AN INTERACTIVE MANAGEMENT SUPPORT SYSTEM FOR PLANNING, CONTROL, AND ANALYSIS

Richard E. Fikes, et al

Stanford Research Institute

Prepared for:

Office of Naval Research

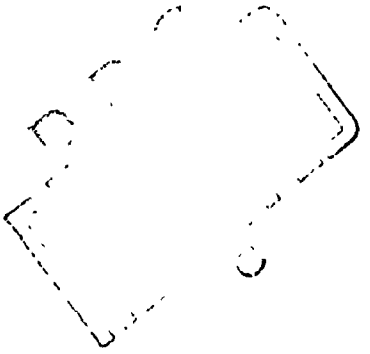
November 1975

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

Technical Report 12

AN INTERACTIVE MANAGEMENT SUPPORT SYSTEM FOR PLANNING, CONTROL, AND ANALYSIS



By. RICHARD E. FIKES and MARSHALL C. PEASE

Prepared for:

OFFICE OF NAVAL RESEARCH
DEPARTMENT OF THE NAVY
ARLINGTON, VIRGINIA 22217

Contract Monitor MARVIN DENICOFF, PROGRAM DIRECTOR
INFORMATION SYSTEMS BRANCH

CONTRACT N00014-71-C-0210

STANFORD RESEARCH INSTITUTE
Menlo Park, California 94025 · U.S.A.

STANFORD RESEARCH INSTITUTE
Menlo Park California 94025 U S A

Technical Report 12 *November 1975*

AN INTERACTIVE MANAGEMENT SUPPORT SYSTEM FOR PLANNING, CONTROL, AND ANALYSIS

By: RICHARD E. FIKES and MARSHALL C. PEASE

Prepared for:

OFFICE OF NAVAL RESEARCH
DEPARTMENT OF THE NAVY
ARLINGTON, VIRGINIA 22217

Contract Monitor: MARVIN DENICOFF, PROGRAM DIRECTOR
INFORMATION SYSTEMS BRANCH

CONTRACT N00014-71-C-0210

SRI Project 1031

Approved by:

DAVID R. BROWN, *Director*
Information Science Laboratory

BONNAR COX, *Executive Director*
Information Science and Engineering Division

Copy No. 5

A

ABSTRACT

A Management Support System currently being implemented at SRI considers the operational problems faced by the commander of a naval air squadron. It provides a research environment in which to study means of aiding him in planning the use of his resources of pilots, aircraft, maintenance mechanics and facilities, and other limited resources to accomplish missions and other required objectives. It provides a tool for studying how to aid him, also, in controlling the operations both by identifying events which may require replanning and, where it can, by adjusting operations appropriately. Finally, it provides a facility in which to study how he can analyze and evaluate past operations to improve the squadron's performance.

While the experimental system operates in the context of a naval air squadron, the objective of the work is to develop concepts and techniques that will be applicable to a range of managerial situations, independent of the particular type of organization involved, or its functions or objectives. The concept, of a knowledge-based system, that is being implemented appears to be broadly applicable to managerial situations.

CONTENTS

TABLES

I INTRODUCTION

This report is an interim progress report on our research program on large-file management information systems, which we also refer to as management support systems. It summarizes the status of the experimental system being implemented and our immediate plans for future research.

The general objective of the program is the study of large-file management support systems that can serve as intelligent decision aids for management in planning and controlling operations and for the analysis of operations through the analysis of historical data.

To provide a meaningful context in which to study the problems of creating an effective management support system, we are developing an experimental demonstration system that addresses some of the problems of actual naval managers. The context that we have chosen is the management of a naval air squadron or group. We wish to emphasize, however, that the goal of the program is the development of concepts and techniques that can be applied to many managerial situations, independent of the particular types of organization or managerial situations, independent of the particular types of organization or operations to which they might be applied.

Within our general objective, the program has two specific goals:

- Knowledge-based planning systems that will effectively aid a manager in a variety of complex situations are to be developed. Implicit in this goal is the objective that the system have the flexibility to tailor its behavior to the manager's needs and preferences.

- Knowledge-based systems that can aid the manager in evaluating operations through the analysis of historical data are to be developed. Implicit here is the development of a data system and question-answering capabilities that reflect the concerns of the manager.

The experimental system currently being implemented is intended to provide an environment within which these objectives can be pursued. This environment is described in some detail in Section II, "Current Design and Status." The ways in which these objectives will be studied and the problems that are foreseen for obtaining effective solutions to them are discussed in Section III, "Research Program."

II CURRENT DESIGN AND STATUS

A. Introduction

Current project effort is focused on the design and implementation of an experimental management support system capable of assisting in the planning, monitoring, and recording of activities within the manager's organization. The system is being designed to reflect the actual structure of the manager's organization; it consists of a collection of independent modules each responsible for a specific set of the organization's resources, personnel, and activities. For example, in the naval air squadron domain there is a module responsible for aircraft maintenance activities, another responsible for pilot scheduling, another responsible for flying missions, and so on.

When a manager requests some activity, such as flying a mission, the module responsible for that type of activity accepts the request and attempts to create a plan to satisfy it. The planning effort consists primarily of obtaining commitments from the other modules to provide resources and services at specific times. For example, the mission planner sends a request to the aircraft scheduler that an aircraft be assigned to the mission for a specific interval of time, and sends a request to the maintenance coordinator for preflight maintenance on the aircraft to be performed at a specific time. If the system is successful in producing a plan, and the plan is approved by the manager, then the same system module acts as the administrator of that plan, in the sense that it initiates and monitors each of the plan's steps, provides status reports on the plan's progress, responds to events that endanger the success of the plan by replanning or sending alert messages to the manager, and acts,

in general as a human might do who is delegated responsibility for following through on the plan. Finally, the module creates a record of the planning and occurrence of the activity on the system's historical files. These files form a data base that describes the operational history of the organization. They are used both by the system as input to its planning models and by the manager as a basis for analyzing the operation of the organization.

The system concept is based in part on the use of process models[1]* which are generalized plans for the execution of various types of activities. A plan that is developed to meet a particular requirement is an instantiation of the appropriate process model and includes the identification of the particular resources and personnel that will be employed and the starting and ending times of the activities that are required. For example, the process of flying a mission requires that a pilot and aircraft be assigned for the necessary time. It also requires the execution of various activities such as briefing the pilot, executing the preflight maintenance on the aircraft, and fueling and arming the aircraft.

The system concept is also based on the use of what can be called "resource models." Each resource model models the current and expected states of all instances of a given resource type--for example, all pilots in the squadron, or all aircraft. A resource model applies the rules that govern the use of that type of resource--both the rules that are inherent in the nature of the resource involved and those that are set by policy. For example, the aircraft resource model would apply the knowledge that an aircraft can only be assigned to one activity at a time, as well as the policies governing scheduled maintenance.

---

*References appear at the end of the report.

Within the system, as currently conceived, separate program modules are responsible for each model. Those that handle process models are called "planners." Those that implement resource models are "schedulers."

Developing a plan to meet a required objective is the responsibility of the appropriate planner. It knows the resources that it needs and when it needs them. It interrogates the appropriate schedulers to determine if the needed resources can be made available when needed. If there are no conflicts, a plan is generated with the form of a set of commitments among the various modules--for example, that an aircraft will be ready for launch at the time required.

The interactions, both between the various modules and between the system and the user, are executed through a control structure that administers a message-handling discipline. That is, each intermodule communication is initiated as a message that is placed in a message queue within the control structure and delivered by the control structure to the appropriate module. This procedure is important for the implementation of the system since it provides an orderly sequence of operations within the system. It maintains the integrity and autonomy of the various models while facilitating the necessary interactions and negotiations.

The following subsections discuss the principal specifications and behaviors of various components and functions of the system. Subsections II-B and II-C define and specify the implementation of activities, events, and messages and their use in the control structure. Subsection II-D discusses the schedulers and their implementation through the use of "scroll tables," which are defined. Subsection II-E considers the planners and their use of process models. While the discussions in Subsections II-D and II-E are primarily concerned with the planning of operations, Subsection II-F considers the use of the system in the control of operations as plans are executed, including the response of the system to events

that might jeopardize existing plans. Subsection II-G then considers the data structures used for the historical files which are available to the manager for the analysis of operations. Finally, Subsection II-H includes a brief description of some additional features that will permit use of the system as an experimental environment.

B. Activities, Events, and Messages

Three data types are used throughout the system, namely "activities," "events," and "messages." An activity is a data structure that represents a specific operation in the organization. Typically, the operation involves performing a task such as flying a mission or doing a preflight checkout of an aircraft. Activities are hierarchically structured in the sense that any given activity may consist of a sequence of operations, each of which is itself an activity. For example, the activity of flying a mission consists of many subactivities, including preflight checkout of the aircraft, briefing the pilot, and launching an aircraft.

In the current implementation, an activity is represented by a list of property-value pairs. Each activity has a unique number as one of its property values. Typical other properties include "scheduled start time," "name of planning module," and "type of activity."

An event is a data structure that represents a specific state-changing occurrence in the organization. The occurrence happens at a specific time and is considered to be instantaneous. Typical events include beginning an activity, ending an activity, and changing the status of a resource.

In the current implementation, an event is represented by a list of property-value pairs. Each event necessarily has property values specifying a unique ID number, the time at which the event occurs, and a "type" indicator (such as "start activity," or "change status"). An event also

often has a list of functions, which is stored as the value of the property "code." These functions are called when the event occurs, and they typically update the system's model of the current status of the organization to reflect the effects of the event.

A message is a data structure that represents a request or response of one system module to another system module. All intermodule communication within the system is done by sending and receiving such messages. For example, when the mission monitor is ready to initiate preflight checkout of the aircraft, it sends a message to the aircraft maintenance module requesting that the preflight checkout activity begin; the maintenance module then sends a message back to the mission monitor module indicating that the activity has begun, and another message later indicating that the activity has ended.

In the current implementation, a message is represented by a list of property-value pairs. Each message necessarily has property values specifying an ID number, the name of the sending module, the name of the receiving module, the time at which the message is to be sent, and a type indicator.

C. Control Structure

The processing framework within which the system modules and the users of the system operate has been designed and an initial version implemented. This subsection describes the existing implementation.

The top level function is a system monitor that administers a queue of messages and a queue of events. Each message contains the time at which it is to be sent. When that time occurs, the monitor calls the module that is to receive the message, with the message as an argument. This is the only way in which modules are called, and they must return control to the monitor when they have completed their processing. For

one module to communicate with another module, a message must be composed and entered into the monitor's message queue.

This control discipline has many desirable properties, including the following. A module can "suspend" itself by returning control to the monitor, and then be restarted by the occurrence of an event that sends it a message or by receiving a "wake-up" message that it has scheduled to be sent to itself; hence, the system can process and plan multiple activities at any given time, and can respond to events in a straightforward manner. For example, when the mission module requests the initiation of preflight checkout, the maintenance module issues a directive to a mechanic to do the checkout, schedules a message to itself to be sent at some later time if the preflight checkout is not completed when expected, and returns control to the monitor for other processing to occur. When the mechanic reports back to the system that the preflight checkout is completed, the maintenance module will send an "activity ended" message to the mission module, and the mission module will proceed to request the initiation of the next step in the mission. Hence, the interaction between the maintenance and mission modules is suspended while the preflight checkout is in progress, and then is restarted when the activity ended event occurs.

Since the message-passing control structure reflects the communication style of the actual organization, the system can interact with those parts of the organization for which system modules do not exist in the same manner that it interacts with other modules. For example, if there is no aircraft maintenance module, then the system can send start activity requests to the maintenance division in the same manner that it sends messages to a maintenance module. Hence, the boundaries of the system are flexible and, in a sense, invisible.

Events are scheduled by entering the data structure representing the event on the monitor's event queue. At the time chat the event is to occur, the monitor removes the event from the queue, fetches the type of the event, calls a list of "demon" functions that are associated with that event type, and executes the code associated with the event. The demon functions typically look for events with some specific characteristics and, when such an event is found, schedule a wake-up message to some module.

The monitor also accepts interrupts from any terminals that are attached to the system. When a user types a control character to request access to the system, a "wait" message is typed out and an internal flag is set. After the monitor has processed all the messages and events that are queued for the current time (the internal clock is incremented in steps of one minute), control is given to the terminal and an interaction with the user is initiated.

D. Scheduling Using Scroll Tables

Many of the modules in the system will be responsible for scheduling the use of a set of resources (including personnel). To facilitate this task we have designed and implemented a type of data structure called a scroll table that is basically a two-dimensional array, in which each row represents a resource and each column represents a time interval. The primary piece of data that is stored in each element of the table is the expected status of the row's resource during the column's time interval. These tables are designed to represent the present and future only. As time progresses, the table is automatically "scrolled" so that effectively, the time interval represented by the first column of the table always includes the current time. The advantage of this representation is that it simplifies the search for a facility such as an aircraft, that is, for example, unassigned over some specified period of time. Since there

may be many suitable aircraft, it permits what amounts to a parallel search of the current plans for all aircraft simultaneously.

Each scroll table has a time quantum associated with it that indicates the smallest interval that can be represented by the table. Each column's time interval can be any multiple of this quantum. Hence, a scroll table can store status changes scheduled to occur at arbitrary times in the future without the necessity of containing a column for each quantum-sized interval. In the current implementation, the numbers of rows and columns in a table varies dynamically with the module's scheduling requirements, and the last column of a table always represents an "open-ended" interval starting at some specific time and continuing indefinitely into the future. Hence, a table always has a column representing any given future time, and new columns are created when needed by dividing an existing column into two columns, with the new column boundary occurring at the time where a status change is being scheduled to occur. Rows can be added or deleted to indicate the addition or depletion of resources assigned to the module.

When a module schedules a resource to be in a given state for a specific time interval, a list of property-value pairs is created to represent that state so that information in addition to its name can be stored about it. For example, if the name of the state is "assigned," then the list would indicate the ID of the activity that the resource will be assigned to. A pointer to this list is the "value" that is stored in the appropriate elements of the module's scroll table. For example, if aircraft AC3004 is to be assigned to mission M2345 from 08:00 to 15:00, then a list would be created containing property-value pairs specifying "assigned" as the state name, M2345 as the activity ID, and possibly other information. A pointer to this list would be entered in the row of the scroll table that corresponds to aircraft AC3004 in those columns that span the period from 08:00 to 15:00.

Our implementation of scroll tables also contains a demon activation facility that allows functions to be called whenever new values are entered into the table. In particular, there is a list of demon functions associated with each element, each row, each column, and the entire table. Whenever a new value is entered into an element of the table, any of the demons associated with that element, the element's row, the element's column, and the entire table are considered for activation.

Associated with each demon is a list of old state values and a list of new state values. A demon is activated only if the name of the new state being entered into the table is on the demon's list of new state values and the name of the state being replaced is on the demon's list of old state values. Hence, the demon facility allows one to specify a demon's activation conditions in terms of resources, time intervals, and state transitions. For example, one could specify a row demon that would be activated whenever a particular resource is scheduled to enter an assigned state, or a column demon could be specified to be activated whenever any resource is scheduled to become "available" during a particular period.

The typical scheduling module receives resource allocation requests specifying the type of resource desired, the length of time the resource is needed, and the time interval during which the assignment must occur. The module attempts to find a resource that has appropriate availability to satisfy the request by consulting the scroll table for that resource type. If such a resource is found, then an assigned state is created for the resource and entered into the table. If no resource is available as requested, the scheduler has several options, including finding resources that are available in a time interval that is "close" to the interval given in the request or rescheduling lower priority planned assignments to make a resource appropriately available. We are currently completing the design of an initial version of these scheduling algorithms.

E. Planning Using Process Models

The previous section described the use of scroll tables for scheduling resource allocations. The system will also be involved in the planning of activities. Typically, a module will receive a request (from another module or from a terminal) that a specific activity be carried out within a given time interval. The module's task is to create a plan for that activity, including scheduled start and end times for each of the plan's steps or subactivities. If the planning effort is successful, the module will respond with a commitment that the activity will be carried out as planned. If the requester must supply some resource (such as an aircraft or a pilot) before the activity can begin, then the commitment is a mutual one between the requester and the planning module that if the resource is provided at a specific time the activity will be completed as planned.

During the planning of an activity that involves subactivity steps, the planning module becomes a requester to other modules for scheduling the subactivities. For example, when the mission module receives a request to plan a mission, it sends planning requests to the maintenance module for a preflight checkout activity, to the flight deck module for a launch activity, and so on. When a planning module cannot schedule an activity within the requested time period, it will, whenever possible, return alternative schedule(s) that are close to the requested time period. The requesting module can then try to modify the time period constraints so that one of the alternative schedules satisfies them. This may mean interacting with the manager at the terminal, or if the activity is a single step in the plan of some superactivity, then it may mean attempting to reschedule the steps immediately before and after the alternatively scheduled step.

The planning algorithms we have designed and are now implementing proceed by instantiating a process model of the activity. A process model can be thought of as a generalized plan for an activity type, such as a mission or a preflight checkout. It includes the steps that a plan will have, the preconditions and effects of each plan step, a set of variables for which the planner must find acceptable values, and a set of constraints on the values of those variables. Typical variables in a process model are start and end times for subactivity steps, and identities and quantities of the resources that will be used by the activity. Typical constraints are partial ordering relations on the start and end times of the individual plan steps.

Process models are useful for planning in situations where the ordering and identity of plan steps are the same for all activities of a given type. In such situations, the planning task is basically one of obtaining scheduling commitments for each subactivity and for the necessary resources. This style of planning appears to be generally useful to a manager, since many of the operational procedures in an organization have the same steps each time they are carried out. For example, flying a mission can be described by a process model that involves preflight checkout of the aircraft, loading and fueling the aircraft, briefing the pilot, launching the aircraft, a flight to the target, a delivery activity, a flight back to the carrier, a postflight checkout of the aircraft, and a pilot debriefing. Each mission typically involves this fixed set of steps ordered to satisfy a fixed set of constraints.

Variables in our process models are "describable" in the sense that each one has a list of property-value pairs associated with it. One important piece of information stored on a variable's property list is a specification of how a planner can determine a value for the variable. This information typically specifies the name of the module that can

determine a value and the set of parameters needed by that module as input. For example, the identity of the aircraft to be assigned to a mission can be determined by sending a request to the aircraft scheduling module; the request must include the estimated start and end time of the mission and the type of aircraft desired. Other important types of information found on variable's property list include default values and symbolic constraints that restrict the range of possible values.

F. Operational Control of Planned Activities

The system will also have facilities for control of the execution of an activity after a plan has been completed and approved. When a module obtains approval from the manager for one of its plans, it will schedule a wake-up message to itself to be sent at the start time for the first step in the activity. When this message is sent, the awakened module will begin to function as an execution monitor by sending a start activity message either to the module or to the person responsible for carrying out the first step. Before releasing control, the monitoring module will also schedule an "alarm" message to itself to be sent when the completion of the first step would be considered overdue. If the first step ends on schedule, an activity ended message will be sent to the monitoring module, the alarm message will be canceled, and the next step of the plan will be initiated in the same manner. If the alarm message is sent before the first step is completed, then the monitoring module will send out a message requesting an estimate of when the first step will be completed, and upon receiving an answer, will initiate replanning using the new estimated end time for the first step.

In general, modules make commitments to each other during the creation of a plan to perform tasks and to provide resources at specific times. Each module has the responsibility of monitoring those activities, events, and resource usages in the organization that could jeopardize its

ability to keep its commitments. This monitoring is typically done by demon functions set up by the modules at the time the commitments are made. When such a demon is activated, it checks to see if the commitment can still be kept, and if not, sends a message to the involved parties. A module receiving such a message will typically initiate an effort to modify the plan being executed to correspond to the new situation.

G. Data Base Architecture

A major concern in our management support system is the creation, maintenance, and review of large historical files of information about system activity and performance of the organization. We have completed the design and implementation of the basic filing techniques needed to implement these data base facilities. These techniques are similar to ones widely used in current commercial systems, with ours having slightly more generality and correspondingly less efficiency. It is not these filing techniques in themselves that are of interest in this project, but rather their use in the context of a process model. To accommodate process models, we have developed two original features: demons and process model schemas, which we will discuss further. Subsection IV-E gives specific examples of how the process model can be used in answering queries about stored data. In this subsection II-G, we concentrate on the ways in which program modules interact with the data base.

As discussed earlier, our management support system contains a set of independent program modules, each of which is responsible for a specific set of the organization's resources, personnel, and activities. We have chosen to adopt a modular data base approach in which each program module is tightly interconnected to the segment of the overall data base relevant to it, so that the complete data base is "decentralized."

For example, the pilot scheduling module has responsibility for scheduling and monitoring the assignment of pilots. These responsibilities include ensuring that each pilot receives adequate rest time, and that a reserve of pilots is maintained to allow for unplanned events such as a pilot becoming unavailable due to illness or injury. To fulfill these functions, the pilot scheduler needs current information about the number of hours each pilot has been assigned during the preceding time period, the policy that determines how much rest each pilot needs, and the probability (conditional on his past record) of each pilot becoming unavailable.

Another exemplary module is the mission coordinator, which has responsibility for the planning and monitoring of missions. In order to make realistic plans, it must have accurate statistics on the time each mission subactivity takes to execute. For example, it might use estimates that the mean briefing time is 25 minutes with a standard deviation of 5 minutes, obtained from weighted or running averages of past performance.

The information that the pilot scheduler and mission coordinator require in the above examples is stored in local data bases that these modules have sole responsibility for creating, maintaining, and reviewing. The complete data base is simply the union of these local data bases.

One of the primary advantages of this decentralized approach is the ease with which the system can be extended. When new modules are defined, or when particular segments of the data base require restructuring because of a change in the responsibilities or resource characteristics of a particular unit of the organization, an overall restructuring of the global data base is not required. Instead, it is only necessary to change the structure of a local data base.

The decentralized approach introduces several technical problems. Specifically, assuring consistency of information included in more than

one segment, allowing for cross-referencing between local data bases, and providing for answering queries that review the system as a whole. Another problem is providing a set of general-purpose access software that can support all the local data bases without forcing them into a single central organization. Clearly, any such general-purpose software would have to be "content-free," in the sense that its procedures could not be tied to any one local structuring of data.

The solution we have formulated to these difficulties is based on structuring all local data bases in a simple "relational" form, and maintaining local data base declarations, or subschemas, that table-drive general-purpose access functions. The union of the subschemas for independent modules serves as the schema for the complete data base.

The relational data structure has been shown by E. F Codd[2] to be general and to facilitate the maintenance of multiple consistent views of a data base. A relation is simply a table whose columns correspond to particular "data attributes," and whose rows correspond to a collection of "associated data values." Each relation resides in a functional entity that we refer to as a relational file. A simple example of a historical relation recording information for the mission coordinator is shown in Table 1. The data attributes are generic items that take on values for each mission.

Since our management support system is being implemented in INTERSLIP, we have chosen to represent the subschema for a relation as a list structure. Table 2 gives the subschema for the MISSION.SUBACTIVITY.TIMING relation.

Our schema approach has been widely advocated[3] and almost universally adopted in sophisticated data base management systems. In essence, the schema records in a form that can be machine-interpreted, the structure of the data base, in a manner independent of procedures that access data.

Table 1

AN EXEMPLARY RELATION

Relation: MISSION.SUBACTIVITY.TIMING

Mission Id.	Preflight Checkout Time	Briefing Time	Fueling Time	Launch Time
#2931	10 mins.	15 mins.	10 mins.	7 mins.
#2932	15 mins.	25 mins.	12 mins.	6 mins.
#2933	12 mins.	20 mins.	8 mins.	9 mins.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

The subschema for MISSION.SUBACTIVITY.TIMING contains precisely the information about the structure of its local data base that programs other than the mission coordinator should know to use data in this relation. The schema language requires that for each attribute its name and primitive type, PRIM.TYPE, be given. The name is a symbolic reference name by which procedures can refer to each attribute. PRIM.TYPE is the low-level type of the data, which is NUMERIC, TEXT, or LIST.

Several other optional specifications are possible. Statistical type, STAT.TYPE, tells review programs what types of statistical techniques apply, i.e., whether variables have discrete or continuous values, or whether, for instance, they are exponentially or normally distributed. The retrieval feature, RETRIEVAL, of a data item indicates what special "retrieval files" should be maintained by the system so that data used in certain ways can be accessed efficiently. Initially two types of

Table 2

SUBSCHEMA FOR MISSION.SUBACTIVITY.TIMING

```
(MISSION.SUBACTIVITY.TIMING
     ((NAME . MISSION.ID)
      (PRIM.TYPE . NUMERIC)
      (STAT.TYPE . SEQUENTIAL)
      (RETRIEVAL . INDEX))
     ((NAME . PREFLIGHT.CHKOUT.TIME)
      (PRIM. TYPE NUMERIC)
      (STAT.TYPE CONTINUOUS)
      (UNITS . MINUTES))
     ((NAME . BRIEFING.TIME)
      (PRIM.TYPE NUMERIC)
      (STAT.TYPE CONTINUOUS)
      (UNITS . MINUTES))
     ((NAME . FUELING.TIME)
      (PRIM.TYPE NUMERIC)
      (STAT.TYPE CONTINUOUS)
      (UNITS . MINUTES))
     ((NAME . LAUNCH.TIME)
      (PRIM.TYPE NUMERIC)
      (STAT.TYPE CONTINUOUS)
      (UNITS . MINUTES)
      (RETRIEVAL . TRANSPOSED)))
```

retrieval files will be available as well as the primary relational files: an index file that increases the speed of searches, and a transposed file that reduces the time required to process a few attributes of a relation.

The full set of schema language options for declaring process models has not yet been developed. When it is, the schema declaration will not only provide for accessing data, it will also drive planning procedures and facilitate inferences related to queries.

As well as describing all the data items and their grouping into relations, the data base schema stores a set of procedures, or demons, that are executed when data are stored or modified in certain relations. The purpose of these demons is to alert interested modules of occurrences significant to them that have been recorded in the data base, and to maintain useful summary information about particular relations. A typical alert demon might print a message on a terminal, warning the commander whenever the number of available reserve pilots drops below a threshold. A typical summary demon might count the number of missions flown each week. Demons may also be used to check for unlikely or impossible values resulting from data entry errors, thereby helping to improve the quality of the information in the historical files. This implementation of demons on the relational files gives the same programming power for operating on historical data that is provided for planning using the scroll tables.

Three logical groups of general purpose functions are used for creating, maintaining, and reviewing data relations. The first is a set of functions for creating and modifying the data base schema. These functions allow the programmer to add, delete, and rearrange particular attributes among several relations, and they allow an individual attribute's description to be modified. Once a schema is defined, it becomes possible to store, retrieve, and modify data in it with the functions RELATION.PUT and RELATION.GET. RELATION.PUT stores a row of data

specified as an association list of attribute-value pairs in a relation. For example, the first row in the example of Table 1 would have been recorded:

```
RELATION.PUT(MISSION.SUBACTIVITY.TIMING
        ((MISSION.ID . #2931)
         (PREFLIGHT.CHKOUT.TIME . 10)
         (BRIEFING.TIME . 15)
         (FUELING.TIME . 10)
         (LAUNCH.TIME . 7)))
```

RELATION.GET retrieves data after performing a pattern match of a "query template" against the data stored in a relation. This query template is, in essence, a filter through which the whole relation is passed. From a programming point of view, RELATION.GET returns a generator function that can be called on successively to return each of the rows of the subrelation that passes the filter. A third set of functions is provided that allows the set operations "and," "or," and "relative complement" to be applied to generators. The resulting query language is general and convenient to use.

In addition to relational files, we have defined a secondary set of data base entities called "trace files" that are optimized for keeping a complete log of system messages and events. Trace files record sequentially every message and event processed by the management support system. They are used to isolate and debug errors or shortcomings of individual modules, and to verify that each module is capable of handling the requests sent to it by other modules (hence facilitating debugging of module interactions).

At present, the design of the data base procedures is complete, and we have finished the implementation of direct access file procedures for

INTERLISP, which represents about one third of the data base implementation work. The filing procedures allow random access to variable length, arbitrarily nested or circular s-expressions, stored under symbolic keys (which may also be s-expressions). Records can be overwritten by larger records, and the index can be searched with an n-ary tree (logarithmic) search algorithm. The filing procedures also allow for sequentially scanning a file either in key order or reverse key order, and they will generate unique record identifiers sequential to allow record writing.

The data base system being implemented is described in detail in a concurrent Technical Report.[4]

H. Demonstration System

To facilitate debugging, testing, and demonstrating our experimental system, we are designing and implementing a facility for simulating the occurrence of activities and events. Hence, in a situation where a start activity message would be sent to a person in an actual operational environment, our experimental system will send the message to a module that will simulate the occurrence of the activity. These simulation modules will determine the duration and effects of each activity, and will generate the appropriate messages and events to note the occurrence. We will also provide a facility for entering unexpected events, such as a pilot becoming ill or some piece of equipment failing. These events will be either prestored on an event file or entered from a terminal in "real time" while the system is operating.

In general, we will create simulations and scenarios that will exercise as many of the system's features as possible in a reasonable length of time. Also, we will provide appropriate trace and commentary features so that an observer can understand and appreciate the behavior of the system.

III RESEARCH PROGRAM

Future directions planned for the research program include effort in the following areas:

- Improved replanning capability
- Use of alert functions to warn of critical situations
- Use of priorities and value functions during planning
- New maintenance modules
- Retrospective analysis of historical files
- Use of accumulation and summary files during planning.

In the following subsections we describe some particular problems and needs in each of these areas, and discuss how we expect to meet these needs.

A. Improved Replanning Capability

The system being currently implemented does not have the capability to consider modification of existing plans as a means of fulfilling the resource requirements of a new plan. There is need for techniques to remove this limitation as a step toward achieving the general goal of a system capable of modifying existing plans to accommodate a variety of circumstances.

In the system being currently implemented, if an action that is part of a planned process is not completed at the planned time, the system replans the rest of the process as well as it can, but without taking into account the possibility of adjusting other operations to facilitate recovery. Also, a new exogenic demand, such as a new mission, initiates

a planning procedure that seeks to schedule the mission as required by the demand, but again without taking into account the possibility of adjusting the schedules of other operations to make room for the new one.

Additionally, it is desirable to base planning on estimates of the time needed for the various actions that are either "safe" or "tight," depending on the level of activity currently expected of the squadron. For example, safe planning might use for the preflight checkout time the experienced average time plus three times the experienced average deviation, while tight planning would use the experienced average time itself. As the planned level of activity grows, we would like the system to use tighter estimates for all the activities required, and modify existing plans accordingly.

To obtain the capabilities that are desired, the system must be able to evaluate the total situation and to use this evaluation to guide its handling of a given requirement either for a new activity being added to the schedule, or for the replanning of currently scheduled activities. Development of the procedures for evaluating the general state of plans and for adjusting planning strategies according to that evaluation will be an important focus of future work.

B. Alert Functions

It would be desirable for the system to warn the commander when there is danger that a critical situation may be developing that may limit the capabilities of his organization. The planning facility currently being implemented produces information that can be used to recognize such potential threats relating to the exhaustion of some necessary resource.

One of the functions of the system as currently planned is to advise a commander when events happen that force a substantial modification of existing plans, such as a delay in the execution of an approved mission.

This capability can be extended to provide advance warning when the situation becomes such that current plans are sensitive to the possibility of unforeseen events. For example, the system can recognize that some vital resource, such as the pilot pool or the maintenance facility, is becoming saturated. It should warn the commander of this fact so that he is made aware that any trouble in any of the existing plans, or any new requirement such as an additional mission, may create a bottleneck that the system will not be able to resolve.

In the current system it is not difficult to identify times of peak loading for any resource from the corresponding scroll table. However, the need is for an alert capability that is somewhat more discriminating. First, some activities may be easily deferrable, or have low priority, and should be excluded in evaluating loading. Other activities such as scheduled maintenance, may be easily deferrable so long as they have not actually been started, but should not be interrupted. The evaluation procedure should take account of these different possibilities.

In addition, account should be taken of the flexibility that may be present in scheduled activities. For example, maintenance may be heavily scheduled in one period, but only lightly loaded in an immediately preceding period. This does not constitute a bottleneck for any activity that can be done during the lightly loaded period, and should not be reported, except, perhaps, as a suggestion that the schedule should be rearranged.

The alert facilities envisioned here may be said to be a part of the general capability called "crisis management." It is intended to warn the commander of the possibility of a crisis before it actually occurs, enabling him to take appropriate steps to avert the possibility.

C. Priorities and Value or Cost Functions

The current system does not provide for the use of priorities during planning. All demands for missions, and all activities planned by the system have equal weight and are undertaken on a first-in basis. This is undesirably restrictive. There should be a capability through which the commander may set priorities, either for individual activities or for classes or types of activities. We expect in the near future to incorporate the use of priorities, both internally and externally generated, into the system's planning and administrative procedures.

A set of priorities or value functions can serve several purposes that are important to the use of the system. It can describe requirements that have a degree of flexibility, such as that an action shall be done "as soon as possible." It can be used to improve the overall condition of the squadron or of some department in the squadron, reflecting, for example, the desirability of keeping a fairly uniform load on the maintenance facility. It can be a powerful and subtle tool for guiding scheduling in a complex situation.

The assignment and use of priorities is a procedure that often requires assessment of the entire present and future situation. The setting of priorities must remain a command prerogative, and be one tool by which the commander enforces his decisions regarding the future of the squadron. There is need, however, for effective means by which the commander can enter priorities, either as a general policy, or in particular types of situations, and to automate the use of priorities in scheduling.

In addition, there are priorities that may be set by the system itself. For example, in periods in which the maintenance department is very busy, its scheduling may be very tight. It may become critical

that all aircraft reach the maintenance department on schedule. The requirement to get them there on schedule should then be given high priority.

There are problems associated with the introduction of priorities and value functions, and, in particular, important questions regarding the resolution of conflicts. In the absence of a satisfactory resolution algorithm, it is possible either that the system will thrash, or that it will reach a solution that is totally dominated by one consideration and essentially ignore other requirements. Future work with priorities will include consideration of such problems involving the resolution of operational conflicts.

D. Multiple Process Types

The system currently being implemented is concerned with a single type of process, in our particular experimental demonstration context, the flying of missions. It does take into account a variety of conditions that may affect the available resources or that may alter the execution of a planned process. There are additional problems, however, when there are different types of processes competing for the same resources.

In order to study the effect of completing process types, we intend to extend the experimental system to include two additional process types that have some interesting differences in the demands they place on the planning facility.

The first process type is that involved in handling the scheduled maintenance requirements that are carried out by squadron personnel. The second concerns the handling of unscheduled maintenance on the squadron's aircraft. As discussed below, the integration of these two types of activities with flying missions poses some interesting problems.

1. Scheduled Maintenance

Scheduled maintenance performed within the squadron's own facilities represents a long-term and continuing requirement. Its scheduling involves a number of constraints and goals of quite different types from those used in planning missions. First, scheduled maintenance is not tightly constrained in time. In general, it can easily be deferred, although it is likely to become more urgent the longer it has been deferred. Second, scheduled maintenance can be anticipated, and therefore planned well in advance. Third, it can be executed in advance of its normal time. This allows it to be scheduled so as to smooth the anticipated load on maintenance facilities, or in anticipation of a future tactical situation, e.g., prior to engaging the enemy.

These considerations are quite different from those that apply to planning missions. Different procedures for the allocation of resources are required. It is the effect of these different procedures that requires study.

We envision establishing means for identifying scheduled maintenance operations, to keep track of what is required, and to initiate and monitor the actions that are required. The means for identifying the required operations can be incorporated within the aircraft scheduling module through the use of demons acting on the scroll table for aircraft scheduling. Whenever a scheduled maintenance job is done on an aircraft the demon for that type of job is activated and enters the time when it should next be done. Whenever a job is deferred past its scheduled time, a demon is activated that reenters it into the schedule at a later time, with higher priority if appropriate.

Some scheduled maintenance requirements are expressed in terms of flight hours, rather than elapsed time. For such actions, a different procedure is needed since the scroll tables are set up in terms of elapsed

time. One possible procedure is to include the requirement in the property list for the aircraft in terms of the total flight time since the last time the job was done. The current accumulated flight time is also maintained on the property list and updated each time the aircraft is flown. On each update, a demon checks the current value of the accumulated flight time against the threshold values listed there for various scheduled maintenance requirements. When a threshold is passed the activity is scheduled.

If it were convenient to do a scheduled maintenance action prior to the threshold to smooth out the load on the maintenance shop or for tactical reasons, this action would be initiated either by the maintenance scheduler or by the alert module that watches for possible bottlenecks (see Section IV-B).

The actual scheduling should be done according to a set of priorities or a value measure, as discussed in Section IV-C. Given the capability of planning according to a specified set of priorities, or to maximize a given value function, there appears to be no intrinsic difficulty in the inclusion of scheduled maintenance.

2. Unscheduled Maintenance

Unscheduled maintenance differs from scheduled maintenance not only in the fact that it cannot be anticipated in advance, but also because it may lead to some quite complicated priority situations. The priority given to an unscheduled maintenance requirement may be a complex function of the nature of the action required and of the operational demands being placed on the squadron as a whole. If the fault makes the plane unusable or NOR (not operationally ready), then the plane is a lost resource until the action is performed. How critical this may be depends on the anticipated load on the squadron's aircraft. Other maintenance actions, such as preflight checkouts for currently planned

missions, may be more critical. If the fault makes the plane RMC (reduced materiel condition) so that it can still fly certain types of missions, then the urgency of the corrective action depends also on what kinds of missions are likely to be required in the near future.

It does not seem reasonable that these factors should be automated; the commander should retain control. Assuming that the decision is retained as a command prerogative, the system should give the commander the information he needs to exercise command responsibility.

The scope of the system can be extended, then, to include unscheduled maintenance with two objectives in mind:

- The system should provide the commander with information that may help him to determine what priorities should be set for unscheduled maintenance actions.
- The system should schedule unscheduled maintenance actions in accordance with the priorities set by the commander, and should oversee their execution.

The second objective is, in many respects, similar to the planning and monitoring of a mission. It is a process that is initiated by a demand to do the required maintenance action, although the demand may be originated differently. For example, the failure of a system or subsystem discovered in flight and confirmed during the postflight checkout should automatically generate a demand for the appropriate unscheduled maintenance action without requiring a separate exogenic demand.

The principal new factor is the almost automatic requirement that the job be done as soon as possible, consistent with the assigned priorities. This differs from a mission for which, generally, a specific time for execution will be required, and from scheduled maintenance which is executed on a planned schedule.

In Section IV-C, we discussed the use of priorities and value functions in a broader context. The requirements of the second objective listed above can be met through the use of priority planning. The details must be worked out, but no intrinsic difficulty is anticipated.

For the first objective of advising the commander so that he may best judge what priority to assign a job, the critical question is the effect on operations of either doing the job or deferring it. This depends on evaluating the situation in terms of a possible overload of critical resources under various contingencies and alternative decisions. The evaluation of the situation in these terms has been discussed in a more general context in Section IV-B, Alert Functions." Again, ways to implement the required capabilities must be developed, but no intrinsic difficulty is anticipated.

In summary, the need for an unscheduled maintenance action will be initiated either by an exogenic demand, or automatically as a consequence of trouble detected during some other process. The facility provided for the alert capability can be used to assist the commander in determining what priority should be given to the action. Once the priority has been set, the system will schedule the required action and will oversee its execution.

E. Retrospective Analysis

Primary attention has so far been given to developing the basic facilities for planning and monitoring ongoing operations. As part of this work, we have been concerned with the construction of the historical files that will hold the detailed records of past operations. However, we have given little attention to the use of these files in any depth, except to make certain that we retain the ability to determine the connections between data elements that are likely to be required by

anticipated uses. Future work will include the development of a data-base querying facility to verify the utility of our process-model oriented handling of the historical files.

The following are representative examples of the kinds of questions that are important to a squadron commander:*

(1) Why was the mission on date xyz canceled?

(2) How many missions were canceled or delayed last month? Give a breakdown of the reasons for each.

(3) For the last six months, divide all missions into two groups--those for which the preflight checkout took less than the current average, and those for which it took more. What is the percentage of mission aborts for equipment problems in the two groups? What is the number by type of equipment malfunction found during flight or during post-flight checkout in the two groups?

In (1), the manager is looking for a specific event. The system must first find the event. Then it must interpret what is expected as a reason. It is sensible to limit the reasons to those that are implied by the process model or by the scheduling procedures. Specifically, the system can respond through a hierarchical sequence of tests. If there were an inability to schedule some part of the process due to an overload, then that fact becomes the answer; if there were a default of some preparatory action without a satisfactory recovery, then that is the answer; or the cancellation was a command decision.

In (2) he may be trying to determine what critical resource most often causes trouble. The reasons for cancellation or delay can be interpreted as in (1).

---

*While the questions are given in natural language for the sake of clarity, the system will require that they be given in an appropriate formal language. Our concern is with the identification, retrieval, and processing of the responsive data elements, not the development of a natural language capability.

In (3) he may be testing the hypothesis that preflight maintenance is sometimes being cut short, perhaps when there is too much pressure from the schedule. Or he may be hypothesizing that excessive time in preflight checkout is a symptom of a more deep-seated trouble that interferes with, but is not caught by, the preflight checkout procedure.

The responses to these questions require, first, the identification of the missions involved. This can be done from the file of exogenic demands. It cannot be done from the file of aircraft assignments, since some of the questions concern cancelled missions. Searching the exogenic demands will also uncover the delayed missions, since a scheduling delay would be referred back to the commander for his acceptance of the revised plan.

These examples, which are not intended to be exhaustive, illustrate the use of the historical files for the analysis of past operations. They illustrate the ways in which, given appropriately structured data, questions that are of direct significance to the commander can be handled.

The information required to respond to questions such as those listed is contained in the historical data files as they are currently planned. It is an important aspect that the information likely to be important to the commander is associated with the appropriate process model. Regardless of the detailed structure used in the data files, there should be ways to recover data elements that are connected as described by the process model. It is this capability that makes the data useful for retrospective analysis by the commander. This capability is being built into the file system currently being implemented.

Research is needed to determine suitable ways of responding to such queries. It appears that the capability can be implemented through one or more modules introduced for that purpose. The user's query will be directed to a module that will then operate through the control structure

as do the scheduling, planning, and monitoring modules. This is significant in that it permits a uniformity of design through both the operational and analytic functions of the system. It also provides an effective interface between these two types of functions. Finally, it simplifies the problem of adding new capabilities to the system, such as the capability of responding to a new type of query.

F. Accumulation and Summary Files

In the current system, data describing operations are recorded on the historical files. During this process, there is provision in our procedures for the development of various special files that build running accumulations of selected data and summary figures. For example, provision can be made to accumulate total flight time, number of flights, scheduled and unscheduled NOR time, NORS (NOR Due to Supply) time, and total time in squadron for each aircraft in the squadron during the month. At the end of the month, the accumulated totals are those that must be reported in the Aircraft Readiness/Flight Summary report (card code 79).

Another, somewhat more interesting, example is the accumulation of weighted total of the time taken for various activities, together with the correspondingly weighted number of occurrences. The ratio of these two accumulations gives a weighted average that can be used for planning purposes. For example, the average, weighted to emphasize recent experience, of the duration of preflight maintenance can provide a better figure for planning than a preassigned standard value. A similarly weighted normal deviation can be computed. Use of this figure, together with the weighted average, would allow planning to be based on either the average experience, or on the safer figure of the average plus, for example, three deviations. This would permit planning to be either tight or safe, depending on the importance of the missions and on the general load conditions.

IV SUMMARY

The system being implemented provides an environment for the study of the principles and techniques that are useful for the design of knowledge-based systems to aid a manager in planning, control, and analysis of operations.

The work done so far verifies the importance of the concept of a process model as one component in structuring both the system itself and the data that it generates and records. To the extent that it describes the way in which the manager regards the operations of his organization and the way in which the organization itself responds to the demands placed on it, it leads to a system design that will be capable of recognizing significant events and of responding to them in ways that are meaningful in the managerial context.

The concept of a process model is not sufficient, however. In addition, means must be provided for handling the allocations of limited resources to the activities that implement the process models. The means may be regarded as implementing a set of what might be called "resource models." In a sense, the sets of process models and resource models are orthogonal views of the real-world operations. In this sense, the problem of system design is, first, to implement these two views separately, and then to implement their interactions in an appropriate manner.

The system being implemented handles the two types of models separately. The process models will be implemented in a set of "planner" modules which will include also much of the control function and part of the analysis function. The resource models are implemented in a set

of "scheduler" modules, each dealing with a given type or class of resources. The separation of these two types of modules permits the use of data structures that are appropriate to each type and the separation of the data itself according to its use. The interactions between the two types of modules are then achieved through the message discipline and the control structure.

While the implementation of these principles must be further demonstrated, refined, and broadened, the system concept being implemented appears to have great potential.

REFERENCES

1. M. C. Pease, "Application of a Process Model to a Management Support System," Technical Report 9, Project 1031, Stanford Research Institute, Menlo Park, California (July 1974).

2. E. F. Codd, "A Relational Model for Large Shared Data Banks," CACM, Vol. 13, p. 377 (1970).

3. CODASYL Data Base Task Group Report," ACM (April 1971).

4. Stephen Weyl, "An Interlisp Relational Data Base System," Technical Report 11, Project 1031, Stanford Research Institute, Menlo Park, California (November 1975).