine, and the sheer dynamism of interacting with a conversational device.

The remote access interactive mode has been the keystone in making feasible a host of potential data-processing applications that could not bear the test of practicability when constrained to batch mode forms for using a computer. From this perspective, online technology has provided the missing link for application of the computer to areas of data-processing requirements previously not considered seriously for automation. In turn, the advent of automation possibilities for such areas has often wrought a profound transformation of what is deemed feasible, desirable, and necessary in terms of their data-processing support requirements.

Online Systems Technology in a Tactical Context. U.S. Army tactical intelligence processing activities are one among a number of Army information processing activities for which very significant support can be anticipated through use of online interactive computer techniques. A major characteristic of Army tactical environments is the short time periods available for transferring needed information from point to point. Compounding this problem is the absolute necessity for controlling, storing, and retrieving a sizable proportion of the real-time tactical information flow. Computers in a network arrangement with distributed storage and processing of data can meet these dual challenges, but a crucial link in the chain is represented by the human information processors in the system.

Humans are required at many levels in the system where information processing decisions must be made regarding the disposition of unique or complex information.

These decisions are reserved for humans, since making them is beyond the programmed repertoire of the machine. At the same time, the normal mental and language habits of the human decision makers are attuned naturally to modes of communication far removed from those most directly usable by computer systems. To adapt continuously to the unnatural language of the computer requires the use of mental resources that are thereby less available for the crucial work of making information decisions, i.e., doing tactical intelligence analysis. Moreover, the realities of personnel turnover during combat tends to produce situations in which comparatively less trained and experienced personnel may be assigned relatively complex combat intelligence tasks. Under time pressure, such factors in aggregate can effectively strangle user/system communication and deny to the user the processing resources represented by the computer system. In this light, the objective of "naturalizing" the communication interface between the human user and the system of computers is much more than an embellishment or a nicety. In truth, it may well partake of some of the qualities of the fabled horseshoe nail.

A major opportunity for improving user/system communications lies in improving the machine component of the system. In considering the user/system interaction process the designer should make maximum use of the machine's capabilities for forwarding that process. This can be accomplished by having:

- segments of interaction functions usually assigned to the user "off-loaded" on the machine system.
- the common language between user and machine match the user's conceptions and habits, leaving the task of "translation" to the machine.

- the machine with its impeccable memory support the user's fallible, short-term memory.

In other words, the machine must assume the initiatives in improving user/system communication wherever these are possible. Such a set of user/system interaction arrangements can thus be termed a mixed initiative interaction language. User/system interactions conducted through a mixed initiative language involve a continuous mixture of user and system initiatives for forwarding the course of the interaction process.

Project Purpose. The purpose of the project reported here is to investigate the applicability of the mixed initiative concept to an online language for user/system interaction involving U.S. Army tactical intelligence information processing tasks.

While the investigation is experimental in nature and confined to tactical intelligence processing, many of the design concepts and lessons learned are relevant to other online systems in structured environments of the kind found in other Army settings.

Project Task Objectives. To serve the purpose of the project, task activities may be conveniently sorted into four main categories reflecting objectives:

- Develop and elaborate a concept of a mixed-initiative interaction language: Mixed Initiative Query Structure with Task and User Related Elements (Subsection 1.2.)

- For Army tactical intelligence information processing, describe the functional requirements for an online interactive language. (Subsection 1.3.)

- Provide overall functional specifications of a MIQSTURE language adapted to Army tactical intelligence information processing needs, and more detailed specifications of selected subsets of the language to serve as concrete examples. (The overall functional specifications of a MIQSTURE adapted to Army tactical intelligence uses is presented in Section 2.0. Detailed specifications for selected subsets of MIQSTURE are presented in Section 3.0.)

- Evaluate the MIQSTURE language and report the results and conclusions. (Evaluation considerations, philosophy, procedures, results, and conclusions are presented in Section 4.0.)

1.2 The MIQSTURE Concept for Army Tactical Intelligence

The acronym MIQSTURE emphasizes the inclusion of task- and user-related elements in the machine-initiated side of the mixed-initiative interaction language. Together, task and user are sufficiently broad concepts to encompass most features of the user/system interface amenable to machine-initiated aids to interaction. The concept of mixed initiative is straightforward, but its underlying technical implications are not so immediately obvious. In apparency, the mixed initiative concept emphasizes machine-active (in distinction to machine-reactive) components and arrangements of the user/system interface. At the underlying technical support level, however, the concept implies availability of two main kinds of system properties without which the machine-initiated elements of interaction would be ineffective and short-lived. One is human-initiated override for machine initiated elements; the other is a generative capability to build and modify machine-initiated elements more or less "on the

spot." Later we will discuss these as premises for a calculated reliance on machine-initiated interaction activities.

The opportunities for defining machine-initiated activities with significant positive impact on the course of user/system interaction arise out of the points in interaction where the details are highly context dependent. If certain user actions are germane only to specific intelligence tasks, the system can use such contexts to infer and initiate necessary actions that would otherwise be left to the user to accomplish. In this manner, the system can perform the following functions:

- "Volunteer" information to the user, as well as summarized and approximate response in line with the perceived "intent" of a query.

- Prompt the user through complex and variable sequences of task steps he must perform. Figure 1-2 illustrates a segment of a mixed-initiative dialog relating to Task Representation/Control features in MIQSTURE.

- Recognize and diagnose certain categories of errors to which the user may be prone in certain contexts, and signal corrective actions.

- Monitor system input streams for conditions satisfying "templates" depicting situations to which the user must be alerted.

- Undertake ongoing and variable "housekeeping" activities, without being prompted, as these become appropriate in a developing context.

- Perform arduous trend analyses automatically in a "background" mode, presenting the results as points of consideration to the user.

- Store sequences of command actions and accomplish them at a single user- or system-initiated signal.

The achievement of active machine initiation of the elements (segments) of an interaction sequence is obviously a matter of degree. Some major proportion of the total interaction must remain human-initiated, or else the interaction becomes a machine monologue rather than a user/system dialog. Achieving appropriate active cooperation from the machine with unvarying consistency throughout the course of interaction therefore becomes an ideal toward which to attain more or less closely. Every appropriate machine-initiated element of interaction is based on the same fundamental principle: the storing of unequivocal knowledge in machine form. The knowledge may be about what follows what, about what means (stands for) what, about what is functionally equivalent to what, and so on. In each case, the knowledge must involve a stable and accurate prediction about future events and states of affairs. Otherwise the machine's future action will turn out to be ill-informed and faulty. This issue of machine pre-knowledge has several important implications for interaction dynamics, which are discussed in Section 2.1.

1.3 Army Tactical Intelligence Processing Requirements

A range of general (first level) functional requirements for an interaction language for Army tactical intelligence processing was derived from the following sources of information: Functional Area Descriptions for Ground Order of Battle, Enemy Situation, and Collection Management; Intelligence and Operations FMs;

Presenting situation: System has displayed an incoming SPOT REPORT message in a window partition of the user-analyst's text display surface.

USER: "TRUN MSR "EX

User enters three terms by pressing three function buttons so labeled. With this action he initiates execution of Task Run aids for the Monitor Spot Report task.

SYST: PROCESSING JOURNAL ENTRY BLOCK

System responds in another window of display (spot report remains). A tabular figure captioned JOURNAL ENTRY is displayed, which contains a system-assigned Journal Entry Number, plus data extracted and copied over automatically from the message header section of the Spot Report (source, time, message no., etc.). User reviews the machine-initiated entries, makes corrections (if any) via his keyboard and cursor, then moves task aids to next substep of task:

USER: "TMOVE (Single button action by user)

SYST: EVALUATE MESSAGE TASK BLOCK SKIPPED FOR THIS MESSAGE SOURCE TYPE.

SITMAP REFER BLOCK IN PROCESS.

CHECK AUTO-ASSIGNED N.A.I. STEP IN PROCESS.

System responds by skipping next block in normal task sequence (system has pre-stored instructions for mandatory skip of block for source type shown in message.) An appropriate notification is displayed to the user. The system names the task block in process (SITMAP REFER) and the first task step in that block (CHECK AUTO-ASSIGNED N.A.I.). Below the latter caption are displayed the titles of the current Named Areas of Interest for which the system has assigned message relevance to the Spot Report being processed. The user reviews these and may delete or add assignments for the report via keyboard and cursor.

(continued)

Figure 1-2. Task Representation/Control Features of MIQSTURE: Example Interaction

USER: "TMOVE

SYST: SITMAP REFER BLOCK IN PROCESS.

CHECK AUTO-ASSIGNED E.E.I. STEP IN PROCESS.

Same arrangements as for previous step.

USER: "TMOVE

SYST: SITMAP REFER BLOCK IN PROCESS.

CHECK AUTO-ASSIGNED HI-THREAT TEMPLATE STEP IN PROCESS.

NO AUTO-ASSIGNMENTS MADE.

OPTIONAL SKIP STEP? Y / N

System moves task run aids to next step, generates appropriate notifications. It has found no matches in the Spot Report contents to the currently active highest threat template. It so notifies the user and invites the SKIP option allowed in this case. A (Y) response from the user moves task aids to next step. An (N) response causes summary of hi-threat template to be displayed.

Figure 1-2. (Continued) Task Representation/Control Features of MIQSTURE: Example Interaction

Commander's Manual for Intelligence Analyst MOS; Extended interviews with personnel of the U.S. Army Intelligence Center and School, Fort Huachuca, Arizona; drafts of selected course support materials provided by these personnel; and draft documentation from the Intellience Preparation of the Battlefield (IPB) project team within USAICS at Fort Huachuca.

1.3.1 Summary of Findings

A major theme of documents concerned with U.S. Army tactical intelligence processing is the phenomenally increased rate of change of battle situations modern mobile surface warfare, as compared with even the recent past. Although terrain and weather factors change no more rapidly than before, maneuver and logistical aspects move much more quickly. The increased pace generates larger volumes of information than formerly and at the same time reduces the time available to the commander in which to react to changing situations. Thus the required rate of information processing and assimilation is magnified in two ways. Since the human capacities for assimilating information remain essentially constant, technical means must be found for repackaging information in summary forms for G-staffs and commanders in order to provide higher information yield per unit time spent in scanning and assimilating the information products.

To accomplish the necessary summary repackaging, two major processing requirements themes emerge for the automated support system: (a) store pre-analyzable information in ready-to-use form, and (b) maximize the use of semi-automatic processing of real-time data that cannot be pre-analyzed. Examples of pre-analyzable information include mobility corridors, fields of surveillance, and enemy tactical doctrine; summaries of such information can be stored in the form of templates and overlays for rapid call-up and assimilation. Examples of

semi-automatic processing for real-time data include several forms of continuity analysis. Automatic data correlation processes assess different data from the same or different sources and with observation times within prescribed time envelopes. The processes contain algorithms for deciding whether or not various data represent the same or different objects, events, etc. A "track" concept may or may not be involved. Automatic data fusion processes select or construct a data value to represent several input data values (usually from different sensor types) for a single object or event. Automatic alerting processes employ a combination of pre-analyzed data and real-time data; pre-analyzed data in the form of event templates are compared with input data streams by semi-automatic processes, and alerting messages are delivered when approximative matches occur.

In Figure 1-1, depicting the Mixed Initiative Processing System Concept, the top-down oriented intelligence products shown are comprised partly of the results of semi-automatic processes of the kinds described in the preceding paragraph. The online interaction language must have elements and arrangements for defining, loading, overriding, and operating the kinds of semi-automatic processes described; but attainment of highly serviceable and sophisticated end products for intelligence analysis requires that such special processes use adaptive data structures (Figure 1-3) supported by a number of subsystems, each of which requires online interaction language elements and arrangements. In addition to the above conclusions, a number of requirements for MIQSTURE were delineated. The general areas covered in the description of requirements are presented below along with a brief indication of the topics covered. A more detailed discussion of each of these areas is contained in Appendix A.

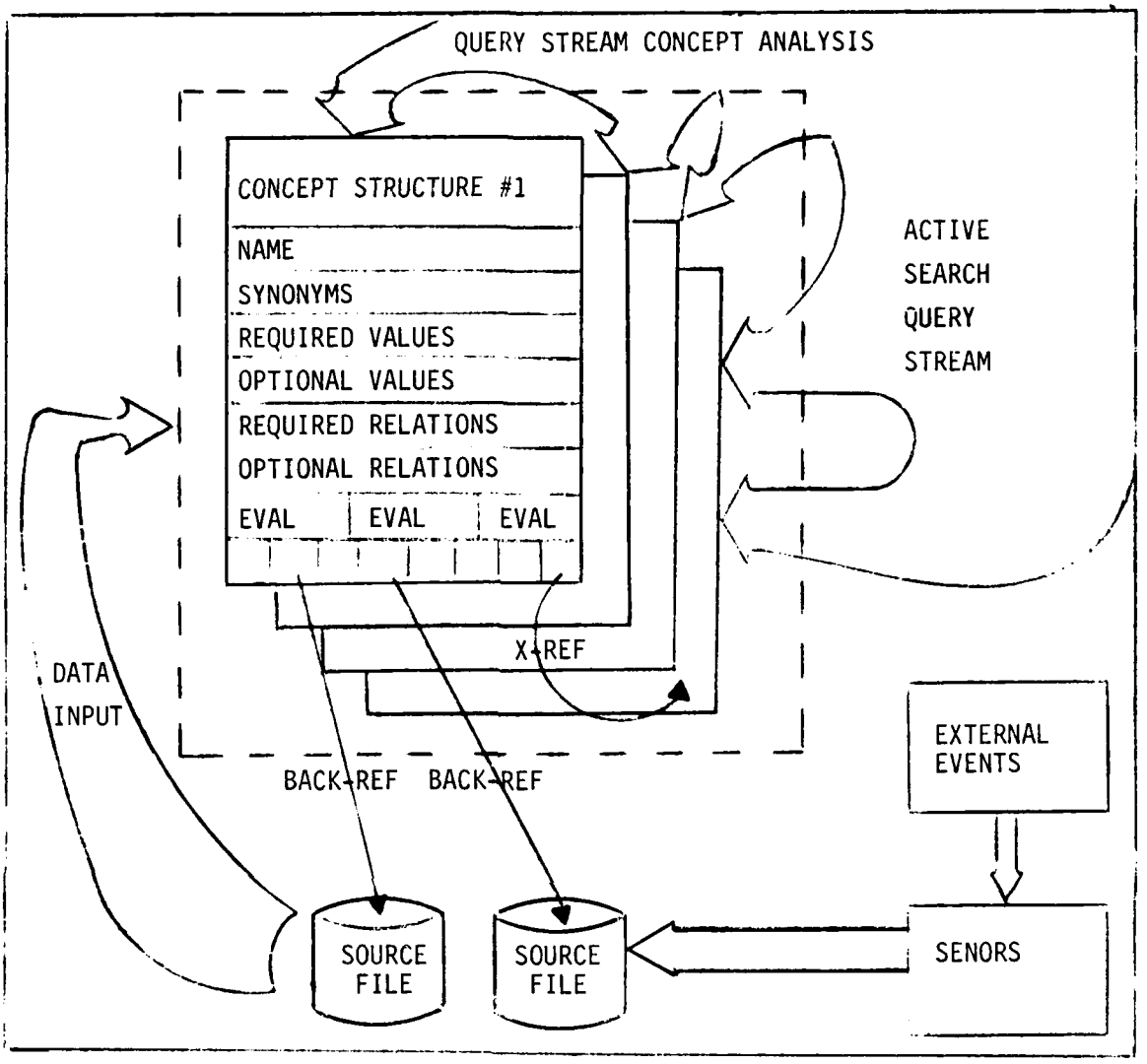


Figure 1-3. Adaptive Data Structures[a]

[a]Records of an adaptive data structure are shown in storage within the dashed line. The records are summarization products based on processed data stored in source files, which in turn have been filled by sensor inputs. The adaptive records carry back-reference pointers to their supporting source data, and cross-reference pointers to each other. The active query stream applied by users against the adaptive file is also concept-analyzed as a background activity to continuously optimize the file structure.

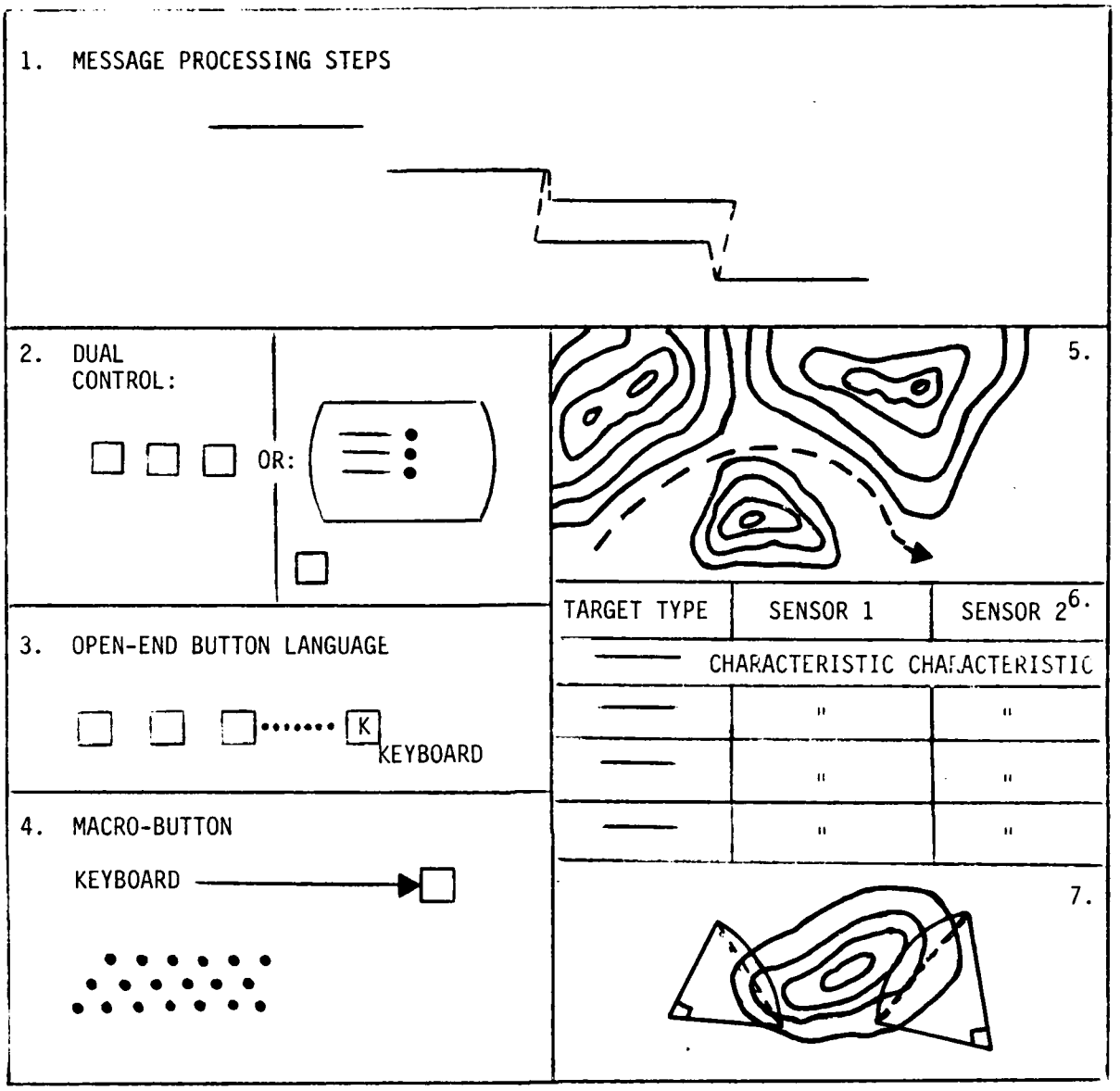


Figure 1-4. Graphic Aids and Input Modes[a]

[a]Sketches are provided for: 1. Task steps display for task representation/control aids discussed later in this report; 2. Illustration of interchangeability of function buttons and menu display for the same sequence of actions; 3. Illustration of combination of function buttons for command names and keyboard following for parameter and data values; 4. Illustration of keyboardable sequence summarized by single function button; 5. Sketch of automatic track calculation functions; 6. Sketch of tabular display for a collection management function; and 7. Example of display using line-of-sight fans.

General Description of Work-Station Components. Display surface requirements are sketched for terrain displays with combat symbology overlays, and surfaces for handling textual and numeric data and graphic aids. Keyboard, function-button modules, and cursor and display controls coordinated with the displays are also described. The descriptions provide context for the rest of the paper (Figure 1-4).

Display Subsystem Support Provisions. Requirements are sketched for types of terrain overlays that summarize forms of pre-analyzed information. Eight types of control language elements for terrain display overlay symbology are described. Online language element requirements are enumerated for screen history, map coordinate functions, and track calculations.

Task Management/Control Support Provisions. Tactical intelligence processing tasks as depicted by Functional Area Descriptions (FADs) are considered. Resulting requirements are sketched for elements (of an online language) that allow construction, modification, and operation of online aids to accomplish complex task sequences. Requirements for task overload, load-smoothing, and task backlog management functions for the language are also described.

Message Files Support Provisions. Requirements for maintenance of message files separate and distinct from data base files are described. Online language elements for defining, loading, maintaining, and querying such messages files are outlined. Several message processing functions amenable to semi-automatic computer applications are sketched. Language requirements for manipulating security and access arrangements for message files are considered.

Data Base(s) Support Provisions. Information flow requirements from message file to data base are sketched. The need for a family of file structures

of varying data precision and completeness is considered. Storage requirements for secondary characteristics of data (security codes, perishability data, reliability, etc.) are noted. The need for cross-reference structures is cited. Requirements are discussed for data continuity analysis of the kind introduced earlier. Implications for design of an interaction language are examined.

Specialized Support Functions. Requirements for computer-based services, to replace or augment hand-held calculators and "whiz-wheels" are exampled. Such functions are highly specialized and need to be separable from the data they involve, since they may be used for "one-shot" types of analyses. Another class of specialized support requirement is for language elements to build and maintain process and event templates that are stored in the computer system for comparisons with input data streams.

System Configuration/Degradation Control Support Provisions. There are requirements for functions ordinarily reserved for computer operators to be accomplished by personnel with only user-level training. These involve setting up the system for possible degradation and reconfiguring it in the face of such problems. Requirements for interactive language for such functions are listed.

Across the range of interactive language requirements an overriding requirement exists; this is for an understandable functional concept of the overall system of language for the user-system interface. Figure 1-5 sketches one such concept. In the figure, the gross functional relations of a MIQSTURE interaction language are diagrammed. Those elements of the language used most often by operational personnel (but, to an extent, by system management personnel also) are depicted

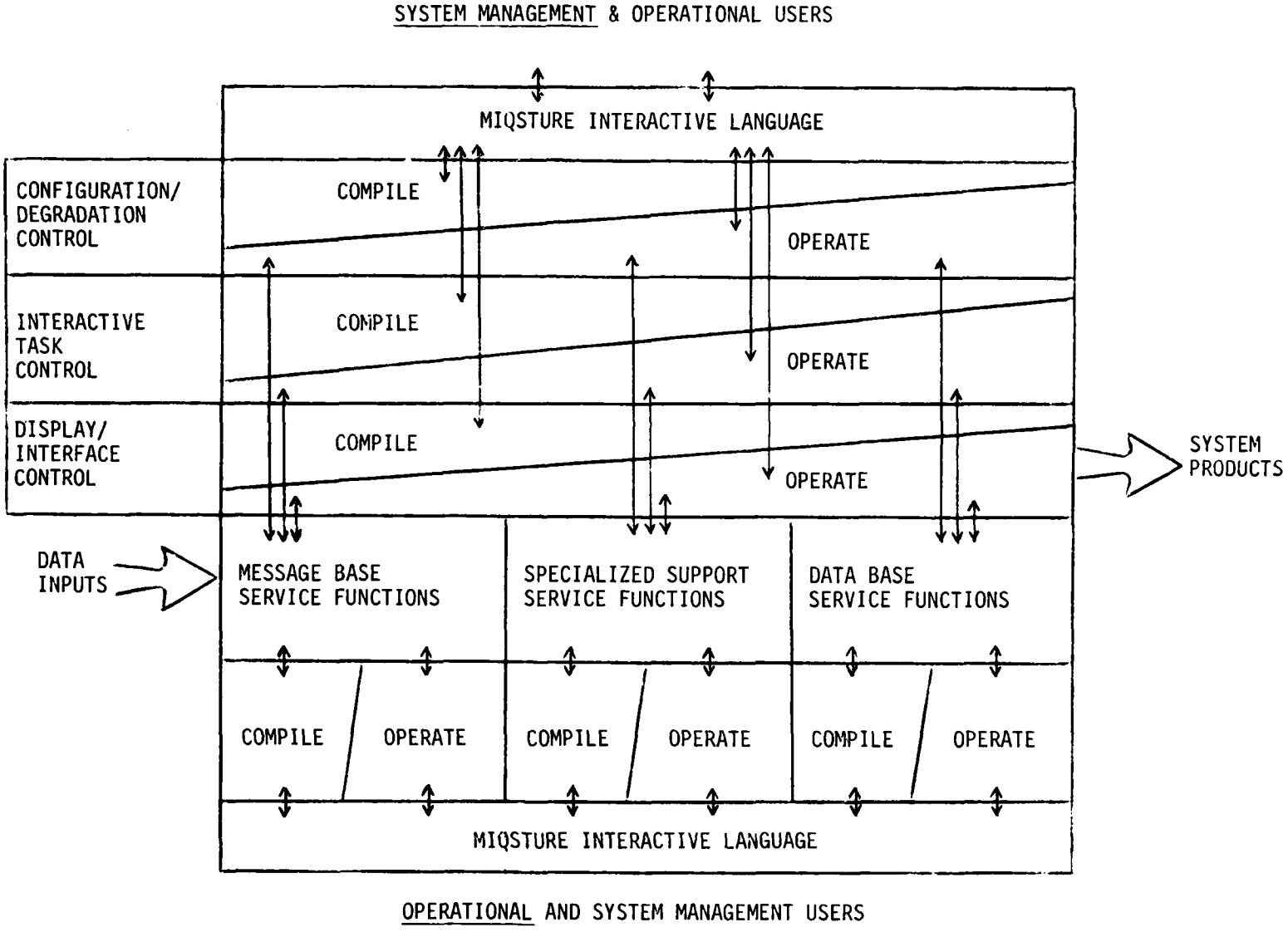


Figure 1-5. MIQSTURE Functional Relations Diagram

by the top-most row-box. The interface between the language and the message base, specialized support, and data base service functions has, in each case, a compiler component and an operational component, since significant changes or reconfigurations to any of these functions will be mediated through compile techniques. A corresponding set of relationships holds between the elements of the language used for interaction with the other three major system functions depicted in the three boxes on the left margin of the figure. The main flow of data is in through the message base functions, on to the specialized service (especially templating) functions, and from thence to the data base functions, where it is very likely to be recirculated through templating functions after it has been stored. The operational features of the display/interface control functions are in direct contact with all three of the above major processing stages and thus conduct the system products to the users. Finally, the operational features of the configuration/degradation and interactive task control functions are also interfaced directly with the three major processing stages, thus affording overall system control.

2.0 SECOND LEVEL FUNCTIONAL DESCRIPTIONS OF POSSIBLE MIQSTURE REPERTOIRE

2.1 General

The services that might be delivered by an automated data-processing system to the user-analyst for Army tactical intelligence are very broad. The scope of a potential MIQSTURE language is correspondingly broad. Both for purposes of designing and for evaluating any subset of such a potentially extensive language system, overall system perspective must be maintained. Each part of the language attains much of its significance in relation to the whole system. One must decide: into which subset of language a particular function most rightly belongs, and why; the functional domain of the language; how to smoothly articulate (interface) the various parts of the language, both from a computer logic and a human psycho-logic viewpoint; and the overall shape and patterning of the various parts of the language repertoire into a whole that best fits the user-analysts' natural perceptions and expectations.

The aim of Section 2.0 is to provide a perspective and framework for MIQSTURE so that the selected MIQSTURE subsets presented in more detail in Section 3.0 can be better understood. In the present section, language subsets with both high and low priority for MIQSTURE are described; in Section 3.0 the subsets selected for detailed treatment are all of high priority.

2.2 An Augmented Definition of Mixed-Initiative

To this point, our definition of mixed-initiative has emphasized that the machine's ability to participate actively in the user-system dialog is dependent upon detailed knowledge of overall processes being served. The knowledge must

be stored in machine-accessible and -interpretable form. At this point, an interesting set of conflicts begin to emerge. The machine can appropriately initiate actions regarding only pre-defined events and conditions, but on the other hand it can "recognize" situations that deviate from the expected (pre-described) events and issue a warning. Issuance of such a warning can be very valuable, but on the other hand most real-world events deviate somewhat from idealized prediction "models," raising the possibility of swamping the user with deviation warnings. Specific aspects and details of interactive task contexts form the basis for the most useful machine-initiated actions, but the detailed aspects of situations and events are the most variable. Machine-initiated interaction activities can indeed "offload" the user and thereby increase the load throughput capabilities of the user/system combination, but the system cannot cope with unpredicted change without having its internalized knowledge modified to fit newly emerging circumstances and conditions, which only the user can define.

A generalization from the above observations is that much of the knowledge of situation and context upon which many of the most valuable machine-initiated actions depend is of a contingent and changeable nature. This fact provides the basis for refining the definition of a mixed-initiative interaction language beyond the concept offered earlier. The original concept is retained, but augmented so that a mixed initiative interaction language is defined as one that provides language elements and arrangements for:

(1) Mediation of active machine-initiated contributions to interaction load-sharing (of the kinds sketched earlier in this discussion.)

(2) Manual (user) real-time backup/replacement for the subset of elements and arrangements in Item (1) which reflect contingent and changeable knowledge about the situation and context defining the data processing tasks to be accomplished by the system.

(3) Revising definitions and constructing new definitions to the system for the elements and arrangements in Item (2).

In this augmented definition of the mixed-initiative concept, both parties to the dialog (the system and the user) provide two main levels of intermeshed initiatives (non real time and real time initiatives).

NON REAL TIME INITIATIVES

A. System Initiated: Provision of a framework for fill-in capabilities that can receive a certain range of knowledge about the potential world of interaction for the particular environmental and task contexts involved.

B. User Initiated: User-formulated fill-in activities for the contents of the above framework, depicting the details (at various levels) of the interaction elements to be carried (initiated) by the machine in this context.

REAL TIME INITIATIVES

C. System Initiated: Interaction elements evoked during real-time interaction, as per information provided in Step B.

D. User Initiated: (1) Elements of real-time interaction (mostly complex information decisions) not assignable to machine through Steps A and B.

(2) Override activities, where machine-initiated interaction elements are judged insufficient to a situation.

(3) Additional (Step B) activities to adjust and refine the machine-initiated repertoire to better fit predicted trends.

In the version of MIQSTURE elaborated in Sections 2.0 and 3.0 of this report, the elements and features described can be summarized as contributing to four main functional goals of the language. These are:

1. Change/Adaptivity Goals
2. Task State Sensitivity Goals
3. User Learning Support Goals
4. Interaction Format Optimization Goals

A particular language element or arrangement often contributes to more than one of these goals, and it is useful to enumerate them in order to sharpen the focus of both design and evaluation thinking.

The following sections will discuss MIQSTURE user/system interface design assumptions under three headings: User Interface Subsystem Adaptivity, Task-Generalized Interaction Arrangements, and Task-Specialized Interaction Arrangements.

2.3 User Interface Subsystem Adaptivity

A number of requirements define the framework within which adaptivity of the interface subsystem must be considered. First, because of differences in assigned responsibilities and in problem-solving styles, Army tactical intelligence user-analysts will develop different patterns of frequently used interaction sequences with the system. This means that provisions should be available to allow (a) building different interaction sequences using the elements of the MIQSTURE interaction repertoire, (i.e., everybody doesn't have to use the system the same way even when the overall goal is the same), (b) storage, maintenance, and re-display on demand of the contents of such sequences, and (c) use of such sequences as inputs to the system via simplified command actions (i.e., a potential for building macros to make these sequences easier to perform for the particular user).

We can assume that there will be new requirements generated as users develop confidence in a system like MIQSTURE. New language elements will be required to support the increased capability. This means that provisions must be available to allow: (d) easy additions and modifications to the elements of the interaction repertoire, and (e) relatively automatic incorporation of the new elements into the explanatory, tutorial, and user-error control service features of the subsystem, and in user manuals.

A design incorporating several features described briefly below can provide the required adaptive user interface.

Keyboardable Substrate Input Language. As a baseline capability, the interaction input string parsing routine will be designed to interpret a

keyboard-oriented command and data language. Command actions taken via the media of function buttons and/or cursor selection from displayed menu formats (or single keystrokes as answering responses) will map into and trigger stored keyboard-oriented command contents to be placed in the command string passed to the input parsing routines (described more fully in Section 3.0).

Command Summarizing Overlays. Two kinds of command summarization capabilities will be provided: public and private. Both capabilities will provide for executing complex command strings (command names, parameters, executions, etc.) via highly abbreviated sequences of button actions and/or cursor selection from a menu display format and/or keyboardable fill-in formats. The public version will be under strict system administration control and will be available for use by all users (i.e., will be part of the standard system repertoire). The private version will be employable by individual users to construct command summarizing overlays for their private use. The elements of the keyboardable substrate input language will not be able to be modified or augmented by these procedures. However, new command summarization names can be produced and can be selected by function button or by cursor and menu display. The command strings associated with the names will be stored for future use.

Decoupled Substrate Language Forms. The form recognition repertoire of the input parsing routine will be stored in tables with pointers to coded entries that constitute the input values used to elicit associated processing routines. System administrator-initiated changes in substrate language forms as well as command synonym sets will thus be easily handled. The tabular organization will also provide ready-made access for "filling" standardized online explanatory and cueing routines for the user, and for generating add and change pages for user manuals.

2.4 Task-Generalized Interaction Arrangements

Generalized functions may be conveniently grouped into five main classes: Interaction Cueing and Facilitation, Interaction Traffic Control, Input Error Forgiveness and Correction, Learning Acceleration Devices, and Individualization and Continuity Aids. Generalized interaction arrangements are important for the support of all system processing tasks and consist of a combination of user interface arrangements and software program and machine hardware capabilities. Generalized functions are central to the establishment of a user interface framework within which learnability, useability, efficiency, and extensibility of the interface repertoire can be realized.

2.4.1 Interaction Cueing and Facilitation. Means for maintaining the integrity, smoothness, and relative effortlessness of the interaction process are important for online interaction interfaces. Without interaction cueing and facilitation, user movement may be frequently thwarted and the user may develop a negative attitude. Means of interaction cueing and facilitation include:

Equipment Status Indicators. The interaction chain may be broken immediately if equipment malfunctions or is improperly set. Equipment status indicator devices and control switches must be prominently located and labeled with the "distracted" user in mind. Status indicators and controls for all equipment affecting user interaction should be grouped in as few locations as possible and prominent to the user's sphere of attention.

Login Error Default Path. Login procedures are noteworthy for more than their share of frustrated interaction processes. Login, User Identification

(USERID), validation, password, terminal or channel identification, system address or facility to which connection is desired, and similar information must often be passed before the user is granted access to the system. Characteristically, the user is in his "warm-up" phase and the system's response to error is an enigmatic "try again," followed by disconnection after repeated errors. (Disconnection is necessary to prevent successful access to the system by unauthorized users via trial and error and feedback sessions of attempted Logins.) A better solution is to provide the Login Error Default Path: Login actions are recorded; repeated error results in a system-initiated switching of the erring terminal to a monitor terminal from which help can be provided.

Accidental Interrupt Recovery. Momentary malfunction of any component in the chain connecting the user to system resources may interrupt interaction. In such a case, the user needs system-initiated: (1) means to diagnose the source of interruption to assess whether he can take actions; (2) means to immediately "buck the problem" to an appropriate correction capability if it is beyond his control; and (3) means to save the record and products of his interaction in order to resume at the correct point when system capabilities are re-established. The advent of "intelligent" terminals allows some diagnostics to be performed from both ends of the system chain and also makes it possible to decrease the proportion of the interaction record that may be lost through an interruption.

2.4.2 Interaction Traffic Control. Message traffic between user and system is a dialog in which each party must have an accurate picture of the dialog status at each step. Even in applications in which communications flow in only one direction at a time (as in many current online systems), "traffic

signals" have been found essential. Most successful programs signal the user regarding whose turn it is to send the next message. Some systems also provide periodic activity signals when processing time exceeds a few seconds. Some provide response-time estimates for large processing tasks, and some deactivate input devices during periods when the system cannot accept further inputs. For work-station terminals with the requirement to process more than one task at a time, a multiple dialog status display is required. System-initiated activities identify each dialog stream, show the last interaction step for each, allow a history review for each, and allow processing status reports for each.

Message Source/Type Distinguishers. System-initiated message source/type distinguishers become important in systems with multiple message sources and message types and help to speed interaction and reduce error. Even systems with only two sources of messages (system and user) have found it desirable to distinguish message source in the display record of the interaction dialog. The three main kinds of distinguishers available are: Print Font (upper and lower case, etc.); Indentation, Spacing, and Underlines; and Explicit Naming. Although modern display technology makes color and gray-scale available for use as distinguishers, these may have some drawbacks: They do not carry over to hard copy printer outputs, are somewhat more vulnerable to masking by ambient light, and are less uniformly discriminable across the population of visual abilities. More research is needed on this topic.

Attentional Devices. For some applications and under certain circumstances, the system must be able to reliably co-opt the user's attention. Audible signals can introduce an attentional device, but must be able to be muffled or

turned off. The main visual cues are position on the display, brightness differentials, and motion. Blinking combines brightness differential and a form of motion. A maximum effective configuration involves a system-initiated pairing of a blinking signal at the display position of interest with a corresponding blinking indicator at a display area reserved for attentional device announcement. This is the suggested attentional display for MIQSTURE.

2.4.3 Input Error Forgiveness and Correction. The most frequent sources of errors in user inputs for experienced users are keyboarding errors and momentary lapses of decisional attention in which an unwanted action is taken and almost immediately recognized as such. (Wrong actions flowing from inadequate knowledge of the system or of the task context are considered part of the user education requirement of the system). Error forgiveness and correction system capabilities focus on "mistakes" that the user is aware of in principle.

Command Keyboarding Errors. For command inputs accomplished by keyboard (not by function button or cursor/menu/select button), there is the ever-present possibility of typographical misspellings of command names or synonyms. A typical system response is to reject the input and produce a brief error message. A more forgiving or "cooperative" system response is based on a command-discrimination mechanism for interpreting keyboard inputs signalled (formatted) as command messages. In this approach, command names, synonyms, qualifiers, and some simple parameters are table-stored, and the contents of the input string are compared with the tabled values. Failing an exact match, the command-discrimination subroutine will branch to another that locates the alpha-numerically most similar tabled value and outputs it to the user in the form of a system-initiated question: "Do you mean ____? Y/N" A single keystroke then moves the interaction sequence forward.

Data Language Keyboarding Errors. For data-specification-dependent operations, the possible input vocabulary diversity is so immense that keyboarding is the predominant mode of entry. If an erring keyboard entry is matchable to corresponding system contents, the system will treat it as a correct entry and proceed accordingly. If the user does not recognize this type of error, nothing more of an immediate nature can be done about it. If the user does recognize the error rather quickly and is well advanced along the course of an instruction sequence of many steps, he may find it highly beneficial to be able to "cancel" or "undo" the last one, two, or three steps in the interaction sequence. For some types of tasks it is feasible to store the command history and the system processing results in coordination so that such canceling actions become possible. On the other hand, when the system finds no match for a keyboarded data specification input, the "no-match" message it returns to the user must repeat the no-match input item, not only to provide exact information on the focus of mismatch, but also to provide a second opportunity for the user to detect a keyboarding error if one has occurred.

Unwanted Actions Errors. In many cases, an unwanted action produced by a lapse of attention or by a rapid change of mind can be allowed to run its course quickly without any greater penalty than a short delay in pursuing desired interactions. In other cases where a long delay for unwanted processing will ensue, a processing interrupt command capability is essential. In some cases where an action is only belatedly recognized as unwanted and is already embedded in a developing sequentially dependent task/structure, the "undo" capability described in the previous paragraph may provide benefits worth much more than its extra system costs.

2.4.4 Learning Acceleration Devices. Learning acceleration arrangements seek to speed up the rate at which a user will assimilate system-related knowledge by sharply reducing the user effort required to acquire knowledge at the point in time when he most feels the need for it. Several types of such system-embedded learning acceleration devices can be distinguished: Output Message Abbreviated Mnemonic Cues, Command Term Mnemonic Abbreviations, State-Linked Hierarchized HELP Materials, Coordinated Online/Offline Tutorials, and Simulation Practice Capability.

Output Message Abbreviated Mnemonic Cues. The power of the user's associative learning capacities will be harnessed in a relatively unobtrusive and effort-free way to help him to learn rapidly to recognize very short mnemonic forms of the standard system output messages used in the course of interactions. In this technique, each standard system message is output in a two-level "mnemonically nested" form, mnemonic term first and corresponding mnemonic phrase last; this is the "paired-associate" pattern of learning. The user can judge when he is able to associate the term with its corresponding phrase sufficiently well to anticipate the phrase reliably; at that point, he takes a simple command action to eliminate output of the phrase from then on. If he should later suffer a lapse of anticipatory recognition, a single keystroke (such as ?) following the message output will reinstate the repetition of the longer phrase for a single cycle. The learning process carried out at the users pace can help to eliminate the need for the phrase, leaving only the short mnemonic and thereby streamline the interaction process.

Command Term Mnemonic Abbreviations. A similar arrangement to that above depends upon both reinforcement (reward) and anticipatory (associative) learning

principles to provide relatively effortless learning of keyboard input command terms and abbreviations. The abbreviations are stored as synonyms in the command recognition table. If a no-match input is keyed, the command discrimination device described earlier (under Command Keyboarding Errors) locates the most similar tabled form, whether abbreviation or main term. It then outputs a system-initiated clarification request message organized very much like the multi-level output messages described in the preceding paragraph. For example:

MEAN: "PT"--"PRT"--"PRNT"--"PRINT"? Y/N

displays all legal abbreviations in ascending order of their length, stimulates associative learning, and allows the interaction to proceed with a single keystroke.

State-linked Hierarchized HELP Materials. If the system maintains a total record of the process for the two or three preceding steps of an interaction sequence, this information can be used to sharply improve the accuracy of the HELP explanations and materials provided to the user when he issues such a command. Without such state-linked capabilities, the HELP-menu first offered the user will be very general and will be the same each time he asks for help. With state-linking, the varied first menus offered can usually contain several well-focused, narrowly pertinent choices. Once the user indicates his particular predicament through the menu choice, help can be delivered in a hierarchized form with levels of elaboration added only if the user indicates his need for them. For example:

SYST:

OMITTED PARAMETER --MORE? Y/N

USER:

(Y)

SYST:

THE (STAGE) COMMAND MAY PRODUCE THE CONDITION YOU INDICATED IF OPTIONAL PARAMETERS ARE INADVERTENTLY OMITTED --MORE? Y/N

USER:

(Y)

SYST:

THE OPTIONAL PARAMETERS ARE: _____, _____, _____, etc.

Aids of this kind are incorporated in MIQSTURE.

Coordinated Online/Offline Tutorials. Continued requests for HELP on a certain menu-selected topic will cause the system to initiate a suggested branching to a full-scale online tutorial covering the topic in a systematic way and discussing its context more thoroughly. Such tutorial material does not exercise the user, but is in effect an "online manual." As such, the material is highly coordinated with the published manual (that is in ring-binder form), and is heavily cross-referenced to it. Thus, a continuous unbroken reference path is provided from user-initiated requests for HELP through to the most detailed and exhaustive treatments of the relevant topics. The user's path is thereby eased to information at the time of his immediate need.

Simulation Practice Capability. Another learning acceleration option that can be incorporated into the user interface subsystem is a special limited purpose CAI capability aimed only at exercising the user in interaction skills. This capability allows lessons to be developed, stored, and executed through

the regular interface facilities. The two main ingredients that are added by this capability are: (1) ability of the system to initiate exercise problems through the work-station interface, to which the user can respond by using the system, and (2) system-initiated monitoring of user responses to the lesson instructions, so as to provide the learner corrective feedback as well as provide system-initiated evaluation of his level of competence in learning the lessons. Ordinarily, lessons are developed for the primary or initial skill levels and for very complex user tasks that have high error-penalties associated with them in the real world. Again, the user's path is eased in acquiring these skills at the precise moment when he most feels the need.

2.4.5 Individualization and Continuity Aids. Such aids take advantage of the fact that the day-to-day work patterns of various users are (a) different and (b) often quite predictable. In effect, the system initiates marshalling of such aids to create a "tailored work environment" for each user, thereby eliminating many of the requirements for him to re-adapt to the system each day and freeing more of his energies for productive work. A number of such aids can be employed: Automatic Command Rename, Stored Command Sequences, Audit (History) File, Work-Save Store, and "Desk-Top" Indexing for User-Analyst Individual Work Files.

Automatic Command Rename. Because command terms are decoupled from processing programs and table-stored, it becomes very easy to provide a RENAME command that allows the user to insert his preferred name for a command into the system, thus making his interactions easier. The renamed command is in effect only for his terminal. An option allows the rename action to be in effect only until he logs out, or else to be re-initiated by the system each time his User Identification (USERID) logs in.

Stored Command Sequences. Sequences of commands, specifications, and parameters for guiding system processing and production tasks can be stored as "macros" once they have been developed and perfected by the individual user. Two varieties of macros can be distinguished: (1) standardized components used in a number of different kinds of tasks so that use frequency is very high, and (2) Major divisions or totalities of task structures used less frequently. The former can be made invokable by a single function-button of which several may be reserved for user individuation. The latter will usually be invoked by selection from a menu display elicited by a command. The inherent flexibility of the MIQSTURE interface subsystem design allows for a wide variety of individual options for this capability.

Audit (History) File. In addition to storing both the complete interaction details and the resulting system response products for several preceding command actions for each user (uses were discussed earlier), a record of command actions taken can be stored for many more (e.g., 100) preceding steps. The user often finds a review of this record very valuable in maintaining work organization for complex task sequences, especially those with unique or untried aspects. It also provides another safeguard against loss of continuity from interruptions.

Work-Save Store. If the user must log off the system before a task process has generated all its products or before all interactive task specification work has been completed, he will often wish to save the entire task structure as contained in the system at that point. An OTL

(Out To Lunch) option stores the material only until the system is taken down or until the user logs on again and claims it within that computer run period. A ICY (I'll Call You) option stores the task indefinitely.

"Desk-Top" Indexing for User-Analyst Individual Work Files. A main measure of effectiveness of a user work-files capability is the ease with which the user can relocate materials he has stored so much earlier that he cannot recall their exact location. "Desk-top" indexing addresses this problem with the goal of making the indexing and retrieval tasks for user work files no more effortful than the visual-access methods used by most persons for printed materials on their desks and in their bookcases. It consists of several arrangements. The user must name each file he creates, and the system automatically appends the creation date. A simple command produces a display of file names and creation dates for existent files. Each record in each file must be given an explicit beginning and ending sign by the user before it can be transferred to be stored in the system. The system automatically appends a unique accession number, date/time group, and file order number to each stored record, which are displayed as part of the record header field. Records can be retrieved by any combination of file name, date/time group, accession number, or file order number. A retrieved record is immediately displayed. A BROWSE command with skip capabilities allows movement back and forward in the file order, locking on each record header with each succeeding button action. Another command allows the order of records in a file to be changed and records to be moved from file to file. In this action, only the file order number of the record changes. Full text-editing capabilities are available for the record

contents only. Record-delete and file-delete commands require double confirmation from the user. A file status display lists the contents of each file via date/time group, accession number, first line of each record, and its status (active, deleted). This display is maintained automatically by the system on the basis of user actions. By command action, the user can also delete listings of deleted records from the display.

2.5 Task-Specialized Interaction Arrangements

2.5.1 Introduction. In a system as complex as a tactical intelligence processing system, a major goal guiding the design of task-specialized interaction arrangements at the user interface must be to develop a descriptive scheme for the arrangements that accords well with the representative user's view of system functions. Each command name, display type, etc., is understood and remembered (and therefore easily used) by the user mainly in terms of its usage context. For users having no great familiarity with computer programming and how computers work, the system processes in which the user participates are understood primarily from the functional point of view, i.e., (a) the process objectives they support, and (b) the types of data involved. By contrast, the computer program analyst's view of the same processes tends to be in terms of the underlying program building blocks.

If the representative user of a system is computer-programming sophisticated, the descriptive schema for interaction arrangements may be (for some, but not all, purposes) better explained in terms of program building blocks. However,

if the main business of a system is accomplished mostly by static combinations of program building blocks, a programming-sophisticated user will find no real advantage in a "higher order" DP language interface except of its initial familiarity to him; in distinction, a progamming-naive user will find major disadvantages to such an interface scheme. For the Army tactical intelligence system it is clear that: (a) most daily business will be accomplished by static arrangements of program blocks, and (b) most users will not be highly computer-programming sophisticated. The choice of user's view to guide development of the interaction element descriptive scheme is therefore obvious.

These considerations suggest the following major kinds of functions (combinations of process objective and data type) that we think are most easily conceptualized by the representative intelligence analyst user of the system:

1. Startup/Shutdown: Includes processes for connecting and disconnecting a work station to the active system. (Sketched in subsection 2.5.2.)

2. System Navigation: Includes processes for guiding the user to information about system capabilities appropriate to particular data processing tasks. (Sketched in subsection 2.5.3.)

DEFERRED TO SECTION 3.0:

3. Simple List Records and Conventional Querying: Includes processes for specifying search and display parameters for identifying and perusing information contained in simple records consisting of categorized lists of data. (Detailed in subsection 3.2 and 3.3.)

4. Alerting and Input Driven Automatic Querying: Includes processes for specifying search parameters for formulating automatic queries for alerting functions and for automatic input-driven file searching functions. (Detailed in subsection 3.4.)

5. Tabular/Hierarchical Database Query and Calculation: Includes processes for specifying search and calculation parameters for identifying, summarizing, and reporting out data stored in tabular formats and in hierarchies. (Detailed in subsection 3.5.)

6. Task Representation/Control: Includes processes for defining task-knowledge structures to the system, and for using such structures as aids to interaction with the system. (Detailed in subsection 3.6.)

7. Terrain Overlay Symbol Data Reference: Includes processes for cross-referencing information in terrain-based displays with information in system files. (Detailed in subsection 3.7.)

8. File Define/Maintain: Includes processes for defining, loading, modifying and purging contents of the information files of the system. (Detailed in subsection 3.8.)

---

9. Data Staging: Includes processes for transferring blocks of data among subsystems, and for transforming data during transfer. (Sketched in subsection 2.5.4)

10. Analytic Processing: Includes processes for invoking and managing specialized on-demand (non-routine) data processing routines. (Sketched in subsection 2.5.5.)

11. Print Products Production: Includes processes for organizing and producing printed products from information contained in system files. (Sketched in subsection 2.5.6.)

12. System Administration Support: Includes processes for recording system activities at several levels, and for summarizing this data in automatically produced reports (Sketched in subsection 2.5.7.)

13. Interface Adaptation: Includes processes for changing and augmenting system navigation aids, system messages, diagnostic, error and learning aids, and USERIDs. (Sketched in subsection 2.5.8.)

Within this framework the repertoire of elements of the MIQSTURE user/system interaction interface is meaningfully organized. Before discussing each function in more detail, four points about the interaction arrangements are reinforced:

(1) Need for Comprehensive Overall Conceptual Framework. A typical sequence of use of types of interaction elements in pursuing a tactical intelligence analysis task ranges across half the above categories. For example: A simple message search might involve a sequence such as STARTUP, QUERYING, DATA STAGING, PRODUCT OUTPUT, SHUTDOWN; a complex data base statistical report by an inexperienced or "rusty" EOB user might involve a sequence such as STARTUP, NAVIGATION, QUERY DEFINITION,NAVIGATION, QUERY DEFINITION, DATA STAGING, QUERY DEFINITION, NAVIGATION, ANALYTIC PROCESSING, NAVIGATION, DATA STAGING, PRODUCT OUTPUT, PRODUCT OUTPUT, NAVIGATION, PRODUCT OUTPUT, SHUTDOWN. The point is that very long, complex sequences of interaction are sometimes necessary to make use of the system's potential. In order for the user to be able to switch confidently and accurately between the different steps in such sequences, a readily comprehensive overall conceptual framework for the system is necessary.

(2) Capabilities for Concatenation and Simplification of Elements. For frequently repeated sequences of element use (such as the simple message

search cited above), a standardized command concatenates a sequence of several elements so that they become "invisible" to the user. For example, the simple search might become STARTUP, QUERY DEFINITION, Terminal Display Command(s). In this case, DATA STAGING and PRODUCT OUTPUT specifications are simple, standard, and have been pre-arranged, so that these steps are effectively skipped in the interaction process. In the same vein, a very complicated set of interactions such as the statistical report example above is standardized and invoked by a function key action to produce a menu display, a cursor/button selection from the menu, and a keyboard fill-in for data ranges to be used for searching. The product might be entitled "Daily Statistical Summary on XYZ." The point is that the user is not "stuck" with the flexibility, but is able to move to very succinct, rapid forms of interaction with the system in the many cases where they are appropriate.

(3) Need for Modifiable or Augmentable Interaction Elements. The kinds of interaction elements enumerated for each of the categories in the discussion that follows should in each case be considered an open rather than a closed set, and modifiable or augmentable as needed due to the adaptivity built into the interface subsystem.

(4) Need for Synonym Capability. The exact form of the command names, qualifiers, and parameters used in the examples presented later are of no major significance, because of the substrate language decoupling arrangements already described, terms can take almost any shape desired and can be easily changed (under control of the system administrator, of course). Many different versions of terms exist in currently operating online information processing systems. For example, words used by some modern systems to initiate activities after Login is complete include: START, RESTART, BEGIN, GO, N.A., and SEARCH,

to name only a few. Words used to terminate activities before disconnecting include: STOP, END, BYE, Z, OFF, QUIT, and DONE. Words used to obtain output from a search-identified set of records include: PRINT, DISPLAY, TYPE, VIEW, SHOW, and LIST. Army developmental personnel and users of the MIQSTURE language hawe systematic opportunities to bring their unique requirements and personal preferences to bear in selecting particular words for the MIQSTURE command vocabulary.

Seven of the major functions are described briefly (sketched) below. The other six are described in greater detail in Section 3.0.

2.5.2 STARTUP/SHUTDOWN Interaction Arrangements. These arrangements are oriented to the un-initiated user rather than the practiced user, because the system's HELP, LEARNING ACCELERATION, and ERROR FORGIVENESS AND CORRECTION capabilities do not become available to a neophyte user unless he is able to successfully effect system startup.

Startup Switch Settings. The MIQSTURE-supporting system is composed of a range of hardware at the interface, some with switching arrangements optionally dependent upon how it is connected to other equipment. All switch locations and their sequencing for startup and shutdown are placard-displayed at a single prominent location on each work station. Standardized labeling for all switches is color-keyed to the placard. Starting diagnostic procedures are placarded, and more detailed versions contained in one section of the user online manual are referenced on the placard.

Login/Logout. After switches-on is completed, the placard instructs the user to depress one key, and the system responds with the message ENTER USERID: _____. The USERID (when valid) grants the user access to the

next step in interaction, which can be a user-optional, user-assigned password known only to himself (but able to be displayed on command by the terminal operating under the supervisor's USERID and password). The password capability works as follows: If the user during a previous session has not supplied the system with a password to use for challenge, the valid USERID immediately provides the user with whatever pattern of system capabilities has been defined as associated with his USERID. On the other hand, if the user has in any previous session taken a "NEWPASSWORD_____ command and received acknowledgement PASSWORD ACCEPTED, the user is expected without prompting by the system to follow the USERID by a space and his password. All invalid USERID and password entry actions are recorded by the system along with date/time and work station number and are available as a supervisor printout. (The valid USERID also "sets" the File/Element/Command pattern permitted to that user.)

Opening Exercises. The system signals successful Login with the message: OPENING EXERCISES Y? Striking the Y key produces the opening exercise menu display, which is also produced at any other time during interaction with the command "OPENING EXERCISE or any of its synonyms such as "OPEX . The opening exercise menu lists titles for available opening exercise interaction sequences (which changes from time to time, depending on system experience). Each title is selected by striking the key corresponding to the letter preceding it, by cursor on the displayed letter plus SELECT button action, or by light-pen or graph-tablet select actions, depending upon the equipment configuration used. With this arrangement a single keystroke moves the user into opening exercises. Simply ignoring the question and proceeding as planned causes the system to "drop" the opening exercises issue.

Selection of any opening exercise title produces either another menu display or the initial display of information associated with the selected exercise. Continuity along the selected exercise path is maintained easily, flexibly, and effortlessly by a continuation query posed to the user: CONTINUE_____Y/N. An N keystroke reinvokes the opening exercises menu display (which the user may again utilize or simply ignore by taking any command action).

Representative examples of Opening Exercise Titles that appropriately appear on the menu display are:

- Neophyte User Instructions
- Late News and Day Calendar
- Mail Addressed to User
- Personal Calendar and Reminder Notes
- Review Requirements Docket and Statuses
  - Assigned for User's Action
  - Assigned for User's Information only
  - Open Information, Pending Action Assignment

Closing Exercises. Upon receipt of a STOP or QUIT command from the user, the system responds with the message: CLOSING EXERCISES Y/N? An N keystroke results in a message: STOPPING NOW, PLEASE SWITCH OFF EQUIPMENT, GOODBYE. A Y keystroke produces a menu display to be used in a fashion similar to the one for opening exercises. The closing exercises menu contains such titles as:

- Shutdown Checklist (system version)

- Late News (items entered in news file after sign-on time of user's current session)
- Late Mail (same)
- Late Requirement Assigned (same)
- Personal Shutdown Reminders (devised by user for himself) These are entered at any time during a session by the user through the command "SHUTDOWN NOTE:_____". The last-entered note is presented first, next earlier next, etc. After each note is presented, the system first delivers the query DELETE?Y/N and after a keystroke answer is received, the system asks NEXT?Y/N, for which an N reproduces the DLO EXER?Y/N and a Y cycles to the next note, or on the last note, back to the CL EX?Y/A system message.

Shutdown Switch Settings. A single state-light on the placard display described earlier remains lit as long as any switch to computer-associated equipment in the work station is set to the on position. The final item on the system version of the shutdown checklist is: State Light is Out, and this message is repeated on the placard.

2.5.3 System Navigation. This includes information and system procedures aimed at helping the user to understand the system, its purposes, and activities that are obtainable through interaction arrangements in this area. (The NAVIGATION area is, of course, referenced by the Neophyte User Instructions contained in the Opening Exercises display.) The navigation

area consists of three main types of contents, and a "NAV" command entered via keyboard or by function button action produces a menu display with three descriptive titles:

- Facilities of the System (Current)
- Task-type Descriptions (Current)
- Work-Planning/Formatting Aids

Menu displays for three areas are obtained via cursor or light pen selection, by keying in the commands "FACILIT "TASKS and "PLANAID or by pressing command function buttons so labeled. (Such entry options are not mutually exclusive. They are enumerated here for illustrative purposes and to highlight the flexibility of the MIQSTURE interface design in providing any needed combination.)

Facilities of the Current System. This title contains the following kinds of subtitles in a menu display:

- General Interaction Arrangements
- Error Diagnosis/Feedback Facilities
- Learning Acceleration Facilities
- File/Record Descriptions
- File Care Facilities
- File Subset Defining Facilities
- Data Staging Capabilities
- Data Staging Requirements, Shortcuts, and Defaults

- Analytic Processing Routines, Uses, Characteristics
- Report Assembly/Output Capabilities, Uses, Characteristics
- System Management Facilities
- System Configuration Adaptation/Control Facilities

Each title in turn elicits another menu representing the table of contents of the topical area of materials available through the interface.

Task-type Descriptions, Current. This title contains descriptions of typical sequences of steps involved in interactive processing of most representative types of intelligence analysis tasks. The descriptions are augmented and modified on the basis of maturing task experiences, new uses of the systems capabilities and expansion of the system's capacities. Cross-referencing these descriptions are user comments entered when a user has something valuable to contribute to the common understanding of task processing within the system.

Work Planning/Formatting Aids. These capabilities are aimed at helping the user to make notes for himself that will measurably aid him in both planning and executing tasks of great complexity with many sequential dependencies. Such a machine-based "note system" has several advantages over paper and pencil or typewriter produced note systems. First of all, it is provided with "machine-based-cut-and-paste" capabilities. Thus, via simple commands, parts of task descriptions are lifted and placed in the contexts of new task processing plans, as are various parts of earlier processing plans the user has developed and stored. Full text-editing capabilities are available for these purposes. The work planning aids can also help the user execute complex tasks with many sequential dependencies by assisting him to "step through" the task.

An example of stepping through a task was given in Section 1.0.

Functions that will receive more detailed treatment in Section 3.0 will not be covered here. They are:

- Display Convention for List-type Records — Section 3.2
- Scenarios for Conventional Querying, List-type Records — Section 3.3
- Scenarios for Alerting and Input-Driven Automatic Querying — Section 3.4
- Scenarios for Tabular/Hierarchical Database Query and Calculation— Section 3.5
- Scenarios for Task Representation/Control — Section 3.6
- Scenarios for Terrain Overlay Symbol Data Reference — Section 3.7
- Scenarios for Defining/Maintaining Data Storage Structures — Section 3.8

2.5.4 Data Staging. Data Staging is the process of transferring data in blocks among large-scale system components (subsystems and major routines) that are capable of being used in different combinations, and interchangeably, to accomplish various processes. Once a particular STAGING PATH is defined by the interaction elements (to be sketched briefly below), the stored instructions for invoking the entire path are elicited by a single function-button action, by menu selection, or by a simple keyed-in command name. (The PRINT command represents an example of a routine using such a stored set of staging paths.) A "STAGESPEC command causes the system to display a menu format upon which is shown the kinds of selections described below. Each select action causes a fill-in format display to appear that is appropriate to one named task:

(1) Name and Scope-Note the Staging Path. This comprises brief free-text sentences answering a set of listed topics on the display (commensurate with system configuration control practices) and provides spaces to assign a unique identification number and path title (if desired). The number is embedded automatically in a context containing the initiation date/time, USERID, and Combat Work Station No. Both this sequence and the optional title are placed in a display file under the USERID, the contents of which can be displayed by a "MYPATHS command. Associated with each entry in the display is a system-assigned unique Path Identification Number (PIN). Each fill-display for staging-path construction described below starts with the item: ENTER PIN: _____.

(2) Identify-Enumerate All Sources, Holders, Receivers.

(3) Source Output Spec (for each source).

Output Record Format

Minimum, Maximums, elements records (bytes)

(4) Source/Hold(s) Linking (for each source).

Source Ident/Holds Ident link(s) spec

(5) Hold File Spec (for each source).

Max. expected hold size (records)

Hold record partitioning spec

x Partition Identifier

x Partition Definition

(6) Partition/Receiver(s) Linking (for each partition).

Partition Ident/Receiver idents link spec

(7) Receiver Transform Spec (for each Partition/Receiver link).

Partition output element spec

Receiver input elements spec

Partition output/Receiver input elements linking spec

(8) Path Construction (using above-produced materials).

Assemble sub-path component identity sequences

Assemble main-path sub-path sequences, contingencies

This flexible, hierarchically organized method of Data Staging allows alternative paths for arriving at the same data flow patterns. Procedures for optimization can be developed for each type of flow path. Special requirements are incorporated; for example, for many paths elements need to be retained throughout the path to allow back-referencing to the original data base for end products. The hold-partitioning capabilities allows record identification elements to be included in all partitions when necessary.

2.5.5 Analytic Processing. The repertoire of analytic processing routines for organizing data in ways meaningful for tactical intelligence analysis can be expected to continue to grow. All resources for analysis require standardized approaches to the interaction arrangements provided to the user if they are to receive widespread usage. An ANALYCAP command causes generation of a menu display listing descriptive titles for each analytic capability. A SELECT action upon the menu produces an "instruction sheet" display for using the selected analysis routine. Included in the instruction sheet display is the CALL NAME employed in a SETUP CALL NAME command that produces a fill-in display for setting-up the analytic processing run. Although this fill-in display is quite self-explanatory, the "instruction sheet" display for the routine being

used also provides detailed back-up instructions for adjusting the analytic routines. The instruction sheet display also describes the routine briefly and references more detailed printed material. It lists the adjustment parameters and the run parameters of the routine, and cross-references the proper fill-in slots on the SETUP fill-in display.

Once a particular staging path and analytic routine setup procedure is satisfactorily defined to the system, the entire sequence can be set to be invoked automatically by a single button action, menu select action, or keyed-in command term. An "ANAREADY command causes a menu display of title of such pre-packaged staging and analysis sequences, and a select action on a title produces a description of the pre-packaged sequence and the control actions needed to invoke it.

2.5.6 Product Output. The variety of tactical analysis products available from the system will increase as a result of use of the system. Most products will be produced on a standardized, periodic basis (e.g., sector/period activity summaries), but others will be standardized aperiodic or non-standard aperiodic. Once set up, periodic and aperiodic standardized products can be produced with minimum interaction load on the user-analyst. The PRODUCT OUTPUT interaction elements consist, therefore, of arrangements both for setting up new products and for executing already set up products. The list of interactive activities involved under product set up are:

(1) Provide products name and scope note

(2) Provide product components manifest

(3) Marshall manifested components from various system sources

(4) Format/Edit each component

(5) Assemble components according to plan

(6) Make proof run, do proofing, correction actions

(7) Initiate production run, specifying quantities, schedule

(8) Develop distribution instructions

2.5.7 System Management Processing. The main function of system management processing is to produce reports and other forms of display that summarize important features of the system's processing activities for managerial purposes. Volumes of system processing activities accomplished for various time periods, distributions of demand types submitted to the system, and system utilization statistics are examples of management-relevant information. As with all types of commands, only authorized personnel have access to commands and displays associated with management processing.

A "MANAGE command from an authorized USERID results in a menu display containing titles of items selectable in the management processing repertoire. The major portion of this display consists of titles for standard management report products, selection of any of which produces a fill-in display that contains slots for parameters necessary to run the report. Other items on the MANAGE menu display elicit fill-in displays that allow specifications for new reports to be developed. The report contents specifiable are limited to the types already contained in the system management data journaling file, and the report formats are selectable only from a set of pre-prepared ones. Report types are prepared to support management decisions in the areas of availability analysis, traffic types analysis, and performance efficiency studies.

2.5.8 Interaction Adaptation Processes. The main function of this type of arrangement is to allow authorized personnel to alter and augment user interface arrangements to accommodate changes in the system capabilities repertoire. These interaction arrangements are not for producing new system programming codes or for making changes in existing codes. Rather, the arrangements are limited to changing and adding: (1) system messages; (2) display contents; (3) command input names and parameters; (4) USERIDs; and (5) their associated authorization patterns. These five kinds of items are decoupled from the system code with which they inter-communicate and are, therefore, produced independently of program code changes and additions. In order to be used with new system code, the interaction elements of the interface must be precisely mated with corresponding program code. This is done through communication tables, which establish the equivalences between user/system interaction elements and the internal input and output message formats of the software programs.

The commands associated with this area are authorized for the system manager's USERID and password and to his delegates. The five initiation commands associated with the areas enumerated above are: "NMESSAGE, "NDISPLAY, "NCOMMAND, "NUSERID, and "NUSCLASS. Each elicits an appropriate menu display with titles for the fill-in displays resulting from a SELECT action.

(1) "NMESSAGE command. Produces menu with the following titles:

- General Interaction Capabilities
- Startup/Shutdown
- Navigation
- File Care

- File Subset Defining
- Data Staging
- Analytic Processing
- Product Output
- System Management Processing
- Interaction Adaptation Processes
- CHANGE fill-in format

A SELECT action on any of the above titles (except the last) produces a display of system messages (including all error and interaction messages) for that area. Associated with each message in the display is its SYSTEM MESSAGE NUMBER. A SELECT action on the "CHANGE fill-in" menu alternative produces a fill-in display with the following kinds of slots:

x SYSTEM MESSAGE NUMBER _____.

x SHORT FORM _____.

x INTERMEDIATE FORM _____.

x LONG FORM _____.

Execution of this display after it is filled causes the system to write over old values in the appropriate communications table slots with the new specified values. The result is to (a) alter the message forms output by the system, and (b) alter the message forms in the tutorial materials supplied online by the system.

(2) "NDISPLAY command. The arrangements for this command are analogous to those just described above, except that the menu display lists titles of display types for each of the ten areas, together with their SYSTEM DISPLAY

NUMBERS. The fill-in formats provide for filling in the SYSTEM DISPLAY NUMBER in the upper left corner of the display, normally reserved for attention device announcements. This action causes the appropriate display to appear in an online-edit-modifiable version. After the changes are effected and checked, an "EX command causes the new display format to replace the older one.

(3) "NCOMMAND command. These arrangements are almost identical with those for changing system messages, with two differences: The result of such change action is to (a) alter the command forms recognized by the system, and (b) alter the command forms in the explanations of commands supplied on-line by the system. In addition, the NCOMMAND fill-in displays allow for entry of command synonyms and variations.

(4) "NUSERID command. Issuance of this command causes a menu display listing the names of all personnel assigned USERIDS in the system, sorted by the system into aphabetical order. The topmost item on this list is a constant titled NEWNAME. Selection of any name results in display of that user's authorization table entry, which contains three values: The user's full name; the current USERID; and the assigned user-class number(s). These values can be edited online, and the result is stored via execution of the EX" command.

(5) "NUSCLASS command. This command elicits a display of the USERCLASS table and allows online modification of that table via the online edit capabilities. The first part of the table consists of a tabular listing of commands not related to files or else automatically related to only one file.

To the right of each command name is space within which can be contained user-class numbers separated by commas. The numbers must be in ascending order from left to right for acceptance by the system. A number in such a slot means that class of users has access to the associated command.

The second part of the table consists of a tabular listing of commands related to more than one file (or potentially related to more than one file). Following each command name (below it) are listed file names; each of these file names is followed by record-type names, and each of these record-type names is followed by element (field) names for that record. To the right of each element (field) name is a space for the userclass numbers.

Together, the "NUSERID and "NUSCLASS commands provide the system administrator with complete flexibility in assigning any particular user to any number of userclasses, as well as complete flexibility in defining as many different patterns of system access (userclasses) as necessary.

2.6 Summary. At the end of the discussion of an augmented definition of mixed-initiative, four main functional goals of the MIQSTURE language are listed:

- Change/Adaptivity Goals
- Task State Sensitivity Goals
- User Learning Support Goals
- Interaction Format Optimization Goals

These form a convenient framework for summarizing the MIQSTURE features just presented or referenced.

Change Adaptivity Goals. Three general features of MIQSTURE contribute to its ability to adapt to change. First, system initiated aids to interaction for DEFINE activities include menu displays showing existing defined commands, parameters, and data structures, fill-in and prompting formats for aid in defining new capabilities or modifying existing ones, and work interruption set-aside features for saving definitional products at intermediate states of completion. These capabilities apply to defining files, records, tables, task components and sequences, and new commands and parameters. Second, system-initiated automatic procedures produce suitable fill-in and menu formats for purposes of maintenance of records and files and for running task sequences, once their reference structures have been defined through DEFINE activities. Finally, fill-in and menu interaction formats are based on a "keyboardable substrate" for the language, so that a straight keyboard version is available as an alternative for the more formatted, effort-reducing forms of interaction. The keyboard version is more effortful than are menu and fill-in formats, but also more immediately flexible and adaptable to unique new requirements.

Task State Sensitivity Goals. A number of features of MIQSTURE contribute to system-initiated interaction services that are keyed to particular states or conditions of task processes. The display format has dual (separate) counters for interaction numbering and query numbering, allowing complete accountability for complex sequences of activities while preserving flexibility. Interaction status and history are shown in separate display windows, and the history display is both searchable by transaction number and browsable through scrolling controls. A multiple dialog (multistream) arrangement at the work

station interface allows for interleaving several tasks simultaneously, producing both task structuring flexibility and better interface throughput capacity. Finally, a very significant capability is provided for defining schemas for task structures, for filling in the schemas with detailed knowledge of task interaction requirements, and for "running" automatically produced task element sequencing aids generated by the system from the schema materials defined to it. In addition, the EXPLAIN and HELP online capabilities are linked to the task state materials, producing better focused and more complete assistance to the user by the system.

User Learning Support Goals. The technique of presenting highly focused learning materials to the user during his normal interaction with the computer is the principle underlying the three main types of learning support arrangements for MIQSTURE. Abbreviated forms of both system messages and command names and parameters can be learned through an associative pairing of long and abbreviated forms. The user is spared laborious separate learning activities in acquiring skill with the abbreviated forms.

For the learning of conceptual materials, an incrementally structured principle is followed, in which earlier stages of conceptual structures and processes provide the user with readiness for more complex later stages. One such form of structuring distinguishes between task-general and task-specific elements and arrangements for interaction, the former serving as the foundation for the latter. Another form of structuring is in the "layered" approach to the query subset of the language, in which the concepts and skills for searching and handling simple list-type records underlie the procedures for handling tabular stored data and hierarchical structures. A third kind of

structuring involves sequence rules (syntax) for interactions; a "process visualization" type of sequence is aimed at the user's natural perceptions of the particular sequence in which the machine would appear to "need" various elements of the instructions presented to it. In this sequence, the interaction elements that are the most constant across a series of interactions are presented earlier in each command entry sequence and can be set to be in effect until changed, thereby eliminating repetitive actions.

For system navigation aids, such as online EXPLAIN and HELP displays and exercise support, the conception of MIQSTURE includes a careful integration of such materials, all supplied both online and in published form. The explanatory materials as well as emergency HELP materials are hierarchically organized from succinct to expanded presentations, and cross-reference each other's online as well as published versions. A simulation run mode is provided for exercising skills without the requirements of a full system function being supported.

Interaction Format Optimization Goals. The MIQSTURE design recognizes that certain formats for interaction are better suited to certain interactive requirements and user skill levels than are other formats. Each format has strengths and weaknesses, costs and benefits. The philosophy is to make a number of formats available, to make them substitutable in action, and to provide as much machine-initiated aid for each format as is feasible.

The major backup format is the keyboard substrate, in which the entire contents of interaction inputs can be expressed, including command names and parameters and names for data structure components such as files, records, and tables.

A system-initiated aid to this form is the command language discrimination approach for analyzing keyboard input: the names and abbreviations of commands, parameters, and data storage structures are immediately checked for correctness, and if a typographical error has been made, the system responds with alternative possible correct versions of the misspelled item. In addition to the definitional capabilities for revising parts of the command language, an instant-reaction RENAME function is able to rename commands and parameters to suit the individual user without disturbing the language arrangements for other users. In addition to tabular and prompting fill-in formats and menu selection type displays, there is also an intermediate "checklist" display that reminds the user of the interaction elements that need to be included in a particular entry, while at the same time permitting him use of the relatively unstructured keyboard entry approach. Finally, standardized interaction sequences are made efficiently by use of assigned function buttons and by programmable (by the user) function buttons.

3.0 DETAILED EXAMPLES OF SELECTED ELEMENTS OF MIQSTURE

3.1 General

In the preceding section, functional descriptions were provided for the range of MIQSTURE language elements deemed applicable to army tactical intelligence information processing operations. The present section will give details of the formats and functions of selected subsets of the MIQSTURE language. Six subsets were chosen as examples on the basis of:

- Coverage of most of the primary functions represented in the larger set.
- Emphasis on the main functions which would receive heavy use in the larger set.
- Inclusion of advanced state-of-the-art features.
- Direct comparability with query languages for relational data management.

In Table 3-1 titles of the six subsets are shown down the left margin of the table, while the titles for types of language elements contained in the subsets are arrayed across the top of the table. The number of elements of each type contained in each subset is depicted in the cell at their row/column intersection. Before providing definitions for the ten titles of the table, some observations regarding the table will be made:

- It is estimated that the less than one hundred (94) elements tallied in the table represent about 70% of all the elements of MIQSTURE that would be needed for a complete Army Tactical Intelligence information processing application. Many specialized users would not find occasion to use all the elements. Viewed as a procedural nomenclature for an Army equipment, the MIQSTURE language does not represent an exhorbitant learning load.

- Elements for USER/SYSTEM INTERACTIONS CONTROL and for QUERY FORMULATION SPECIFYING account for three-fourths of the elements. These comprise the main "flexibility" aspects of the language.

- The last three subsets contain only interactions control elements.

Reading downward through the table, the trend is for fewer language elements required to accomplish each additional function. This is the result of the incremental layering approach to the structure of the MIQSTURE language, to be discussed in more detail later. In this approach, later layers are added as increments to the more basic layers appearing earlier in the table. Since the later layers use some of the elements, logic, and general arrangements of the earlier layers, there is a positive transfer of knowledge and training for the user between earlier and later layers. The user's task of learning and retaining knowledge and skill concerning the language is greatly eased by this positive transfer effect.

SELECTED MIQSTURE SUBSETS	TYPES OF LANGUAGE ELEMENTS INCLUDED			
	USER/SYSTEM INTERACTIONS CONTROL	DATA STORAGE LOCATION/FORMAT SPECIFIERS	QUERY FORMULATION SPECIFIERS	QUERY OUTPUT FORMAT SPECIFIERS
LIST-TYPE RECORDS, NORMAL QUERIES	21	3	18	6
ALERTING & INPUT-DRIVEN AUTOMATIC QUERIES	2	0	5	3
TABULAR/HIERARCHICAL DATA-BASE QUERY & CALCULATION	0	5	12	5
TASK REPRESENTATION/ CONTROL	10	0	0	0
TERRAIN OVERLAY SYMBOL SUPPORTING-DATA REFERENCE	2	0	0	0
DATA STORAGE STRUCTURES DEFINE/MAINTAIN	2	0	0	0

Figure 3-1. Distribution of Language Elements in Example Subsets

Subset for Normal Querying of List-type Records - This is the basic subset around which the bulk of user-analyst interactions with the system will be oriented. List-type records, consisting of records for messages, citations, and other entities describable by a list of descriptive categories, are handled by this subset. Since this type of data comprises a major proportion of system traffic, most of the generalized user/system interaction control arrangements used in all subsets are introduced in this subset.

Subset for Alerting and Input-driven Automatic Queries - This subset allows for formulating automatic (system-initiated) queries on message records. To the elements of the basic normal-querying subset is added an increment of ten elements which, together with the basic elements, provides for two kinds of automatic queries. In an alerting query the system scans the message input stream, picking out and routing to the user those that match a standing query formulation. In input-driven querying, the message file is searched automatically for records similar or identical to new records arriving in the input stream.

Subset for Tabular/Hierarchical Database Query and Calculation - This subset provides the main capabilities for dealing with database information, which is most frequently stored as tables of data, sometimes arranged in a hierarchy of tables. To the basic subset is added an increment of twenty-two MIQSTURE language elements. Together with some of the basic elements they comprise the database portion of the language.

Subset for Task Representation/Control - This subset allows for storing detailed knowledge about tasks and their structures in the machine, thus providing the main basis for machine-initiated cooperation in interaction with the user-analyst. The subset includes elements for constructing schemas for tasks, for filling the schemas with detailed information about data relations between task steps and contingency shortcut rules between task steps, and elements for stepping-through the developed task structures in using them as aids to accomplishing the task.

Subset for Terrain Overlay Symbol Supporting-Data Reference - This subset includes language elements for defining cross-reference linkages between symbology on enemy situation displays and the stored data that supports the assignment and positioning of the symbols. It also includes elements for eliciting retrieval of the cross-referenced data at the convenience of the user-analyst.

Subset for Defining/Maintaining Data Storage Structures - This subset includes elements for defining to the system the structures of record types and table types and hierarchical relations between tables. It also includes elements for defining the contents of files (records and tables). Elements for maintaining (loading, updating, purging) records and files are also part of this subset.

Element Type for User/System Interaction Control - This type of element provides means for maintaining user/system communication about the status and progress of user/system interactions themselves. These elements are used in conjunction with the other three types to accomplish processing task goals.

Element Type for Specifying Data Storage Location/Format - Most of the data flowing through the system is stored at one or more points in the processing cycles. In characterizing or describing to the system particular data involved in any of the many processes, storage format and location of the data must be specified.

Element Type for Specifying Query Formulations - The actual combination of data values for which the machine is to search must be expressed to it. The expression may specify a narrow or broad search, complex or simple.

Element Type for Specifying Format for Query Output - After data are identified by the machine that match a query, instructions must be provided to the machine as to how to treat the data. Ordinarily, it will be output in one or more formats for the user-analyst's consumption.

In the remainder of this section, examples are given of the arrangements and usage of language elements for each of the six subsets listed in the table. Contained in each subset description is a table listing each of the language elements in the subset. Corresponding to each such table is a section of Appendix B, in which a definition of each element is given. While reading the examples, the reader may wish to consult Appendix B for clarification of the meaning and functions of some elements.

3.2 Display Convention for List-Type Records

A list-type record consists of a list of descriptive categories of data, which taken together cover the important aspects of the description of an entity or event. Message records for army tactical intelligence processing are a prime

example of list-type records, and contain such data category fields as Originator, Date-time group, Geographic coordinates, etc. Ordinarily, list-type records describe only one general type of information within a single file. Thus the file might contain message records for a certain geographic area and time frame. While MIQSTURE can be used in the one-record-type-to-one-file manner, it is not confined to that mode; different types of records may be contained in one file.

The conventions by which a data structure such as a record is displayed can affect the ease of comprehension for the user. Close observation of users of conventional bibliographic retrieval systems (which employ list-type records) find them habituated to certain kinds of displays. When these users attempt to learn to use an online data management report program system, the data formats sometimes confuse them. The columnar format of the numerical data sometimes constitutes an early stumbling block in their developing rapid facility with the tabular organization of data.

An army tactical intelligence processing system has a ubiquitous requirement for handling data in table form, and an equally strong need to handle messages that consist essentially of a list of categories of data. The MIQSTURE solution to this problem is to provide users with early training in the tabular or columnar type of presentation, by displaying the list type message as a horizontal sequence of categories, similar to the row of a table.

Thus with respect to the list type record, the MIQSTURE language convention is to display the record format with the information category names distributed horizontally rather than vertically, and with the data instances

distributed vertically under the category names. This provides two important benefits:

- It reduces negative interference in display-reading habits as they will apply to alternating between table and list record structures that the user will encounter at more advanced stages of skill and responsibility.

- It provides better spatial utilization and cursor manipulation for the display surfaces in "TABing the cursor to fill-in slots and menu choices.

A typical list-type message record display format would appear as shown in the first window (labeled) of Figure 3-1. In the display, the first record field contains the information category MESSAGE TYPE, the second record field QUERY REQUEST NUMBER, etc. Category Field Identifier (CFI) abbreviations are contained in parentheses above each category name. The fill-in slot for data in each field is indicated by dashes. Fill-in instructions are below the dashes; the MESSAGE TYPE category is filled with a mix of four alphabetic and numeric characters, while the VELOCITY (VEL) value requires three numeric characters.

The alphanumeric/textual display is automatically partitioned into appropriate sized display "windows" separated by dashes as shown in the figure. Up to four interaction windows (labeled first, second, etc.) may appear on a single display surface. In addition, there are two fixed-function windows, the INTERACTION HISTORY and INTERACTION STATUS windows. The STATUS window is <u>not</u>

INTERACTION HISTORY:

INTERACTION STATUS:

First Window

(METY) MESSAGE TYPE	(QRNU) QUERY REQUEST NUMBER	(TRA) TRACK #	(EDTI) EVENT DATE/TIME	(DIRE) DIRECT REPORT	(TYSU) TYPE/ SUBJECT	(TACA) TARGET CATEGORY	(LOC) LOCATION	(DIR) DIR	(VEL) VEL	(QTY) QTY	(TROR) TRACK ORIGIN
4A/N	4A/N	6A/N	11A/N	2A/N	6A/N	7A/N	8A/N	3A	3N	5N	18A/N

CONTD.

(TRDE) TRACK DESTINATION	(REM) REMARKS
18A/N	30A/N

END

Second Window

Figure 3-1. Message Record Display; Horizontal List Format

scrollable, but automatically moves down the screen if the HISTORY window is expanded. The STATUS window contains various kinds of interaction cueing messages to be described later in this report. As interaction messages in the interaction windows are executed and scrolled upward, they are passed "under" the STATUS window and appear in the HISTORY window, still scrolling upward. Thus the STATUS window provides a natural separator between current interaction materials and the interaction history.

3.3 Scenarios for Conventional Querying, List-type Records

The first example will depict the querying of a file of messages similar to the one formatted in Figure 3-1 earlier. In the INTERACTION STATUS window of the text display the program shows current information on the status of interaction between user and system. Assuming for the example that the user has logged in and was automatically connected to the OPENING EXERCISES file, and that he has dispensed with opening exercises (news, notes from others, new system capabilities, warnings, etc.), the user is prepared to do querying. The system generates an interaction status message:

INTERACTION STATUS:T1;C?--TRANSACTION 1; ENTER COMMAND?

First Window

USER:_

"FILE MESSAGE "BLANK SOTAS "EXECUTE (long form)

or

"FI MESSAGE "BL SOTAS "EX (short form)

Explanation:

System generates the short and long forms of interaction message, in "associative learning" pairing pattern of MIQSTURE. In the first interaction window below the STATUS window, it also generates the message source cue USER:, positioning the cursor (—) in the immediately following character spase. This indicates that the source of the next interaction message is expected to be the user (rather than the system, in which case the message source cue would be SYST:).

(User response is dropped down on page for convenience in exposition.) He issues three commands in one entry, using keyboard for each character, or else function buttons containing individual words. Each command name is preceded by command introduction character, chosen arbitrarily as ("). The first command connects him to the message file, the second asks for the fill-in blank form for the SOTAS (Stand Off Target Acquisition System) record type, the third directs execution of the entry. System responds in Figure 3-2a.

INTERACTION HISTORY:

INTERACTION STATUS: FILE IS MESSAGE, RECORD TYPE IS SOTAS; FORMAT IS FILL-IN BLANK, T2;C?/QF1? --TRANSACTION 2; ENTER COMMAND?/QUERY FORMULATION 1?

Explanation:

Interaction Status display increments transaction counter, adds query formulation option (since a file is now connected), increments query counter to (1), names file, record type, interaction format type.

First Window
USER:

(METY) MESSAGE TYPE	(QRNU) QUERY REQUEST NUMBER	(TRA) TRACK #	(EDTI) EVENT DATE/TIME	(DIRE) DIRECT REPORT	(TYSU) TYPE/ SUBJECT	(TACA) TARGET CATEGORY	(LOC) LOCATION	(DIR) DIR	(VEL) VEL	(QTY) QTY	(TROR) TRACK ORIGIN
4A/N	4A/N	6A/N	11A/N	2A/N	6A/N	7A/N	8A/N	3A	3N	5N	18A/N

CONTD.

(TRDE) TRACK DESTINATION	(REM) REMARKS
18A/N	30A/N

END

Explanation:

This fill-in display is actually generated immediately below the STATUS window, without intervention of the explanation window. Message source cue is USER:; cursor is positioned on first character space of first date category field (MESSAGE TYPE) so user can immediately begin entering values. A "TAB action positions cursor to first character of second DCF (QUERY REQUEST NUMBER) for user to fill, and so on.

Figure 3-2a. BLANK Fill-in Format for Message Record from SOTAS File.

INTERACTION HISTORY:

INTERACTION STATUS: FILE IS MESSAGE: RECORD TYPE IS SOTAS: FORMAT IS FILL-IN BLANK
T2;C?/QF1?--TRANSACTION 2; ENTER COMMAND?/QUERY FORMULATION 1?

Second Window

USER:

(METY)

MESSAGE TYPE
SO2
OR
SO4
OR
DA4C

Explanation:

User may "Tab over a field without entering any value. He may enter more than one message type value in the first field (or any appropriate field, each separated by an OR,) and entry will appear as in second window.

Each "TAB action is also interpreted as a boolean AND for the query. At end, user enters "EX action to cause query action.

Figure 3-2b. BLANK Fill-in Format for Message Record from SOTAS File

To continue with the explanation of fill-in sections for the fill-in blank for the SOTAS message in Figure 3-2.b on the preceding page: When the user reaches the fourth field of the fill-in format (EVENT DATE/TIME) he may enter an exact time or a time range. A between times range is entered as:

BET YYMMDDTTTTZ YYMMDDTTTTZ
or
BET DDTTTTZ DDTTTTZ
or
BET YTTTTZ YTTTTZ
or
BET TTTTZ TTTTZ

The single Y directly preceding the hours-minutes group signifies yesterday. The default date value is today. A time range earlier (less) than a certain time is entered as:

LES YYMMDDTTTTZ
or
LES MMDDTTTTZ
or
LES TTTTZ (etc.)

A time range later (more) than a certain time is entered as:

MOR MMDDTTTZ
or
MOR YTTTTZ (etc.)

If the alpha-character time-zone designator (Z) is omitted, the default zone is that of the terminal location.

In each case, the computer program automatically stacks the keyed-in entry or entries into the appropriate fill in box, since the BET, LES, and MOR prefixes signal the system that a time range value is to be entered. The current day and yesterday abbreviations are especially useful for constructing "standing" queries that may be elicited by a single button action.

INTERACTION HISTORY: QF1=(METY) SO2 OR SO4 OR DA4C AND (QRNU) XXXX AND (EDTI) BET YTTTZ TTTTZ

INTERACTION STATUS: FI=MESSAGE; RECTYPE=SOTAS; FRMT=BLANK
T2;QF1-14RECS--TRANSACTION 2; QUERY FORMULATION 1 FINDS 14 RECORDS
T3;C?/QF2?

First Window

T2;QF1-NREC XXXX(QRNU)
(no records found for (QRNU) value in query)

T2;QF1-NONE
(no records found by QF1)

Examples of alternative responses for line 2 of interaction status display

Explanation:

The fill-in format of QF1 is transformed to the keyboardable substrate form and transferred to the interaction history window. In the status display, the short form is used for the first and third lines, the long form (combining short and long) is used for the second message line to illustrate a new message. The third line increments transaction and query counters and invites continuation. Other possible system responses for line two are given in the first window, with their interpretations.

In real display, explanation window is absent. User is now free to formulate T3 in the first window.

Figure 3-3a. Query Result Messages

INTERACTION HISTORY: QF1=(METY) SO2 OR SO4 OR DA4C AND (QRNU) XXXX AND (EDIT) BET YTTTTZ TTTTZ

INTERACTION STATUS: F1=MESSAGE: RECTYPE=SOTAS; FRMT=BLANK

T1;QF1-14RECS--TRANSACTION 2; QUERY FORMULATION 1 FINDS 14 RECORDS

T3;C?/QF1?

First Window

USER:_

"VI 5SK 5 "EX (short form)

or

"VIEW 5 SKIP 5 "EXECUTE (long form)

Explanation:

User directs system to display the 6th through 10th records identified by QF1. If query counter were incremented to, say 10, user could insert a QF1--10 after the "VI command name, thereby printing from any one of the 10 queries. If no QF is specified the default is the last successful query.

"VI 5SK (TACA) (LOC) "EX (short form)

(Display 4th through 9th records target category and location values only)

"PR 4 SU (TROR) (REM) "EX (short form)

or

"PRINT 4 SUPPRESS (TROR)(TRDE)(REM) "EXECUTE (long form)

(On slaved printer output first four records, include all categories except those suppressed)

Explanation:

For both the "VIEW and "PRINT commands, selective output of category fields is available. Inclusion of categories (other than ALL) requires NO INCLUDE NAME in the command sentence, but once a category has been specified (e.g., (TACA)), the system includes only specified categories.

Figure 3-3b. "VIEW and "PRINT Command Examples

The ranging qualifiers BET, LES, and MOR can be used on any appropriate numeric values. The boolean OR may separate multiple values in any appropriate field. The user may specify search values for any or all of the data category fields, or he may omit all but one by "TABing through them. When he has entered all search values he wishes evaluated for the search, he finishes with an "EX action, and the system responds with the display shown in Figure 3-3 a:

When the user takes the "EX action, the system accomplishes several things. It evaluates the query statement just entered through the fill-in blank display format. It presents the results of the evaluation in the INTERACTION STATUS window. And, it moves the query formulation from the active interaction window to the INTERACTION HISTORY window, but expresses the query in a modified form. The query shown in the interaction history window is the keyboardable substrate form of the language, so named because the query can be keyboarded in this form directly into the system, and because this form of the language underlies the function button, menu display, and fill-in blank display entry forms of the MIQSTURE language. All these types of entry forms are converted by the system to the substrate form before they are submitted to the command and instruction parsing routines in the computer software program; that is the definition of the keyboardable substrate language.

An potential advantage to the user of displaying the keyboard substrate form in the HISTORY window is that the user can view innumerable "paired" examples of

queries and other interactions presented in the substrate form and one of the other forms, and thus learns the MIQSTURE language "in depth" in a reasonably effortless manner. This is very important because new, unusual or complicated action formulations may not always be available in fill-in or menu form, but can be immediately expressed in the substrate form through the keyboard. Also, the formulation of complex sequences of command instructions to be stored for elicitation by a single function button action will often need to be done at least partly through the keyboard form of the language.

The INTERACTION STATUS display separates the INTERACTION display (First Window, Second Window, etc.), from the INTERACTION HISTORY display as a deliberate strategy. This gives the user an instant recognition of what is presently active and what is the immediate past in interaction, and the separator itself carries the rest of the story.

In the second alternative feedback message from the QF1 action, (the no records for (TYSU) and (LOC) values message), the non-match search values are repeated as keyboarded on the assumption that the user will want to know which portions of his query formulation need to be revised (either because no such values reside in the file, or because he has made a previously unnoticed keyboarding error in entering the value). The program may issue more than one kind of message as feedback for one query action.

Query Results Output. The "VI 5 SK 5 "EX command of Figure 3-3.b produces a display on the textual display surface in an INTERACTION window. The records retrieved are ordered in the sequence in which they were inserted into the file,

and the user can scroll the window to observe each record in turn. (The figure presents and explains some more complicated VIEW and PRINT output options.) For many purposes, a simple "VI button or "PR button command will produce a standardized default output pattern suitable to the user. Default patterns are adjustable.

Query Modification. If the user-analyst discovers an error in a query formulation he has already had executed (evaluated), he can redo the entire query or he can use the MIQSTURE query modification capabilities. As an example, assume the user-analyst has entered a query and received back the following message:

```
-------------------------------------------------------------------------
            (T4;QF3-14;) NREC XXAX(TYSU) XXYYAYXX(LOCT)
-------------------------------------------------------------------------
```

He wishes to change both values in the query. He responds:

```
-------------------------------------------------------------------------
First Window
USER:QF3 MOD (TYSU) XXAX TO XXXX SEP (LOCT) XXYYAYXX TO XXXYAYXX "EX
-------------------------------------------------------------------------
```

This response illustrates several MIQSTURE elements. Re-keying of the query formulation number term (QF3) followed by MOD indicates that the remainder of the entry is a modification of the designated query. A second point is that a CFI abbreviation (Category Field Identifier abbreviation) such as (TYSU) qualifies or identifies each of the data values following it, but only until intervention of a boolean or ranging operator, or another CFI abbreviation. On the other hand, a CFI abbreviation following a data value modifies only that data value, but it has power to override a preceding

qualifier if one exists. This allows for very flexible query modification with an economy of action, but it also means that a separator sign must be imposed between each modification instruction (if there are more than one) in a query modification entry such as above. Note that the sign (SEP) is placed after the second data value of the first instruction, and before the CFI abbreviation for the second instruction.

Flexible Format Options. The point was made earlier that the keyboardable substrate form of the MIQSTURE language preserves an always-available flexibility of expression that is difficult to achieve with menu or fill-in displays or with function buttons. (The latter sacrifice some flexibility for great economy of expression, display aids to the user's memory, and elimination of typographical errors). The other advantage of the keyboardable substrate that places it in the interaction formats of MIQSTURE is that a series of sequentially dependent command actions can be expressed in one command entry. This feature is especially valuable for formulating user-defined function button elicited processing instructions. As a very simple example, both the query formulation and query result output instructions of the kind entered via separate entries in the preceding two illustrations can be input to the system in a single keyboard entry:

First Window

```
USER:(METY)SO2 OR SO4 OR DA4C AND XXXX(QRNU) AND (EDTI) BET YTTTTZ TTTTZ "VI
     5 SK 5 "EX
```

In this example, the query asks for any of three message types occuring with a certain query request number (XXXX) and with a certain event time range between yesterday and today, while the output portion asks to view the

sixth through tenth records retrieved. Several points can be made from this example:

- In the straight keyboard format, boolean AND must be inserted when needed because it is not done automatically as in the fill-in blank entry format. (The automatic linking of AND to the "TAB action is an arbitrary convenience for the fill-in format, under the assumption that most users and uses of fill-in formats will require boolean AND between values from different category fields.)

- In the straight keyboard format the user must also insert CFI abbreviations to quality the search term values. They are inserted automatically in the fill-in blank format.

- There is no practical limit to the length of a keyboard formulation since carriage return is automatic and the program only executes the entry when given the "EX command.

- As in all other input formats, the character and line erasure capabilities of the full EDIT facilities are in effect and can be used during the process of input formulation.

- If a query formulation value is entered without a CFI abbreviation qualifier, the program will search <u>all</u> category fields for the value. If it finds the matching value in more than one category field (e.g., in one category field in one record and in another in a different record) it sends the search result message:

```
MM--MULTIMEANING: XXXX: 1.(METY) 2.(QRNU) 3.(EDTI)
SPECIFY NUMBER OF CFI QUALIFIERS YOU DESIRE, OR TERM ALL
```

Appendix Definitions. The preceding scenarios have illustrated the types of interaction formats available in MIQSTURE, and the general nature of the mixed-initiative interaction arrangements for normal querying of list-type records. Some of the commands, command parameters, and system message types were covered, but certainly not all elements of the MIQSTURE subset of normal querying of list-type records were mentioned. Table 3-2 lists the long forms of the language elements for this subset, categorized as to the four types of elements depicted earlier in Figure 3-1. Each element is also tagged for its "part of speech" or grammatical function in the MIQSTURE language (command name, command parameter, boolean operator, etc.). The elements in Table 3-2 are the ones tallied in the first row of Table 3-1. For the reader who wishes an in-depth picture of this basic subset, Table 3-2 can be used as a summary reference to Appendix B, which contains a corresponding section in which is provided a more detailed definition of each element.

3.4 Scenarios for Alerting and Input-Driven Automatic Querying.

The elements of this subset are depicted in Table 3-3. As can be seen in the figure, this second layer of the query formulation capabilities of MIQSTURE requires only a relatively few additional elements for performing the automatic querying formulation functions. As mentioned earlier, the main purpose of the layering arrangement of the MIQSTURE elements is to ease the load of learning and retention for the users of the language. The task

Table 3-2. Elements Supporting Conventional Querying, List-Type Records

Controlling User/Program Interaction		Specifying Storage Format/Locations		Specifying Results-Output Actions	
TRANSACTION__:	(sm)	"FILE___	(c)	"VIEW	(c)
COMMAND?	(sm)	RECNAME	(cp),(sm)	"PRINT	(c)
QUERY FORMULATION__	(sm)	(XXXX)	(CFI Abbrev.)	QF__	(cp)
" (Command Intro. Char.)		Formulating Queries		____	(cp)
"COMMANDS	(c)	"QUERY	(c)	SKIP___	(cp)
"EXECUTE	(c)	"BLANK	(c)	SUPPRESS___	(cp)
"HOLD	(c)	"TAB	(c)	(Syntax)	
"ERASE	(c)	"CHECKLIST	(c)		
"CANCEL	(c)	"CONSTANT	(c)		
"ESCAPE___	(c)	NO RECORDS FOR	(sm)		
"TRANSACTION___	(c)	___RECORDS	(sm)		
"VERSION___	(c)	NONE	(sm)		
"AUTOSET	(c)	QUERY ___ MODIFY	(cp)	(sm)=System Message	
"WHAT?	(c)	QUERY ___ NESTING	(cp)	(c)=Command Name	
"HELP	(c)	"SAVQUERY	(c)	(cp)=Command Parameter	
"DISCUSS	(c)	AND	(bo)	CFI=Category Field Identifier	
"NAMESTREAM	(c)	OR	(bo)		
"SWITCHSTREAM	(c)	BUTNOT	(bo)	(bo)=Boolean Operator	
"CLEARSTREAM	(c)	BETWEEN	(ro)	(ro)=Ranging Operator	
"SIGNOFF	(c)	LESSTHAN	(ro)		
		MORETHAN	(ro)		

Table 3-3. Elements Supporting Alerting and Input-Driven Autosearching

Controlling User/Program Interaction		Formulating Queries		Specifying Results - Output Actions	
"MATCHPICK	(c)	# :	(cho)	ROUTECODE____ ____	(cp)
"DUPSEARCH	(c)	OLE, OMO, OBE	(oo)	ACTIONCODE ____ ____	(cp)
Specifying Storage Formats/Locations		_____		DELTIME____	(cp)
No added elements		(cho) = Character Operator			
		(oo) = Offset Operator			

specialization profiles of all user-analysts will include requirements to work with files containing simple list type records. Some analysts will have additional duties involving the formulation of automatic pickoff and input driven automatic queries. Others will be additionally tasked to query data stored in tables in data bases, and to formulate calculations on the data. In each case of further specialization, the user-analyst will find the new layer of language he uses to be a simple extension of the list type record handling capabilities of MIQSTURE. There will be no requirement to learn a total new system of language to handle the added specialized tasks; the fundamentals he learned in dealing with list type records will carry over directly or else by close analogy to the handling of automatic queries and tabular data.

Automatic Querying Functions. The autoalerting query is used to continuously scan the input streams of specified files, identifying each newly input record that matches the query. Various actions can be specified for disposition of the identified records. MIQSTURE's "MATCHPICK command allows formulation of such queries.

The query for the input-drive autosearch is also used to continuously scan the input streams of specified files, identifying each newly input record, that matches certain features of the query. Here the similarity to the autoalerting query ends. In essense, the automatically generated input-driven query is looking for records in the file that duplicate or are very similar to the new record in specified ways. To generate an input-driven query, specified category field values of the incoming record (usually EVENT DATE/TIME and LOCATION) are extracted and used as anchor values.

To these anchor values are applied Offset values, which are obtained from the query formulated by the user-analyst. The result is to produce ranging query terms, the exact values of which are dependent on the values in the new record identified in the input stream. The ranging query is then automatically (by system initiative) applied against the file for which the new record is a candidate member. MIQSTURE's "DUPSEARCH command allows formulating of queries for input-driven autosearching.

Scenarios. Assume the user wishes to formulate a query to be matched against a file input stream, and when matches are found, for certain routing actions to be taken. Of course, his USERID is one that grants him access to the commands that allow such actions to be taken, and he has already reviewed a file of such input-matching query formulations, to make sure he will not be duplicating ones that are already in effect (we will not describe this review process here; suffice to say that the queries themselves are contained in a file that can be both queried and browsed or scrolled). The action begins:

INTERACTION STATUS:
T21;C?/QF13?

First Window

USER: "MA "BL "EX

Explanation:

For transaction 21, user chooses set of commands rather than a query: "MATCHPICK "BLANK "EXECUTE. System will generate a fill-in blank format for formulating a query for autoalerting function. Fill-in format is shown in Figure 3-4. (See p. A-7 for "BLANK and "CHECKLIST definitions.)

INTERACTION HISTORY: "MA "BL "EX

INTERACTION STATUS: MATCHPICK FORMULATION
T22;C?/QF13?

First Window

USER:

"FILE NAME	RECORD NAME(S)	ROUTECODE(S)-----DELTIME(S)	ACTIONCODE(S)----DELTIME(S)
---------	----------	--------- -------	---------- -------
	----------	--------- -------	---------- -------

(contd)

QF__ OR QF:

Explanation:

Each "MA formulation requires: one file name; one or more record names; one or more routecodes, actioncodes, or both; associated with each such code a delay time (DELTIME); and a Query Formulation reference number or a query formulation.

The user-analyst "TABs the cursor through the fill-in blocks, th[illegible]B producing an automatic AND for this display also. If he wishes to enter more than on[illegible]or record types contained in the file, he follows entry of the first name with an OR, whic[illegible] the cursor to the blank below as already described for an earlier fill-in format. The pro[illegible]ll not process the entry unless routecodes and actioncodes are accompanied each by their delay time value.

Figure 3-4. Fill-in Blank Format for "MATCHPICK Command

The explanation of the fill-in format for the "MATCHPICK command shown in Figure 3-4 is contained in the figure. The experienced user has alternatives; the user could have employed the keyboard substrate form of entry by not ordering the fill-in blank ("MA "EX). In this case, the interaction would appear as follows:

INTERACTION STATUS: MATCHPICK FORMULATION: ENTER MATCHPICK
T22;C?/QF13?

First Window

USER: "FI MESSAGE AND SOTAS AND RO 124,89 DE DDO1MM AND AC 16 DE DDHH05
AND QF10 "EX

Explanation: File name MESSAGE record name SOTAS Routecodes 124,89
Delay time one hour; Actioncode 16, Delay time five minutes,
Query Formulation Number 10 Execute.

It is clear from this example that the fill-in format provides a very great aid to the user for entries that require rather exact formatting of input. A strength of MIQSTURE is that it allows appropriate interaction formats geared to task and user requirements.

The "DUPSEARCH command elicits the same kind of fill-in format as does the "MATCHPICK command described above. The only difference is that for some values of the query formulation (usually the only values filled in) the Offset Operators (see page B-14, Appendix B) are employed. Usually these will be for the EVENT DATE/TIME and/or LOCATION values. The method of employing offset operators is described in the definitions Appendix.

3.5 Scenarios for Tabular/Hierarchical Database Query and Calculation

This comprises the third layer of the querying capability of the MIQSTURE language. A considerable amount of information must be stored in tabular structures for Army Tactical Intelligence processing: Message contents may be semi-automatically transformed to row entries in tables of data bases; Enemy Order of Battle files are mostly tabular in nature.

The twenty-two added MIQSTURE elements required to query data stored in table formats, to "chain" queries through a hierarchically organized series of tables, to perform basic arithmetical operations on the data, and to display and/or print the results are listed in Table 3-4 and defined in Appendix B.

Table 3-4. Elements Supporting Table/Hierarchical Query and Calculation

Controlling User/ Program Interaction		Querying/Calculations		MUL	(ao)
		FROM QF__	(qto)	MAR	(ao)
No Added Elements		COUNT	(ao)	Specifying Results - Output Actions	
Specifying Storage Format/Locations		MIN	(ao)		
		MAX	(ao)	"RETAIN (auto)	(c)
INTABLE___	(cp)	SUM	(ao)	COL	(cp)
UNDTABLE___	(cp)	AVG	(ao)	ROW	(cp)
INSUBTBL___	(cp)	GRP	(ao)	"TEMPTABLE__	(c)
UNDSUBTBL___	(cp)	ADD	(ao)	"MELD ___,___	(c)
INSUBSUBT___	(cp)	DIV	(ao)	(qto) = Query Transfer Operator	
		SUB	(ao)	(ao) = Arithmetical Operator	

Table Oriented Transaction Statement. Earlier, the concept of a transaction entry (or statement), and the concept of a query formulation were described for list-type records. For treating data organized in tabular records (tables) these two concepts need to be reviewed and elaborated slightly.

A table-oriented transaction statement contains four main sequential steps, the first and last mandatory, the second and third optional. These are:

(1) Specifying table(s) in which data are to be found.

(2) Formulation of query statement.

(3) Formulation of calculation processing instructions.

(4) Specifying of output actions.

All four of the parts may be contained in one transaction statement entry followed by an "EX. Let us consider each part briefly:

Step 1: Specifying Tables - The five command parameters defined in the preceding section under Specifying Storage Format/Locations, plus the TEMPTABLE__ parameter are the means for specifying tables. These parameters operate as MANUAL CONSTANTS, i.e., once invoked they remain in effect until specific action (a different table name, record name, file name, or file connection cancelling command) is taken. Unless a table name is already in effect or is invoked by a proper action, attempted execution of the rest of the transaction statement will result in an error message to the user.

Step 2: Formulating Query - Each row in a table record may be thought of as essentially a list type record. The function of the query (when one is used in a table-oriented transaction statement) is to identify table row(s)

that match the query. These are identified in the same way that list records are identified by queries. The purpose of identifying rows of the table is to provide a restricted set of data upon which to operate the calculation processing steps and/or output actions which follow. In essence, the preamble to the processing instructions (step 3) and output specifications (step 4) each read: "Within the rows identified by thc query step, or if no query step, then within all rows of the table."

Step 3: Formulating Calculation Processing Instructions - The MIQSTURE elements for formulating calculation processing instructions are those identified as Arithmetical Operators (ao) in Table 3-4 presented earlier
If no calculation processing is desired, these instructions can be omitted from the table-oriented transaction statement.

Step 4: Specifying Results Output Actions - Output action specification is mandatory: If a table name is in effect (has been invoked), and if no "VIEW, "PRINT, TEMPTABLE, or "MELD command (or in their absence "RETAIN command) action is taken prior to "EX, an error message and a prompt, indicating the output action choices available, will be delivered by the system. Processing of the transaction statement will not be started until the User chooses one or more of them. When data values are output (through "VIEW, "PRINT, "RETAIN, "TEMPTABLE, "MELD) the Column Identifier Abbreviations associated with each are automatically output with them. Values derived from the various arithmetical operations are also given appropriate labels automatically.

For purposes of illustrating the use of some of the MIQSTURE language table-oriented elements, two tables (greatly simplified) of the kind used in Army Tactical Intelligence processing are presented below:

TABLE EQUIPT

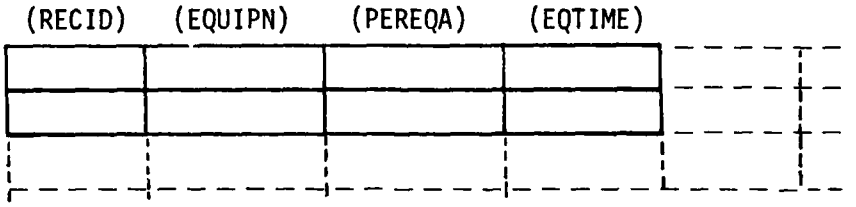
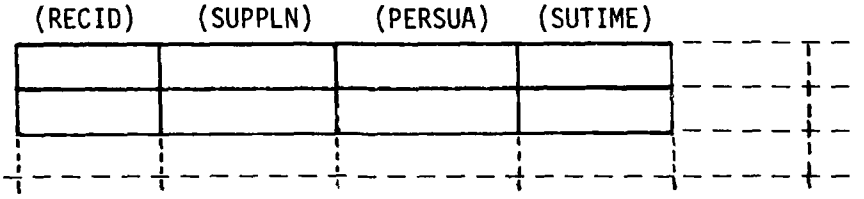


TABLE CRITSUP



In both the EQUIPMENT and CRITICAL SUPPLIES tables, the dashed lines are meant to indicate that the tables would, in reality, have many more rows and several more columns than depicted here. The examples presented here will be confined to Column Identifier Abbreviations shown. Also, the examples will be for single named tables, and will not use the UNDERTABLE and UNDERSUBTABLE parameters that allow processing specifications to be directed at several subtables or sub-subtables with a single statement. However, the only difference for multi-subtable transaction statements is in Step (1), where the subtables are specified.

The Column Identifiers associated with the parenthesized abbreviations are as follows for the EQUIPT table reading left to right: Record Ident; Equipment Name; Percent Equipment Available; Equipment Record Entry Time.

For the CRITSUP table the abbreviations are: Record Ident., Supply Name; Percent Supply Available; and Supply Record Entry Time. The percentage values are calculated from an authorized full strength number. The entry time is the last update time of the record. The record ID consists of a code name for a friendly military unit.

Table Manipulation Primitives. Before considering the display formats for the MIQSTURE table manipulation subset, the functional primitives it makes available for querying and processing must be defined. The definitions will be accomplished by using examples. The examples which follow consist of ten problem statements, each followed by a solution using the MIQSTURE table-oriented elements. In the solution, the elements used in the four steps of the transaction statement are depicted, followed by an explanation of each step.

The reader should bear in mind that the four steps are separated out on different lines in the examples in order to clarify the exposition; in operation the steps could be entered one directly following another for as many lines as required. To make this point clear, the transaction for the first problem only is presented in both the open and formatted step forms. The transaction and query formulation counters will be incremented in the example as they would be in an operational system.

Problem One: Output all the values in specified Row(s).

SYST:T4;C/QF1? (Transaction 4, enter Command or Query Formulation 1? It is also assumed that the name of a file is in effect that contains both the EQUIPT and CRITSUP tables.

USER: INTABLE EQUIPT (RECID)XXXX OR YYYY OR ZZZZ "VI "PR ROW "EX

1. INTABLE EQUIPT
2. (RECID) XXXX OR YYYY OR ZZZZ
3.
4. "VI "PR ROW "EX

Explanation:

1. "With reference to the table named EQUIPT:
2. find row(s) with (Record Ident.) values xxxx, yyyy, or zzzz
3. skip step
4. display, print, (and retain) the full row(s) from above results Execute."

Problem Two: Output all the values in specified column(s).

SYST:T5;C/QF2? (Transaction 5, enter Command or Query Formulation 2?)

USER:

1.
2.
3.
4. "VI "PR COL (RECID)(SUPPLN)(PERSUA) "EX

Explanation:

1. "Still referencing CRITSUP table."
2. skip step
3. skip step
4. display, print, (and retain) columns (Record Ident.), (Supply Name), and (Percent Supply Avail.) Execute."

Problem Three: Find row(s) meeting boolean and ranging criteria and retain.

SYST:T6;C/QF3?

USER:

1.
2. LES XX(PERSUA) AND MOR TTTTZ(SUTIME)
3.
4. "RE "EX

Explanation:

1. "Still referencing CRITSUP table
2. find row(s) with less than XX (percent supply avail.) and more than (i.e., later than) TTTTZ (Supply Record Entry Time)
3. skip step
4. retain the above results Execute."

Problem Four: Create and name a temporary table.

SYST:T7;C/QF4?

USER:

1. INTABLE EQUIPT
2.
3.
4. "VI "PR COL (RECID)(EQUIPN)(EQTIME) "TEMPTABLE QUIPDATE "EX

Explanation:

1. "With reference to the table named EQUIPT
2. skip step
3. skip step
4. Display, print, (and retain) column(s) (RECID), (EQUIPN), and (EQTIME), and create from them a temporary table to be named QUIPDATE Execute."

Problem Five: Use the arithmetical operator GROUP, and retain the results.

SYST:T8;C/QF5?

USER:

1. INTABLE CRITSUP
2.
3. GRP (SUTIME)
4. "RE "EX

Explanation:

1. "With reference to the table named CRITSUP
2. skip step
3. Within constraints imposed by Step (2) if any, group row(s) by column (SUTIME) values
4. retain the above results Execute."

Problem Six: Carry results of an earlier query to the calculation processing step of a later query formulation, and use arithmetical operator AVERAGE on the carried values; display, print, retain the results.

SYST:T9;C/QF6?

USER:

1.
2.
3. AVG QF5 GRP (SUTIME)
4. "VI "PR "EX

Explanation:

1. "Still referencing CRITSUP table
2. skip step
3. Within constrains imposed by Step (2), if any, AVERAGE the values from Query Formulation 5 results, groups (SUTIME)
4. Display, print (and retain) Execute."

Problem Seven: Perform a boolean operation (set intersection) on the outputs from two query formulations, and display, print, and retain results.

SYST:T10;C/QF7?

USER:

1.
2. (SUPPLN) XXXX OR YYYY AND LES XX(PERSUA)
3.
4. "RE (RECID) "EX

Explanation:

1. Still referencing the CRITSUP table
2. find row(s) with (Supply Name) values XXXX or YYYY and with less than XX (Percent Supply Avail.)
3. skip step
4. retain (Record Ident) values from the above results Execute."

SYST:T11;C/QF8?

USER:

1. INTABLE EQUIPT
2. (EQUIPN) XXXX OR YYYY OR ZZZZ
3.
4. "RE (RECID) "EX

Explanation:

1. With reference to the table named EQUIPT
2. find row(s) with (Equipment Name) values XXXX or YYYY or ZZZZ
3. skip step
4. retain (Record Ident) values from the above results Execute."

SYST.T12;C/QF9?

USER:

1.
2. QF7 AND QF8
3.
4. "VI "PR "EX

Explanation:

1. Still referencing the EQUIPT table
2. find the intersection (AND) of the sets of values from the results of Query Formulation 7 and Query Formulation 8
3. skip step
4. display, print, (and retain) the above results Execute."

Problem Eight: Use the output of an earlier query formulation as part of the query formulation in a later transaction statement, and modify it by a ranging operator (more than)

SYST:T13;C/QF10?

USER:

1.
2. (RECID) XXXX OR YYYY AND MIN (PEREQA)
3.
4. "RE (PEREQA) "EX

Explanation:

1. Still referencing the EQUIPT table
2. Find row(s) with (Record Ident) values XXXX or YYYY and with the smallest (MIN) (Percent Equipment Avail.) value.
3. skip step
4. retain the (percent Equipment Avail.) value Execute."

SYST:T14;C/QF11?

USER:

1.
2. MOR FROM QF10 (PEREQA)
3.
4. "VI (RECID) "EX

Explanation:

1. Still referencing the EQUIPT table
2. find row(s) with values more than that from query formulation 10 output (Percent Equipment Avail.) value
3. skip step
4. display (Record Ident) values from the above result Execute."

Problem Nine: Do calculations involving values from different tables, using the data coordination marker operator; display, print, retain results.

SYST:T15;C/QF12?

USER:

1.
2.
3.
4. "RE (RECID) (PEREQA) "EX

Explanation:

1. "Still referencing the EQUIPT table
2. skip step
3. skip step
4. retain (record ident.) and (Percent Equipt. Avail.) values Execute."

SYST:T16;C/QF13?

USER:

1. INTABLE CRITSUP
2.
3. MAR (RECID) MUL (PERSUA) QF12 (PEREQA)
4. "VI "PR "EX

Explanation:

1. "With reference to the table named CRITSUP
2. skip step
3. marker operator is (Record Ident.) multiply (Percent Supply Avail.) values with corresponding values from query formulation 12 output (Percent Equipt. Avail.) column.
4. display, print (and retain) Execute."

Problem Ten: Combine the results from different query formulations into a temporary table.

SYST:T17;C/QF14?

USER:

1.
2.
3.
4. "MELD QF2 QF13 "TEMPTABLE YZYZ "EX

Explanation:

1. "Still referencing the CRITSUP table
2. skip step
3. skip step
4. combine the outputs of query formulation 2 and query formulation 13 into a temporary table to be named YZYZ (and retain) Execute."

Format Aids for Table-Record Transactions. The example transactions just presented demonstrate some of the capabilities of the MIQSTURE table-transaction subset and the logic of its use. The range of interaction format options for the table-transaction subset includes those already described for the list-record oriented MIQSTURE capabilities, plus some additions. The "BLANK and "CHECKLIST options (see page A-7) for table transactions produce displays of the column identifiers of the table named, similar to the display of category field identifiers for list-type records. In addition, a prompting display is available for making the four steps in each table-transaction explicit to the user. This prompting display is provided automatically below both the "BLANK and "CHECKLIST format options: When either "BLANK or "CHECKLIST is invoked along with an INTABLE___ parameter entry, the prompting display is also automatically presented. The user employs the "TAB button command and carriage return "CR to step himself through the entry process. Figure 3-5a provides an illustration of a table-transaction aids display using the fill-in "BLANK option.

The Figure depicts several points of information. First, the "BLANK format for table-oriented transactions uses both menu-select and key-in forms of entry; steps one and two can use either form of entry, steps three and four (Figure 3-5b) are confined to the key-in format because of the nature of the entries. Second, the entire text of the entry is reproduced on the Interaction History window in the continuous key-in form; this "paired-associates" type of display provides an effortless way for the user to learn the flexible and economical key-in format which most highly experienced users seem to prefer. Third, the "SELECT action results in automatic OR between

INTERACTION HISTORY:

T11;QF4/C? User: Transaction entry reproduced here in continuous key-in format.

INTERACTION STATUS: FILE IS FRENSIT; TABLE IS EQUIPT; INTERACTION FORMAT IS "BLANK: STREAM IS A__.
T12;QF4/C?

STEP ONE: TABLE EQUIPT
TABLE CRITSUP
TABLE POLYGON
SUBTABLE NAI

STEP TWO: QUERY FORMULATION:

TABLE EQUIPT COLUMN IDENT. ABBREVIATIONS

(RECID)	(EQUIPN)	(PEREQA)	(EQTIME)
XAXAXY	ZDZEZDF		
YAYAXA	DEFZFZD		
BACAXA	EEFFZZD		
	DDZEZZF		

Explanation:

Table EQUIPT is referenced from earlier transaction, format "BLANK is in effect; therefore the four-step fill-in format for table transaction is generated. The cursor is positioned under first letter of first alternative not in effect (TABLE CRITSUP). Alternative TABLE POLYGON can be scrolled up to replace TABLE CRITSUP, and then a "SELECT action automatically replaces TABLE EQUIPT with TABLE POLYGON, writes INTABLE POLYGON into the interaction history window, and (since there is only one column in step one) jumps the cursor to the initial position for step two.

Alternatively, the entry INTABLE POLYGON can be keyed-in; the first keystroke jumps the cursor to open space to right of step one column. In this case, either "TAB or "CR may be used to move to step two. Since more than one (RECID) and (EQUIPN) value can be specified by "SELECT button (automatic OR inserted between selections), cursor is moved to next column by "TAB action. The (percent equipt. avail.) and (equip record entry time) values are inserted via keyboard. "TABing the last column moves cursor third step. To skip an entire step, use "CR. The interaction entry is reproduced in interaction history display. As described earlier, interaction history can be scrolled and searched.

Figure 3-5a. Fill-In/Menu Interaction Format for Table Transactions

INTERACTION HISTORY:

T12;QF4/C? User: Transaction entry reproduced here in continuous key-in format.

INTERACTION STATUS: FILE IS FRENSIT; TABLE IS EQUIPT; INTERACTION FORMAT IS "BLANK: STREAM IS A__.
T12;QF4/C?

STEP THREE: ARITHMETIC PROCESSING: (COU, MIN, MAX, SUM, AVG, GRO, ADD, DIV, SUB, MUL, MAR.)

STEP FOUR: RESULTS OUTPUT: ("VI, "PR,AUTO-"RE, COL, ROW, "TEMPLATE___, "MELD)

Parameter names and command names in parenthesis in steps three and four are reminders, these step entries are keyed in.

Figure 3-5b. Fill-In/Menu Interaction Format for Table Transactions

data values entered under (qualified as) a single column heading; the "TAB command results in automatic AND between column headings; the "CR (carriage return) results in the cursor skipping to the initial entry position for the next step. The net result of these arrangements is to provide flexibility in the interaction choices; the user can take advantage of memory aids and keystroking shortcuts for those portions of the task for which they are appropriate, and can shift to more flexible key-in tactics whenever they become necessary.

3.6 Scenarios for Task Representation/Control

General. As a participant in a dialog, what most clearly distinguishes the interactive computer program from the human user/communicator is the system's lack of knowledge about either the ends or the related means of the dialog. While the human user has in mind purposes and possible outcomes and past history of an ongoing task process, the program is devoid of such knowledge and must be continuously "reminded" afresh of each step in the task process. Ordinarily the computer program is not expected to share any of the burden of understanding the implicit continuity among the steps of the task. Nevertheless, this easily explained shortcoming of the machine takes its toll in complicating the user/system dialog. The neophyte user, deeply accustomed to dialogs with humans, is chronically thrown off stride and disappointed by the machine's failure of understanding, even when the reasons for the problem are intellectually quite clear to him. As an ameliorative measure, means can be developed to allow the machine to accept and store task-structure knowledge and to use it in the course of interactive dialogs.

Task knowledge is knowledge about a particular task. Task knowledge has several characteristics important from the point of view of system design:

(1) It must be provided by persons intimately familiar with the details of the complexities and variations in ends and means, contingencies, qualifications, etc., that apply to each identifiable task.

(2) To be imparted to the machine, task knowledge must be constrained to reliable generalizations about relations between given sets of conditions and the task steps and substeps they imply or involve. Variations in relations between conditions and task steps must be expressed in unequivocal terms or else cannot be part of the machine's "task-knowledge" -- though such "soft" knowledge may be part of the user's understanding.

(3) The quality, level of detail, and usefulness of task knowledge for the system is dependent upon the effectiveness of the means provided by which the task-knowledgeable person communicates his task knowledge to the system.

The design goal, therefore, is to provide for: 1. interaction tools which allow easy and effective incorporation of task knowledge by the system, 2. ready utilization of the knowledge during task interaction, and 3. easy updating of tasks. The above three points must temper the thinking about the use of task knowledge in forwarding an interactive dialog. In order for the system to make use of task knowledge it requires two main language components:

(a) those for specifying task knowledge (developing task representation structures or frameworks); and (b) those for using task representations as aids during the actual course of a task. They are listed in Table 3-5, and defined in Appendix B, (p B-22).

Table 3-5. Elements Supporting Task Representation/Control

"TSCHEMA "TLOAD "TSAVE "TCALL__ "TSUBMIT	Commands for developing task representations	"TNAMES "TRUN__ "TCONTEXT__ "TDETAIL__ "TMOVE__	Commands for displaying and running task representations

Describing Task Frameworks to the System. A completed task framework is comprised of a task schema plus its fillings, which consist of data input to the system via a "TLOAD fill-in format. A task schema is a three-level hierarchical structure of labeled boxes, the uppermost called task BLOCKS, the middle ones task STEPS, and the bottom ones task SUBSTEPS. Although a BLOCK can subsume any number of STEPS, and a STEP any number of SUBSTEPS, ordinarily two to five subordinate boxes will be found convenient under either a BLOCK or STEP. An example of one display subset of a task schema is shown in Figure 3-6. The figure does not show the entire hierarchy for the task "MONITOR SPOT REPORT"; it shows a subset. ALL BLOCKS are shown, all STEPS under the BLOCK "SITMAP REFER", and all SUBSTEPS under the STEP "CHECK/AUTO HI TEMPLATE". This type of subset display shows the overall task context of the substeps in a particular step. The total task

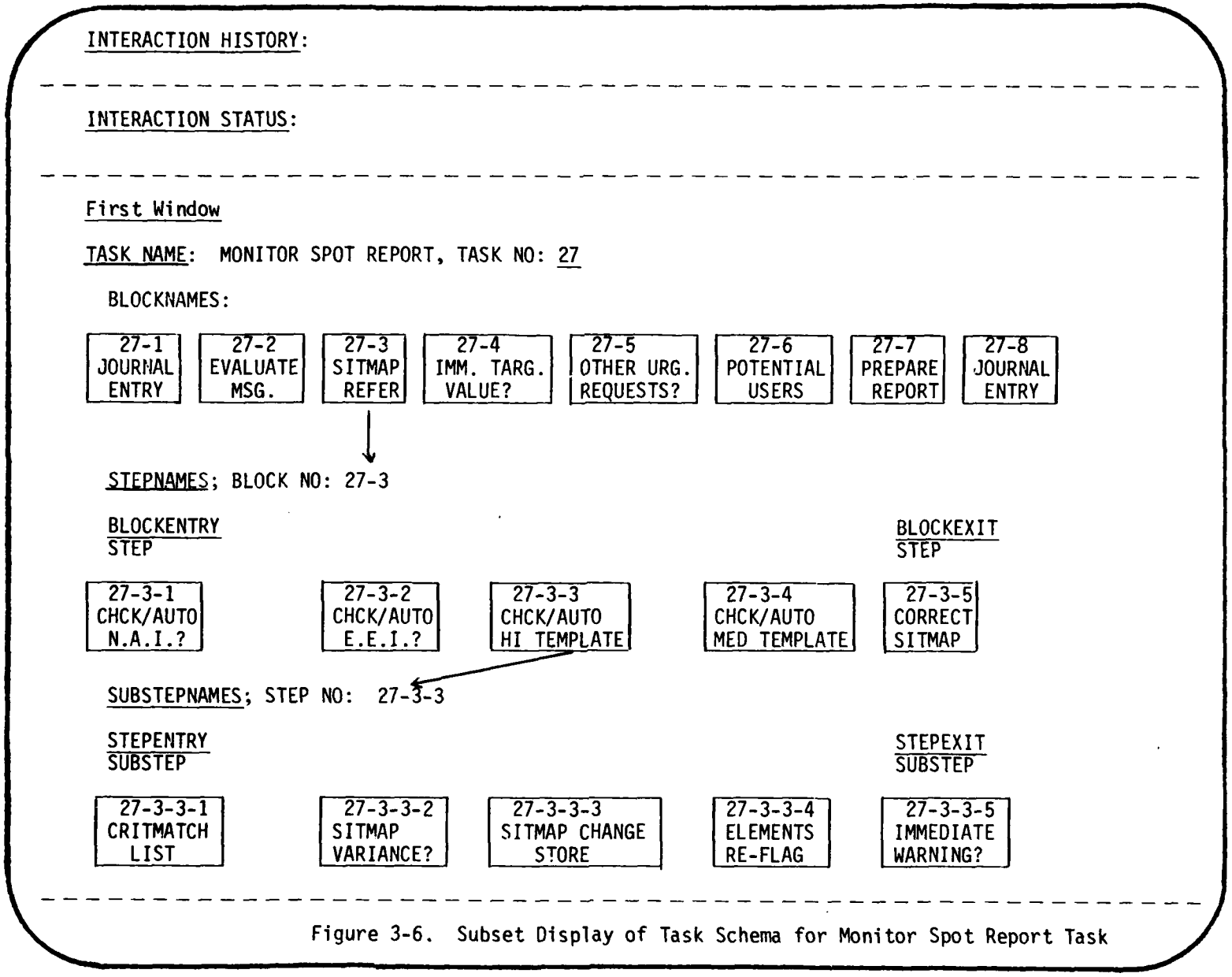


Figure 3-6. Subset Display of Task Schema for Monitor Spot Report Task

hierarchy for this task might contain between 75 and 200 boxes. The degree of detail of task-related information depends upon the particular task, greater detail requiring more boxes in the task schema.

Building the Schema. The schema for a task must be developed before it can be loaded with detailed information by use of the "TLOAD fill-in format. The task schema is developed through fill-in formats invoked by the "TSCHEMA command. The series of display formats for the "TSCHEMA command are depicted in Figure 3-7, which continues for four pages. In developing a task schema, the responsible parties would confer, develop the concept of the task, make detailed notes, organize the material into the proper sequential outline, and then compose the task schema on the system.

The MIQSTURE task schema capabilities are completely flexible or "transparent" with respect to the descriptive patterning of tasks: BLOCKS may be large, medium or small, as may be STEPS and SUBSTEPS: Elements (BLOCKS, STEPS, SUBSTEPS) of the task may be omitted contingently; Iterative or recursive loops may be defined in task processing sequences; Optional and mandatory paths may be defined. The rationale for building the task schema is to try to provide an all-inclusive framework for describing the task, within which variations of the kind just mentioned as well as many others can then be "declared" at a greater level of detail. The task schema provides the inclusive naming and cataloging function for the task structure, and is in fact completely analogous to a work breakdown structure diagram. Superimposed on the ordinary WBS hierarchy, however, is a numbering system that implies a linear sequencing of task parts, at least at the first-order level of description. Linear sequencing becomes the default condition of the schema, with deflections requiring extra specification information.

The value of modeling task frameworks for the system will depend upon the quality of the modeling inputs, which is an extensive subject onto itself. Experience with modeling tasks should lead to improved modeling inputs. Task modeling for tasks supported by an interactive computer capability has much in common with the preparation of Functional Area Descriptions (FADs) with the added aspect of requiring a firm grasp of the potentialities of the machine system.

Filling the Schema. After the schema has reached a prescribed level of acceptability, to be defined administratively on the basis of experimental and operational experience, it is ready to be filled with detailed task knowledge specifications. It is assigned to a special work file which is accessible to two commands: the "TCALL__ command and the "TLOAD__ command. The former, which was described as part of the "TSCHEMA command explanation, displays the schema, or part of the schema, designated by the number inserted in the slot. Scrolling and edit capabilities are available on the display.

The "TLOAD__ command will operate only on a schema stored in the special work file. Execution of the command with a schema part number in the slot results in the fill-in format displays shown in the four-page Figure 3-8. The BLOCKS, STEPS, and SUBSTEPS of the schema can each be loaded with specifications by the "TLOAD__ command. Normally, only the SUBSTEPS would be so treated, since they compose the STEPS, which in turn compose the BLOCKS. But there will be instances in which entire STEPS reference other STEPS, and BLOCKS reference other BLOCKS or STEPS. Although some consistency checks regarding the specifications loaded into the schema can be envisioned,

INTERACTION HISTORY:

INTERACTION STATUS:

PROVISIONAL TASK NUMBER IS 27 (ASSIGNED BY SYSTEM)

ENTER TASK NAME, (MAX 25 CHAR/SPACES):_____(DO "TAB)

ENTER NAMES OF TASK BLOCKS, ONE ON EACH LINE, MAX 25 CHAR/SPACES AND "TAB.

AFTER LAST BLOCK NAME, ENTER "EX.

BLOCK NUMBER:	27-1	27-2
BLOCK NAME:	27-3	27-4
	27-n	

Explanation:

User employes cursor (␣) and keyboard with full edit capabilities to fill in indicated entries. After last entry, command "EX causes the next stage of the display to be generated as shown in the second window of this display. A "TSAVE command at any point in the process stores the interaction record and results (e.g., the user may wish to accomplish only part of the schema definition at one time).

Figure 3-7a. Fill-in Format Display for "TSCHEMA Command.

INTERACTION HISTORY:

INTERACTION STATUS:

Second Window

PROVISIONAL TASK NUMBER IS 27; TASKNAME: MONITOR SPOT REPORT

BLOCK NUMBER IS: 27-1; BLOCKNAME: JOURNAL ENTRY

ENTER STEP NAMES FOR BLOCK 27-1, MAX 25 CHAR/SPACES EACH LINE, AND "TAB.

AFTER LAST STEP NAME, ENTER "EX.

STEP NUMBER STEP NAME:	27-1-1	27-1-2
	27-1-3	27-1-4
	27-1-n	

Explanation:

TCALL___ (with number in the blank) provides a scrollable display of all saved parts of schemas. Full split-screen edit capabilities allow schema parts to be shuffled and recombined at will. Step name and command "EX causes next stage of display as seen in Figure 3-7c, next page. The underlined values (TASK NAME, 27-1, JOURNAL ENTRY, etc.) are filled in by the program on a

Figure 3-7b. Fill-in Format Display for "TSCHEMA command.

INTERACTION HISTORY:

INTERACTION STATUS:

First Window

PROVISIONAL TASK NUMBER IS 27; TASKNAME: MONITOR SPORT REPORT

BLOCK NUMBER IS 27-1; BLOCKNAME: JOURNAL ENTRY

ENTER SUB-STEP NAMES FOR STEP 27-1-1, MAX 25 CHAR/SPACES EACH LINE, AND "TAB.

AFTER LAST SUB-STEP NAME, ENTER "EX.

SUBSTEP NUMBER:	27-1-1-1	27-1-1-2
SUBSTEP NAME:	27-1-1-3	27-1-1-4
	27-1-1-n	

Explanation (cont'd)

standard hierarchical progression basis. The user may key over the number and name slots, which causes the numbers in boxes to follow the new key-in value. Any step can be returned to or done out of sequence via use of the "TCALL___ command and the keyover capability. If a duplicate box number is entered, the system rejects it with an error explanation.

Figure 3-7c. Continuation of Fill-in Format Display for "TSCHEMA Command.

INTERACTION HISTORY:

INTERACTION STATUS:

Second Window

Prog: "TCALL?

Prog: "SAVE?

Prog: "SUBMIT?

Prog: "CANCEL?

Explanation:

After the last sub-step name of the last step defined, the "EX causes the first of the four messages above to be generated. A "SELECT response causes a display of the entire schema, organized in hierarchical progression and scrollable, with full edit capabilities operative. At the end of the task schema, the three other messages are displayed. "SAVE keeps the schema in its present work file location. "SUBMIT moves the schema to a review and edit file for signoff by higher authority. "CANCEL removes the schema from the work file.

Figure 3-7d. Continuation of Fill-in Format Display for "TSCHEMA Command.

INTERACTION HISTORY:

INTERACTION STATUS:

PROVISIONAL TASK NUMBER IS 27; TASKNAME: MONITOR SPOT REPORT

BLOCK NUMBER IS 27-1; BLOCKNAME: JOURNAL ENTRY

STEP NUMBER IS 27-1-1; STEPNAME MSG TO POST AREA

SUBSTEP NUMBER IS 27-1-1-1; SUBSTEPNAME: SELECT MSG,JR NOS

LOADSCHEMA ACTION ITEM IS SUBSTEP 27-1-1-1; SELECT MSG, JR NOS)

A.) CAN EXECUTION OF THIS ELEMENT BE OMITTED? NEVER
FULL USER DISCRETION
CONDITIONAL

ELEMENT AUTOMATICALLY OMITTED IF:

	OPTIONAL OVERRIDE RETAIN?
ELEMENTS EXECUTED EARLIER IN TASK ARE: (SEPARATE BY COMMAS)	
ELEMENT NUMBERS: _____	Y / N
ELEMENT NUMBERS: _____	Y / N
ELEMENTS OMITTED EARLIER IN TASK ARE: (SEPARATED BY COMMAS)	
ELEMENT NUMBERS: _____	Y / N
ELEMENT NUMBERS: _____	Y / N

Explanation: "TLOAD___ slot can specify BLOCK, STEP, OR SUBSTEP number, the most common being SUBSTEP number. The introductory lines to the left go down the hierarchy only to element specified. Question (A.) is answered by cursor and "SELECT. If CONDITIONAL, the remainder of item (A) lines are presented for user response. The element numbers are BLOCK, STEP, or SUBSTEP numbers. Movement of cursor and display lines to next item is triggered by first or second answer alternatives to question (A), or by "EX after last conditional entry. The "TAB moves cursor to new number pattern.

Figure 3-8a. Fill-in Format Display for "TLOAD___ Command.

INTERACTION HISTORY

INTERACTION STATUS

B.) COMPLETION OF THIS ELEMENT IS SIGNALED BY: NO SPECIFIABLE EVENT

USER INPUT ACTION(S)

SYSTEM OUTPUT ACTION(S)

USER INPUT ACTION CODE(S) (USE BOOLEAN OPERATORS BETWEEN CODES):

_____

SYSTEM OUTPUT CODE(S) (USE BOOLEAN OPERATORS BETWEEN CODES):

_____

Explanation:

Selection of first option in Item B.) causes exit to Item C.) on next page of this display figure. Selection of other two options causes display of appropriate fill-in lines. Exit to next item (C) is by "EX.

Figure 3-8b. Fill-in Format Display for "TLOAD___ Command.

INTERACTION HISTORY:

INTERACTION STATUS:

C.) CAN NEXT ELEMENT OPTIONALLY BE THE SAME AS ANY PREVIOUS? YES NO

IF YES, SPECIFY NUMBERS OF OPTIONAL NEXT ELEMENT(S) (COMMA SEPARATES):

_____

D.) USER CUE/EXPLANATION RECORD NUMBER FOR ELEMENT:____

E.) USER HELP RECORD NUMBER FOR ELEMENT____

Explanation:

Item (C) provides allowance and preclusion of recursion from element. The entries ALL and SUPPRESS___ can be used following a YES option, as in "VIEW and "PRINT commands. The cue/explanation and help records are loaded via file maintenance commands and referenced here by record ident. number. This allows for flexibility and avoidance of duplicative effort for Item D.) and E.), since cue/explains and helps will be standardized and used repeatedly.

Figure 3-8c. Continuing Fill-in Format Display for "TLOAD___ Command.

INTERACTION

INTERACTION STATUS:

F.) CAN DATA VALUES FROM PRECEDING ELEMENTS BE COPIED TO THIS ELEMENT? YES / NO

PRECEDING ELEMENT NUMBER	DATA CATEGORY IDENTIFIER	COPY TO:	DATA CATEGORY IDENTIFIER	COPYING AUTOMATIC OR OPTIONAL?
				AUT / OPT
				AUT / OPT

G.) CAN DATA VALUES FOR THIS ELEMENT BE CHECKED FOR CONSISTENCY, CONTINGENT ON DATA VALUES IN PRECEDING ELEMENTS? YES / NO

PRECEDING ELEMENT NUMBER	DATA CATEGORY IDENTIFIER	VALUE OCCUR RANGE	DATA CATEGORY IDENTIFIER	VALUE ACCEPT RANGE

Explanation:

The tabular fill-in format for Items F.) and G.) both require identification of preceding element and its data category plus data category of current element. Copying action is automatic, or optional to user. Consistency check requires in addition the value occurrence ranges for both preceding and current data categories. Range specifications use MOR, LES, BET, and of course, simple matching for exact consistencies.

Figure 3-8d. Continuing Fill-in Format Display for "TLOAD___ Command.

they will not be treated in this paper, because of their complexity and because of their dependence on particular task structures.

The main forms of task knowledge loadable through the fill-in formats concerns: the automatic and optional omission of elements (BLOCKS, STEPS, SUBSTEPS); contingent recursion to earlier elements on optional bases; cueing, explain, and help materials for any element; automatic and optional data copy-over from preceding elements; and, data consistency checking that is contingent on values in preceding elements of the task.

The administrative and system control aspects of online interactions related to task frameworks will not be described in this report, except for the following general observation:

> The functions of moving Task Frameworks (when finished) into and out of active status will be accomplished by separate commands (not shown here) that are administratively restricted. The command language for these functions is less complex than that just described, and employs the same general strategy.

Interacting with Task-Knowledge Aids. To use stored knowledge of a task during online interaction aimed at performing the task, commands and interaction formats are needed for five main kinds of functions: Initiating Task Aids; Eliciting task-aid related Displays; Stepping through the Task, Management of Data Continuity/Redundancy; and, Use of Task-state sensitive Explain and Help Facilities.

Initiation of Task Aids - Tasks may, of course, be accomplished without task aids. If task aids are to be used in performing a task, they may be initiated in any of three ways: by independent command of the User; by User selection from a Task Queue routed to him for accomplishment; and, as a continuation, upon his reinstatement of a dialog stream. The user-initiated "TRUN__ command, followed by task number, task name, or both, may be accomplished by keyboard, function button, or by menu selection from the alternatives in a "TNAME display. This depicts the range of task names with which the individual having the USERID logged in to the terminal is to be familiar, and indicates which tasks he is authorized to run. The system prevents the user from running the task aid on tasks he is not authorized to perform.

The user may also use the cursor and "SELECT button to initiate the task aid run for some kinds of tasks waiting in his Task Assignment Queue. As part of the load distribution and load smoothing functions of the system, his supervisor may have forced certain message inputs to his work queue, with names of recommended actions appended. In such cases, the "SELECT action can initiate the task aid run for the task referred to him.

A task-aid run can also be reinstated as part of a reinstated dialog stream. The stream commands (pages A-4,5) operate to retain all aspects of the ongoing dialog on an interaction stream that is temporarily set aside by the user, and these aspects include a task-run aid if it is active. A "SWITCHSTREAM command specifying a previously set aside stream containing an active task-run aid will cause that stream to be reinstated, and the task-run aid to take up precisely where it was interrupted by the earlier set-aside action.

Task-Aid Related Displays - Three types of displays are available as adjuncts to the task-aid function. The first is a simple expansion of the information contained in the Interaction Status window of the display. This window always contains the Transaction number, and a Query number if querying is being done. It contains names of the file, table, record, and interaction format in effect whenever such there are, and also contains the letter designation of the stream currently active on the display. When a task-aid is run, the window is expanded to also include designation of the active task name and number as well as the name and number of the block, step and substep currently in effect.

If the user wishes to refresh his memory regarding the task element context of the particular task element currently in effect on his display, he may issue a "TCONTEXT command via button and an "EX. The display generated is essentially the same as the one in Figure 3-6 shown earlier; the task element in effect will be displayed within the row of elements subsumed by its immediate superordinate. In turn, the superordinate element is displayed within the row of elements subsumed by its immediate superordinate, etc. In addition, the connections are depicted between the element of interest and its superordinate elements. To display the context of an element other than the active one, the element number is inserted after the "TCONTEXT command name, followed by "EX.

The user can also review the detailed information loaded for an element by issuing a "TDETAILS command and an "EX. The command works the same way as the one just described (regarding the active element as default choice if no element number is specified).

Stepping Through the Task - When a "TRUN__ command is taken, the first substep of the first step of the first block becomes the active task element. If sufficient information has been loaded for this first element (see earlier description of "TLOAD command), upon sensing completion of the element, the system will automatically advance the Interaction Status window display of active task name and number to the next element in succession. If the system has information that the completion of the element cannot be sensed by the machine, the Interaction Status window will instead show a blinking message: MOVE TASK MANUALLY, signaling that the user must take a "TMOVE button action to advance the task-aid to the next TASK element in sequence.

Given the MOVE TASK signal (delivered manually or automatically), the system will either:

(1) Move the task state display to the next-numbered element;

(2) Skip the next-numbered element(s) and provide a notification message in the Interaction Status window;

AUTOMATICALLY SKIPPED TASK ELEMENTS NUMBER:___

(3) Provide an option to skip the element:

OPTIONAL SKIP ELEMENT (Element Name, No.) Y / N?

The user may recurse to an earlier step by the command MOVE__ "EX with the task element number to be targeted in the slot. The system will honor this command or refuse it, depending upon the data that has been loaded into the task frame (see earlier description of "LOADSCHEMA command).

Finally, when the last substep of the task has been signaled as completed, the system will message:

TASK (taskname, no.) IS COMPLETE

Managing Data Continuity/Consistency/Redundancy - For some task elements, task specification data may have been loaded which allows for copy-over of data from earlier elements, or consistency data checking on the basis of data values from earlier elements. For the data copy-over operation, if automatic copy-over was specified, the involved data slots in the active task element will appear already loaded with data from the earlier element, and accompanied by a symbol indicating automatic copy-over. If the user wishes to view the copy-over source, he moves the cursor to the appropriate copy-over symbol and takes a "VIEW "EX action. The data input format for the earlier task element is displayed, along with its data values and a blinking attention device on the appropriate copy-over value. A "TMOVE action returns the user to the task element he just left for the digression.

In other cases, the system offers an option regarding copy-over. The system displays the option message in the appropriate fill-in space on the data fill-in format for the active task element: AUTOFILL? Y/N. Again, the cursor can be moved to the first letter of the message and the "VIEW "EX action used in the same fashion as just described above. The "TAB button is used to move the cursor to the autofill symbol for automatically filled categories, and to the Y/N fill space for the optional autofill categories. The "VIEW "EX action automatically receives addresses from these two cursor points.

For automatic data consistency checks based on data values from earlier elements in the task, MIQSTURE provides a simple interaction format. If the user enters what is sensed as an inconsistent value in the input format of the active task element, a message is generated:

DATA REJECT accompanied by a blinking symbol.

If the user positions the cursor on the symbol and takes a "VIEW "EX action, the system provides a two-part display consisting of:

(a) The contingent conditions and rules forming the basis of the rejection, and,

(b) The identification and value of the data value(s) that triggered the rejection.

Task-state Sensitive Explain and Help Aids - The "WHAT? and "HELP commands are used in the same manner as described in Appendix B, page B-3, except that the information available to these commands through the specific task element is normally much greater than without the task aids running. Consequently, the responses to the "WHAT? and "HELP commands will be more focused and more thorough. The information and advice contained in the system response materials can be based on user manuals augmented by operational experience with a given task or task element. Experience will suggest changes in task schemas, in the fill for schemas, and in the associated explanatory and help materials.

For explanatory materials, repetition of the "WH command will elicit first an activities checklist for the currently active task element. A "SELECT action from this menu will produce an explanatory discussion. For "HELP materials, each task state element is analyzed (on the basis of experience) for "predicaments" and their associated "solutions". Each predicament is described briefly but understandably in the "HELP display for the element. A "SELECT action from this menu produces the list of possible solutions for the predicament. By adding the task element number to the "HELP command

("HELP__"EX), the user can look at help materials for non-active task elements; the same arrangement applies to the "WHAT? command.

3.7 Scenarios for Terrain Overlay Symbol Data Reference

General. The MIQSTURE subset for symbol/data cross-referencing is aimed at allowing accomplishment of user/system interactions by which: (a) symbology of situation displays are linked by cross-references to their supporting backup data; (b) reciprocal linkages are also established; (c) the benefits of cross-reference linkages are immediately available to users online; and, (d) maintenance of linkages is accomplished dynamically in a mixed-initiative mode. The MIQSTURE approach to these functions is intended to be adaptable to any detailed pattern of symbology and of commands to manipulate the symbology. A MIQSTURE subset for symbol manipulation (i.e., creating, assigning, moving symbols) will not be presented here since it is beyond the scope of this report.

A number of general assumptions underly the MIQSTURE subset to be illustrated here. They are:

*Generation of some of the situation display symbology representing information in message bases and data bases will be essentially automatic; a significant part of the remainder will be semi-automatic with manual override; a minority of symbol generation functions (speaking numerically) will be initiated manually.

*For most types of situation display symbols there will be a requirement for reciprocal cross-reference linkages between the symbol and its supporting data, stored in one or more message or data bases.

*Flexibility for defining new types of cross-reference relations will be needed.

*Cross-references building as well as access and usage must all operate as "real-time" processes in the system.

The displays containing symbology to be cross-referenced to supporting data are mainly concerned with enemy situation (ENSIT) rather than friendly situation (FRENSIT). Since the trustworthiness of friendly situation data is comparatively much higher than that of enemy situation data, the supporting bases for FRENSIT symbology are more uniform and therefore more immediately apparent to the user viewing the symbols. Correspondingly, the need for cross-reference from symbol to supporting data is much less for FRENSIT data.

Among the variety of detailed concepts that can be evolved for ENSIT situation displays, three general types emerge with some consistency: The Detailed Action (Maneuver) display; the Organization position (Units) display; and, the Summarizing (Commander's) display. A type of display that contains aspects of each of the other three is an Event Template comparison display. In it, displays of pre-stored battle scenarios can be compared for their match with a display of the current history of an ongoing action. A brief elaboration of these main situation display types is contained in Appendix C.

Reference Linkage Program Mechanisms. The requirements for cross-reference linkages generated by the above problem statement can be met by a number of different kinds of arrangements and mechanisms. As a basis for positing a MIQSTURE subset to satisfy the four functions enumerated at the beginning

of this section, the following computer program capabilities are presented to indicate how this might be accomplished:

(1) The basic machine-initiated capability is automatic coding that assigns a unique date/time/sequence code to every symbol, message and table row entry handled by the system. The code also reflects the type of symbol, message, or record entry.

(2) The second capability is formation of a set of linkage or association sub-files to contain records in which the above codes operate both as searchable keys and associated data. A record associating one code with another (e.g., a symbol code with one or more message codes) may be created by either automatic or manual (online) assignment actions.

(3) A third capability makes all referenced items (symbols, messages, table entries) searchable by their reference codes. (A "retrieved" symbol will be made to blink and momentarily show its reference code and the associated reference code that evoked its retrieval.)

(4) A fourth "machine initiated" capability is an automatic "chaining" search from any item command-identified on a display (symbol, message, table entry), through the associative linkage record(s), to retrieval of its associated items (symbols, messages, table entries).

INTERACTION HISTORY:

INTERACTION STATUS:

First Window

MANUAL CROSS-REFERENCE LINKAGE ASSIGNMENT FOR _ _ _ _ _ _ _ _ ? Y / N

(Text of message in question presented here)

(System-proposed automatic assignment presented here)

Explanation:

Cursor is auto-positioned to space after question mark, must be moved for either Y or N selection. Alternatively, Y or N can be keyed.

Second Window

ACTION IS: CROSS-REFERENCE LINKAGE ( )MAKE ( )BREAK

SYMBNO: MSGNO: TABENTNO:

( )READY

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Explanation:

Cursor selection of ( )MAKE or ( )BREAK is usual first step. When desired slots have been filled as described in accompanying text, cursor is moved to ( )READY and "SELECT "EX action is taken (a single button). Linkages defined by the fill-in values are then immediately available for use.

Figure 3-9a. Cross-Reference Linkage Related Fill-in Diplays

INTERACTION HISTORY:

INTERACTION STATUS:

Third Window

CROSS-REFERENCES FOR _____ NO:_____

SYMBNO(S):	MSGNO(S):	TABENTNO(S):
_____	_____	_____
_____	_____	_____

Explanation:

Cursor selection from the values displayed in this "CREF command display will retrieve corresponding items for viewing. The name and number of the reference items are presented in the first line. Any command other than "SELECT, "VIEW, or "PRINT, or a query, will cancel the display.

Figure 3-9b. Cross-reference Linkage Related Fill-in Displays

The MIQSTURE language elements needed to accomplish cross-referencing capabilities as they have been outlined above are of three kinds: those for default review of automatic reference linkage establishing routines when they call for help from the user-analyst; those for manually (interactively) establishing and removing linkages; and, those for retrieving cross-referenced materials. Two commands ("LINK and "CREF) are required (see p. B-24). They will be described in scenarios below.

Assigning Cross-References.

Default Review arrangements - These include a system interaction message (MANUAL CROSS-REFERENCE ASSIGNMENT FOR _____? Y / N AND "EX) as shown in Figure 3-9 a first variable sized window. The name slot contains the input message number that has been automatically assigned by the system for cross-reference purposes. Below the interaction message appears the corpus of the input message in question, (with which the system's automatic assignment procedures have had difficulty). The user-analyst reviews the text in question and selects an N answer if the suggested automatic assignment made by the system is satisfactory, and a Y answer if it is not. A Y answer produces the manual "LINK fill-in display to be discussed next.

Manual (interactive) Linkage Action Arrangements - These include the "LINK command and the fill-in display it produces. The display is shown in Figure 3-9 a second variable-sized window. (As indicated above, the "LINK command can be invoked by a Y response to the default review message; it can also be taken directly via function button, keyboard, or a select action the "COMMANDS menu.) In addition to generating the display shown in the

second window of Figure 3-9a the first "LINK command following any other action also arms the cursors (on both the terrain display and the textual display) for additional repetitions of the "LINK command. Immediately following each "LINK action (including the first one) both cursors are automatically moved to neutral corners. This prevents accidental mis-linking from a cursor being left unintentionally on a symbol or message or table entry number. The second and subsequent executions of the "LINK command are coordinated with both cursors; if the terrain display cursor is on the point feature of a symbol, and the textual display cursor is on the first character of the message number (MSGNO) of an input message, a second or subsequent "LINK action will cause the symbol number (SYMBNO) and message number (MSGNO) to be copied into the appropriate first-row blanks on the display shown in the second window. In this fashion, any pattern of linkages can be built up. Alternatively, the user-analyst can key number values into appropriate slots using the regular edit mode.

When the appropriate linkages have been defined in the fill-in table, the user-analyst moves the textual cursor to the ()READY position on the display and takes a "SELECT "EX action. The declared linkages are immediately stored and are ready for use. If, however, the user has not already taken a "SELECT "EX action on either the ( )MAKE or the ( )BREAK option in the display, a reminder message will be produced. When one of the options has been selected, the READY action will then be processed. To erase an established linkage pattern, the parts to be altered are declared using the "LINK command in the same manner as just described, and the ( )BREAK option is selected. If a ( )BREAK option is taken on a non-existent linkage, an error message is

produced for the linkage(s), but existing linkages are broken as per the command instructions. Typographical and other errors are treated through the edit mode which is operational on this command.

While individual linkages can be purged via the ( ) BREAK option, most linkage patterns will be purged automatically on a batch basis according to standardized time criteria.

Arrangements for Retrieving Cross-reference Materials. After cross-reference linkages have been established as described above, they are available for retrieval to aid the user in many ways. The command eliciting the chained retrieval mechanism described earlier is the "CREF" command. When taken (via button or keyboard action or selection from the COMMAND menu) both cursors are "homed" (moved to neutral corner) automatically. The user-analyst can then move the cursor to the location point of a symbol or to the first character of the reference number of a message or table entry that is being displayed, and take the "EX" action. The program response is a display such as that shown in the third window of Figure 3-9b in which are displayed the reference numbers of all items linked to the query item identified by the cursor. The user-analyst then sets the cursor on the first character of any displayed number and takes either a "SELECT" or "VIEW" or "PRINT" action followed by an "EX" to receive a display of the identified item. (The "SELECT" command is interpreted as the "VIEW" command in this context.) Following each transaction, the interaction status display will increment the transaction count but will not increment the query count. The interaction cueing message will offer the standard option of a query or command for the next transaction.

3.8 Scenarios for Defining/Maintaining Data Storage Structures

General. The MIQSTURE subset for defining and maintaining files includes language elements for defining new record types to the system, within the constraints of the three formats discussed earlier in this report. These formats are:

*List format: The record consists of a prescribed sequence of data category fields, the fields being displayed in a horizontal sequence on CRT displays. The record ordinarily contains data for a single event or reported phenomenon. Formatted military messages fit the list format.

*Table format: The record consists of a "stack" of rows, each row having essentially the same structure as the list format. The columns of the table correspond to data category fields of the list format. The row ordinarily contains data for a single event or reported phenomenon. The status of a certain type of equipment (within a certain unit) fits the row of a table format.

*Hierarchical Table format: The record consists of a set of table names which have been declared (defined) as being in a hierarchical relation. The hierarchy may correspond, for example, to the hierarchical arrangement in military units, or to a classification hierarchy in a weapons system. The individual tables in a hierarchy may all have the same format or they may not. Three levels of hierarchy are available; Table, Subtable, and Sub-subtable.

The definitions provided to the system through the MIQSTURE element subset establish the format and contents allowed in the records, the relations between records, and the record types composing each file.

The language elements for maintaining files include those for loading records manually (interactively) and automatically, for modifying/updating records already in the file, for deleting individual records, and for batch purging of records. The two commands required are "DEFINE and "INPUT (p. B-25).

Several aspects of file definition and maintenance activities that are important are beyond the scope of this experimental study, and will not be covered in this report. They include MIQSTURE support for the administrative arrangements for controlling the relations between "system" or "public" files and individual work files of user-analysts, which may have similar or identical formats to some public files. They also include means of defining and maintaining the instructions to the system by which it effects control of user access to data.

File-Defining Elements of MIQSTURE. The program cannot include an undefined record type in a file, or an undefined data category field in the definition of a record. Thus definitions are built up from category fields to records, and from records to files. The issuance of a "DEFINE command produces a menu display of the kind illustrated in the first variable-sized window of display (Figure 3-10(a). The user selects the define action he wishes to accomplish from among the alternatives presented on the menu: DEFINING: ELEMENT, RECORD, TABLE, HIERARCHY, or FILE. His selection elicits the appropriate fill-in display for the task he wishes to accomplish.

INTERACTION HISTORY:

INTERACTION STATUS:

First Window

DEFINING: ( )ELEMENT ( )RECORD ( )TABLE ( )HIERARCHY ( )FILE

Explanation:

Response to "DEFINE command. User "TABs cursor to desired parentheses and takes "EX action.

Second Window

DEFINING ELEMENT: ( )READY AND DOUBLE-CHECKED

DICTIONARY ELEMENT NO. XXXXX ELEMENT NAME _ _ _ _ _ _ _ _ _ _ _ _ _ _

NAME ABBREVIATION_ _ _ _ _ INPUT EDIT RECORD NO._ _ _ _ _

MAX NUMBER VALUES_ _ _ _ _ MAX CHARACTERS PER VALUE_ _ _ _ _

Explanation:

Response to "EX action in first window choosing ELEMENT alternative. The data dictionary number is temporarily assigned by machine initiative; user fills in remaining blanks. When satisified with entries, he moves cursor to ( )READY paren., takes "EX action. Longer cursor move reduces errors.

Figure 3-10a. Definition-Related Fill-in Displays.

INTERACTION HISTORY:

INTERACTION STATUS:

Third Window

DEFINING RECORD: ( )READY AND DOUBLE-CHECKED

DICTIONARY RECORD NO. XXXXX RECORD NAME_____

ELEMENT NUMBERS FOR THIS RECORD'S DATA CATEGORY FIELDS:

F1_____ F2_____ F3_____ F4_____ F5_____ F6_____

F7_____ F8_____ F9_____ F10_____ F11_____ (etc.)

Explanation:

Response to choice of RECORD in action in first window. Record No. supplied automatically is temporary. User "TABs through each fill-in slot. When he has filled the last (F)ield slot required by the record he is designating, he avoids the next slot that is automatically presented, and moves cursor to ( )READY, taking "EX action.

Fourth Window

DEFINING TABLE: ( )READY AND DOUBLE-CHECKED

DICTIONARY TABLE NO. XXXXX TABLE NAME_____

MAX TABLE ROWS_____ ELEMENT NUMBERS FOR TABLE'S COLUMN HEADINGS:

C1_____ C2_____ C3_____ C4_____ C5_____ C6_____

C7_____ C8_____ C9_____ C10_____ C11_____ (etc.)

Explanation:

Response to choice of TABLE in window one. Procedures are analoguous to those in above windows. Program does check on element, will accept only those with MAX NUMBER VALUES___ 1. When MAX TABLE ROWS is blank, it means no limit.

Figure 3-10b. Definition-related Fill-in Displays

INTERACTION HISTORY:

INTERACTION STATUS:

First Window

DEFINING HIERARCHY: ( )READY AND DOUBLE-CHECKED

TABLE LEVEL: DICTIONARY TABLE NUMBER:_ _ _ _ TABLE NAME _ _ _ _ _ _ _ _ _ _

SUBTABLE LEVEL:

DICTIONARY TABLE NUMBER_ _ _ _ _TABLE NAME_ _ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _ _ _ _

etc.

Explanation:

Response to choice of HIERARCHY in first window, previous figure. Table names and numbers are filled in by user, who "TABs cursor to first character position in each slot. Table level and subtable level are treated as a unity; when entries have been checked user moves cursor to ( )READY and takes "EX action. This produces display in second window.

Second Window

DEFINING HIERARCHY: ( )READY AND DOUBLE-CHECKED

SUB-SUBTABLE LEVEL:

FOR SUBTABLE TABLE NUMBER XXXX, TABLE NAME XXXXXXXXXXX:

SUB-SUBTABLES:

DICTIONARY TABLE NUMBER:_ _ _ _ TABLE NAME _ _ _ _ _ _ _ _ _ _ _ _ _

_ _ _ _ (etc.) _ _ _ _ _ _ _ _ _ _ _ _ _

Explanation:

Response to exiting from display in first window. System initiates generation of first table number and name entered for SUBTABLE LEVEL in display in first window. When entries have been checked user moves cursor to ( )READY, takes "EX action. This produces display in third window.

Figure 3-10c. Definition-Related Fill-in Displays

INTERACTION HISTORY:

INTERACTION STATUS:

Third Window

FOR SUBTABLE TABLE NUMBER YYYY, TABLE NAME YYYYYYYYYYY:

SUB-SUBTABLES:

DICTIONARY TABLE NUMBER_____ TABLE NAME_____

_____ (etc.) _____

Explanation:

Response to exiting from display in second window 3-10c. System retains first two lines of previous display, changes third line to second table number and name entered for SUBTABLE LEVEL in display in first window. Process is cycled until all subtable level names and numbers have been presented.

Figure 3-10d. Definition-Related Fill-in Displays

The fill-in display for defining an ELEMENT is presented in the second window of Figure 3-10(a). In this display, the data dictionary number for a new ELEMENT is temporarily (provisionally) assigned by system-initiated generation of the number into the appropriate display slot. (All newly defined structures, including elements, records, tables, hierarchies, and files are stored in a review file. After they have passed review, they are declared active and the provisional numbers become permanent.) System-initiated number assignment is used for elements, records, tables, and files. The INPUT EDIT RECORD NUMBER references a record containing detailed data and instructions for filtering data values input to this element. If no filtering actions are desired, no such record is created and the slot is left blank.

While operating in the "DEFINE mode, each finished piece of definition (signaled by (x)READY "EX) is shifted from the window it is occupying to a new window created immediately above. This window is the SDEFINE, or Storage for Definitions window. It automatically scrolls upward as new pieces of work are shifted into it. The user-analyst can move the cursor into this window and then use the scrolling control to review his finished definitions. The contents of this particular display storage window remain stored even after user LOG-OFF. When the same USERID is again logged in and the "DEFINE action is taken, the SDEFINE window is reinstated, containing the last piece of work shifted to it. The full edit capability is available for the fill-in slots of the finished pieces of definition stored in SDEFINE: alterations can be made. In addition, a "PURGEBAK command erases the SDEFINE display file starting with the definition currently in the window and including all definitions deposited earlier.

The third window of Figure 3-10(b) contains the fill-in display for defining a RECORD. After filling in the record name, the user-analyst "TABs the cursor progressively through as many fields as he wishes his record to contain, inserting the number of the already defined element that he wishes to occupy each field. He can refer to the SDEFINE window, scrolling it as necessary, to keep track of the element numbers he wishes to enter, since their definitions were stored there earlier.

The fourth window of Figure 3-10(b) is almost identical with the display in the third window, except that the element definitions inserted in the slots depict the contents of columns of the table. The other difference, as compared to a record definition, is that a table cannot accept multiple values within one row/column intersection. Therefore the program will reject the insertion of any element whose definition includes MAX NUMBER VALUES greater than one. It will therefore also reject elements where this slot has been left blank, which signifies no limit on number of values.

The windows depicted in Figures 3-10(c) and (d) contain fill-in displays for defining a HIERARCHY among already defined tables. The display in the first window results from selection of the HIERARCHY alternative from the menu display elicited by the "DEFINE command. All fill-in slots are keyed in by the user, the cursor being "TABed into first-character position on each slot. The name and number of an already defined table are inserted into the TABLE LEVEL slots, this table to serve as the trunk (superordinate position) of the inverted-tree hierarchy. The numbers and names of other tables (fitting the user-analyst's concept) are keyed in to the SUBTABLE LEVEL of the hierarchy, to act as the immediate subordinates to the first table. When all tables at the SUBTABLE level have been identified, the user-analyst moves the cursor to

( )READY and takes an "EX action. The result is to store the completed part of the hierarchy definition in the SDEFINE file and move it to the SDEFINE window, and to place the display shown in the second window of Figure 3-10(c) into the active formulation window.

The display shown in the second window of Figure 3-10(c) permits fill-in of the table names for the SUB-SUBTABLE LEVEL of the hierarchy (the "bottom" of the inverted tree). At this last level, the system-initiated fill-in of the previously inserted SUBTABLE name and number of each mid-level table is accomplished; the system presents the first subtable, defined earlier, for insertion of table numbers and names to serve as SUB-SUBTABLEs (subordinates) to the SUBTABLE. When entries are complete and the ( )READY "EX action is taken, the next previously inserted SUBTABLE is presented to receive subordinates (Fig. 3-10(d)). If no subordinates to a subtable are to be defined, the user takes the ( )READY "EX action without making any entries, and the system skips to the next subtable.

The final structure to be defined is the FILE. Figures 3-11a,b devote all three windows to defining a file. In the first window, there is a system-initiated generation of a provisional file number. The user-analyst "TABs the cursor to the slots to be filled with record numbers or table numbers, according to his wishes regarding file contents. He exits the display via a cursor-select of ( )READY and "EX action, and enters the display in the second window.

System-initiated fill-in produces a file name, and number and name for the first record or table for which ELEMENT STATUS is to be defined, plus number and name for each of its elements. The user "TABs the cursor to the select

INTERACTION HISTORY:

INTERACTION STATUS:

First Window

DEFINING FILE: ( )READY AND DOUBLE-CHECKED

DICTIONARY FILE NO. XXXX FILE NAME _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

RECORD/TABLE NUMBERS FOR THIS FILE'S CONTENTS:

R/T1_ _ _ _ _ _ R/T2_ _ _ _ _ _ R/T3_ _ _ _ _ _ R/T4_ _ _ _ _ _

R/T5_ _ _ _ _ _ R/T6_ _ _ _ _ _ R/T7_ _ _ _ _ _ R/T8_ _ _ _ _ _ (etc.)

Explanation:

Response to choice of FILE alternative from "DEFINE command menu display. The file number is provisionally assigned by system initiative. User inserts name for file, "TABs to record/table number slot (1), enters number from previously defined record or table. Same as preceding examples.

Second Window ( )READY AND DOUBLE-CHECKED

DEFINING ELEMENT STATUS FOR RECORDS/TABLES IN FILE XXXXXXXXXXXXXXXXXXXX

FOR RECTAB NO. XXXX, RECTAB NAME XXXXXXXXXXXXXXXXXX:

FF1:ELNO XXXXXX ELNM XXYSSXXXXYXX XXYYSSXX ( )PR ( )VI ( )IND _ _ SCODE

FF2:ELNO XXXXXX ELNM YYSSXXXXYYY XYXYX ( )PR ( )VI ( )IND _ _ SCODE

FF13:ELNO XXXX XXYSSYXX YYXXSS ( )PR ( )VI ( )IND _ _SCODE

Explanation:

Response to exist from display in first window, by moving cursor to ( ) READY and taking "EX action. System-initiated fill-in of file name and of number and name of record or table, and of number and name of each of its elements. For each element, user-analyst "TABs cursor and selects (or doesn't select)whether element is to be printable, viewable, indexed, and may also assign it a security code.

Figure 3-11a. Definition-Related Fill-in Displays

INTERACTION HISTORY:

INTERACTION STATUS:

Third Window

FOR RECTAB NO. YYYY RECTAB NAME YYYYYXXYYYWWWSS SS:

FF14:ELNO XXXXX ELNN XXYYYYZZS ZZZ ( )PR ( )VI ( )IND __SCODE

FFNN: (etc.)

Explanation:

Exit from second window via ( )READY position of cursor and "EX action retains first line of previous display naming defining function and table, and changes remaining lines to describe next table or record to be processed for element status. Note that the FF (File Field) number begins with 14 for the field containing the first element in the new record or table.

Figure 3-11b. Definition-Related Fill-in Displays

positions for making each element PRintable, VIewable, INDexed, and assigned a Security CODE. If the "SELECT action is taken with the cursor so positioned, the element will be processed accordingly into the file definition. The cursor positioned on ( )READY and an "EX action repeats the cycle for the next record or table defined as in the file in the first window.

File Maintenance Elements of MIQSTURE. MIQSTURE formats for loading a new record and for maintaining or updating an existent record on a manual (interactive) basis will be treated here. Language elements for defining more complex automatic file loading and purging operations are beyond the scope of this experimental study and will not be covered in this report.

Since both loading of a new record and maintenance/updating of one calls for input to the file, the "INPUT command is used to initiate all these kinds of actions by the user, after which system-initiated actions prevail. If an "INPUT command is issued and there is no "FILE command that is in effect, i.e., the user is not connected to a file, the program first sends an interaction message asking for one: PLEASE SPECIFY FILE NAME_ _ _, and when the slot has been filled and "EX taken, proceeds as it would have if that file had already been connected when the "INPUT command was issued. The response of the program is to generate a menu display of the numbers and names of all records and tables contained in the file. The user moves the cursor to the description of the table or record type for which INPUT work is desired and takes an "EX. The system responds with another message: ( )LOAD NEW ( )MAINT/PURGE, to which the user makes a selection.

(If the user commonly loads a certain kind of table or record he might take advantage of the alternative approach afforded by the MIQSTURE language's keyboardable substrate construction. He might type in an entire command sequence (to this point): "FILE XXXX "INPU TA23XYZP LOAD NEW "EX, and store it for elicitation by a single function button action.

When the user selects either the LOAD NEW or MAINT/PURGE alternative, the program produces essentially the same fill-in display for each, shown in an example in Figure 3-12a first window and explained in Figure 3-12b second window. This display constitutes the horizontal kind of format used both for list type records and for rows of tables. The same fill-in outline is used both to query and to load, as well as to maintain list-type records and table rows.)

INTERACTION

INTERACTION STATUS:

First Window

{LOAD NEW / MAINT/PURGE {RECORD NO XXXYX NAME XXYYXXZX: / TABLE

(METY) MESSAGE TYPE	(QRNU) QUERY REQUEST NUMBER	(TRA) TRACK #	(EDIT) EVENT DATE/TIME	(DIRE) DIRECT REPORT	(TYSU) TYPE SUBJECT	(TACA) TARGET CATEGORY	(LOC) LOCATION	(DIR) DIR
4A/N	4A/N	6A/N	11A/N	2A/N	6A/N	7A/N	8A/N	3A

CONTD.

(VEL) VEL	(QTY) QTY	(TROR) TRACK ORIGIN	(TRDE) TRADE DESTINATION	(REM) REMARKS
3N	5N	18A/N	18A/N	30A/N

END

Explanation:

Program response to selection message in "INPUT" command interaction sequence. Display specifies whether new load or maint/purge, record or table, and gives identification of type and name.

The fill-in format is identical with that used for the query function, illustrated in an earlier section. User has very little new material to learn.

User fills in values for new record/table row, or fills in values identifying an existent record/table as the first step in maint/purge procedure.

Figure 3-12a. Load New, Maintain/Purge Related Fill-in Display

INTERACTION HISTORY:

INTERACTION STATUS:

Second Window

(For MAINT/PURGE operations, the user employs query procedures to identify a single record/table row, which is displayed in the fill-in format as shown in the first window. The user has full edit capabilities available to do maintenance modifications on the record contents. When finished, he selects the ( )RESTORE option from the message line (shown below) which is displayed immediately below the active fill-in display of window above. To PURGE the record/table row, he selects the ( )PURGE option.

( )RESTORE ( )PURGE

Figure 3-12b. Load New, Maintain/Purge Related Fill-in Displays

4.0 EVALUATION OF MIQSTURE LANGUAGE

4.1 Alternative Approaches to Evaluation

Methods for evaluating online interactive languages vary widely from mathematical modeling of selected features, to "black box" testing in an applied setting in which user acceptance is often used to infer the quality of the design. The methods selected for a specific evaluation should be appropriate to the technical goals of the development situation and compatible with the resources available for conducting the work.

As shown in Table 4-1, evaluation methods can be conceptualized along two major dimensions: (1) Concreteness, i.e., the contextual completeness of the materials and arrangements which represent the interface and its language; and, (2) Predictive validity, i.e., the equivalence of the evaluation measures to the criterion situation for which the evaluation predictions are being made. The effectiveness of an evaluation method is

		PREDICTIVE VALIDITY		
		Lower		Higher
		Expert Analytic Opinion	User Reaction	Performance Measurement
Lesser	Design, Specifications, Concepts	xx	xx	
CONCRETENESS	Non-hardware Simulations	xx	x	xx
	Parts Implementations in Hardware	xxx	xxxx	xxxx
Greater	Integrated Systems Operation	xxxx	xxxxx	xxxxxx

Table 4-1. Effectiveness of Evaluation Methods

defined by its combination of concreteness and predictive validity. Thus, the effectiveness of the methods in Table 4-1 increase as one progresses

downward and to the right. Nevertheless, the lease costly methods in terms of time and money are in the upper left-hand corner of the table.

Although the mode costly methods cannot be dispensed with, early directions in developmental research can be guided most efficiently by less costly methods, while more costly techniques are reserved for the highly focused evaluations used to fine-tune the developmental product.

Given this philosophy, the method of evaluation used in the present project is represented by the left-most (shaded) cell in the top row of the table. The artifacts evaluated are all documented design specifications and concepts, and the evaluative measures used are all derived from expert analytic opinion. In summary, the evaluation methods used fit the present formulative stage of the MIQSTURE development effort, and are also within the scope of the resources allocated to the effort.

4.2 Focus of Evaluation

Two main targets for evaluation of MIQSTURE can be identified. One is the overall approach embodied in the MIQSTURE concept presented earlier, and also evidenced in the second-level functional repertoire described in Section 2.0 of this report. The other is the selected subsets of the language presented in more detail in Section 3.0.

Because of the scope of functions embraced by MIQSTURE, meaningful comparisons of the overall MIQSTURE approach with that of other online languages for support of intelligence data processing are not possible. No other intelligence

processing online language has been designed to: (a) give maximum emphasis to the mixed initiative concept, and (b) provide the framework or armature for a complete functional repertoire for handling tactical intelligence information processing. Thus MIQSTURE shows, as would be expected, more mixed-initiative features and wider functional coverage than its predecessors.

An evaluative question more pertinent to the goals of the project is, "How well does the overall approach meet the two design goals mentioned above?" In this regard, it is the (perhaps biased) opinion of the project personnel that the introductory concepts of mixed initiative, and of a holistically unified language for the system repertoire, have both been satisfactorily enunciated and exampled. On the other hand, it is clear that neither the concept or the repertoire is likely to be perceived as totally complete in the eyes of future developments in these directions.

4.3 MIQSTURE Subsets Evaluations

Evaluation of the subsets of MIQSTURE (presented in more detail in Section 3.0) will be considered separately for each subset. The subsets, in the order in which they will be considered here are:

(1) Task Structure Representation/Control
(2) Terrain Symbology Overlay Cross-Reference
(3) Files Define/Maintain
(4) Basic Interactions, Querying, and Automatic Querying
(5) Tabular Data Query and Calculation

Evaluation of Task Structure Representation/Control Subset. The machine-modeling of process-related knowledge for support of machine-produced

inference has a range of applicability.* The requirements for success in such applications are discussed in the introduction to the Task Structure Representation/Control Subset (Section 3.6). The goal of this subset is not to provide a detailed knowledge model of any particular task process or class of task processes. Rather, the goals are to: (a) provide a framework within which detailed knowledge about computer-assisted processing of tactical information can be stored and organized in ways that are naturally understood by intelligence analysts; and (b) provide the interaction elements of MIQSTURE to fill, maintain, and utilize such frameworks in an online tactical intelligence processing context. The hierarchically structured flowchart format was chosen for the framework because of its similarity to task representations in Functional Area Descriptions (FADs) of tactical intelligence processing tasks. The particular generic types of task knowledge called for in filling the task schema formats were chosen on the basis of intensive analysis of a number of FADs for tactical intelligence tasks, combined with an assumed general structure support system. These interaction formats follow the mixed-iniative philosophy of MIQSTURE.

---

*Moore, J.A., Mann, W. C., Levin, J.A. "A Goal-Oriented Model of Natural Language Interaction." Information Sciences Institute, University of Southern California ISI/RR-77-52 January 1977.

Charniak, E. "Organization and Inference in a Frame-like System of Common Knowledge", Theoretical Issues in Natural Language Processing, Cambridge, Mass.,: Bolt Beranek and Newman, 1975 pp. 42-51.

Rumelhart, D., "Notes on a Schema for Stories". In Bobrow, D. and Collins, A. (Eds) Representation and Understanding, New York. Academic Press, 1975, 211-236.

Worthy, R., Montgomery, C., and Dwiggins, D., Synthesis of Inference Techniques, Operating Systems, Inc. Final Report, RADC-TR-73-219, August 1973.

Our opinion is that the goals cited above have been reached to a satisfactory first approximation which can serve as a good basis for refinement and extension of the concepts.

Terrain Symbology Overlay Cross-Referencing. The need for cross-reference capabilities is evident,* and the goal is to conceptualize an approach to the problem that utilizes mixed-initiative capabilities at several levels. Machine-initiated actions are incorporated in the design of this subset to (1) assign indexing numbers, (2) create and erase indexing number linkages automatically (with user override capabilities), (3) indicate the number and nature of cross-referenced backup items linked to a symbol in response to a single user button action, and (4) to deliver the backup materials on the basis of user menu selections. The cross-indexing arrangements are generalized, so that they apply to all kinds of symbols and to the various types of data stores.

In our opinion, the goals cited for this subset have been reached to a satisfactory first approximation, which can serve as a good basis for refinement and extension of the concepts.

Files Define/Maintain. The elements and arrangements for functions to define and maintain files in MIQSTURE are conventional, but they incorporate several machine-initiated features. A prompting and selection response format is used to focus task interactions for both defining and maintaining

*It is not the purpose of the materials and discussion developed for this subset to: (a) prove the ubiquity of the need for cross-referenced supporting data for terrain overlay combat symbology, or (b) rank-order the estimated utility of cross-references backup for different types of symbols.

files. Once a new record or table type has been defined to the system, a corresponding fill-in display format is automatically generated by system-initiated activities. To maximize positive transfer of user knowledge and skill, fill-in formats for maintaining and for querying a particular type of record or table are made as identical as is feasible. System initiated data carry-over features from earlier to later elements in a task sequence are also part of the file maintenance capabilities, but were discussed in Section 3.6.

In sum, our opinion is that the outline of mixed-initiative features for file definition and maintenance provides a good baseline concept for the next level of development.

Basic Interactions, Querying, and Automatic Querying. This subset of MIQSTURE consists primarily of an online information retrieval system for *list-type record files* that are structurally most similar to bibliographic citation records files. Good comparison models exist for these online language functions in the form of commercially viable information retrieval systems such as IBM STAIRS, Battelle BASIS 70, Informatics RECCON, Lockheed DIALOG, and System Development Corporation ORBIT. Of these, the latter two* support two of the largest and most successful publicly available

*A Brief Guide to ®DIALOG Searching September 1976
Guide to ®DIALOG-DATABASES Vol I, Files 1-25 Aug. 1977
Guide to ®DIALOG-DATABASES Vol II, Files 26-50 Aug. 1977
Lockheed Information Systems, Palo Alto, California

*ORBIT® User Manual
SDC Search Service March 1976
2500 Colorado Ave., Santa Monica, CA 90406

search services accomodating a full range of users from neophytes to seasoned professional searchers. The interface language and arrangements of these two systems are generally conceded to be top state of the art within their field, and their competitive success with a range of customers confirms such estimates.

The MIQSTURE subset being considered here was patterned partly after the best features of the DIALOG and ORBIT languages, and in its basic properties, it may be considered to be -- at a minimum -- comparable to them. However, the MIQSTURE subset has additional properties and capabilities aimed specifically at the mixed-initiative concept and at Army tactical intelligence processing, of which none are addressed in the comparison languages. These are:

(1) Separately incremented transaction and query counters, making selective recovery of extended transaction histories possible without requiring hard copy output of all transactions;

(2) Complete storage of transaction history/product for three transaction steps, making possible "state sensitive" HELP capabilities separate from and supplementive to the Task Representation/Control HELP capabilities;

(3) Overlays for the basic keyboard-oriented language which allow sequences of command terms and parameter names to be summarized by a single function button or menu selection action;

(4) Fill-in display formats for reducing the keystroking requirements for standardized types of query structures as well as providing prompting aid to the user;

(5) Capabilities for formulating automatic MATCHPICK queries for input message stream alerting functions;

(6) Capabilities for formulating input-driven automatic querying statements employing offset operators.

On the basis of all the considerations reviewed above, it is our opinion that the MIQSTURE subset for basic interactions and list-type record manipulations exceeds the capabilities of any comparable language. This subset directly meets certain special requirements of tactical intelligence processing, and is strongly exemplary of the mixed-initiative concept.

Tabular Data Querying and Calculation. Languages for querying and performing computational problem-solving on data stored in tabular (relational) formats go under a number of names: Data Management language; Report Generation language; Relational Data Manipulation language, etc. Comparisons can be made for this particular subset of MIQSTURE, since there exist suitable comparison model languages, for which evaluative studies have been reported in the literature*: RENDEZVOUS, QUERY BY EXAMPLE, SQUARE, and SEQUEL.

---

*RENDEVOUS VERSION 1: AN EXPERIMENTAL ENGLISH-LANGUAGE QUERY FORMULATION SYSTEM FOR CASUAL USERS OF RELATIONAL DATA BASES. Codd, E. F., Arnold, R. S. Cadiou, J. M., Chang, C. L., Roussopoulos, N. IBM Research Laboratory, San Jose, California 95193 1/26/78

Query By Example. Zloof, Moshe M. Proceedings of National Computer Conference, 1975, 44, 431-438.

A Psychological Study of Query by Example. Thomas, J. C., Gould, J. D. Proceedings of the National Computer Conference, 1975, 44, 439-445.

Specifying Queries as Relational Expressions: The SQUARE Data Sublanguage. Boyce, R. F., Chamberlin, D. D., King III, W. F., Hammer, M. M. Communications of the ACM, Vol 18, No. 11, Nov. 1975.

RENDEZVOUS is a data base language patterned after unrestricted English. Many arguments have been advanced in favor of the use of unrestricted natural language in a data base management interface. Basically, it is argued that by using a full natural language capability the user does not need to learn a special query and control language; this, in turn, widens the effective utility of the system, especially among "casual" users. However, a meeting with Codd at San Jose cast serious doubts on the achievement of full natural language capability via the RENDEZVOUS methodology. As has been the case in the past with several natural language data processing projects, crippling problems had been encountered when the RENDEZVOUS system's lexicon was expanded to a few hundred words. Undesirable interdependencies among lexical items and higher (syntactic and semantic) rules in the system not only limit the ultimate size of the vocabulary that can be effectively handled, but create almost insurmountable complexities for rapidly updating the language in the face of changing situations. For this reason RENDEZVOUS was eliminated as a comparison model.

QUERY BY EXAMPLE (QBE) query data values are entered into fill-in table formats corresponding to the tabular relations of the elements in the data base. This is the strong point of QBE which has been incorporated into MIQSTURE in the use of fill-in blank formats for standardized querying. The weak point in QBE is the fill-in use of processing operator symbols, the meaning of which can be obscure with respect to both their function and

---

*SEQUEL: A Structured English Query Language, Technical Report RJ1394, IBM, May, 1974.

Human Factors Evaluation of Two Data Base Query Languages: SQUARE and SEQUEL. Reisner, P., Boyce, R. F., Chamberlin, D.D., IBM Technical Report, RM 1478, 1974.

the significance of their positionings. This aspect of QBE was not included in MIQSTURE. QBE was therefore eliminated as a comparison model.

SQUARE and SEQUEL are directly comparable. The language SQUARE employs a combination symbol and positioning notation. SEQUEL uses natural language like terms in a sequence syntax. Both cover the same domain of functions as the MIQSTURE subset being considered here. In comparisons of SQUARE and SEQUEL, SQUARE was found to be consistently more difficult for both programmers and non-programmers, especially for more complex operations. For this reason, SQUARE was eliminated as a comparison model, and SEQUEL was selected as the most powerful user-oriented language which can be logically compared against MIQSTURE.

MIQSTURE and SEQUEL are quite directly comparable in the query and data manipulation and calculation functions they support. The design concept of the MIQSTURE subset differs from the SEQUEL comparison model in three main ways:

* The MIQSTURE subset is part of the "layering" arrangement of the MIQSTURE language; the subset is an extension of a full list-record handling subset of the language. SEQUEL has no layering concept.

* The MIQSTURE subset uses a four-step syntax involving a "process visualization" sequence for the data processing operations the user instructs the system to perform to accomplish a desired end.

* The MIQSTURE subset provides for chaining of operations through hierarchically organized tiles.

Despite these differences, the two languages are very similar in concept, as can be seen in the following example:

Problem: Output all values in a specified row.

MIQSTURE

```
INTABLE EQUIPT (RECID) XXXX "VI ROW "EX
```

SEQUEL

```
SELECT EQUIPT WHERE RECID = XXXX
```

Problem: Output all values in a specified column.

MIQSTURE

```
INTABLE EQUIPT "VI COL (RECID) "EX
```

SEQUEL

```
SELECT RECID FROM EQUIPT
```

A formal comparison of MIQSTURE and SEQUEL, done by PERCEPTRONICS indepently of Operating Systems, Inc., is reported next.

4.4 Multi-Attribute Comparison: MIQSTURE vs SEQUEL

4.4.1 Methodology

Evaluation Procedure. MIQSTURE was evaluated on seven factors: level of development, training factors, speed factors, power factors, accuracy factors, staffing ease, and user acceptability. Because of the level of technical information about MIQSTURE and SEQUEL which was available for this evaluation, we took a conservative approach by making only an ordinal comparison between the two languages with respect to each of the seven attributes defined in the evaluation model. For each attribute, therefore, an evaluation was made whether MIQSTURE or SEQUEL was considered to be the better candidate in terms of potential effectiveness. Furthermore, importance weights were not assigned to the respective attributes since such a level of analysis was not considered appropriate to the objectives of this evaluation.

The comparisons were made on the basis of a consensus of agreement between a panel of experts. These individuals compared the two languages independently at first, and then discussed their evaluation with the others in a group setting. In all evaluations presented below, the judges were in complete agreement on the final determination of which language was better than the other on a particular attribute. The judges were three scientists on the Perceptronics technical staff: two are human factors engineers with several years of experience in the area of man-computer interaction, and the third is a computer scientist with considerable experience in interactive software.

4.4.2 Evaluation Results. The results for the analytical comparisons between MIQSTURE and SEQUEL, and their explanations, are presented in this section. Also included are definitions for each of the evaluative attributes. The discussion is organized according to the hierarchically structured evaluation model.

Implementation. This category of attributes refers to the ease of implementing a language so that it becomes operational on a new system. This is decomposed into two attributes -- level of development and training factors.

Level of Development. This attribute focuses on the design features of a language and is concerned with the level of risk associated with the principles upon which the language is based. At the low end of this attribute would be a state of the art system employing innovative concepts, but with little knowledge about the risk contingencies associated with its use. At the high extreme, would be a language based upon established principles with little risk involved in its use.

Comparison. At present, SEQUEL has been implemented and subjected to some test and evaluation, while MIQSTURE is in the conceptual stage of development. In a comparison on this attribute, SEQUEL would be ranked higher because its use carries less uncertainty and risk than that associated with MIQSTURE. The two languages, however, appear to be based on the same principles, and the innovations found in MIQSTURE are refinements of these concepts directed at improving the general usefulness of query languages. Any increase in risk associated with level of development should therefore be weighted against the gain to be made in terms of improved language features.

Training Factors. This attribute expresses the ease with which a typical operator can acquire facility with the language. Considerations which compose this attribute are the learnability and retention rates associated with the use of the language. Learnability addresses the ease with which the potential user may gain the skill needed to successfully employ the language. Retention concerns the ease with which the user may recall the elements and organization of the language after a period of non-use.

Comparison. Because MIQSTURE is more natural in its procedures than SEQUEL, it is ranked better in terms of training factors. MIQSTURE should be comprehended and retained more readily by users because its processes are conceptually more compatible with natural though processes. In particular, the layering structure and logical organization of MIQSTURE offers an advantage which should lead to increased learnability and retention. Other MIQSTURE features such as format prompts and adaptive feedback also contribute to its superiority in terms of training ease.

Operability. This category of attributes concerns the level of performance that a user can attain while operating with the language. The attributes included in this category are speed factors, power factors, and accuracy factors. Because of its importance within the current scope of evaluation, the attribute of "power factors" is further broken down into sub-attributes of precision, flexibility, and expandability.

Speed Factors. This attribute concerns the time required by the user to complete the processes necessary to input queries to the system. This interval includes both the time it takes him to formulate the query and to key it into the machine. In other words, speed refers to the length of time required for the user to translate a question in his mind into a user-based command recognized by the system.

Comparison. MIQSTURE is judged to be rated higher than SEQUEL in terms of speed. Although SEQUEL appears to employ more natural language than MIQSTURE, the logic behind its terms are more obscure and the formatting rules are more complex. MIQSTURE, on the other hand, more naturally and logically represents the thought processes required to generate and enter queries, and this feature should enhance the speed of query inputs.

Power Factors. This attribute measures the general capability of the language to enable the user to accomplish his information-related task goals. As enumerated below, power takes into account both the precision and flexibility of the language, as well as its potential for future expansion. These factors are identified as sub-attributes of the "power factors" attribute, and they are employed individually in the comparison of MIQSTURE with SEQUEL.

Precision. This sub-attribute covers the extent to which the language allows the user to specify the information format and level of detail he desires.

Comparison. Both languages appear to be relationally complete and non-procedural in nature, with the main difference between them appearing in syntax. MIQSTURE appears to be more compact in various ways. It allows the user to save particular queries (SA) which can be referred to by name (i.e., QF3) in subsequent query formulation. Another facility for compactness is the ability to use "macro" button actions in place of command parameters. The main advantage of MIQSTURE in the area of precision appears to be the ability to display a menu for each message type to aid the user in relating the appropriate Category Field Identifier. MIQSTURE is therefore more suited for implementation on a visual display terminal than SEQUEL. Both languages provide functions used for analysis of the aggregate data. One restriction of MIQSTURE not found in SEQUEL is that the user can operate on only one table at a time. However, this feature makes MIQSTURE more procedurally explicit and logically understandable than SEQUEL.

Flexibility. This sub-attribute concerns the extent to which the language allows the user to adapt the language to his needs. Ability to tailor synonyms, abbreviations, and command procedures for specific tasks all contribute to flexibility. For a high level of flexibility, the user should be able to adapt the system so that tasks frequently performed can be made easiest to use.

Comparison. MIQSTURE receives a higher ranking on this sub-attribute than SEQUEL due to its adaptability to user preferences. This flexibility includes choice of full or abbreviated commands, messages, input procedures, and provisions for setting in task-specific constants. SEQUEL, on the other hand, does not provide flexible features of this kind.

Expandability. This sub-attribute refers to the extent to which the language is amenable to future development. At the low end of expandability would be a language which is so logically and tightly organized that additions to it would destroy the logic and constancy of its structure. At the high extreme, would be a totally modular language.

Comparison. The main body of the MIQSTURE language appears more modular in nature than that of SEQUEL. SEQUEL does include an open-ended list of library functions, but most of these also appear in MIQSTURE. The fundamental operation in SEQUEL is represented syntactically as a SELECT-FROM-WHERE block, and most user queries must conform to this structure. A similar structure appears in MIQSTURE for table type query formulations. Formulations are made in four sequential steps, but parameters for the first three steps may be left off in which case the last values entered for each category are still in effect. Each formulation step is functionally independent and therefore modular in nature. Also, the system commands outside of query formulations are modular. The elements of the query transactions of MIQSTURE are layered with each outer layer adding more capabilities, and this structure facilitates possible future expansion of the language.

Accuracy Factors - This attribute measures the extent to which the language permits the user to perform his tasks in an error-free manner. Such questions which are addressed by this attribute are: Is the language prone to clerical errors and errors of commission, omission, and sequence? How easily can the user detect an error in his input? How simply can detected errors be corrected? And, to what extent can undetected errors degrade system effectiveness?

Comparison. It is difficult to evaluate a query language with respect to accuracy factors without a sufficient amount of empirical error-date collected from operational tests. However, language constructions and organizations do lend themselves to a predictive analysis of the kinds and frequency of errors that can be expected to be observed. Based on such an analysis, MIQSTURE appears to offer an advantage over SEQUEL in terms of being less prone to operator-originated errors. This conclusion has been formulated from several language characteristics covered in the discussion of other attributes. The increased naturalness/logic, learnability/retention, and precision/flexibility of MIQSTURE over SEQUEL all contribute to the prediction of its increased accuracy. More specifically, such MIQSTURE features as format prompts, adaptive feedback, function buttons, one table-at-a-time operation, personalized abbreviations/commands, step-saving storage, should combine to make usage errors less frequent, and more easily detectable and correctable.

Personnnel Factors - This grouping deals with the characteristics of the human users that impact upon the functioning of the system. This category is separated into two components - staffing ease and user acceptability.

Staffing Ease - This attribute refers to the human skill and performance levels that must be addressed in the operator-selection process. To what degree are specialized skills and proficiency levels required for proper language utilization? What level of effort or work-load must be maintained by a sufficiently skillful user so that he may consistently achieve adequate task performance with the system?

Comparison. In terms of promoting staffing ease, MIQSTURE appears to have an advantage over SEQUEL. The reasoning behind this judgment is that it would appear to be easier to train operators to gain and maintain proficiency with MIQSTURE. Furthermore, the potential improvements in speed and accuracy factors and overall better human factors orientation should enable less skillful users to satisfactorily perform intelligence information processing tasks. Also, since MIQSTURE is more flexible and user-adaptable than SEQUEL, the perceived effort involved in its use can be expected to be reduced.

User Acceptability - This attribute refers to the user's perception of the relevance and value of the language for his purposes. A language which is consistent with the logic, experience, and task needs of the user is perceived to be comprehensible and usable. At the low end of this attribute is a language that is so obscure that it is judged unintelligible and unacceptable.

Comparison. The judgment of user acceptability should be primarily based upon the matching of user characteristics with system characteristics. Since the principal users of MIQSTURE can be expected to be intelligence

TABLE 4-2 Multi-Attribute Comparison Study

CATEGORY	ATTRIBUTE	EVALUATION
Implementation	Level of Development	SEQUEL > MIQSTURE
	Training Factors	MIQSTURE > SEQUEL
Operability	Speed Factors	MIQSTURE > SEQUEL
	Power Factors	MIQSTURE > SEQUEL
	Accuracy Factors	MIQSTURE > SEQUEL
Personnel Factors	Staffing Ease	MIQSTURE > SEQUEL
	User Acceptability	MIQSTURE > SEQUEL

analysts and not computer-oriented personnel, MIQSTURE would clearly have the edge over SEQUEL. This is obviously because MIQSTURE is task-based and should therefore be seen as much more relevant and usable by intelligence analysts than the more general, mathematically-based SEQUEL. Furthermore, the structure and organization of MIQSTURE elements should appear more acceptable to the analysts because of their increased logical and natural orientation.

Summary. The results of the multi-attribute comparisons are summarized in Table 4-2. On all attributes, except "level of development", the capability of MIQSTURE was judged to exceed that of SEQUEL. Thus, on a pretty much overall basis, MIQSTURE rather than SEQUEL is considered to be characterized by more potential effectiveness for accomplishing intelligence information processing tasks. These results are not surprising since MIQSTURE was designed to improve upon the functional characteristics and human factors interface of existing interactive query languages such as SEQUEL. If "level of development" is considered as a cost parameter, its impact can be weighed against the cumulative advantages to be gained by the subsequent development and implementation of a MIQSTURE-type language. Our own view would be that any risk to be incurred by this development effort would be well worthwhile in light of the potential improvement to be realized for interactive intelligence processing.

APPENDIX A

DESCRIPTION OF REQUIREMENTS FOR MIQSTURE

1.0 GENERAL DESCRIPTION OF WORK STATION REQUIREMENTS

1.1 Figure 1 contains a sketch of a generalized command-button matrix keyboard appropriate for the kinds of user work-stations used in a computer-based tactical intelligence system. No claim is made that this diagram is either complete or in a properly human-engineered form; it is for conceptual and illustrative purposes only. The figure reinforces several points. The first is the multiplicity of main types of functions for which keyboard and button matrix provisions must be available. There are eight main types shown in the figure, corresponding to the main blocks of the figure:

(1) TASK CONTROL button matrix in the top block of the figure;

(2) MAIN SEARCH ELEMENT elements in the left block of the second row;

(3) SPELL-OUT KEYBOARD (hunt and peck) right block of the second row; this hunt and peck keyboard will be augmented by a standard keyboard for touch typists;

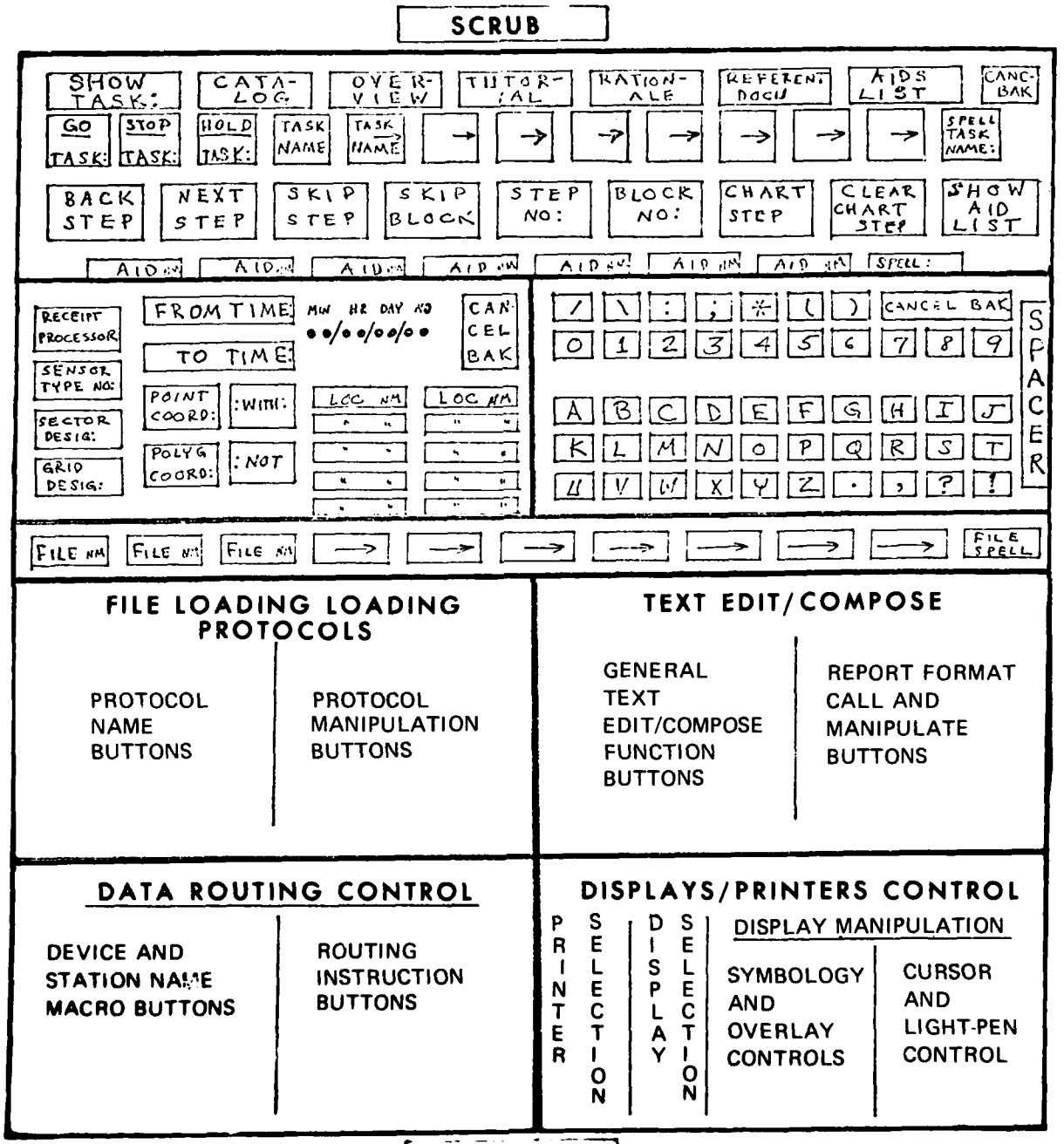
(4) FILE NAME button matrix in the third row-block;

(5) FILE LOADING PROTOCOLS matrix in the left block of the fourth row;

(6) TEXT EDIT/COMPOSE button controls in the right block, fourth row;

(7) DATA ROUTING CONTROL buttons in the left block of the bottom row;

(8) DISPLAYS/PRINTER CONTROL buttons in the right block, bottom row.


SCRUB
SHOW TASK:
CATA-LOG
OVER-VIEW
TUTOR-IAL
RATION-ALE
REFERENT DOCU
AIDS LIST
CANC-BAK
GO TASK:
STOP TASK:
HOLD TASK:
TASK NAME
TASK NAME
SPELL TASK NAME:
BACK STEP
NEXT STEP
SKIP STEP
SKIP BLOCK
STEP NO:
BLOCK NO:
CHART STEP
CLEAR CHART STEP
SHOW AID LIST
AID NM
SPELL:
RECEIPT PROCESSOR
SENSOR TYPE NO:
SECTOR DESIG:
GRID DESIG:
FROM TIME:
TO TIME:
MN HR DAY MO
••/••/••/••
CAN-CEL BAK
POINT COORD:
:WITH:
POLYG COORD:
:NOT
LOC NM
CANCEL BAK
SPACER
FILE NM
FILE SPELL
FILE LOADING LOADING PROTOCOLS
PROTOCOL NAME BUTTONS
PROTOCOL MANIPULATION BUTTONS
TEXT EDIT/COMPOSE
GENERAL TEXT EDIT/COMPOSE FUNCTION BUTTONS
REPORT FORMAT CALL AND MANIPULATE BUTTONS
DATA ROUTING CONTROL
DEVICE AND STATION NAME MACRO BUTTONS
ROUTING INSTRUCTION BUTTONS
DISPLAYS/PRINTERS CONTROL
PRINTER SELECTION
DISPLAY SELECTION
DISPLAY MANIPULATION
SYMBOLOGY AND OVERLAY CONTROLS
CURSOR AND LIGHT-PEN CONTROL
EXECUTE

1.2 The second point made by the illustration is use of the function button approach, which greatly minimizes the number of keystrokes or button actions needed to pass frequently occurring interaction messages to the system. At the same time, the spell-out buttons at the right-most position for several of the button matrices for a command area allow for additional commands to be included by spell-out entry through the keyboard. A third point is represented by such buttons as the AID-name buttons of the first row-block, the LOCation-name buttons in Block 2 (as numbered above), the PROTOCOL-NAME buttons of Block 5, and so on; most of these buttons represent programmable MACRO buttons, depression of which invokes the operation of program software routines that can be modified to suit the application.

1.3 Figure 2 contains sketches (not to scale) of two types of computer-driven displays that might be found at most tactical intelligence work stations. Depending upon the work station functions and workload, the station may support more than one of either type of display. The topmost display represents a back-projected translucent plasma surface. All cartographic, weather/terrain dependent, mobility corridor, etc. information ordinarily storable on maps and map overlays is back-projected onto it. That is, all information is projected that can be gathered and stored on the basis that it can be pre-determined and has highly predictable properties. All dynamic information (movements, concentrations, etc.) is computer-generated as combat symbology onto the plasma surface. The back-projected and plasma-generated elements of the display are, of course, aligned for scale and position.

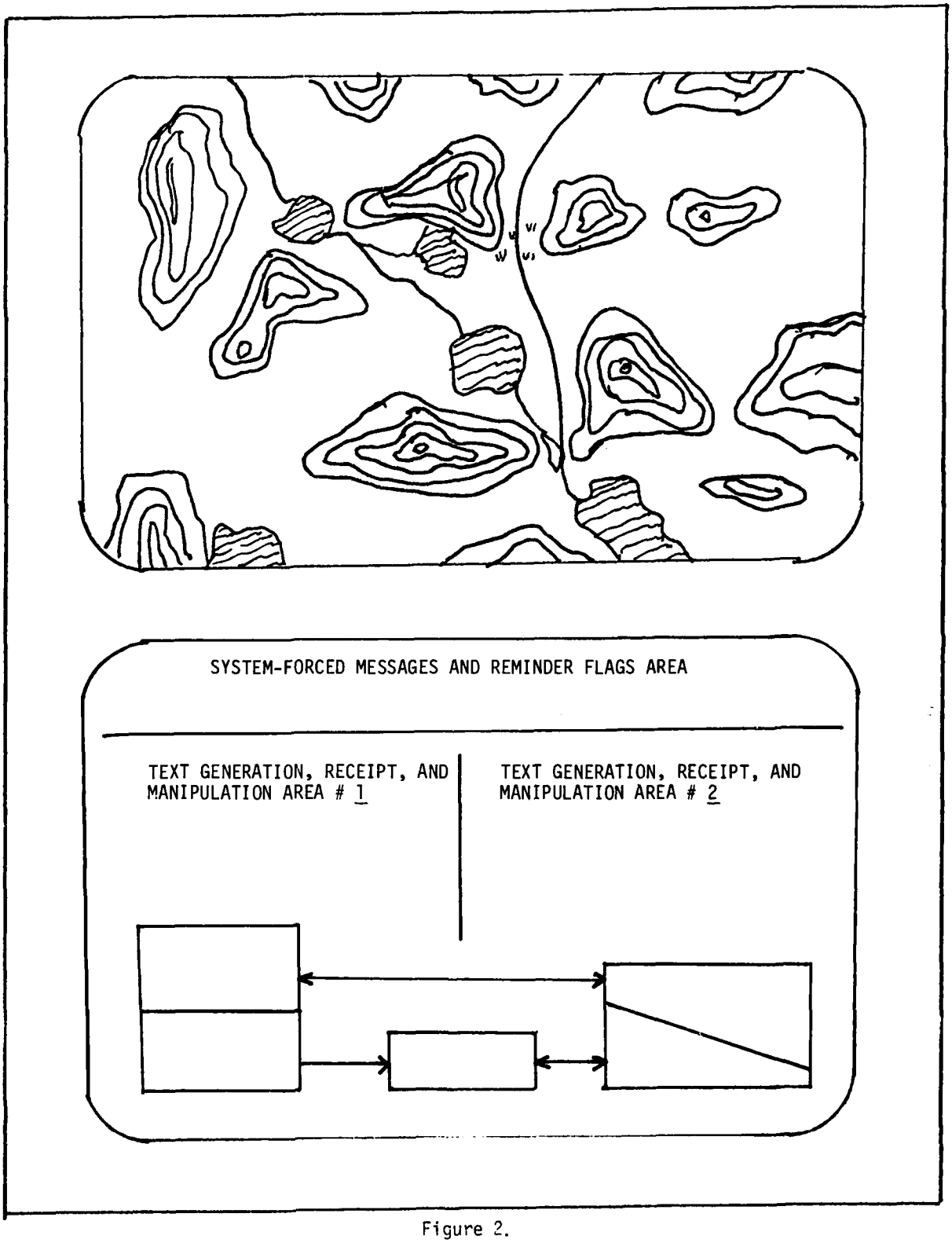

SYSTEM-FORCED MESSAGES AND REMINDER FLAGS AREA
TEXT GENERATION, RECEIPT, AND MANIPULATION AREA # 1
TEXT GENERATION, RECEIPT, AND MANIPULATION AREA # 2

Figure 2.

1.4 The bottom display in Figure 3 is meant to signify that several text display support functions must be available. First, there must be an area where the system can force messages to the user without writing over other material that the user has on his scope. Second, there must be more than one text-handling area, so that composition and text-comparison work can be done without straining the user's memory. Third, the text-handling areas must incorporate optional box-display and highlighting capabilities which allow the operator to produce memory aids and special display lists for himself that are similar to the notes and diagrams he may produce with paper and pencil during composition and editing work, but which are easier to use in the computer-based context because the contents are machine stored and machine readable.

1.5 The topic of interconnections of work stations needs only brief mention here. The conception we wish to employ for the present judgmental exercise is that of a Tactical Intelligence system in which the pattern of connections operating between various work stations is restricted administratively, but in which the system has the potential for any desired pattern of connectivity. The other assumption we wish to make is that a very large proportion of the data inputs to the system will be in machine-readable alpha-numeric forms; i.e., the presently predominant type-written and teletyped messages will be somewhat more regular in format and will be automatically deposited in machine-readable input buffer files. Data arriving in other forms (voice, hard copy) will be keyboarded directly into the same kind of buffer file.

2.0 DISPLAY SUBSYSTEM SUPPORT PROVISIONS

2.1 Considerations for the display subsystem (the displays and their associated button matrix and keyboard components) will be presented first, because both user/system interaction activities and intelligence product output activities take place in the arena represented by the display subsystem. The display subsystem is a very important system component, since through it many of the finished intelligence products for the commander are delivered. In essence, the other data processing activities of the system culminate mostly in information products flowing through the display interface. The most recent advanced exemplar of this point of view comes from the Intelligence Preparation of the Battlefield (IPB) project team within the USAICS at Fort Huachuca. Their draft document TC30-27, April 77 has been used as the main initial design baseline for the display subsystem subset of the functional provision descriptions for the MIQSTURE interactive language. We will not attempt to mirror all aspects of their current design concepts but will adopt their general conceptual framework. We take the liberty of rephrasing some of their concepts and of changing some emphases for our purposes.

2.2 In our view the IPB approach contains three major premises that have ramifications for automated support for tactical intelligence information processing:

2.2.1 Rate of Change and Information Overload - A major theme is the phenomenally increased rate of change of battle situations in modern mobile surface war. Although terrain and weather factors change no more rapidly than before,

maneuver and logistic aspects of the battle move much more quickly than in earlier times. The accelerated changes generate much larger volumes of information per unit time than previously experienced, and also sharply reduce the commander's available time in which to react properly to changing situations. Thus, the required rate of information processing and assimilation is increased from two directions. Since the human capacities of G-staffs and commanders for assimilating information remain essentially unchanged from those of the past, technical means must be employed to help repackage the large volumes of data into forms that contain all the essential information but are much easier to assimilate (i.e., provide a higher information yield per unit of time spent in assimilation).

2.2.2 Preparation versus Reaction Orientation - To achieve such high levels of information assimilability, much data processing activity must be done and this takes valuable time. IPB emphasizes doing as much early preparation as possible, in the form of pre-analysis of information relevant to future battle decisions; sufficiently stable forms of information such as terrain features having combat significance, as well as their interactions with types of weather, are pre-analysed for their potential combat decision relevance. The analytic results are stored in relatively inexpensive media such as maps and overlays for maps. Thus, for example, information on mobility corridors and fields of surveillance and fire under various weather conditions are pre-stored. Somewhat less stable, more widely variable information may also be treated in this fashion; enemy doctrine regarding many aspects of battle behavior is known to within a certain measure of accuracy, and can therefore be used to

pre-analyse alternative probable enemy actions on a specific piece of terrain under prescribed conditions. The various enemy doctrinal maneuvers as well as the courses of counter-actions open to the friendly commander can be stored in the form of templates. Similar templating techniques can be applied to GOB-oriented information about the complex patterns of enemy electronic emissions that must occur (within some margins of variation); sparse patterns of enemy emissions may be compared with such templates to yield very profitable inductions and deductions about ongoing enemy actions. Again, the time-consuming analysis of various alternatives is accomplished ahead of time and the results stored for future use.

2.2.3 Information Compression into Decision-Oriented Displays - The IPB approach emphasizes the use of displays that summarize large amounts of information in highly processed, finished forms. Information about the enemy, which is received in almost exclusively non-graphic form is subjected to several levels of intelligence evaluation processing and summarized mostly in SITMAP-oriented graphics for the commander's consumption. Highly dynamic information is combined with results from the pre-analysed information described earlier by being superimposed graphically. The combined result is a display containing much of the combat decision-relevant information needed for timely decision by the commander.

2.3 In the following brief descriptions of display subsystem interaction provisions to be considered for inclusion in MIQSTURE, it is assumed that the pre-analysed types of information described above will be stored non-digitally in the form of images back-projected through the plasma display

surface. It is also assumed that the dynamic information will be stored digitally in various raw and partially processed forms, and that summaries of the dynamic information will be presented via digital driving of the plasma display (upon which the pre-analysed images are also being back-projected). The above technical arrangements or functionally equivalent ones represent a plausible set of baseline assumptions for a future Tactical Intelligence System. The exact line of demarcation between the types of information to be stored digitally and to be stored on transparent media is a matter for detailed analysis of data volumes and change rates for different types of information, as compared to the overall developmental and operational resources to be committed to such a system, and is therefore not of concern at this point.

2.4 Similarly, at this juncture it should be pointed out that this paper does not require any firm assumptions to be made regarding the exact technical properties of the display system (or the computer system, for that matter). Issues such as color versus black and white displays; automatic blinking and underlining capabilities; character matrix size, overall screen capacity and resolution are not of immediate concern.

2.5 Display/Overlay Adjustments - Means must be provided for adjusting the scale factor on the computer display to the map and overlay projection transparencies of various scales that may be projected in conjunction with the plasma display. Means for x,y movement of the computer display for coordinating reference marks on the display and the projections will also be needed.

2.6 Symbology - In the operational system, different terminals will be "enabled" to allow them to perform different functions with respect to the common symbology of the system. The following kinds of function capabilities must be available for symbology:

2.6.1 Combat Symbol Catalog - To display and define all available symbols (e.g., weapons, organizations, etc.) on command of the operator.

2.6.2 Allowed Actions Display - To show operator at a particular terminal what actions are available to him with respect to a symbol he specifies and/or what terminals are allowed a particular symbol/action combination he specifies.

2.6.3 Symbol Creation - Capabilities will be available to a very restricted set of users, and will allow a certain range of symbol production and also allow modifications of symbol/action combinations available to various classes of terminals.

2.6.4 Symbol Movement/Positioning - Three modes of symbol Movement/Positioning Control will be available, and symbols will display the mode they are under at all times: (a) Manual - in which each change of symbol position is effected by a command executed from a specified terminal(s), and the record of which terminal issued the latest update command is available through a status interrogation command; (b) Auto-Extrapolate - in which symbol position is updated on the basis of an algorithm running off the computer clock; and (c) Auto-Data-Driven - in which symbol position is updated on the basis of an algorithm triggered by changes in data values from a data base. (These

three positioning methods are to be reserved -- conceptually -- for symbols representing real-time movements of enemy and friendly forces, etc., not to display symbols depicting event and decision templates).

2.6.5 Update Status - Two kinds of capabilities will be available for keeping the operator informed about the update status of a symbol: (a) a light-pen/botton command will cause the date/time group for the latest update of that symbol to be displayed, and (b) an optional update-monitoring feature will allow an update age to be specified for any particular symbol on any particular display, which when exceeded will force an attention device.

2.6.6 Designator/Assignment - Commands must be available to allow a symbol to be placed on the display and removed from it. No symbol placing or removal action will be possible without fulfilling the requirements of designation (creation of a name/description scope note referenced in the computer to that symbol; e.g., "111th Brig.", "Unknown Bn", etc.) and Assignment (place/remove authority).

2.6.7 Scaling/Distortion - Means should be available to change the physical size and shape of a symbol for an organization to allow conformance to topography (or else, shaded shaping-drawing capabilities on the CRT should be available to which can be appended unit designator symbols).

2.6.8 Symbol X-Ref - Means must be available for the operator to provide cross-reference pointers between one symbol and another, and between a symbol and certain data values or records in the data base or message base. A cross-reference notation that allows for automatic user-initiated reference-link searching would be very desirable.

2.7 Screen History - Means must be provided to store the bit-streams generating an ongoing display, with ability to stipulate the time period (moving window) for which such storage is desired. A related command will allow retrieval of the dynamic display beginning at any specified time within the storage window.

2.8 Map Coordinates Output - As part of the map-transparency/computer-display synchronization steps required for use of each new transparency, the map scale factor and two bench-mark coordinates must be given to the display subsystem. As a result, it will be possible to use light-pen or cursor to designate any position on the display for which a button action will cause its map coordinates to be displayed. (These will be very useful for communication on voice channels between terminals, etc., and also as input for formulating "polygon" searches to be carried out on the data base.)

2.9 Track Calculations - A track on the CRT display will be able to be defined by two methods: (a) by light-pen or cursor-point "drawing" of a regular or irregular line, or (b) by drawing of a regular or irregular line by the "draw-point" feature of any moveable symbol, (the point on the symbol that maps points on the drawn line). In the case of using the draw-point feature, the Screen-History option must be in effect. For any track so defined, a complete set of time-distance calculation services will be available, referencing the particular map-scale synchronized to the display:

2.9.1 Distances between designated points on the track, (i.e., following the irregular line).

2.9.2 Average rates of movement along the track, given departure-arrival times at points on the track (the times may be supplied automatically by the moving symbology, or be manually input).

2.9.3 Extrapolatory ETAs for moving symbols given a drawn extension of their line and a point designated on the line, on the basis of calculated average movement rates, or such rates input manually. The same calculations for manually drawn lines, points, and rate values.

2.9.4 Track-cluster Statistics given a set of designated tracks on a scope, distribution statistics (means, sigmas, etc.) can be calculated for the set of tracks for values type 1, 2, and 3 above.

3.0 TASK MANAGEMENT/CONTROL SUPPORT PROVISIONS

3.1 Block one (1) of Figure 1, described on Page 1 as the "TASK CONTROL matrix" hints at the kinds of interaction functions that must be supported for intelligence data processing task management and control. These functions will now be considered in more detail.

3.2 Task Management/Control Aids Construction and Modification - Provisions must exist for defining to the machine system the identifications and sequencing of substeps comprising an identified intelligence processing task. These provisions will be used by specially designated Intelligence personnel to construct and to modify task definitions for use by the system in supporting the user in pursuing the tasks. The constructor will use his own task and task-requirement experience in approaching this duty, and may be aided by analytic techniques consisting of formats for analysing the pre-conditions of sub-steps in each task for which the system is to provide user-analysts machine support. Figure 3 sketches an example of one such format that could be useful in analysing the sub-steps of a processing task. (Such tasks are similar to those depicted in flow diagrams in FADs.) In the figure, the rows (labeled on the left) depict the nature of the output of a completed task substep, showing three main types. The columns (labeled on top) show three types of relationship between the preceding step output and the succeeding step initiation input requirements. The row/column intersection cells depict the kinds of activities the system could be expected to take under different combinations of conditions. Data of the kind depicted in Figure 3 can be used

CURRENT STEP (CS) INPUT DETERMINATION PRECONDITION

PRECEDING STEP (PS) COMPLETION OUTPUT CONDITION	NECESSARY CS is "no-go" although "enabling" PS output is available.	SUFFICIENT CS is "go", provided "enabling" PS output is also available.	NECESSARY & SUFFICIENT CS is "go" without any further conditions.
A single machine-interpretable output is produced.	Machine stores PS output for future reference.	Machine checks enabler conditions and if available initiates CS.	Machine initiates CS.
Multiple machine-interpretable outputs are produced.	Machine accomplishes above step for each PS output.	Machine accomplishes above step for each PS output, and for those that are "go", applies priorities or queries operator for priorities, then initiates actions accordingly.	Machine applies priorities or queries operator for priorities and then initiates CSs accordingly.
One of above conditions holds, and outputs not interpretable by machine are also produced.	Machine accomplishes above step(s) as appropriate, and also: (1) Identifies uninterpretable material, and (2) Produces "human-help" reminder checklist associated with PS, for future reference.	Machine accomplishes above steps and also outputs displays based on materials stored for Items (1) and (2) in box to left of this box.	Machine accomplishes above steps and also outputs displays as described in box to left.

Figure 4.

to design prompting-sequence and fill-in displays to aid the person who is constructing new task sequences or modifying existing ones. Figure 4 provides an example of such a display.

3.3 Task Representation Displays - The identification, description, and sequencing of substeps in a task must be displayable to the user by the system. As suggested by the task-control block of Figure 1, tasks will be composed of a series of blocks, which in turn are composed of a series of substeps. Task substep preconditions (such as those suggested in the previous item) should be displayable on command of the user. The displays should be hierarchized; the root display showing tasks and their interrelations, each task display showing blocks and their interrelations, and each block display showing steps and their interrelations, etc. The user should be able to browse these displays. Step and substep descriptions should indicate whether or not they are optional, and under what conditions. The explanations (as all explanations provided by the system), should be available in a very short "reminder-refresher" version, and a more complete "tutorial" version. Figure 6 provides an example of such a display for a hypothetical case in which the task of monitoring the automatic processing of a SPOT REPORT is sketched. Block 27-3 (sitmap refer) of Task 27 (monitor spot report) expands to 5 steps, one of which (27-3-3: CHEK/AUTO HI TEMPLATE) in turn expands to 5 substeps.

3.4 Task Substep Sequence Control - In addition to the system-initiated automatic task substep sequencing suggested by Item 1 of this section, the user must be able, by issuing commands, to start a task, progress from step to step, retrace or skip one or more steps, abandon a task, suspend a task,

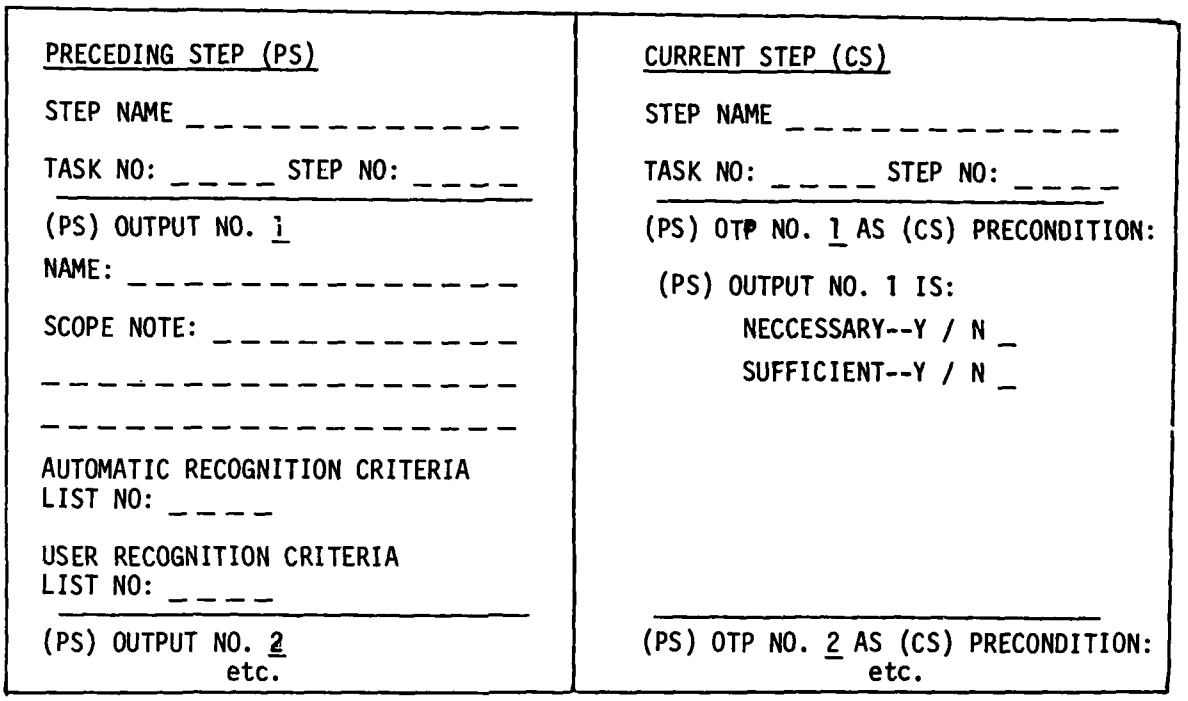


Figure 4 PROMPTING DISPLAY

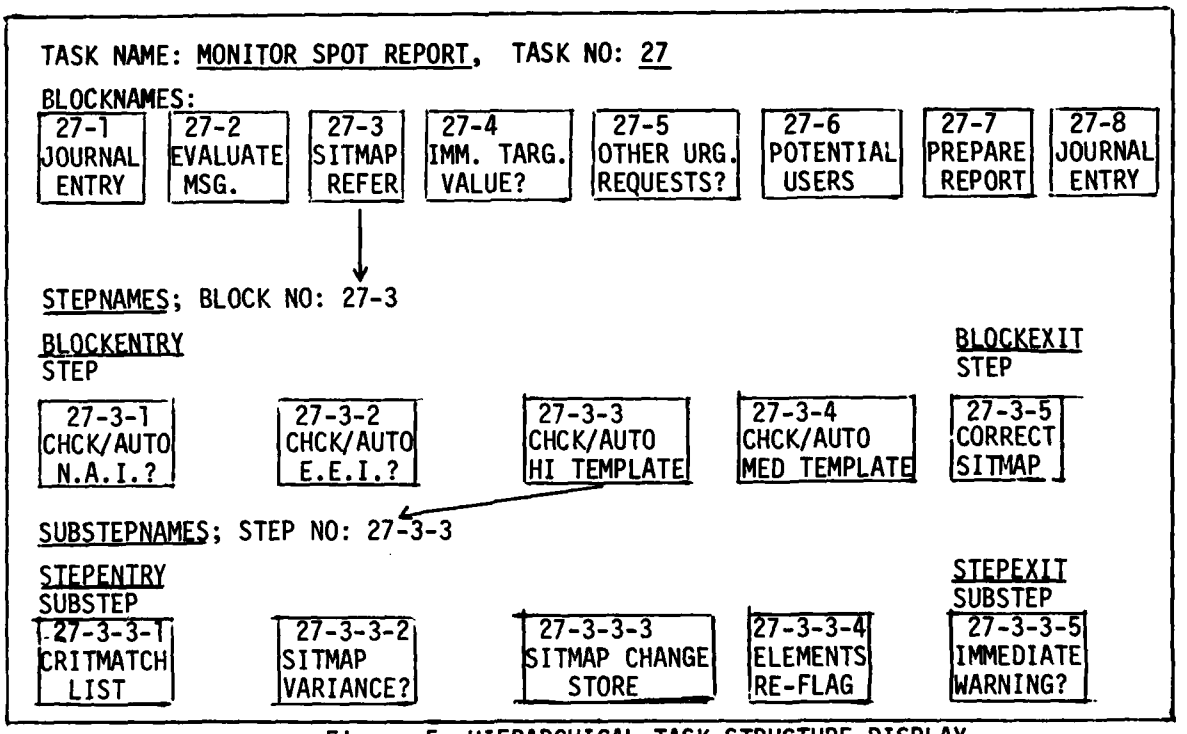


Figure 5. HIERARCHICAL TASK STRUCTURE DISPLAY

and signal completion of a task. In addition to the automatic or semi-automatic task sub-stepping support offered by the system for some types of tasks, the system should be capable of reminding the user about "set-aside" actions that he has taken, complying with the user's instructions as to conditions and timing of the reminder.

3.5 Task Interactions Control - Related to but distinct from the previous item, task decision points not infrequently branch to more than one sequence block or to more than one other task; the user must return to the branch point after completing work on one sequence and pick up the other. System provisions for prompting the user to return and complete unfinished sequences may need to be flexible as to timing, resetting, and change of priorities. Means for suspending and dropping prompted sequences must be available.

3.6 Task Overload Smoothing Control - The human operator or the system or both may receive too many inputs per unit time for the customary processing steps to be accomplished for each task input. Numbers of research studies have shown that under such circumstances the system will fall below its peak performance unless the overload condition is quickly and explicitly recognized, and a pre-planned overload mode of processing is adopted. Three methods for handling load surges are: (a) invoke a priority scheme that defers processing on certain types of inputs, (b) eliminate certain processing steps from some or all of the task sequences, and (c) shift part of the processing load to other elements of the system that are not at peak load condition. The best combination of these tactics will depend upon the circumstances.

The important point is that whatever the combination of tactics is to be, it should be:

- worked out ahead of time and tested
- instituted on the basis of pre-defined levels of input condition
- practiced ahead of time in terms of the details of the procedures to be used.

Interaction provisions must be furnished to facilitate the alternative patterns of processing, once they are established. It is also very likely that the machine system can be programmed to recognize various types of overload condition automatically on the basis of lengthening queues, dynamic buffers approaching overflow, etc. In some cases the system may be able to do a better job of overflow sensing than any combination of users, because each user normally sees only part of the total picture. Finally, given the recognition (machine or human) of a certain type of overload, the system can be programmed to flag deferrable task steps, block eliminable steps, redistribute task loads, etc. Interaction provisions for such capabilities would be required.

3.7 Task Backlog Management - This is not the same as overload smoothing control. Overload smoothing activity assesses each new input for processing decision at the moment of its arrival; backlog management activity assesses each delayed processing requirement at a later point in time, when its relevance to ongoing events may have changed. At that point, deletion of processing steps for outdated deferred inputs may occur, the main consideration usually being to achieve a rapid return to complete processing of current

inputs. Means must be provided for the user to call up inputs for deferred tasks or task steps or blocks, either by priorities, or by time, sector, etc. In some cases, deferred tasks could be automatically dropped by the system if not recalled within a specified suspense time. Interaction provisions for such capabilities would be necessary.

3.8 Task Steps Aids Management - For many task steps the system will be able to offer the user aids to his processing activities. A generalized, standardized set of provisions, applicable to all steps, for displaying the user's aid options for the step, for allowing him to indicate his choices, and to manipulate the aids, will be required. This must be designed to readily accept new aids as they are developed.

3.9 Task Process Trail & Status - For most standardized tasks that follow their abstract structure and sequence closely, and with which a user is well familiarized, no task process trail and status recording needs to be done. For such tasks the provisions of 3.3 will suffice. However, for very complex, new, or non-standardized tasks with which the analyst-user-operator is not familiarized, such a recording aid may be of significant value. Such an aid would keep a running record of what was actually done, step by step, for a task for which the user had commanded the system to do task process recording. Such an aid may also prove valuable for use by user-analysts whose duties force them to interleave several task processes at one time.

4.0 MESSAGE FILES SUPPORT PROVISIONS

4.1 In current manually-based Army Tactical Intelligence processing procedures a large proportion of messages are stored in their original form, either completely or partially. In addition, data from the messages are extracted and placed in summary data files, tables, data bases, etc. For many processing steps as depicted in the FADs, there is a requirement to refer to one or more different files of messages. On the other hand, as data from messages are extracted and placed in data base files to facilitate the systematic manipulation of some types of data, storage of the original messages or parts of messages containing the re-stored data constitutes a duplication of storage. Whether this is wasteful duplicative storage in any particular case depends upon the extent to which the two forms of storage offer the same kinds of access and processing capabilities to the user for that particular set of data. Often this is not the case. It also depends upon whether the message context of the extracted data was important or otherwise useful, and if so, whether it was preserved in the new form of storage. Often useful contextual information is lost when data are extracted from messages and stored in data bases. Thus, the manipulative advantages and summarization advantages of data base technology can in some cases bring with it the disadvantages of undersirable information loss in constraining data to fit the data base.

4.2 For the above kinds of reasons, it is the position of Operating Systems, Inc. that data base technology is unlikely to advance so rapidly that all functions currently being served by message files can be adequately

subsumed by data base applications. Even as data base technology is able to assume more and more of the message file functions in an acceptable manner -- thus eliminating the need for duplicative storage of larger and larger proportions of the input message stream -- message file provisions will still be essential for some proportion of the messages, and some data from such messages will receive essential, non-wasteful, duplicative storage in both message base and data base. Though the message files may grow smaller and smaller as system design advances are made, there will remain a requirement for one or more message bases, and a full message base handling capability will remain a necessity.

4.3 File Definition - User/system interaction provisions will be needed to allow users to define to the system various message file properties such as file name, record structure, indexing strategy and elements, required and optional elements, etc.

4.4 Message Logging, Validation - As indicated earlier in this paper, it will be assumed that a very high proportion of incoming messages for the future ATI system will be input in machine-readable format. This point is reiterated here because the current state-of-the-art for machine-based automatic input analysis for logging and validating a range of intelligence message types is very far advanced. In fact, many kinds of highly formatted alpha-numeric messages can be handled automatically in their entirety without human intervention. Other mixed messages with formatted and unformatted portions can be processed to required standards with occassional very limited human interventions to aid the machine in interpreting very unique or unlikely contents or

expressions. It will be assumed for the present purposes that the future ATI system will take full advantages of automatic input technology which will continue to advance.

4.5 There will be a requirement for interaction provisions to allow users to modify and augment the recognition and processing repertoire of such automatic procedures. Provisions to allow interventions in automatic processing will also be required. Finally, a full manual capability for doing input analysis on messages may be required for messages not received in machine-readable form, e.g., handwritten or verbal. The interaction provisions for the manual activities will be of the automatic prompting variety.

4.6 Message Indexing, EEI, NAI Flagging - The procedures for message indexing and for flagging of EEI and NAI-related information contents will involve the same technology discussed in Item 6.4 above. Much of the process will be done automatically by the system as it reads the contents of incoming messages, with the analyst/operator/user in a direct monitoring and intervention loop with the process at all times. Thus, the analyst assigned this duty will perform mostly checking and quality control functions. Paragraph 4.5 above applies equally to the present item.

4.7 Message Extraction and Rapid Routing - Once the combination of machine and human activities described in 6.6 above have identified and tagged message contents, the way is cleared for almost completely automatic electronic extraction of messages and/or parts of messages, and instantaneous dissemination of the extracts to appropriate recipient work stations, where they are queued for display. (For high priority items an alarm may have

already been forced automatically during Step 4.4.) Thus, current message dissemination lag times can be substantially reduced. Automatic message dissemination will operate for both routine "standing" dissemination requirements as well as for unique, one-shot, transitory requirements. For most routine requirements both the recognition criteria and the routing address instructions will be fully machine interpretable and will require only sporadic intervention aid from the personnel assigned this duty. For unique, short-notice requirements the automatic procedures will be applicable in some cases and not in others: no difficulties would be experienced with automatic routing instructions, but in some instances there might be insufficient time to adequately describe the recognition criteria to the system. In this case the recognition criteria are quickly keyed into the system where they are stored as a rapidly callable display by input monitor personnel, and keyterms from the display are deposited in an alert list at the monitor station. As the human monitor scans the "not assignable" stream display from the automatic input analysis procedures, he can pick off such unique materials and invoke the automatic routing capabilities to have them immediately sent to the proper places. Paragraph 4.6 above applies also to the present item.

4.8 At this point we digress momentarily to note that for all such interaction provisions for instructing the system, menu and fill-in-blank displays will be used, which greatly ease the interactive task for the human operator. Such displays will be invoked either automatically, or by depression of one or two MACRO buttons of the kinds depicted in Figure 1.

4.9 In moving toward a maximum of advantage to be gained from the mixed initiative approach being investigated as part of the MIQSTURE project, "teachable" system components may also be considered. In such components, the system offers an analytic solution to (for example) an input analysis of a message, and the human input processing monitor overrides the machine solution, and substitutes one of his own. A teachable component will automatically set aside and store both its mistake and the corrective action, and will make them available for deeper consideration at the user's convenience. Thus, for example, an ongoing analysis of machine system errors comprising "false negative" and "false positive" recognitions for EEI and NAI can be conducted, and the results used for the recognition criteria correction cycle. This cycle serves to add synonyms, refine ranging expressions, tighten and clarify boolean relations, etc. The system can aid in these processes by providing the human correction cycle monitor with automatic analyses of the differences between erring criteria it has used and corrective criteria inserted "on the fly" by (for example) input processing monitors.

4.10 Yet another kind of "machine initiative" element that is feasible is the bookkeeping activities that can be automatically accomplished in support of extraction and routing dissemination functions. If desired, parts of messages that are routed to recipients can have automatically attached to them the notation that they are only part of a message, and the referrent message number. For individual or unit function dissemination profiles, selection criteria with high priority can be noted to the system, and resulting messages to the recipient(s) can be forced to the head of their display queues by alarms or other appropriate means. Under some circumstances,

receiving back acknowledgment of receipt of a message extract may be important to the originator or intermediate analysis section; automatic "mailbox" features can be built into the system for this purpose.

4.11 Security and Access Classification - In conjunction with input processing and dissemination routing there is the simultaneous requirement for assigning security and access classifications to incoming information. To the extent that classification assignments are based on machine-interpretable criteria such as certain sources and certain explicit contents, those assignments can be done automatically, with human quality-control monitoring. The system also can supply many aids to the user for manual classification assignment activities. For example, prompts and aids consisting of automatic display of special standing orders, late changes, etc. can be made available, as well as callable displays of classification criteria refreshers. Security control logging number management can be handled through semi-automatic means. The interaction provisions associated with all the above types of arrangements would be part of a MIQSTURE.

4.12 Message Contents Storage Analysis - After the activities described in the preceding five items have been executed with respect to an incoming message, the question arises of how much or what parts of the message should be stored, where and in what form, and for approximately how long? Basically, all such decisions are premised on a prediction of the probable useful life of the various types of contents of a message. In turn, most kinds of usefulness of information for tactical purposes is dependent upon the estimated data validity time decay rate. Decisions regarding dissemination priorities and

storage strategies depend upon estimated durability of the validity of data contents. Data durability estimates are also important for planning for semi-automatic purging activities conducted on the contents of message and data base files. Both manual and semi-automatic means may be employed to assign data durability estimate values to message contents, and interaction provisions must be available to accomodate both approaches. In addition, the system may be expected to furnish some system initiated services that can aid in data durability and use-life estimation activities. Callable displays can contain use life distributions per data type, situation, and use made, based on usage statistics gathered by the system from querying and printout activities. The system may also feature special forced alert displays to warn of current exceptions to standard estimation practices.

4.13 At this juncture of the message input processing, decisions must be made concerning the storage of message contents. For each unit of contents the basic decision is whether to store it in message file(s), the data base file(s) or in both. An ancillary decision is how long the data are to be stored in each type of repository. The provisions for entering data into the message files will be discussed in the item following the present one. The provisions for storage of data in data base files will be treated starting in Item 5.4 of the following section.

4.14 Message File Maintenance and Access - The term maintenance includes the functions of file load, update, and purge; access includes search and retrieval activities. Interactive provisions for maintenance and access comprise the main body of what is usually thought of as a query language.

The MIQSTURE interactive language contains query subsets for message file functions as well as for data base functions (which are treated in the next section). For the message files, storing a message in its entirety would often be as simple as viewing the message on the display and pushing "send to storage" and "enter file x (y,z)" buttons. Storing parts of messages would be only slightly more complicated, and would involve using the text edit and compose capabilities in conjunction with the storage command buttons. Updating a stored message (a low-frequency event) involves retrieving and displaying the message, altering it via the text edit and composition capabilities, and again releasing it for storage, while the earlier copy is automatically purged from storage. Purging of messages can be done on a class-batch basis or individually. In each case, search formulations are made to identify the to-be-purged record(s), the retrieved records may (optional) be examined to verify appropriateness of purging, and a single-button delete command can then be issued against that purge batch, whether it contains one or many records. Moving records to secondary storage would be handled similarly.

4.15 As suggested by Figure 1, data entry, updating, and purge formats for a particular type of file can each be made callable by a single MACRO button action. The system reponds by presenting menu and fill-in formats that the user employs by moving the display cursor and taking a combination of button actions and keystrokes. Data entry error correction and validation problems can be minimized by button-oriented language techniques. Automatic validation routines can check that purely numeric fields contain no alphabetical characters, etc., and this can be accomplished interactively, so that the user can correct detected errors immediately. Block 2 of Figure 1 shows some of

the button elements associated with the search formulation capability that would be used in searching message files. In the button-oriented language, elements of a message search formulation can be entered in any order, so long as the sequence of data within each element is in proper order. For example, after depressing the "from time" button, the user would use the keyboard to the right to enter up to eight decimal digits to represent minute, hour, day and month for the beginning time bound for which he wished the search to be carried out. If he only wished to specify hour and day, he would enter null characters for the first two and last two keystrokes, and the system would use the beginning of the specified hour as the mark, and would assume by default that the user wished the present month to be employed. Location name MACROS can be entered as single search elements into a query by a button action. By pressing the "point coordinate" or "polygon coordinate" buttons, the system can be prepared to properly interpret the numerical insertions which follow, and can offer the appropriate fill-in or menu capabilities on the display. If the message receipt processing station, sensor type, map sector involved, or map grid are known for a message to be retrieved, these also can be stipulated by pressing appropriate buttons and following them with the specifications data. Of course, the file to be searched can be specified by pressing a file name button, or more than one. The same logic of interaction provisions would apply to the use of other search capabilities such as point/radius specifications, content indexed terms, etc.

4.16 Cross-Referencing - Cross-referencing functions are a very important general feature of data entry as well as retrieval operations, and need to be considered separately. Cross-references may be needed between any of three main kinds of entities: Messages, Data Records, and Display Elements. Messages may require cross-references to other messages in the message base. Data records may require cross-references to their source message(s), or to other data records. Display elements showing summaries of information contained in the message file(s) and/or the data base file(s) may require cross-reference to messages, data records, or other display elements. User/system interaction provisions must be available to allow the user to easily establish as well as delete any of the above types of cross-reference relations. In addition, the system should provide automatic provisions that can be invoked by the user to cause the system to establish certain types of cross-reference pointers without further intervention from the user, on a system initiated basis.

5.0 DATA BASE(S) SUPPORT PROVISIONS

5.1 As discussed earlier, a considerable proportion of data base contents will derive from message contents. Some proportion of the overall flow of message contents entering the system will probably not be stored in either the message base or the data base files, having been filtered out or used as highly persishable combat information.

5.2 File Definitions - Many aspects of ENSIT and FRENSIT data structures lend themselves to a partially integrated data base approach. This is especially true of GOB data during periods of low-intensity activities. During such periods there is time to impose a fine classificatory structure on input data, and the result is a well-structured data base representing complex organizational entities with many associated properties, and all with a minimum of data storage redundancy. As the pace of activities quickens and is compounded by enemy deception, the completeness of information in messages tends to be reduced and thus the bases for well-founded classificatory decisions are diminished. (Classificatory activities per se, as well as system-initiated means for shunting message contents into hold files, are discussed later in this section.) Thus high rates of activity may require the use of less refined "default" data base structures that better match the degree of precision obtainable from the data input stream represented by messages. At the same time, the degree of integration of the data base is likely to be reduced, with the data coming to reside in a series of separate files, much as the manual files of the present tactical intelligence processing

activities are organized. The main implications of the above are that a family of file structures gauged to different levels of data fineness and completeness may need to be devised, in which the different file levels have continuity capabilities built into them. In this fashion data storage operations can, for example, move from high refinement periods, through very "gross" storage periods, and back to higher refinement periods without loss of information continuity or the necessity to switch to different files covering sequential periods of time. In this light, the File Definition capabilities of the system are indeed a major design issue. The file definition and file utilization features of the MIQSTURE interactive language will need to be based on sound assumptions with respect to the above issues.

5.3 File Structures for Secondary Characteristics of Data - In addition to storing data values per se, other characteristics associated with a datum will often need to be stored; e.g., classification, dating of source message, source reliability, data time durability estimates, alternative or approximative indexing, cross-reference pointers, etc. One important secondary characteristic of data is its sampling error as a function of size of the data aggregate. An example is running strength estimates for the enemy on the basis of observed attrition during high-intensity combat. For enemy losses (troops, material, etc.) identified with a certain geographical area, confidence that the losses can be included within the domain of a certain enemy Field Army is high, within a certain enemy Corps the confidence is lower, and assigning losses to a certain Division or Brigade or Battalion progressively lower. In essence, for organizationally aggregated data the probable error

of estimate becomes a decreasingly small fraction of the estimate at higher and higher levels of the enemy organization. Since information on concentration of force is important at all levels of friendly command, and equally so at the small unit level of engagement, this "gradient of uncertainty" in the data base aggregate values needs to be reflected in data base secondary characteristics values. The ability to accurately reflect the parameter of uncertainty of data values as a function of the range of decisions for which the data may have implications is especially important for lower levels of command. For example, taking an average of two identical sightings that show a variation in reported position may have no implications at Brigade level, but may make a good deal of difference at Battalion or Company level. For the Company Commander, it may be important to preserve the fact of two possible positions for the sighting. Thus, it may be important in some cases to provide secondary probability values associated with some kinds of equivocal or conflicting data entering the data base, with cross-reference pointers as well so that the equivocal values can be identified and considered as a unit for certain decision-making purposes. Especially for data base use (but sometimes for message file use) this type of cross-reference pointer system may require features that invoke automatic "see" and "use for" notifications to be included in retrieval and/or report program outputs when such operations are carried out on the data. Also, class purging activities may block such linked values (with notification to user) unless specific override instructions are given to purge linked values. Ordinary cross-reference patterns between message base, data base, and display-base may thus be augmented by secondary cross-references for uncertain or incomplete data,

the entire cross-reference network being available for the more detailed analyses sometimes required by lower level commanders. A MIQSTURE interactive language will include provisions for handling the types of structures described above.

5.4 Data Base Input Operations - In these operations, data (taken almost exclusively from message contents) is deposited in one or more structures in a data base. The two closely-linked sub-operations involved are message analysis followed by data storage decision and definition.

5.5 Message analysis is accomplished in a mixed initiative mode, with both user and system-initiated message analysis aids operating on the corpus of the message in its machine-interpretable form. Message analysis for data base storage is an activity that grows naturally as a next step to the message logging, indexing, and storage activities described earlier. Message analysis uses the results of the earlier steps as preparation for making the data base storage decisions and for constructing the data base storage definitions required for entering data into the data base. Since message analysis is the main focus of processes by which data about the enemy is put into (stored) in forms or structures that relate the data to other data, message analysis is one of the main steps involved in transforming data to information. That is, the data are placed in the data base to facilitate their usage in context. Message analysis is the intelligence processing stage perhaps most heavily concerned with the employment of "discovery operators" as defined by P. W. Keen in his paper on the intelligence cycle. In effect, unless the data depicting new and changed circumstances requiring new adaptive responses is detected and properly handled at the point of message analysis, it is for the most part lost to system-wide use.

5.6 The user-initiated contributions to message analysis need little elaboration here beyond a brief enumeration. The user brings his entire range of experience and his current understanding of the battle situation to bear in message analysis. In addition, he brings his knowledge of the Army Tactical Intelligence System resources as they can aid in message analysis; the use of the message file(s) and the use of various data base service functions. The user doing message analysis must meet the challenge of fitting the data from each new message into the overall context or "picture" of developing affairs.

5.7 The system-initiated contributions to message analysis activities are most often made in interactive cooperation with the user, and consist of support for content recognition activities. Such system-initiated user support depends upon programmed-in abilities to recognize certain conditions in messages. The aid provided the user in content recognition support is in the form of suggestions offered by the system as to where certain portions of messages might be stored in the data base. As described earlier, the system's interpretation of a message for purposes of message base indexing and EEI and NAI dissemination functions requires such content recognition capabilities. These capabilities are augmented for the message analysis activities related to data base entry; the system reads the message, matches or identifies various content terms, phrases, and special structures, and provides the user an exhaustive list of suggestions as to where each datum might be stored in the data base files. (Making such decisions is part of the "transform operation" investigated by Baker et al in BESRL TRN 212,

August 1969* and was properly identified as an area of special cognitive complexity for the users of automated data base systems.) The operator is free to follow one or more of the suggestions, and/or to correct the machine with feedback about its suggestions by keying in, in effect, counter-suggestions. Such feedback is stored by the system and can be used to improve the system's ability to make pertinent suggestions.

5.8 Data Continuity Analysis - In addition to the linguistic-based types of analyses used for content recognition as outlined above, there is another class of analyses for some types of contents that follows on the heels of the linguistic analysis. These analyses involve the functions of data correlation, of data fusion, and of alerting. For each of these functions, data from the message is compared with data already deposited in the data base or in a special file, to decide: (a) whether the new data belongs to particular data sets (because it is data about the "same" entity as is represented by the other data) and (b) if so, how it is to be incorporated into the existing data set(s). Research on automatic correlation and fusion techniques is well advanced, (e.g., the investigations of use of expanded Munkre's algorithm for making correlation and fusion decisions). Correlation processes assess data from the same or different sources and with observation times within a prescribed time envelope, to decide whether or not the data represent the same events, objects, etc. Fusion processes select or construct a data value to

---

* The Transform Operation in TOS: Assessment of the Human Component. Baker, Mace, and McKendry. Technical Research Note 212, U. S. Army Behavioral Science Research Laboratory, August 1969.

represent more than one separate data values (usually from different sources) that have been determined to represent the same event, object, etc. Alerting processes may compare a "fused" datum with the same "identity type" of datum, perhaps fused earlier, and/or compare it with other types of data, fused or otherwise, to detect a significant pattern of events. This pattern of possible events has been pre-specified in the form of a template, consisting of either a sequence of selected events with approximate timings, or else a prescribed statistical rate-of change or values limits. The template, stored in the system, provides the basis by which the system can assess incoming data values. (These are discussed more fully in Section 6.0.) The analysis functions described above, both user and system initiated, will require the kind of interactive provisions to be embodied in MIQSTURE.

5.9 Data Storage Definitions - Following immediately on input analysis activities are those providing the data storage definitions that must accompany storage of each piece of data. The preceding discussions have already suggested that a certain amount of such decision-making and specification of storage definitions can be done automatically by the system, under the monitorship of the user. However, there will be many instances in which the user must make some of the data entry decisions that select the data storage format(s). The early study of Baker et al demonstrated that this particular set of processing steps pose considerable cognitive difficulties for users. The difficulty level of such manual steps is likely to be greater in a mixed-initiative system, since the high-frequency, well-understood message contents and data base formats will be handled mostly by system-initiated procedures,

leaving the more rare and more complex message instances for the manual efforts of the user. The user's main problems for such comparatively infrequent tasks are related mostly to insufficient practice because of their low frequency of occurence. Ordinarily, the user can recognize the right alternative if it is presented to him, but he cannot recall it without aid. This observation constitutes the main guide in providing user/system interactive aids for the recognition function. The user's easily recallable behaviors in such situations are used as inputs to the system, which provides "mapped-to" displays of alternative possible data formats from which the user recognizes the correct one(s). Again, MIQSTURE interactive provisions must support this range of functions.

5.10 Data Base Structural Monitoring, whether performed by user-initiated or system-initiated functions, operates on data after it has been stored in the data base. Although monitoring activities can be oriented toward detecting individual data instances fitting certain criteria, ordinarily it is used in a class-batch mode to identify sets of data records fitting certain criteria. In structural monitoring statistics may be gathered from such sets, or certain automatic actions taken upon records in the identified sets, or both. Among the most commonly implemented criteria for monitoring operations are data use-life estimates and/or durability estimates, which are involved in semi-automatic purging activities. Two other types of criteria that can be used to gather statistics on the current level of data-gathering certainty are: percentage of "equivocal" data identification assignments, and, percentage of missing-data flags in the data base or in one of its logical file(s). Algorithms for detecting redundant data of various kinds can be devised and

used to control the level of redundancy to a cost/effective level. Ongoing statistical analyses of various categories of data can also make it possible to identify both average and extreme values for data, and to identify various instances of each when needed. Finally, given certain "models" for organizational (TOE, composition, etc.) aspects of the enemy, criteria for signalling conflict or disproportion in data aggregates representing the enemy can be specified and made to serve for automatic monitoring functions, (such as those tied in with the system capabilities for keeping running strength estimates in relation to reports of enemy losses). Data structure monitoring functions will require MIQSTURE language interaction provisions to support them.

5.11 Data Base Retrieval and Report Generation - These functions consist of the familiar ones normally associated with a Data Management System application, and will therefore be treated only briefly for present pruposes; MIQSTURE provisions will be needed to specify data sets and the various data processing operations to be carried out on those sets, and for specifying formats for display of results. Among the many facets that must be considered are: multi-file searching; pre-defined queries; range searching; synonym and phonetics (sound-alike) control; data element dictionary access; string matching; repeating groups specifications; standard general arithmetic and statistics functions; sorting and cross-sorting for analysis and for output, and report generation format controls. The MACRO capability of MIQSTURE must be available for such functions, so that one or two-button actions suffice to allow user initiation of common data base functions.

6.0 SPECIALIZED SUPPORT PROVISIONS

6.1 In distinction to support provisions for the display, task management, message base and data base subsystems, all of which are highly generic to most intelligence data processing tasks supported by the system, the support provisions for specialized functions may serve only a part of one task, or of a few tasks. Such specialized functions commonly provide both computational processing and information-supply services in support of the making of information decisions of a frequently recurring type. (An information decision assigns a context of interpretation, and thus, a selected meaning, to a datum.) The computational processing and information services supplied may be combined with new incoming or stored real-world data to produce new information relevant to decision making. From the point of view of designing an interactive language such as MIQSTURE, several aspects of such specialized functions are especially noteworthy:

- Specialized support provisions may often require highly specialized technical knowledge for their use, and will have a comparatively small population of users.

- Although the functions they perform may often be tied intimately to the use of "real-world" data, the procedures themselves may be separable from such data, and may be able to operate in a "simulation" mode; i.e., they may be tools for conceptualization.

- The pattern of their use more frequently involves deviations or detours from an expected sequence or pattern of actions then would be the case for the more generic provisions; they may sometimes need to be used together in unanticipated combinations.

- The functions in some cases are highly sensitive to change, and may need to be modified or augmented to a degree that would be less often necessary for the more generic functions.

6.2 The specialized provisions may consist of pure versions or combinations of three main types of elements: Handbook Information Supply, Computational Support, and Framework or Templating Services. Regarding the first of these, tables from technical references, charts, etc. that are not appropriate for "whiz wheels" but may frequently be needed quickly, can be stored for rapid display and automatic interfacing with computational processes. MIQSTURE elements for handling such functions include provisions for storage definition, loading and maintenance, indexing and calling, retrieval and display, creation of catalogs and/or tables of contents, creation of data interfaces with various computational routines, and means for initiating the passing of data to computational routines.

6.3 Computational support provisions consist of frequently used procedures that would ordinarily require a laborious sequence of actions on a conventional calculator. These may be pre-programmed in a fill in the blank mode and be manipulable through MACRO-button actions. In addition to eliminating procedural steps necessary with hand-held calculators, such aids also have direct access to reference data and to real-world data stored in the data

base, thus eliminating many data entry keystrokes or button actions and sharply reducing the task time requirements as well as the probabilities for errors in supplying data to the computations. A MIQSTURE interaction language would support the interactive use of such functions, including that of the automatic storage and/or display of the products of the computations, or their routing as input to yet other computational procedures.

6.4 The framework or templating services are presented last because of their rapidly growing importance and because of their complexity; a more extended discussion is appropriate. The framework/templating concept is used here in its broadest sense, to include machine-stored patterns for support of correlation, fusion, and alerting functions as defined in Section 7.8 Data Continuity Analysis. As suggested earlier, such functions share in common the comparison of incoming data values with other incoming values and/or with already stored values, this comparison being made within a pre-established framework, and for the purpose of making or supporting an information decision. As indicated earlier, an information decision assigns a context of interpretation, and thus a meaning, to a datum. Automatic data correlation routines may signal their decision by assigning (or not assigning) an incoming datum to a certain entity or "track" represented by other data acquired earlier. Automatic data fusion routines may signal their decision by choosing between data, to select one to represent an entity signalled by several data values, or by calculating a single new value to represent several. Automatic alerting functions may be of many types and are considerably more complex than correlation or fusion routines. Here, machine or transparency-stored patterns of potential data values

are compared with the incoming data stream. The stored patterns represent the predicted symptons (data-emission signatures) of pre-analyzed patterns of likely enemy activities. An automatic alerting function may signal its decision by showing comparative degrees of match between an incoming pattern of data and several such pre-stored patterns. The patterns may include maneuver models, electromagnetic emissions models, expected levels of specified activity-sensing data, etc. By thus fitting sparse data into appropriate interpretive contexts, powerful inferences about real-world events and processes can be made.

6.5 MIQSTURE interaction language provisions for framework/templating services must include those for allowing users to construct frames or templates, to define data interfaces for the templates, to specify display and other outputs from the use of the templates, to create storage and retrieval for the templates that are created, to facilitate the user-system intractions required for consistency and error checking in the construction and maintenance of templates, and to call the templating functions into action at appropriate times. As compared to display, task control, message base, data base, and configuration management functions, the templating function promises to be of equal or greater importance; the interpretation of data (i.e., the transformation of data into problem-solving oriented information) is the single one of the functions just named for which it is hard to imagine an excess of system capability. That is, each of the functions provides a necessary component of system services, without which the system would be likely to fail. But it is possible to imagine system designs for which excess resources are

expended on display, task control, message base, data base, or configuration management functions. On the other hand, how could a system have too much capability for interpretation of whatever data it processed?

6.6 The system importance of the templating functions can be depicted from two additional points of view. First, the templates comprise, for the most part, the system-incorporated definitions of the top-down -- oriented information product requirements levied by the commander. Second, the template-controlled processing functions are central to much of the system's functioning. Because of the importance of the templating activities for the system as a whole, it seems very likely that for most system users, the image of the future system will be pervasively centered on the top-down, decision-oriented, information products viewpoint that is so clearly embodied in the templating concepts. This has very important implications for the design of a MIQSTURE interaction language; experience shows that users of online systems are greatly aided in adapting to an effective use of the system when they are able to attain a conceptually consistent, organized view of the total system. Experience also shows that the main channel through which the user obtains his impressions of the system as parts and as a whole is through the interactive language. In an absence of a "semanticization" of the system (through the interaction language) that creates a consistent whole in users' perceptions, new users will often imagine or invent relations between parts of the repertoire that are based on an assumed organization of the system. If their assumed organization is at very great variance from the actual organization, the users will have difficulty in adapting to using the system.

6.7 The problem of accurate semanticization of the system through its interaction language is not mainly concerned with hardware and software design as they affect internal processing. It is concerned with inputs, outputs, and perceived flows of information and processes through the system. It is also concerned with the perception of interdependencies, sequencing, and priorities of the various aspects of system functioning.

7.0 SYSTEM CONFIGURATION/DEGRADATION CONTROL SUPPORT PROVISIONS

7.1 Manual Backup Support Maintenance - Communication outages and equipment malfunctions or damage may necessitate switching to alternate communication channels, and may require manual interfacing between activities normally interfaced through computers. At certain points in the network, computer processing may be interrupted, and computer-mediated access to message bases and/or data bases may be lost. Provisions for manual continuation of certain crucial data base access functions must therefore be able to be instituted without delay. At the same time, the possible continued use of partial automated capabilities must be able to be decided on the basis of rapid analysis of remaining system capabilities, interference factors, etc. As preparation for manual backup functions, hard-copy production for selected data base contents and access procedures must be able to be invoked and rapidly performed during periods when system degradation may be threatened. MIQSTURE provisions for accomplishing such backup preparation and for utilizing the results will be necessary.

7.2 Auto-Backup File Reassignment - Equipment may automatically switch to backup files in the event of sudden communications or other kinds of outages that destroy first-choice file availability. Commands and displays in MIQSTURE must be available to configure such automatic backup facilities.

7.3 System Recovery Sequencing - Return to conventional operation will require sequences of file reloading, switching of communication connections, etc. that may need to be performed in a certain order that is dependent upon

the particular nature of the system outage that was experienced. At the same time, such system recovery must be able to be performed reliably by a significantly large proportion of the personnel who will utilize the system, many of whom will not have extensive technical training in computer operation, programming, or system set-up that could be involved in such activities. Therefore MIQSTURE language interactive provisions should be developed to bring such actions within the range of the user personnel who will be operating the system.

APPENDIX B

MIQSTURE Language Element Definitions

LIST-TYPE RECORD NORMAL QUERIES

USER/SYSTEM INTERACTIONS CONTROL ELEMENTS (21)

Short Version	Long Version	Explanation
T__;	TRANSACTION__ ;	(System message) Transaction number is assigned in blank by system; (left-most element in the Interaction Cueing Message.)
C?	COMMAND?	(System message) Command request message; says user is allowed to enter a command(s).
QF__	QUERY FORMULATION__	(System message) Query formulation number is assigned in blank by system. Unsuccessful queries that yield no records (NONE) do not cause query count to be incremented.
"	"	(Command Discriminator Character) Signals to system that characters following are first word in a command sequence (" is arbitrary).
"CO	"COMMANDS	(Command Name) The command "CO "EX causes a display of command names, organized by function areas. Cursor action selection on the names causes explanations similar to those presented here.
"EX	"EXECUTE	(Command Name) Used only at the end of an entry sequence to signal the system to take action on the command or query formulation preceding it.
"HO	"HOLD	(Command Name) Used to hold a query formulation without "EXecuting it. The query increments the query count, but can not be executed as a value entry in a later query. Convenient for formulating queries for "MATCHPICK and "DUPSEARCH.

Short Version	Long Version	Explanation
"ER	"ERASE	(Command Name) Used to "type over" a space into a filled character position on the display. Coordinated with the cursor. Can be used in conjunction with the cyclic forward and cyclic reverse on the cursor. Same action available on keyboard space-bar, forward only
"CA "EX	"CANCEL "EXECUTE	(Command Name) Used to erase a transaction from the display, either before or after the transaction has been executed. Entry of a RECNAME has same effect but also qualifies following query. If no erasure is done, entry is eventually forced off top of scroll.
"ES__	"ESCAPE__	(Command Name) Blank contains escape code; meaning is conventional meaning of escape code.
"TR	"TRANSACTION__	(Command Name) The command "TR__ with a number in the blank causes the record of the specified transaction to appear in the transaction history window of the CRT display. Scrolling or paging are available in this window.
"VE__	"VERSION__	(Command Name) Used to establish the short or long versions of system message to be delivered to the work station; (using the command has no effect on other work stations). The start-up default value for all messages is set at long, unless automatic VERSION-setting is in effect through the USERID code (mechanisms for this function are discussed elsewhere; the "VE__ command is used to control both automatic setting and manual setting). The entry "VE SH ALL "EX will cause all system messages to appear in the short form for that interactive session (substituting the parameter name LO ALL will cause long form of all messages). The entry "VE SH "EX entered immediately after delivery of a system message will cause that message to be delivered henceforth during that session in short form (substituting LO will provide long form for that message).

Short Version	Long Version	Explanation
"AU	"AUTOSET	(Command Name) Used in conjunction with certain other command names to store the current settings for those commands, and automatically set them as an opening exercise at LOGIN for that USERID. Thus the command "AU "VE "FI "EX will cause the work station to be configured to the VERSION pattern and FILE attachment in effect at the time of the command, whenever that user does a LOGIN.
"WH	"WHAT?	(Command Name) If entered immediately after a system message delivered in short form, will cause the message to be repeated in long form, but will not cause the VERSION setting for that message to be altered. If entered under any other circumstances, will be interpreted as a "HELP command.
"HE	"HELP	(Command Name) The response to the "HELP command is, wherever possible, state-sensitive. If the "RUNTASK command is not in effect (see later command definitions) the program will scan the record of the last three transactions, noting their types, and on the basis of the combination will select a list of possible circumstances descriptions and present them to the user. The user may select one or more of the circumstances (via cursor and "SELECT button), and receive back associated solution suggestions. If the "RUNTASK command is in effect, the system is able to sense the task state, and both the circumstance descriptions and solution suggestions will be much more pointed. Included as part of some suggestions will be the advice to review certain discussions through the "DISCUSS command.

Short Version	Long Version	Explanation
"DI	"DISCUSS	(Command Name) The discussion contents are an online mirror of the User operating manuals. Each topic is organized hierarchically, from overview and common refresh points, down through more detail and examples. The user scrolls down through the discussion, at the end of which is the system message MORE? Y/N The command "DI "EX will cause a display of the discussion topic names, which can be selected by cursor. Insertion of a topic name between "DI and "EX will also cause the named discussion to start.
"NA	"NAMESTREAM__	(Command Name) An interaction history and progress is arranged so that it takes place in a stream. The work station capabilities can be hooked to only one stream at a time, but can interleave interactions on several different streams from one work station, with each stream maintaining complete independence. The program automatically labels streams, A, B, C, etc., and the label of the currently in-progress-on-work-station stream is displayed along with the stream name, if one exists, continuously in the interaction status display window. A "NAMESTREAM command name followed by a series of up to 30 characters and spaces will cause the data portion of the entry to be stored in a stream name table opposite the alphabetic label of the stream within which the command is taken. A repeat of the command will cause the earlier "name" to be overwritten by the new one. Use of the command is optional.
"SW	"SWITCHSTREAM	(Command Name) The command "SW "EX will cause the program to select the alphabetically earliest stream label (A,B, etc.) not in use, assign it to a newly established stream, switch the work station components to attachment to the new stream and generate the interaction cueing message: T1;C? (cont'd)

Short Version	Long Version	Explanation
		The command "SW__ with an alphabetic character in the blank will, (if that character labels an already established stream), switch the work station attachment to the specified stream. The processing activity on the previous stream will continue until outstanding commands on that stream are satisfied, and the results will be stored. Interaction on the switched-to stream will pick up at the point at which activities ceased when it was previously left.
"CL	"CLEARSTREAM	(Command Name) This command stops all action on the stream within which the command is taken and erases the interaction history file for the stream. Before taking this action it seeks a confirmatory message: CLEARSTREAM Y/N?. It then takes the actions if signalled Y, and generates the interaction cueing message T1;C?
"ST	"STREAMS?	(Command Name) Generates a display of stream labels and names (if any) currently not cleared.
"SI	"SIGNOFF	(Command Name) Generates the message: CLOSING EXERCISES? Y/N The N response causes clearstream action on all uncleared streams, and signs the workstation off the system. A Y response generates display options discussed elsewhere (including a "ST display).

LIST-TYPE RECORD NORMAL QUERIES

DATA STORAGE LOCATION/FORMAT SPECIFIER ELEMENTS (3)

Short Version	Long Version	Explanation
"FI	"FILE	(Command Name) Used preceding the name of a file the user wishes to select for actions to follow, (such as search, or manual loading or maintenance). The user can be "connected" to only one file at a time, and the "FILE command remains in effect until superseded by another "FILE command, or by an "ESCAPE command.
RECNAME	RECNAME	(Display heading) in Interaction Status Display window, showing the name of the record type (if any) that the user has currently specified for actions to follow. Since a record type name cannot be specified unless a file name is in effect, the name of a record type is inserted without the command discrimination character, and without any preceding command name. It may be inserted as part of a "FILE command sequence (see related scenario) either alone or in conjunction with the "BLANK or "CHECKLIST commands.
(XXXX)	(XXXXXXXXXXXX)	(Category Field Identifier) Used to qualify query value entries as to which category field they are intended to be found in. The explicit version is the category field name, consisting of up to three words of up to 15 characters each. The symbolic version is the four-character category field name abbreviation. Either name or abbreviation are entered surrounded by parenthesis. The rules for use of CFI qualifiers are described in the related scenario.

LIST-TYPE RECORD NORMAL QUERIES

QUERY FORMULATION SPECIFIER ELEMENTS (18)

Short Version	Long Version	Explanation
"QU	"QUERY	(Command Name) Informs system that user entry actions to follow constitute a query formulation. This is the single command in MIQSTURE that may be ignored by default. That is, if any keyed-in characters are presented to the system without being preceded by the command discriminator character, they are assumed to be a query formulation. (The exception to this rule is if the program has posed a required-response message to the user, e.g., Y/N?, etc.). (Since in the command context repertoire of the system one context can, logically, remain unnamed for interaction purposes, the context chosen should be the one most frequently used across all type of tasks; this is the query formulation context.)
"BL	"BLANK	(Command Name) Used preceding a record type name to evoke a fill-in display format for the record type.
"TA	"TAB	(Command Name) Conventional TAB function. Is used to "step" through fill-in formats for records and tables.
"CH	"CHECKLIST	(Command Name) Used preceding a record type name to evoke a display format of the record type similar to the fill-in display, but not having fill-in capabilities. Cursor is positioned in an unprotected key-in partition of display window containing record format.
"CO	"CONSTANT	(Command Name) Used preceding a "BL or "CH command name to establish the follower command as "standing" for the entire interaction session, or until retracted by the "CANCEL command. In the constant "standing" condition, the fill-in or see display is automatically evoked by insertion of a record type name.

Short Version	Long Version	Explanation
"CA__"EX	"CANCEL___"EX	(Command Name) Used just like "CO, purpose explained in preceding item.
--NREC	NO RECORDS FOR	(System message) No records found for the terms designated in message; short version in full would be: Tx;QFy--NREC XXXX( ), where x is transaction number, y is query formulation number, XXXX is term value and ( ) contains category field identifier abbreviation.
--NONE	NONE	(System message) No record found fitting any terms in designated query; full short version is: Tx;QFy--NONE
___	___RECORDS	(System Message) ___records found satisfying query. Full short version is: Tx;QFy--__
QF__MOD TO SEP	QUERY FORMULATION__ MODIFY TO SEPARATOR	(Query Modification Entry) Used to modify a partially unsuccessful query formulation. The QF number is entered in the blank; QFxx MOD XXX( ) TO YXX( ). The entry is completed with "EX. If a second value is to be modified in the same entry, a SEP replaces the "EX, and the two values separated by a TO as before, are entered. This can continue for as many value changes as necessary. Changing a value to a blank (one or more character spaces) results in its deletion from the query, with the following rule regarding Operators: Operators that apply to the deleted value (i.e., all operators preceding the value, if there is one) are auto-deleted along with the deleted data value. QF count is not incremented.

Short Version	Long Version	Explanation
QF__ AND, OR, BNOT	QUERY FORMULATION__ AND, OR, BUTNOT	(Query Nesting Entry) Used to incorporate an earlier Query Formulation as a single value entry in a later Query Formulation. The QF__ must precede all other value entries in the new query formulation. The blank contains the QF number of the Query Formulation to be incorporated. QF count is incremented.
"SA_ _	"SAVEQUERY__ __ __	(Command Name) Query Formulation numbers entered after the command name separated by spaces will specify queries, the outputs (identified lists of records) of which will be saved. All other query outputs will be erased, and the saved queries will be renumbered starting with number (1) in the order in which they occurred in the transaction log. In addition, the query transactions will be recopied into the transaction log, with renumbered query (1) in the transaction number following the transaction carrying the "SAVEQUERY command, etc. The saved queries can be used as components of later queries; the erased queries cannot. The formulations for the erased queries are still available in the transaction history file and can be viewed through the transaction history window. If saved queries refer back (contain as single values) earlier queries not specified for save in the command, these will be automatically added to the command formulation and treated in the fashion already described.
AND	AND	(Boolean Operator) Used between query value entries in a query formulation. Produces the Venn set intersection between records identified by the value entries surrounding AND.

Short Version	Long Version	Explanation
OR	OR	(Boolean Operator) Used between query value entries in a query formulation. Produces the Venn set UNION of records identified by the value entries surrounding OR
BNOT	BUTNOT	(Boolean Operator) Used between query value entries in a query formulation. Produces the Venn set complement of records identified by the value entry it precedes.
BET	BETWEEN	(Ranging Operator) Used preceding two query value entries separated by a space (which define the range) and qualified by a CFI of a category field on which ranging is permitted.
LES	LESSTHAN	(Ranging Operator) Used preceding a query value entry (which defines the upper limit of the range) and qualified by a CFI of a category field on which ranging is permitted.
MOR	MORETHAN	(Ranging Operator) Used preceding a query value entry (which defines the lower limit of the range) and qualified by a CFI of a category field on which ranging is permitted.

LIST-TYPE RECORD NORMAL QUERIES

QUERY OUTPUT FORMAT SPECIFIER ELEMENTS (6)

Short Version	Long Version	Explanation
"VI	"VIEW	(Command Name) Used to bring the contents of data records identified by queries to the CRT display for user examination. The default value (with no following parameters) shows a pre-established maximum number of the identified records, in a pre-established format, for the latest-numbered successful query preceding the "VIEW command. The records are retrieved and made available to a window on the display surface which can be scrolled and/or paged.
"PR	"PRINT	(Command Name) Has the identical action of the "VIEW command explained above, except that: (a) The output is directed to the hard-copy printer associated with the work station, and (b) The total output specified is printed immediately, with no paging or scrolling actions being involved.
QF__	QF__	(Command Parameter for "VI and "PR) A number entered in the blank sepcifies the QF number of the Query Formulation from which the "VI or "PR processing is to take its data. May be entered more than once in a single "VI or "PR command. All parameters following it (up to another QF__ or the "EX) are interpreted as related to that query output.
__	__	(Command Parameter for "VI and "PR) A number entered that is not preceded by either QF or SKIP (see below) specifies the maximum number of records to be output (if available from that query).

Short Version	Long Version	Explanation
SK__	SKIP__	(Command Parameter for "VI and "PR) A number entered in the blank specifies the number of records to be skipped (ordered by file accession time, latest records presented first) in the query product before beginning output.
SU__	SUPPRESS ( ) ( )	(Command Parameter for "VI and "PR) CFI abbreviations following the parameter name specify category fields not to be included in the outputting of each record. To specify inclusion of category fields, CFI abbreviations may be entered without a parameter name: if this mode is used, only category fields specified will be output.
	Discussion of Syntax	A single instance of both "VI and "PR commands can be included in one transaction entry. Command parameters may be entered in any order.

USER/SYSTEM INTERACTIONS CONTROL ELEMENTS (2)

Short Version	Long Version	Explanation
"MA	"MATCHPICK	(Command Name) The commands "MA "EX cause generation of a fill-in display in which the user may enter a required combination of information specifying each "MATCHPICK (automatic alerting MATCH). File name(s) and Record name combinations are entered; ROUTECODE, ACTIONCODE, and DELTIME values are entered; either a QF number or a Query Formulation is entered; and, finally, an "EX. These elements are defined below and illustrated in a scenario.
"DU	"DUPSEARCH	(Command Name) The purpose of a "DUPSEARCH is to search for records already stored in a file that are in some sense(s) duplicative or similar to a record that is a candidate to also be stored in that file. (Functions of redundancy control and of providing data for correlation and tracking algorithms will depend upon the "DUPSEARCH function.) The commands "DU "EX cause generation of a fill-in display with all the same contents slots already described above for the "MATCHPICK command. In the "DUPSEARCH command, however, the ROUTECODE and ACTIONCODE values entered in the fill-in refer to both the initial identified record from the input stream and the records from the file found by the offset query to be duplicative or similar

ALERTING AND INPUT-DRIVEN AUTOMATIC QUERIES

DATA STORAGE LOCATION/FORMAT SPECIFIER ELEMENTS (None)

QUERY FORMULATION SPECIFIER ELEMENTS (5)

Short Version	Long Version	Explanation
#	#	(Character Operator) Single-character "don't care" character (choice is arbitrary). Available for conventional query formulation, but presented here because it is more likely to be used in a "standing query" context. Character Operators can be used only in data value entries, not in commands, command parameters, or storage format/location names, or other types of operators.
:	:	(Character Operator) Variable length right-continuing "don't care" character.
OLE	OLESSTHAN	(Offset Operator) May be used only in conjunction with the "DU command. Precedes the data value to which it refers, with only spaces intervening. Specifies that the data value following is a deviation or offset value to be added to a reference value in an identified message to obtain the upper limit of a ranging value for search.
OMO	OMORETHAN	(Offset Operator) Same first five lines as OLE, be subtracted from a reference value in an identified message to obtain the lower limit of a ranging value for search.
OBE	OBETWEEN	(Offset Operator) Same first five lines as OLE, be subtracted from a reference value in an identified message to obtain the lower limit of a random value for search, and added to the reference value to obtain the upper limit of the range for search.

QUERY OUTPUT FORMAT SPECIFIER ELEMENTS (3)

Short Version	Long Version	Explanation
RO_ _	ROUTECODE___ ___	(Command Para) Values entered in the blanks are coded identifiers for routing lists to system and organizational addresses. Any number of codes may be entered for one "MATCHPICK.
AC_ _	ACTIONCODE__ __	(Command Para) Values entered in the blanks are coded identifiers for action patterns (other than routing) to be taken with regard to the identified message records; (suspend for consideration before loading into file; load into file and also create cross-reference record to geo-display support files, etc.)
DE_ _	DELTIME DDHHMM	(Command Para) Value entered in the blank is the maximum delay allowed after "MATCHPICK identification of a message record before routing and/or action processing must be completed. In effect, the DELTIME provides the "batching interval" instructions to the system. A DELTIME of 000005 would require a five-minute batch reporting cycle. A value of 010000 would require a 24-hour cycle.
Syntax	Syntax for OLE, OMO, OBE.	Both OLE and OMO values will ordinarily be specified for a particular reference value. They are entered consecutively, with the CFI abbreviation preceding them: (XXXX)OLE XXX OMO YYY AND etc.
	Interpretations of Offset Instructions	For reference values from CFI (LOCATION), any of several convenient conventions could be adopted. The numbers in the offset values inserted represent kilometers. The OBE offset instruction can be interpreted as a radius; the OMO instruction as the 180 semicircular fan with its center oriented toward the FEBA, the OLE as the fan oriented away from the FEBA, etc.

TABULAR/HIERARCHICAL DATA BASE QUERY/CALCULATION

USER/SYSTEM INTERACTIONS CONTROL ELEMENTS (None)

DATA STORAGE LOCATION/FORMAT SPECIFIER ELEMENTS (5)

Short Version	Long Version	Explanation
INTBL ___	INTABLE___	(Command Parameter) Treated as part of the "FILE" command, since it specifies storage format location in the same fashion that a RECNAME does. Used immediately following a file name (if, as is commonly the case, a file name has been invoked in an earlier command, the INTABLE parameter name can be the first word in a transaction entry). Used immediately preceding the name of a table. If the table name specified is not contained in the file currently invoked for the terminal, the system will refuse specification and issue an appropriate error message. Once invoked, the table name remains in effect (i.e., "CONSTANT) until superseded by another table or record name.
UNDTBL ___	UNDERTABLE___	(Command Parameter) Used in a fashion identical to the INTABLE parameter. A table name is inserted in the slot. This parameter name specifies all subtables formatted as subordinate to the named table. If certain subtables are to be omitted from the specification, the entry BNOT _____,_____ with subtable names in the slots is placed immediately after the table name.
INSUBTBL___	INSUBTABLE___	(Command Parameter) Used in a fashion identical to the INTABLE parameter. A subtable name is inserted in the slot, and only that subtable is used as a data source for the operations specified to follow.
UNDSUBTBL___	UNDERSUBTABLE___	(Command Parameter) Used in a fashion identical to the UNDTABLE parameter. A subtable name is inserted in the slot. This parameter name specifies all sub-subtables that are formatted as subordinate to the named subtable. The BNOT operator may be used as in the UNDTABLE parameter.

Short Version	Long Version	Explanation
INSUBSUBT__	INSUBSUBTABLE__	(Command Parameter) Used in a fashion identical to the INSUBTBL parameter. A sub-subtable name is inserted in the slot, and only that sub-subtable is used as a data source for the operations specified to follow.

TABULAR/HIERARCHICAL DATA BASE QUERY/CALCULATION

QUERY FORMULATION SPECIFIER ELEMENTS (12)

Short Version	Long Version	Explanation
FROM QF__	FROM OF__	(Query Transfer Operator) Used to transfer the result of an operation on a table to the position in which the result operates as a query on a table. Thus, for example, the data values identified in one table are used as search values on another table. The query formulation number of the query containing the result is placed in the slot, and the entire query transfer operator takes the place of the search formulation portion of the transaction in which it is used.
COUNT	COUNT	(Arithmetical Operator) Used preceding the CFI abbreviation(s) of the Processing instructions of a transaction. Specifies that the results output desired is a simple count of the number of data values (within the specified Category Field) that map from the search formulation portion of the transaction statement.
MIN	MIN	(Arithmetical Operator) Used in a manner identical to the COUNT operator. Specifies that the results output desired is the MINIMUM data value among the data values, within the specified Category Field, that map from the search formulation portion of the transaction statement.
MAX	MAX	(Arithmetical Operator) Used in a manner identical to the MIN operator, except that the results output desired is the MAXIMUM data value among the data values.
SUM	SUM	(Arithmetical Operator) Used in a manner identical to the COUNT operator. Specifies that the results output desired is the SUMMATION of the data values, within the specified Category Field, that map from the search formulation portion of the transaction statement.
AVG	AVG	(Arithmetical Operator) Used in a manner identical to the COUNT operator. Specifies that the results output desired is the AVERAGE of the data values, within the specified Category Field, that map from the search formulation portion of the transaction statement.

Short Version	Long Version	Explanation
GRP	GRP	(Arithmetical Operator) Used in a manner identical to the COUNT operator. Specifies that the results output desired is the GROUPING of the data values, within the specified Category Field, that map from the search formulation portion of the transaction statement. Identical values are grouped and the groups sorted alphanumerically.
ADD	ADD	(Arithmetical Operator) Used on more than one CFI abbreviation, but otherwise in a manner identical to the COUNT operator. Specifies that the indicated column values are to be added together to produce a results column. Only values that map from the query formulation (if one is included in the transaction statement) are to be processed.
DIV	DIVIDE	(Arithmetical Operator) Used in a manner identical to the ADD operator, except that DIV is used on pairs of CFI abbreviations (if an odd number of CFI abbreviations are entered following the parameter DIV, an error message will result). Specifies that the data values in the first CFI of each pair are to be divided by those in the second CFI of the pair, and a new results column produced.
SUB	SUBTRACT	(Arithmetical Operator) Used in a manner identical to the DIV operator. In each pair, the second value is subtracted from the first.
MUL	MULTIPLY	(Arithmetical Operator) Used in a manner identical to the ADD operator. If more than one pair of columns is to be specified, the CFI abbreviation pairs are separated by commas. If the same CFI abbreviation is repeated in a pair, the values will be squared. If repeated in a triple, the values will be cubed.
MAR	MARKER	(Arithmetical Operator) Used ahead of Col ID Abbrev. to designate it a coordination marker category in inter-table computations.

QUERY OUTPUT FORMAT SPECIFIER ELEMENTS (5)

Short Version	Long Version	Explanation
"RE	"RETAIN (auto)	(Command Name) Used to retain the output of a table transaction processing, whether the output is displayed, printed or not. If the table transaction formulation is successful (does not result in error message or null return message) the result will be automatically retained. Use of a VIEW or PRINT command will also cause an automatic retain. The purpose of naming the command and describing it here is to highlight this automatic action.
COL	COLUMN	(Command Parameter) Parameter of "VIEW and of "PRINT and auto-"RETAIN commands. Used after the command names (inferred in case of auto-REtain) and just preceding the CFI abbreviation alone (without any data value). Specifies that the entire column of values identified by the CFI abbreviation(s) in the transaction statement is to be output (printed, viewed, retained). Multiple columns may be specified.
ROW	ROW	(Command Parameter) Parameter of "VIEW, of "PRINT and of auto-"RETAIN commands. Used after the command names (inferred in case of auto-REtain) and just preceding a single data value followed immediately by a single CFI abbreviation. Multiple rows may be specified by adding more data value/CFI abbreviation pairs. Specifies that the entire row of values identified by the data value/CFI abbrev. combination is to be output.
"TE__	"TEMPTABLE___	(Command Name) Has the same role as the "VIEW, "PRINT, and "RETAIN commands (that of specifying results output actions). Here the output is specified to be a temporary table with the name of the table inserted in the slot. Such tables are for convenience in calculations and in outputs. They are not part of the regular table structures of a file. The format location parameter for such a table, once established, is INTEMPTBL___. Temporary tables may not be assigned sub-tables or sub-sub-tables.

Short Version	Long Version	Explanation
"ME	"MELD___,___	(Command Name) Specifies a type of output action. In the slots are inserted QF___, QF__,; the query formulation numbers of the table-oriented transaction statements containing results outputs that are to be melded together. In the melding process, column identifier abbreviations and operator processing labels associated with data values are retained and carried forward.

USER/SYSTEM INTERACTIONS CONTROL ELEMENTS (10)

Short Version	Long Version	Explanation
"TSC	"TSCHEMA	(Command Name) Used to produce a fill-in display by which the user defines the structure of a task schema to the system.
"TLO	"TLOAD	(Command Name) Used to produce a fill-in display by which the user loads detailed task-related knowledge into an already defined task schema.
"TSA	"TSAVE	(Command Name) Used to store (temporarily) a task schema at any stage of completion. When issued, the task schema currently in process of being defined or loaded is saved.
"TCA___	"TCALL__	(Command Name) Number inserted in blank is provisional task number assigned by systems to a task schema. That schema is retrieved from SAVE temporary storage, displayed in scrollable window of user's display. Full edit capabilities are operative.
"TSU	"TSUBMIT	(Command Name) Used to move a completed task framework (a fully loaded schema) from temporary working (SAVE) storage to a review cycle.
"TNA	"TNAMES	(Command Name) Used to invoke a menu type display of task names for which task-run aids are available. Each name is accompanied by a permanently assigned task number.
"TRU__	"TRUN__	(Command Name) Used to start operation of a task-aid run. The permanent task number is entered in the slot, or the command is used in conjunction with the "TNA command, after the cursor has been positioned on the desired task name.
"TCO__	"TCONTEXT__	(Command Name) Used to invoke a display of the structural context of a specific element of a task for which a task aid exists. If a task element is currently active in a task-run, "TCO can be used without filling the slot to obtain a display for that element. Otherwise, the permanently assigned element number is entered in the slot.

Short Version	Short Version	Explanation
"TDE__	"TDETAIL__	(Command Name) Used in a manner identical to the "TCO command in the preceding description. The display produced shows the detailed data loaded through the "TLOAD command for the specified element.
"TMO__	"TMOVE__	(Command Name) Causes the task-aid run mechanism to advance the task-aid display to the next task element to be accomplished. When used while a task element is active, it advances the task aids to the next element in sequence. If a task element number is inserted, the task aids are moved to the indicated element.

TASK REPRESENTATION/CONTROL

DATA STORAGE LOCATION/FORMAT SPECIFIER ELEMENTS (None)

QUERY FORMULATION SPECIFIER ELEMENTS (None)

QUERY OUTPUT FORMAT SPECIFIER ELEMENTS (None)

TERRAIN OVERLAY SYMBOL SUPPORTING-DATA REFERENCE

USER/SYSTEM INTERACTIONS CONTROL ELEMENTS (2)

Short Version	Long Version	Explanation
"LIN	"LINK	(Command Name) Causes generation of the "LINK fill-in display. This display provides slots for numbers automatically assigned to symbols, to messages, and to table entries (rows). When a combination of symbol number and message or table entry number are placed in one row of the display, either a linking or de-linking action may be performed on the pair by specifying which through a select action on the display.
"CRE	"CREF	(Command Name) Used to retrieve the cross-referenced data items that back up (are the basis for) display of a particular symbol on the terrain display. The cursor specifies the desired symbol. A list of cross-referenced item numbers are presented, the numbers depicting the type and approximate time of the item. Selection actions on the item numbers cause the associated items to be displayed.

TERRAIN OVERLAY SYMBOL SUPPORTING-DATA REFERENCE

QUERY FORMULATION SPECIFIER ELEMENTS (None)

DATA STORAGE LOCATION/FORMAT SPECIFIER ELEMENTS (None)

QUERY OUTPUT FORMAT SPECIFIER ELEMENTS (None)

DATA STORAGE STRUCTURES DEFINE/MAINTAIN

USER/SYSTEM INTERACTIONS CONTROL ELEMENTS (2)

Short Version	Long Version	Explanation
"DEF	"DEFINE	(Command Name) Used to invoke a series of fill-in blank displays by which the user can define to the system the structures of records, tables, hierarchical relations between tables, and contents of files. From these definitions the system initiates processing actions that create the defined entities, as well as corresponding appropriate fill-in blank formats for displays involved in querying and maintaining the various structures.
"INP	"INPUT	(Command Name) Used to invoke a series of fill-in blank displays by which the user performs various maintenance actions on records, tables, files and hierarchies.

DATA STORAGE STRUCTURES DEFINE/MAINTAIN

DATA STORAGE LOCATION/FORMAT SPECIFIER ELEMENTS (None)

QUERY FORMULATION SPECIFIER ELEMENTS (None)

QUERY OUTPUT FORMAT SPECIFIER ELEMENTS (None)

APPENDIX C

TERRAIN DISPLAYS DEFINITIONS

The Detailed Actions (maneuver) display will be assumed to consist of symbols overlaid on a geographic terrain display. Symbol scale and map scale are coordinated, and several display scales are provided. (These design factors are currently in a state of flux, however, with the successful advent of new large-screen display technology such as the LCLV.) Overlays for several different types of symbology can be superimposed simultaneously on the basic terrain display. Relatively static overlays include mobility corridors, line-of-sight corridors, meteorology-dependent facets such as river fording depths, cloud cover and terrain condition indicators, and artifacts such as mine fields. Highly dynamic overlays include symbology for enemy combat elements, often summarized as shooters, movers, and emitters (SME). Symbols allow for further discrimination within these three categories.

For the SME symbology, it will be assumed that backup data may be stored in an in-process message buffer, in a recent message file, in a message archive, or even as a row of a table in a data base. For SME elements, the positing of symbols to the ENSIT displays on the basis of message contents will be assumed to be semi-automatic (with user review and default override) and will operate from the in-process message buffer contents. The display symbols for the relatively more static overlays (corridors, meteorology, etc.) may have backup data in message files or archives, and will often reference data in the EOB data base. Thus the cross-reference requirements for symbology in the detailed action-type situation display will include pointers to and from all types of data storage.

The Organization Position (Units) display will be assumed to consist of symbol overlays on a geographic terrain display. The military unit symbols used may be those in current usage or a revised set based on researches now in progress. Graphic symbols depict the type and category of the unit and its location on the terrain map display. Alphanumeric notes beside or inside the symbols depict the unit number. The symbol location point device is placed on the known or estimated unit headquarters or on the center of its estimated area of occupation.

Compared to SME symbology, the Unit symbols are relatively static. Since identification of enemy units is sporadic in incoming messages (at best), and since usually more than one message with currency will contain information about any enemy unit's location, assignment of enemy Unit situation display symbology requires a human interpretation step based on all the currently available information. However, the preparatory pickoff and grouping of messages by identified enemy Unit designation is done semi-automatically by the system. These grouped messages comprise the data "backup" for the particular enemy unit symbol, the assignment and positioning of which have been accomplished by the user via online interaction with the system. As each position update of the symbology for a particular unit is accomplished, the composition of the backup information referencing the symbol changes, and the cross-reference linkages must also be updated. The symbol's linkages to earlier messages are erased; linkages to later messages are added; and, in some cases linkages are established to notices of extrapolatory movement of the unit symbol, done by a user via interaction with the system.

The backup data for unit position is contained in a message file. Identification of the specific enemy unit on the symbology (when this is possible) opens up the way to cross-referencing data in the ENSIT file and in the ENEMY ORDER OF BATTLE file. This allows the system to provide standardized reports from these two files regarding any identified enemy unit, when the user moves cursor to the unit symbol and takes a button action identifying the desired report.

The Summarizing (Commander's) display is assumed to be a set of symbology overlays superimposed on a geographic terrain display. In this case, the symbols used are aggregating symbols which cluster units into larger force symbols, provide indications of main movement directions, and show demarcation lines such as the FEBA (forward edge of the battle area), etc. The symbols are assigned and positioned entirely by user online interaction with the system, and updated the same way. The backup supporting data for the symbology consists of the symbology of the maneuver and units displays, which analysts have used as bases for constructing the summarizing display for the commander.

Since the immediate or first-order supporting data for the symbology of the summarizing display are maneuver display and units display symbols, the cross-reference linkage problem here is straightforward; the commander's symbols will cross-reference the particular aggregates of units and/or maneuver symbology used by the analyst in summarizing each feature of the battle situation for the commander.

The Event Template Comparison display is a concept in which it is assumed that scenarios of feasible enemy battle tactics and strategies, reflected in

maneuvers, would be stored in the system. A set of alternative scenarios are developed for each important piece of terrain and each phase of an anticipated battle. The scenarios could be "run" on the MIQSTURE language interface subsystem in a manner very similar to the displays resulting from a real battle. The scenarios would each include maneuver, units, and summarizing overlays coordinated to the developing action, and would consist of a series of time-phased "snapshots" of a developing action, perhaps at quarter-hour intervals. In this fashion, a scenario history encompassing eight battle hours could be displayed in slightly over two minutes, allowing four seconds per "frame". In running the scenarios, branching alternatives for each phase of the developing action would be incorporated, so that a choice at an earlier stage constrains and limits the actions that are feasible at later phases. The scenario symbols at any stage may reference discussion notes, materials, assumptions, considerations, main factors taken into account in the simulation, etc.

The counterpart display for comparison with the event template display would be the Summarizing History display. This would provide the same series of "snapshots" at a given time interval, including the same three types of overlays. In the comparison mode, two juxtaposed displays would be used, one containing the Summarizing History display, the other the Event Template display. The history display would contain cross-references to messages that in many cases have been relegated to archive tapes. The alternative scenario segments would be compared with the historical summary of the real current developing situation, and the most similar scenario identified. The backup materials for this scenario segment would then be accessed, to provide previously prepared situation analysis and decision aids.

In the four types of situation displays just sketched, various symbol-to-backup data cross-reference linkages were enumerated. A reciprocal linkage (backup data-to-symbol) may also be required in some instances, though the frequency of use of this type of linkage is obviously much less than the symbol-to-backup data type. Its main use may be to monitor semi-automatic assignments of SME symbology in maneuver displays; in the flow of messages a later message may show an earlier correlation or fusion decision to be possibly erroneous. The analyst at this point must have the ability to find the symbol assigned to any earlier message he may bring into question. One symbol may reference a number of messages, as in the case of a unit position. Several symbols may reference a single message if that message reports, for example, both shooters and movers.