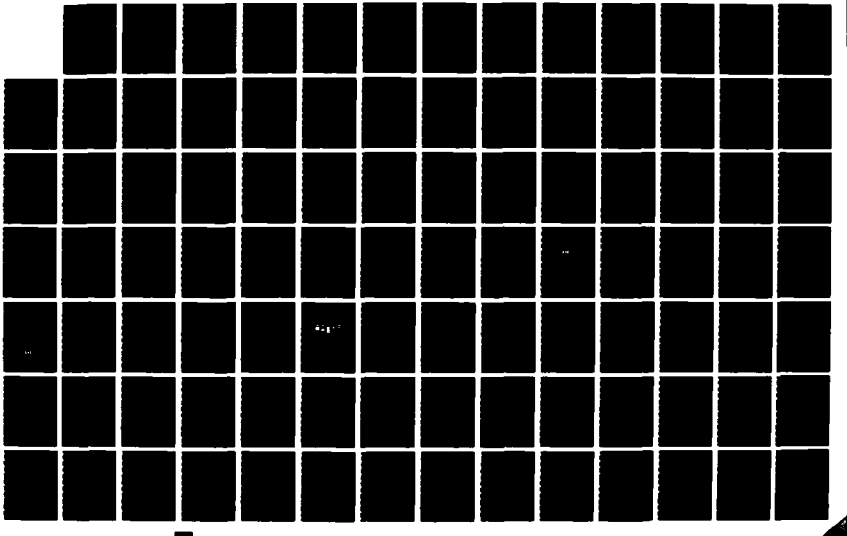
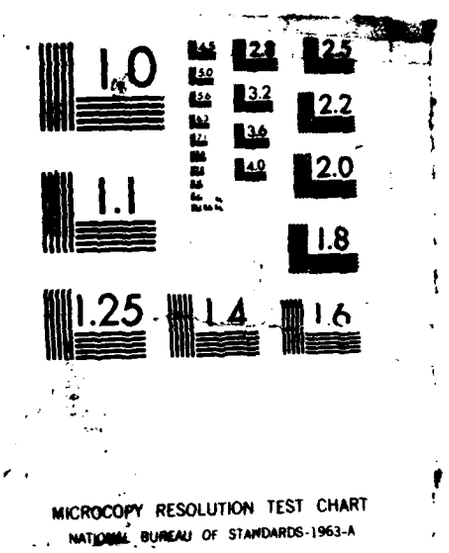
AD-A180 265 DESIGN EVOLUTION OF A FIGHTER TRAINING SCHEDULING DECISION SUPPORT SYSTEM(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI 1/3

UNCLASSIFIED J W GRECHANIK ET AL MAR 87 F/G 5/8 NL





AFIT/GST/ENS/87M-6

DESIGN EVOLUTION OF A FIGHTER TRAINING SCHEDULING DECISION SUPPORT SYSTEM

THESIS

Paul E. Trapp
Captain, USAF

Jeffrey W. Grechanik
Captain, USAF

AFIT/GST/ENS/87M-6

Approved for public release; distribution unlimited

6c. ADDRESS (City, State, and ZIP Code)		7b. ADDRESS (City, State, and ZIP Code)			
Air Force Institute of Technology Wright-Patterson AFB, Ohio 45433					
8a. NAME OF FUNDING / SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER			
8c. ADDRESS (City, State, and ZIP Code)		10. SOURCE OF FUNDING NUMBERS			
		PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO	WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

DESIGN EVOLUTION OF A FIGHTER TRAINING SCHEDULING DECISION SUPPORT SYSTEM

12. PERSONAL AUTHOR(S) Jeffrey W. Grechanik, B.S., Capt, USAF
Paul E. Trapp, B.S., M.A., Capt, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
MS Thesis	FROM _____ TO _____	1987 MARCH	240

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Adaptive design; design evolution; scheduling; decision support system; fighter training sched-
01	03	06	
05	01		uling.

The ~~primary~~ objective of this research effort was to trace the evolution of a decision support system for a fighter training unit scheduler. The purpose of a Decision Support System (DSS) is to assist the cognitive processes of judgment and choice as performed by a squadron scheduler.

This DSS was built using the adaptive design process. The adaptive design process includes development of a concept map, analysis of tasks and data, and identification of the kernel to select the central decision process. The designers employ this central decision process to develop a feature chart and storyboards, the starting point for the DSS construction.

Ideally, the DSS begins with a small prototype that decision makers use and evaluate. Builders then customize and modify the DSS as a result of user feedback. This process repeats, incorporating user needs and requirements. Thus, the DSS improves with each successive iteration.

This particular DSS automates the squadron scheduling decision process without interrupting the schedulers ability to concentrate on the task at hand. This DSS interfaces with an automated wing procedure. The procedure currently used at Holloman AFB supplies each fighter squadron with daily flying information on a computer disk. This DSS processes the data residing on the disk and automatically displays the scheduling framework, eliminating the need for grease boards. The DSS database tracks the availability of personnel, thus replacing two more scheduling grease boards. Elimination of the grease boards frees the scheduler from the time-consuming process of manually updating them. As a result of this DSS, more quality flying training will be accomplished.

AFIT/GST/ENS/87M-6

DESIGN EVOLUTION OF A FIGHTER TRAINING SCHEDULING DECISION SUPPORT SYSTEM

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science In Operations Research

Paul E. Trapp, B.S., M.A.
Captain, USAF

Jeffrey W. Grechanik, B.S.
Captain, USAF

March 87

Preface

This work originated from the joint scheduling experience of Captain Paul E. Trapp and Captain Jeffrey W. Grechanik. Our work in the field as squadron and wing schedulers showed us the inefficiency of current scheduling procedures.

We are deeply indebted to our thesis advisor, Lieutenant Colonel John Valusek, for his support and encouragement. His insight and guidance were instrumental in this thesis effort.

We would also like to thank Capt Votipka and Lieutenant Colonel Kunzman at Holloman AFB for their cooperation and support.

Capt Trapp would like to thank his supportive wife, Leesa, and children, Jonathon and Katrina, who gave him the motivation to accomplish what he otherwise could not have done on his own.

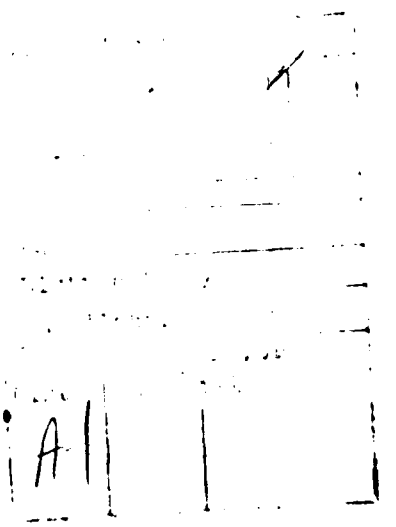


Table of Contents

List of Appendices

List of Figures

List of Tables

List of Abbreviations

AFB	Air Force Base
AFIT	Air Force Institute of Technology
AFORMS	Air Force Operational Resource Management System
ASCII	American Standard Code for Information Interchange
BCP	Back Cockpit
CAPT	Captain
CDR	Commander
CONFIG	Configuration
CP	Crew Position
CPT	Captain
DBM	Database Manager
DEV	Deviation
DNIF	Duty Not Involving Flying
DSS	Decision Support System
ES	Expert System
FCF	Functional Check Flight
FCP	Front Cockpit
FLT	Flight
FLTLD	Flight Lead
GST	Graduate, Strategic & Tactical Sciences
ILP	Integer Linear Programming
INST	Instructor
INSTR	Instructor
IP	Instructor Pilot
IW	Instructor Weapons System Officer

LTC	Lieutenant Colonel
MAJ	Major
MCDT	Multi-Criteria Decision Making
MOE	Measure of Effectiveness
MSN	Mission
MTGS	Meetings
OS	Operational Sciences
OSC	Oscura Bombing Range
PCS	Permanent Change of Station
RCO	Range Control Officer
RDIP	Red Dot Instructor Pilot
ROMC	Representation, Operation, Memory aids, Control Mechanisms
RRIP	Red Rio Bombing Range Instructor Pilot
SA	Studies and Analysis
SAS	Statistical Analysis System
SCHED	Schedule
SIM	Simulator
SOF	Supervisor of Flying
TAC	Tactical Air Command
TDY	Temporary Duty
TFTS	Tactical Fighter Training Squadron
TL	Time Line
TTW	Tactical Training Wing
UIPIP	Upgrading Instructor Pilot
UIPRR	Upgrading Red Rio Instructor Pilot
WX	Weather

Abstract

The primary objective of this research effort was to trace the evolution of a decision support system for a fighter training unit scheduler. The purpose of a Decision Support System (DSS) is to assist the cognitive processes of judgment and choice as performed by a squadron scheduler.

Fighter squadron daily scheduling is a complex and time consuming task. Currently, squadron scheduling is accomplished using plexiglas grease boards and manual tracking systems. It is a manpower intensive task requiring vast amounts of cross-referencing and a great deal of attention to detail.

This DSS is built using the adaptive design process. The adaptive design process includes development of a concept map, analysis of tasks and data, and identification of the kernel to select the central decision process. The designers employ this central decision process to develop a feature chart and storyboards, the starting point for the DSS construction.

Ideally, the DSS begins with a small prototype that decision makers use and evaluate. Builders then customize and modify the DSS as a result of user feedback. This process repeats, incorporating user needs and requirements. Thus, the DSS improves with each successive iteration.

This particular DSS automates the squadron scheduling decision process without interrupting the schedulers ability to concentrate on the task at hand. This DSS interfaces with an automated wing procedure. The procedure currently used at Holloman AFB supplies each fighter squadron with daily flying information on a computer disk. This DSS processes the data residing on the disk and automatically displays the scheduling framework, eliminating the need for grease boards. The DSS database tracks the availability of personnel, thus replacing two more scheduling grease boards. Elimination of the grease boards frees the scheduler from the time-consuming process of manually updating them.

Elimination of the grease boards saves time, thus improving scheduling efficiency. Currently, near the end of the day the scheduler may not have time to correct last minute changes to an already complex schedule. Lack of time may result in disastrous consequences. Mistakes usually occur at this point, in which case crews may not fly missions. This DSS minimizes the time necessary to update a complicated schedule accurately, thus giving the scheduler more time to assign individuals. As a result of this DSS, more quality flying training will be accomplished.

DESIGN EVOLUTION OF A FIGHTER TRAINING SCHEDULING DECISION SUPPORT SYSTEM

I. Background

Problem Description

The squadron scheduling job at the 434th and 435th training squadrons at Holloman AFB, New Mexico is a time consuming, hard, complicated, and thankless job. The schedulers favorite saying is, 'a perfect job merits no increase in punishment.' With two classes of about 25 - 30 students each, the schedulers work with a confusing maze of syllabus and squadron rules. The 435th works with seven different syllabi; the 434th requires only four different syllabi. Near the end of the day, when time is a factor, small changes result in catastrophes for the next day's schedule. The following five elements are part of the scheduling problem:

1. Manual Scheduling.
2. Deconfliction.
3. Lack of Alternatives.
4. Unaccomplished Prerequisites.
5. Mismatched Student/Instructor.

Manual Scheduling. The first of these elements deals with the intensive manual tracking of all duties on several grease boards. The scheduler writes the duties manually

onto the boards. Table 1.1 lists the grease boards a scheduler works with on a daily basis.

TABLE 1.1

Duty Scheduler's Grease Boards

1. Today's Duty Schedule
2. Tomorrow's Duty Schedule
3. Monthly Instructor Duties
4. Student Training Board
5. Deconfliction Board
6. Squadron Schedule Board

Today's Duty Schedule. The schedulers keep the current day's flying schedule to note any deviations that happen during the day. Due to late takeoffs, changed range times, or lack of fuel, the current day's schedule may change. Tomorrow's duty schedule takes into account any discrepancies from the current day's schedule

Tomorrow's Duty Schedule. The duty scheduler writes on a portable plexiglas board what each student and instructor is doing the next day. Figure 1.1 depicts a typical daily duty schedule found on a squadron grease board (range and area times have been left out for clarity).

T/O	FCP	BCP	T/O	FCP	BCP	19 Sept RCO Ltc Becker SOF Ltc Franzel 0620-1200
0700	Smith	Jones	1100	Smith	Jones	
"	Becker	Bohan	"	Becker	Bohan	
0715	Dawson	Deaux	1115	Doelp	Donald	DUTY HOG 0530 Valusek 0800 O'Connell 1100 Trapp 1500 Grechanik
"	Freil	Fussel	"	Gable	Gross	
0745	Huffor	Hunsuk	1145	Kline	Kokal	
"	Allard	Aller	"	Casey	Daniel	
0800	Linnel	Marvin	1200	May	McGraf	SIMS 0845 Frederick/ Heart 1145 Faix/ Valusek 1345 Shirasago/ Fella
"	Miller	Minear	"	Miller	Minear	
0815	Hoser	Adams	1215	Lutz	Charpy	
"	Briand	Schoek	"	Conors	Evert	
1000	Laurel	Hardy				ACADEMICS Class A 0700-1000 Rowell
"	Big	Little				
1030	Larry	Moe				
	Harry	Jack				
						MEETINGS 0900 Franke Dentist 1200 Flight Commanders 1730 Aircrew Meeting

REMARKS		Hoser	F4	Dawson	SA3	Briand	F3
Smith	B6	Becker	B6	Huffor	A2	Linnel	A2
Miller	A4	Laurel	F4	Big	F3	Larry	SA4
Allard	A2	Freil	SA3	Smith	B7	Doelp	I4
Gable	SA4	Kline	SA4	Casey	SA5	May	A3
Miller	A5	Lutz	I1	Conors	I3		

Figure 1.1. Duty Schedule

Monthly Instructor Duties. The scheduler writes the instructor pilot monthly duties on another grease board. Instructor Pilots (IPs) are those experienced pilots that have at least 400 hours in fighter-type aircraft. An example of his weekly duty might be Supervisor of Flying (SOF), Range Control Officer (RCO), or Duty Scheduler (see Appendix K). Each duty requires from half to a whole day's effort to accomplish.

Student Training Board. Another plexiglas board that requires intensive manual tracking is the student training board. Students going through Lead-In Fighter Training (LIFT) must complete 80 events during their 43 training days. The student training board tracks the completion or failure of student training events. With 30 students in training, the scheduler manually tracks about 50 events each day on the board. Figure 1.2, depicting this board, shows the students and the events they must accomplish. Upon event completion, the scheduler enters a date in the appropriate square.

Deconfliction Board. The scheduler enters the event and time onto the deconfliction board when he schedules any personnel. The scheduler references the deconfliction board to see if he scheduled more than one event at the same time. This board ensures that the squadron schedule does not contain any conflicts.

C TRACK	TSM1R	TR1R	TR2R	TR1	ST1	TSIM1	F1-2	TR2	F1	F2	F3	F4	TSIM2	I1	F5	F6	TSIM3	I2
BRUGNOL	X	X	X	X		9		11	X	14	15	16	21	21	22	24	22	25
DUPUIS	8	9	10	11		11		14	16	17	21	22	22	13	23	23	8	19
ERICKSO	X	X	X	X		9		11	X	14	15	16	21	21	22	24	22	25
ESTRELL	8	10	16	17		18		21	22	23	24	24	28	4	25	28	21	
IWANIVK	8	9	10	11		11		14	16	23	24	25	23	15	28	29	13	19
JOHNSTO	X	X	X	X		9		11	X	14	15	21	22	22	23	24	11	13
LARA	8	10	16	17		15		21	22	23	24	25	28	15	28	29	11	20
LEMIEUX	X	X	X	X		X		9	X	X	11	18	21	29	22	23	24	X
SETTER	X	X	X	X		9		11	X	15	21	23	23	5	23	24	11	
B TRACK																		
VERNACK	X	X	X	X		X		9	X	X	4	18	21	14	22	23	11	X
BOBBITT	X	X	X	X		9		11	X	15	21	22	23	6	21	25	14	20
CAVUOTI	X	X	X	X		11		16	X	17	21	22	23	30	24	28	18	21
KELLEY	X	X	X	X		15		16	X	17	21	22	23	7	25	28	11	21
KRAPF	X	X	X	X		X		11	X	X	15	21	22	5	23	25	15	X

Fig. 1.2 Student Training Board

Squadron Scheduling Board. The squadron scheduling board is a large plexiglas board displaying both today's and tomorrow's schedules. This board shows the complete schedule for all squadron personnel. It incorporates all the meetings and appointments for everyone.

The day prior, the scheduler must know anything that will keep one of the assigned squadron personnel from participating in the next day's activities. The scheduler manually tracks special notes on small cards. If a student or IP cannot fly for medical reasons (e.g., a cold or flu) he is put into Duty Not Involving Flying (DNIF) status. The scheduler does not consider DNIF personnel for the next day's flying schedule. For example, an IP with a future dental appointment writes the date on a small card and gives it to the scheduler.

Manual tracking of all the above procedures can result in inaccuracies. A typical day's manual tracking procedure follows. Upon completion of the day's flying, the duty scheduler writes the events on the student training board. An event is an occurrence of an aircraft ride or other required procedure for an individual. Instructor pilots that accomplish SOF or RCO are also annotated. Throughout the day, the duty scheduler works the next day's schedule as if all of the current day's events were accomplished. Once the next day's schedule is complete, administrative personnel copy it to the squadron scheduling board for all of the

squadron personnel to check. Since the scheduler manually updates each board daily, there is a chance for error.

Deconfliction. The duty scheduler maintains a deconfliction board to prevent scheduling two events at the same time for a single individual. During the day the duty scheduler continually updates the deconfliction board. In theory, the board should provide deconfliction for everyone. Yet, near the end of the day, when the scheduler makes last-minute changes to an already complex and interwoven schedule, he can easily make mistakes. Many IP's are already scheduled for two daily events and any change to their schedule often goes unnoticed until it is too late.

Lack of Alternatives. Many times the duty scheduler has woven a tight knit schedule that comes unraveled at the last minute due to a 'ripple effect' through his schedule. Last minute changes to the schedule require the duty scheduler to look for alternative IPs or students to fill slots. By 1300-1500 hours each day the duty scheduler has matched instructors and students with takeoff times. The schedule for the next day must change if a failed ride occurs or an unknown meeting arises after 1500. The scheduler consults several boards to make a change to this intricate schedule.

The scheduler must rapidly evaluate:

1. Which students are available and at what time?
2. Which instructors?
3. If the duty scheduler changes the event time of one person, does it affect an earlier event?
4. Changes may violate crew rest rules and thus affect flight safety.

Unaccomplished Prerequisites. A student must accomplish about 80 different events in the proper order during his stay at Holloman. For example, to fly his first ride a student must accomplish six hours of life support training, one hour of squadron briefing, and one hour of simulator training. If the scheduler puts a student in his first flight without having accomplished these specific events, he is unprepared to fly and may endanger his life. Therefore, to ensure that the students do their prerequisites before the flight, the scheduling shop tracks the events on another large plexiglas board (Figure 1.3). This tracking requires daily intensive manual effort so no one flies without completing their prerequisites.

Mismatched Student/Instructor. Schedulers must properly match students and instructors before flying together. Squadron supervisors screen the students upon arriving at Holloman to determine if they have any special instructional needs. An example would be if a student had spent one-and-a-half years at AFIT, then four years at the Pentagon in Studies and Analysis. Because of his extended absence from flying, the student would assume 'red dot' status and could fly only with red dot instructors.

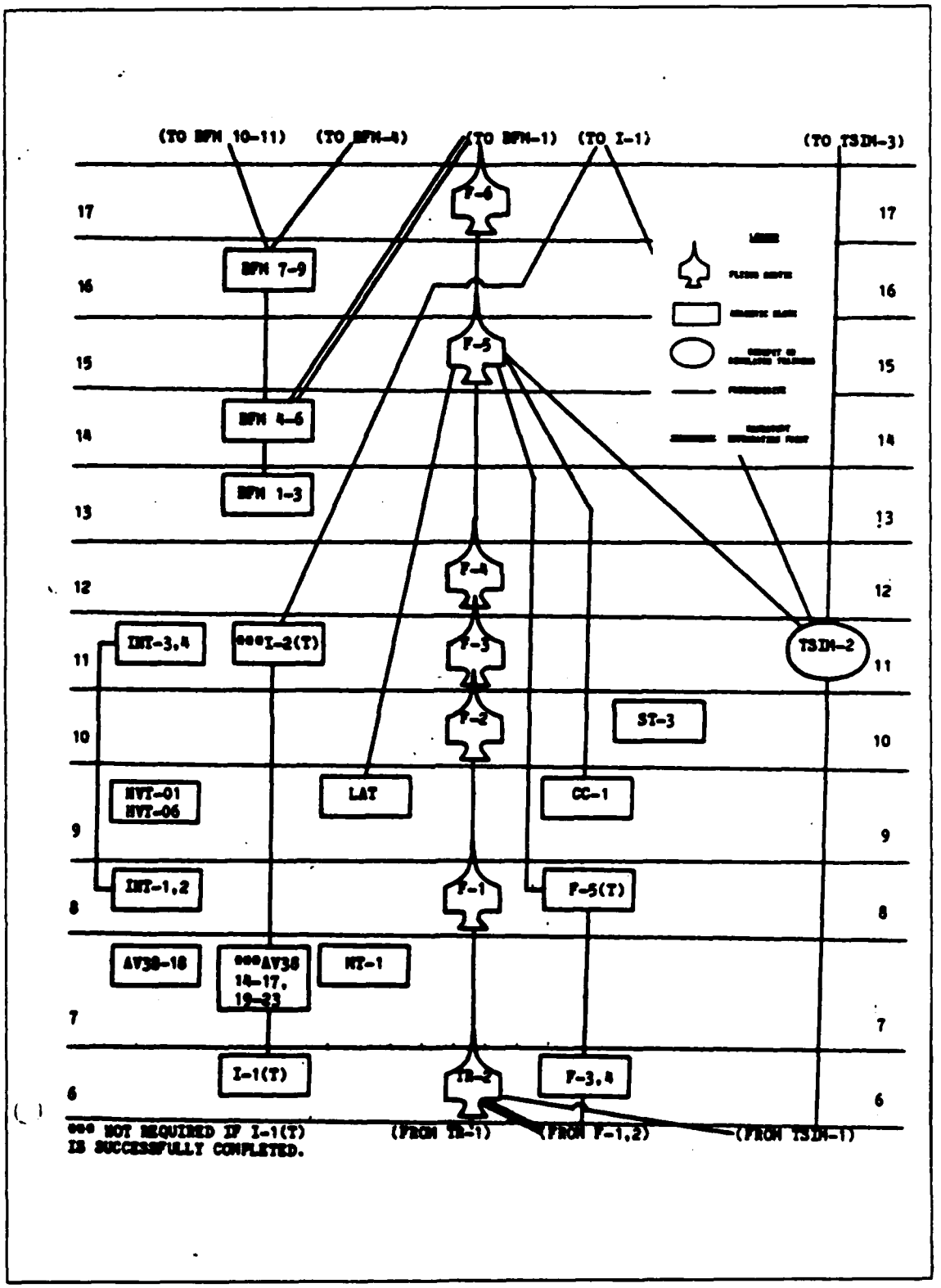


Fig. 1.3 Syllabus Student Training Flow

The operations officer selects experienced instructors to be red dot instructors. They usually have had at least one year of flying with students. If the scheduler allows a red dot student to fly with a non-red dot instructor pilot, he compromises flight safety and a dangerous situation results.

The problem presented in this chapter provided a reference point for the research. To explore avenues and approaches to the problem, it was necessary to perform a literature search. This search reviewed some past approaches, efforts, or solutions to the scheduling problem.

Literature Review

The discussion presents previous research efforts in the area of scheduling in four parts. The first part deals with a general review of the various types of scheduling problems and solution techniques. The next portion deals with the network approach to solving scheduling problems. The third part overviews the use of simulation as a scheduling tool. The final part looks at the decision support system (DSS) approach.

Scheduling Research. A general review of the various types of scheduling problems and solution techniques yielded the following problems: Job shop scheduling, maintenance scheduling, and cyclical scheduling.

Job Shop Scheduling. Job shop problems scheduled items moving through a shop. The purpose was to schedule

each job in a specific order at the proper time or in a random order (depending on the job shop). Each item may have required a visit to each machine a certain number of times, or it may only have visited some subset of the machines in the shop. Moreover, there may have been several types of jobs which required different subsets of machines or different types of flow patterns. Solutions to job shop problems usually used integer linear programming (ILP) or heuristic techniques. Often, these solutions were developed for specific problems rather than providing a general solution for all job shop problems. The literature in this area dealt with finding algorithms or exploring how to solve the same problem using different solutions.

Hosios used a heuristic algorithm that scheduled a minimum number of personnel for completion of a set of activities every scheduling period (6:749). The activities occurred at various times and locations. The algorithm assumed that the personnel can complete the activities in any order.

Student scheduling at Holloman requires the completion of a set of activities every scheduling period. However, the order of completion is important and adds to the student scheduling problem. Although Hosios does not consider completion order in his article, all personnel complete all activities. This is similar to the student scheduling problem.

Maintenance Scheduling. Problems in this area dealt with repair crew assignment, optimal maintenance facility flow, or manpower requirements (13:333-340; 23:2770-2775). The objective of maintenance scheduling was to minimize cost by either reducing manpower requirements or reducing repair costs. This minimization required an optimal allocation of repair crews and money to the equipment being maintained.

Both maintenance and student scheduling required the assignment of scarce resources. Both types of scheduling must consider a structured schedule of events (e.g., tech-orders for maintenance and syllabi for student scheduling). In addition, both techniques dealt with manning requirements. The drawback of maintenance scheduling was that it was not flexible enough to deal with the constant changes in the training arena.

Cyclical Scheduling. This involved the scheduling of people for shift work or a schedule of on- and off-days, such as nurse scheduling. Warner developed a nursing schedule that considered weekdays off, work stretches (consecutive work days), single days off, and undesirable work patterns (back-to-back shifts). Integer linear programming and heuristics were the main techniques used to solve these types of problems.

However, cyclical scheduling fell short of providing all the answers for the student scheduling problem in two

areas. First, users must schedule more than one type of activity for students. The Holloman schedule includes classroom lecture, simulator training, and flying sorties. Second, cyclical scheduling literature does not consider leave, DNIF, TDY, and other disruptions. Therefore, cyclical scheduling will not be used (3:1-16).

Goal Programming. Arthur and Ravindran proposed a goal programming approach to the nurse scheduling problem (2:55-60). Goal programming incorporated several goals into the objective function, seeking to simultaneously optimize all the goals. This optimization technique provided an added degree of flexibility to the scheduling program in that it would substitute different goals into the program if desired.

Warner's use of a set of weighted criteria was very similar to goal programming. Multi-criteria decision theory (MCDT) used a set of weighted measures of effectiveness (22:842-856).

Network Approach. Roege used integer programming, based on branch and bound techniques, to solve pilot scheduling in a fighter squadron (16:50). Roege used TAC Manual 51-50 training requirement minimums as lower bound constraints. The model developed considers crew rest restrictions and absences from duty. His model ensured that each pilot received at least a minimum and no more than a maximum number of flights per week.

The main drawback of Roege is that he considered only experienced pilots (there is no student training or instructor/student matching). In addition, he did not consider any extra duties such as SOF, RCO, or Duty Officer (see Appendix K). Roege did start his formulation with a 'shell.' A shell is a listing of takeoff times, number and type of sorties, and configuration of the aircraft. The shell is a very important starting point for the student scheduling problem at Holloman AFB.

Simulation Approach. Berg used simulation to assist the scheduling of missile crews (3). Berg took several user input factors and automated them to produce a schedule. Missile crews flowed through a network that completed events at specified times. MCDT techniques and Response Surface Methodology provided a worth assessment of each schedule.

Berg's idea of assisting the scheduler is important in the student scheduling problem at Holloman. However, he still proceeded with the 'push-button approach' to scheduling. That is, the machine or algorithm scheduled for the user at the push of a button. This concept made it inappropriate for the student scheduling problem at Holloman. Schedulers must know the reasoning behind the decision process. In addition, scheduler interaction is essential to have a flexible and robust scheduling system.

Holloman's Efforts. Capt Votipka used LOTUS 1-2-3

on a Z-100 computer in an attempt to schedule students and instructors. He scheduled students and instructors on a time line type screen display. Capt Votipka attempted this at Holloman AFB, New Mexico at the 479th Tactical Training Wing. His attempt found limited success for the following reasons:

1. The computer screen could not show all flights on one screen.
2. The refresh rate of the screen was too slow for the users.
3. The computers were unfamiliar and hard to use for the untrained user.

Decision Support Systems

The above techniques did not specifically concern themselves with the decision maker and the information required to make effective decisions. Gonin and Moffett 'argue that complex decision making requires human interaction with .... graphical displays that are in the spirit of the Decision Support approach (10:9).' According to Maj. Valusek, a DSS is a system, manual or automated, that assists the cognitive processes of judgment and choice (20).

A DSS would allow the scheduler to recall needed information from the database, consider the data presented, make a decision, and input the decision into the schedule. He updates the schedule to determine the effect of the decision. In this way users build schedules that reflect the decision maker's desires in an efficient manner.

Attempts to solve the student scheduling problem with the push of a button would be distrusted. Even if a computer algorithm solved the problem instantly, the unit commander would be wary of a solution with no human in the decision loop. Humans solve the problem quite well every day by heuristic decision processes. A DSS assists in the decision process rather than making decisions.

Why Adaptive Design

Adaptive design is an approach to problem solving for semi-structured and unstructured problems (structure is the amount of definition a problem has). A small 'kernel system' grows or changes as requirements are added or modified. A kernel system is that part of a complete decision process considered to be the essential core of that process.

This small kernel provides the starting point about which the system will grow. The system will expand around user needs and requirements. As user needs change, the system adapts to these new requirements. This adaptive ability is an advantage over the complete system approach. Steps in the adaptive design process are:

1. Select the Right Problem
2. Identify key kernel
3. Iterative Design
4. Implementation

The complete system approach 'freezes' user requirements before the start of construction. Because a 'complete

system' takes so long to develop, user needs change. If these needs have changed, the system, when fielded, no longer reflects current user requirements. In addition, the user's perceptions or environment may change. The chance that these views match the 'complete system' characteristics is remote. Therefore, the complete system method is inadequate for a dynamic, flexible problem such as scheduling.

Student scheduling at Holloman is still a time consuming, complex task. Schedulers routinely make mistakes which may affect flight safety. This is normally the result of either scheduling someone for two different events at the same time or violating one of the many rules that exist. An example might be, in the former case, scheduling students simultaneously for class and a simulator training ride. Another example is scheduling a student to fly the night before an early morning flight, as in the latter case. Not only does this affect flying safety, it is a large price for the duty scheduler, his commander, and the squadron to pay for many hours of careful scheduling should an accident occur.

The problems of manual tracking, deconfliction, unaccomplished prerequisites, mismatched instructor/student crews, and insufficient alternatives make it a hard, but possible, task to computerize. However, allowing the squadron itself to create a scheduling program that it <u>can use</u> and <u>evolve</u> is a new approach. With the introduction of

powerful Zenith model 248 (Z-248) microcomputers available at the unit level (8), each squadron will be able to adapt a kernel scheduling system to meet its own needs. Of course, this project needs technical expertise. The wing must be able to provide support to change and adapt the program as new requirements or problems arise. Squadron and wing scheduling at Holloman should work closely to further the computerization of their scheduling system.

Research Problem

Schedulers currently have no means to adequately and quickly manipulate a vast database to produce an accurate and timely student flying training schedule. Although scheduling rules and restrictions are quite clear, time constraints introduce problems. The scheduler must precisely and quickly interpret a large amount of information.

Research Objective

The research objective was to trace and document the evolutionary design process of a DSS that assists in scheduling daily student flying training. The effort used appropriate software on a Z-248 microprocessor in the 434th Flying Training Squadron at Holloman AFB. In addition, this effort showed how end users can use off-the-shelf software on Z-248's to build their own decision aids.

Subsidiary Objectives

1. Document the adaptive design process of the DSS.
2. Use a kernel identification process to identify and test the core of the decision process.
3. Create a picture of what the program should look like before development ("Storyboard").
4. Maintain a future goals section that are not currently attainable ("Hookbook").

Scope, Limitations, and Assumptions

This thesis focused at the unit level duty scheduler at Holloman AFB in the 434th and 435th Squadrons. The authors designed the program specifically for the Z-248 microprocessor. The scheduling system did not make any decisions for the scheduler, but assisted him and supported decisions he made. Emphasis of this research focused on the evolutionary design approach to build the DSS kernel.

Chapter II deals with the approach selected. Chapter II also covers the reasons for DSS use and software selection. Chapter III shows how the techniques described in Chapter II apply to a specific DSS. Chapter IV describes the resulting system and how it varied from the planned system. Chapter V addresses conclusions and recommendations of the scheduling system and the adaptive design process in general.

II. Problem Approach

Initial Conception

The initial idea was to build a scheduling tool to assist the squadron scheduling for the training squadrons at Holloman AFB, New Mexico. The idea seemed appealing for the following reasons:

1. Previous Scheduling Experience;
2. Microcomputers were in all of the squadrons;
3. The problem seemed like it should be solvable;
4. Operations Research 'optimizes', so Operations Research should be able to help in the scheduling area;
5. All TAC squadrons schedule essentially the same way (man-hour intensive process);
6. The topic was unclassified; and
7. There is nothing currently in the field to help at the squadron level.

Approach Selection

After investigating approaches, there were two strong candidates, expert systems (ES) and DSS (14;20). Turban and Watkins compared ES and DSSs in their article. Table 2.1 highlights the differences between ES and DSS.

TABLE 2.1

The Differences Between ES and DSS

	DSS	ES
Objective	Assist human	Replicate (mimic) human and replace him/her
Who makes the decision?	The human	The system
Major orientation	Decision making	Transfer of expertise (human-machine-human)
Query direction	Human queries the machine	Machine queries the human
Clients	Individual and/or group user	Individual user
Manipulation	Numerical	Symbolics
Problem Area	Complex, integrated, wide	Narrow domain
Database	Factual knowledge	Procedural and factual knowledge

(19:141)

The authors selected DSS because it offered an approach to complex, integrated problems. Machines assist, but do not replace, the decision maker. ES, on the other hand, suggested rather than supported decisions. 'ES typically involves a closed-world assumption, that is, the problem domain is circumscribed, and the system performance is confined within those boundaries' (9). In DSS contexts, the world was open. A DSS must be flexible and adaptive to meet the changing conditions in the environment and the evolving needs of the user (18).

Data Collection

A TDY to Holloman AFB, New Mexico (23-26 Aug 86) revealed that the scheduling problem still existed. Interviews with several wing and squadron schedulers revealed the scheduling problems shown in Table 2.2.

TABLE 2.2

Scheduling Problems at Holloman AFB

1.	Undocumented DNIF, Leave and TDY
2.	Last-minute changes wrought havoc
3.	All flights not shown on one screen
4.	Deconfliction of schedule often inaccurate
5.	No sorting or prioritization capability
6.	Intense manual tracking was time consuming and often produced errors

The 479th wanted the project automated to correct the above problems. In fact, the 479th wing scheduling had already automated their portion of the daily flying schedule. The 479th wing scheduling sends the daily sortie information, generated by a computer program, to the squadrons on a floppy disk. Table 2.3 shows an actual copy of a file used on 3 March 1987 to give sortie information to the 434th TFTS.

TABLE 2.3

479th Wing Scheduling File

Line #	Unused	Takeoff time	Land time	Unused	Area time	Msn type	Config	Area
		(minutes past midnight)						
401	0	510	610	1	520	550	SA	B-1 OSC
402	0	510	610	1	520	550	SA	B-1 OSC
403	0	510	610	1	520	550	SA	B-1 OSC
404	0	510	610	1	520	550	SA	B-1 OSC
405	0	555	655	1	565	605	B	E-1 BK-C
406	0	555	655	1	565	605	B	E-1 BK-C
407	0	570	670	1	580	610	SA	B-1 OSC
408	0	570	670	1	580	610	SA	B-1 OSC
409	0	570	670	1	580	610	SA	B-1 OSC
410	0	570	670	1	580	610	SA	B-1 OSC
411	0	690	790	1	700	730	SA	B-1 OSC
412	0	690	790	1	700	730	SA	B-1 OSC
413	0	690	790	1	700	730	SA	B-1 OSC
414	0	690	790	1	700	730	SA	B-1 OSC
415	0	735	835	1	745	785	B	E-1 BK-B
416	0	735	835	1	745	785	B	E-1 BK-B
417	0	750	850	1	760	790	SA	B-1 OSC
418	0	750	850	1	760	790	SA	B-1 OSC
419	0	750	850	1	760	790	SA	B-1 OSC
420	0	750	850	1	760	790	SA	B-1 OSC
421	0	870	970	1	880	920	F	E-1 BK-B/BK-C
422	0	870	970	1	880	920	F	E-1 BK-B/BK-C
423	0	870	970	1	880	920	F	E-1 BK-B/BK-C
424	0	870	970	1	880	920	F	E-1 BK-B/BK-C
425	0	915	1015	1	925	965	B	E-1 BK-C
426	0	915	1015	1	925	965	B	E-1 BK-C
427	0	930	1030	1	940	980	F	E-1 TL-E/TL-W
428	0	930	1030	1	940	980	F	E-1 TL-E/TL-W
429	0	930	1030	1	940	980	F	E-1 TL-E/TL-W
430	0	930	1030	1	940	980	F	E-1 TL-E/TL-W

The 479th wing computer manager, Capt Votipka, mentioned that the wing had ordered six Z-248's, two for wing and one for each squadron. Capt Votipka also mentioned ENABLE, an integrated software package, came with the Z-248's (21).

Methodology

The method used to solve this particular problem stems from an evolutionary design process and its application to a DSS capable of assisting unit schedulers at Holloman AFB. The entire scheduling process must satisfy user needs when builders construct the DSS. The authors used four essential steps to construct this system and achieve the objectives.

The first step involved documenting the daily decision process schedulers use to generate a workable schedule. This task involved building a conceptual map, or network, of the scheduling process followed by a task and data analysis. These analyses trace the flow of information used in building the daily flying lists. Once builders understand this network, it is necessary to organize it in as simple a way as possible.

The next step involves storyboarding, or designing initial computer screen representations of the decision process (1). The scheduler should have all of the information needed to construct the daily flight agenda. On the other hand, this screen format must present enough of the 'big picture' of the next day's tasks to minimize errors and oversights.

The third step is to encode a kernel program, using acceptable software, which will implement the storyboard output and provide as much assistance to the scheduler as needed. The software must support the storyboarded format.

In addition, it must be powerful enough to support changes or modifications as user requirements evolve.

Finally, once the builders construct the kernel, assessment by the users at Holloman is necessary to obtain feedback needed for DSS evolution. This will entail actual 'hands-on' operation of the program to stimulate user comments and critiques of the ability of the DSS to aid scheduling. Taking the feedback into account, further modification or adaptation of the scheduling DSS may take place. In this way, builders may track successive generations of the system to trace the design evolution of a DSS.

Problem Definition

Problem definition is perhaps the most difficult part of building a DSS. The unstructured nature of a DSS problem complicates matters because the true problem may lie beneath layers of information or heuristic processes. Keeping this in mind, the two steps to follow in defining the problem are recognition and identification.

The first step, problem recognition, is the easiest of the two steps. When procedures or tasks within an organization do not run smoothly, it is easy to recognize that some problem exists. If this difficulty surfaced recently, comparison with past events or procedures provides the reference with which to compare the problem's nature and extent. On the other hand, if the organization is looking to improve

(i.e., trying to discover any problems) or streamline their operation, the problem may not immediately expose itself. At this point, knowledge of the specific organizational procedures is necessary to understand the scope of the problem. It may be helpful to interview the novice employee first, as he does not have the expertise required to overcome or circumvent daily operational hurdles. Problems, in his eyes, are magnified and easily recognized. The interviewer must exercise caution, however, because the novice often has little experience with his environment (the way his work interacts with the organization). As such, his perception may exaggerate the problem or prove inconsequential to the decision process. Collaboration with the experts, then, should reveal whether the problem is valid. Chapter III covers problem recognition for the specific scheduling DSS. Once the interviewer is certain a problem or area of improvement exists, the next step is to find out just what that problem is.

Problem identification normally requires the aid of the problem system experts due to the complex nature of an organization's specific decision requirements. These experts are necessary to interpret the problem fully and understand it's impact on both the decision and the environment. Through successive interviews and queries with these experts, the DSS builder gains insight about the problem facing the decision maker.

For this specific DSS, the builders are the scheduling experts. Each author is experienced in fighter training squadron scheduling. Therefore, the authors drew upon their experience for problem identification. The next step in problem identification is the concept map.

Concept Map Development

Conceptual mapping outlines each portion of the decision process using words or concepts and linking words or phrases in a hierarchical depiction (12:3). Conceptual mapping captures the components of the decision-maker's process and provides a rough definition. Figure 2.1 graphically depicts the general method of developing a conceptual map. Chapter III describes the DSS developed for scheduling.

First-cut of the Decision Process. This process starts by independently interviewing each user to establish a common reference to the decision process. The interview focuses upon the general decision process, trying to identify clearly the methods each expert employs. This first-cut of the problem concentrates upon general topics for both simplicity and understanding. The users complete what they think to be the decision process using only word phrases and linking words. Once the interviewer annotates all of the decisions, the next step is to note the important ones.

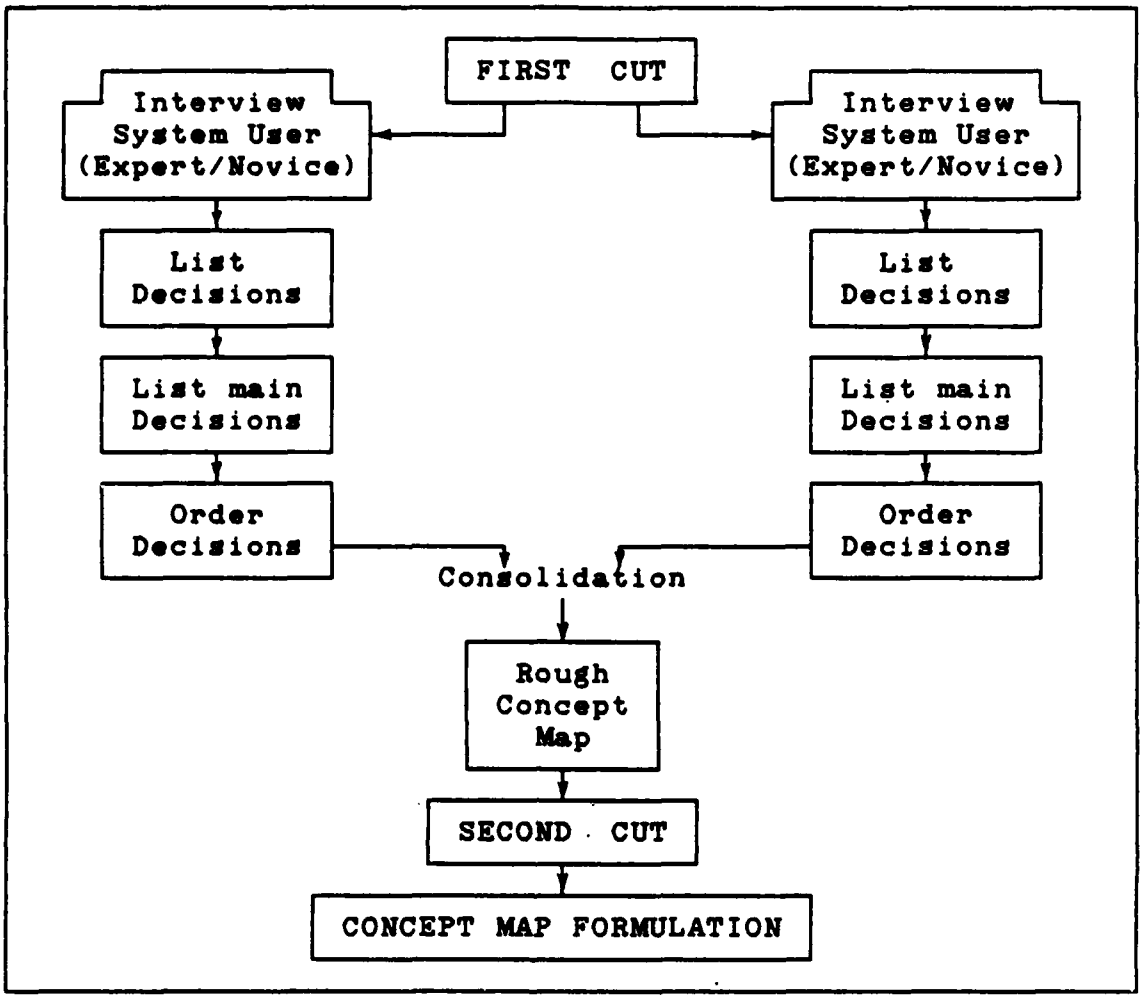


Figure 2.1. The Method of Developing a Conceptual Map

List Main Decisions. The users separate the main decisions from the first-cut list of all the decisions. The lists from each user are essentially the same as the first-cut lists but divided into main topics (decisions). Once the interviewer categorizes the decisions, the next step is to order the decisions.

Order Decisions. From these lists, the experts rank the concepts in a time-ordered manner (i.e., assign a number

to those concepts which must be performed in a certain order). The interviewer combines all the expert's decisions at this point.

Rough Concept Map. From all of the ordered lists, the interviewer consolidates the expert's individual lists into one network (Chapter III explains the actual networks). From these initial concept maps, a rough idea of the decision process structure emerges. An analysis of all the maps together provides a basic idea of the true decision process.

Second-cut of Decision Process. Prior to this point, the interviews were independent to reduce possible bias. Now the users meet as a group to critique the rough concept map. From this network, the users dissect each concept to arrive at an understandable decision process. More interviews with personnel of varying experience levels provide different results.

Concept Map Formulation. From the analysis done previously, the concept map reduces to major ideas. From this simplified concept map, the interviewer may identify the kernel(s). Later sections expand upon this concept.

Task Analysis

The task analysis represents the detailed steps needed to accomplish the decision. Patterned after the concept map, this analysis takes the form of a network and includes each task the scheduler (in this case) must perform and the

order he performs them in. This network, once completed, allows identification of the kernel as well as pinpoints possible bottlenecks within the system. Through experience, testing, and monitoring, the interviewer uses these choke-points to find the shortest path through the system. Once the process finds the path, the builders may create procedures to assist the DM.

Data Analysis

A further test of the task analysis validity is the data analysis. This procedure traces actual data throughout the task analysis to ensure compliance with the decision process. This procedure also helps map the data flow to ease construction of databases used in the DSS. Appendix D contains the database relations discovered by the tracing procedure.

Feature Chart Definition

With the task and data analyses completed, the next step in the system development was to construct the feature chart (17). This chart is a representation which encompasses the tasks and the features necessary to accomplish that task. The DSS should be able to assist users with varied experience and individual techniques, yet not be cumbersome to interpret or manipulate. At the same time the DSS

must be powerful enough to support any decision sequences the user may require. The latter requisite may be accomplished using an iterative approach over a period of time with the user employing the DSS and relying on the structure of the kernel to assist him. As such, this feature chart concerns the development of the screen output format based on the decision process and task/data analyses identified in the previous section.

The builders employ the Representation, Operations, Memory aids, and Control mechanisms (ROMC) user-builder interface technique (18:101-106) to develop the feature chart. Each of these tools enables the designer to translate user requirements into DSS components. This permits the translation of the user's needs to the builder's design requirements. Consideration of these aspects help build a single effective output screen in the storyboard.

Representation, the first tool, depicts the actual screen presentation needed by the user. This data representation must satisfy the user's needs in a clear and concise manner. The order of the representations must be logical, conforming to the user's decision or thought process sequence. This is especially important because the DSS should not distract the user as he thinks, rather it should ease his decisions by presenting him the next piece of information when needed. Should the user require closer look at a particular piece of data or focus on a portion of

the screen , manipulation of the representation may be necessary.

The second tool, operations, enable the user to manipulate the representations on the screen to suit his individual technique. This may take the form of actual data manipulation (e.g., sorting), scale change (e.g., viewing a larger or smaller portion of the screen), or adding/deleting various amounts of data for clarity or interpretation. The user must easily accomplish these operations at his convenience without interrupting his thought process.

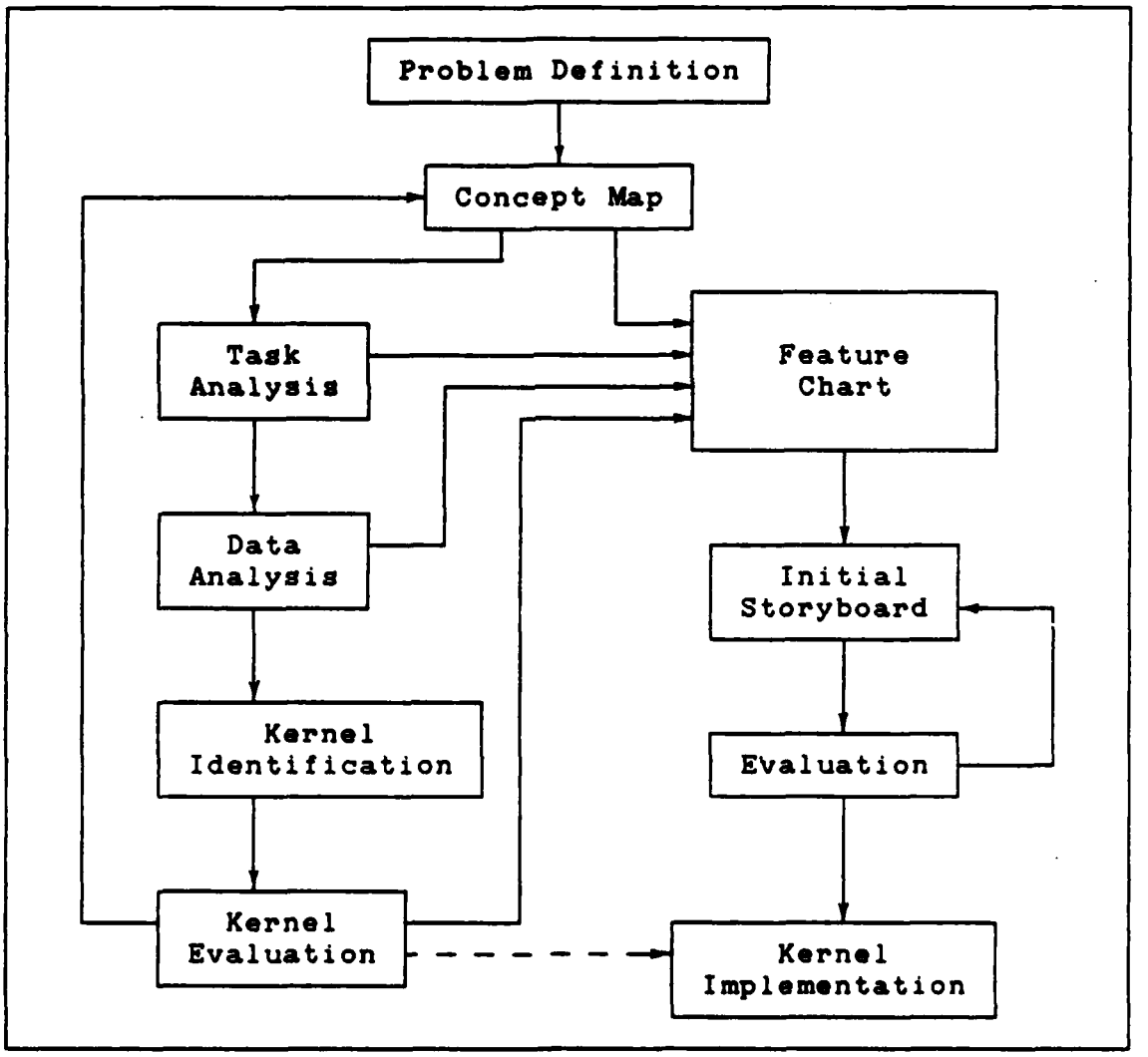
The next tool used in constructing the feature chart is memory aids. These helpful reminders guide the user through his decision process with the use of icons, windows, highlighted (colored) information, or various flags. All these methods serve to trigger or jog the user's memory, reminding or warning him of certain decisions.

Finally, control mechanisms are interwoven throughout the entire system, allowing any user to skip tedious or familiar processes. These mechanisms support the user's decision process, allowing ease of movement to any part of the system. Control mechanisms may take the form of selectable menus or predefined function keys.

Chapter III shows how the techniques described in this chapter were used to develop a scheduling DSS.

III. Specific DSS Development

The specific system development takes the problem described in Chapter I and uses the framework outlined in Chapter II to determine the key kernel. Figure 3.1 depicts the steps necessary to implement the key kernel. To ensure



(12:5)

Figure 3.1. Decision Support System Development Steps

the DSS provides the correct information, the builders use the steps described above. The process starts with the concept map development.

Concept Map

Conceptual mapping, as applied to this specific DSS, deals with the scheduler's decision procedure. It is the relationship (18:225-226) between the scheduler and the schedule he produces - a road map from blank paper to completed schedule. It outlines the scheduling decision process using words or concepts and linking words or phrases in a hierarchical depiction (12:3).

To construct the concept map, the builder must first understand the nature of the decision process. System experts provide the best source of information about an organization's specific processes. Since the builders for this specific scheduling DSS were also the experts, they already knew what was important. Thus, this characteristic is important to end-user application. Figure 3.2 depicts the initial decision process the authors envisioned. Capt Michael McFarren, conducting graduate research in the field of concept mapping and cognitive development, guided the authors through the mapping and analysis portions of the system development via a series of interviews.

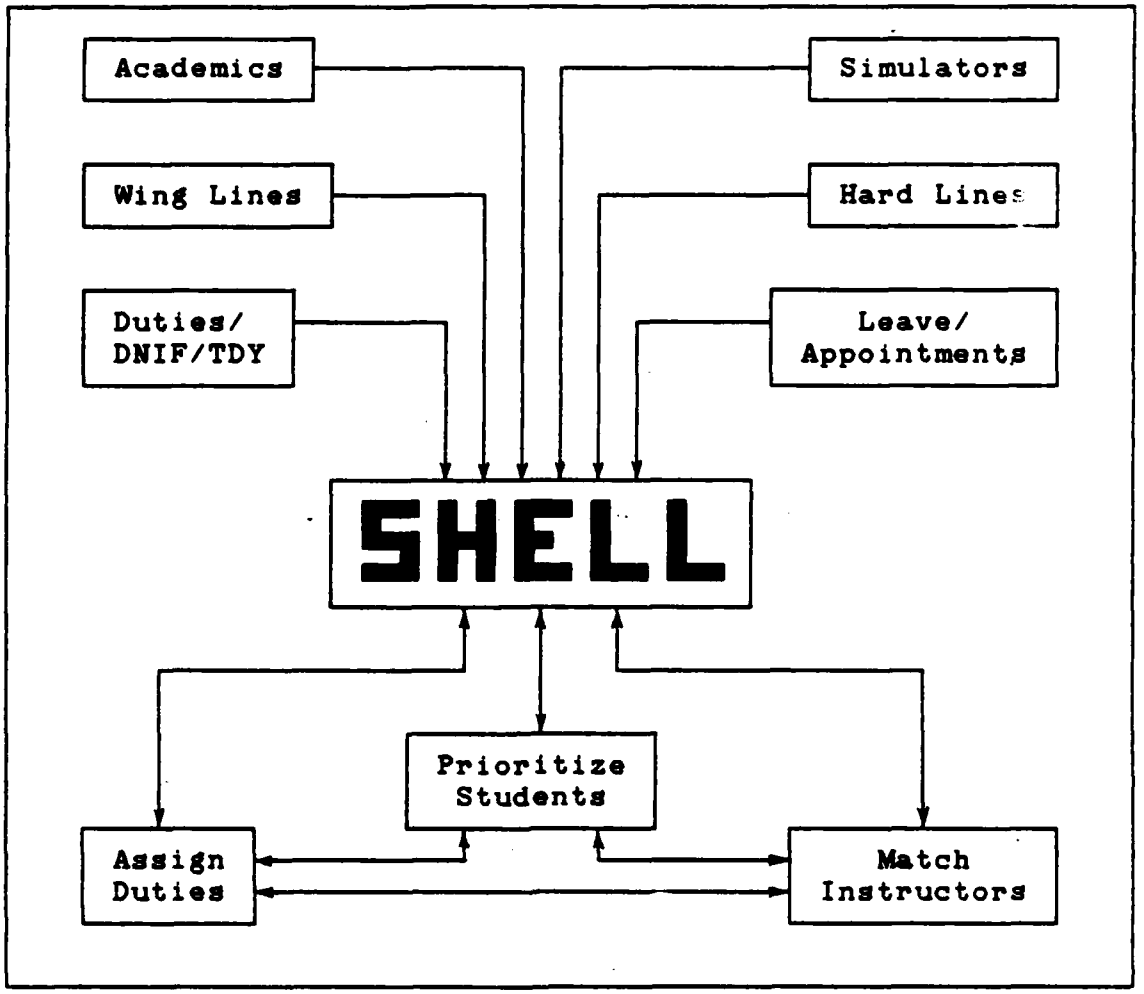


Figure 3.2. Scheduler's Decision Process

First-cut of the Decision Process. This process started by independently interviewing each author to establish a common reference to scheduling. Both authors completed what they thought to be the daily scheduling process using word phrases and linking words (Table 3.1).

TABLE 3.1

First Cut Conceptual Map Word Phrases

1.	Check yesterday's schedule for deviations.
2.	Gather tomorrow's DNIF inputs.
3.	Gather tomorrow's academic inputs.
4.	Gather tomorrow's TDY inputs.
5.	Gather tomorrow's leave inputs.
6.	Gather tomorrow's simulator inputs.
7.	Gather tomorrow's appointment inputs.
8.	Gather tomorrow's hard line inputs.
9.	Gather tomorrow's duty inputs.
10.	Fill Shell flights with students.
11.	Fill Shell flights with matched instructors.
12.	Fill Shell academic classes with students.
13.	Fill Shell additional duties.
14.	Fill Shell appointments.
15.	Fill Shell meetings.
16.	Deconflict throughout.

Appendix B contains the actual lists. Once the scheduling decisions were annotated, the next step was to annotate the important ones.

List Main Decisions. The authors, under Capt McFarren's supervision, separated the main decisions from the first-cut list of all the scheduling decisions. The lists from each author were essentially the same as the first-cut lists (see Appendix B), but divided into main topic areas. Table 3.2 summarizes the main tasks and decision areas found in both author's lists.

TABLE 3.2

Main Tasks/Decisions from First-cut Word Phrases

1. Check yesterday's schedule for deviations.
2. Gather tomorrow's inputs - determine priority.
   a. DNIF
   b. Academic
   c. TDY
   d. Leave
   e. Simulator
   f. Appointments
   g. Hard Lines
   h. Additional Duties
3. Fill Shell flights with students.
4. Fill Shell flights with matched instructors.
5. Fill Shell academic classes with students.
6. Fill Shell additional duties.
7. Fill Shell appointments.
8. Fill Shell meetings.
9. Deconflict throughout.

Order Decisions. From the above lists, Capt McFarren instructed the authors to rank the concepts in a time

TABLE 3.3

Ordered List of Main Decisions

1. Check yesterday's schedule for deviations.
2. Gather tomorrow's inputs - determine priority.
   a. DNIF
   b. Academic
   c. TDY
   d. Leave
   e. Simulator
   f. Appointments
   g. Hard Lines
   h. Additional Duties
3. Fill Shell academic classes with students.
4. Fill Shell additional duties.
5. Fill Shell flights with students.
6. Fill Shell flights with matched instructors.
7. Fill Shell appointments.
8. Fill Shell meetings.
9. Deconflict throughout.

ordered manner by assigning a number to those concepts which must be performed in a certain order (Table 3.3). Appendix B contains the actual lists. Note Table 3.3 varies only in order form Table 3.2. Capt McFarren combined both concept structures at this point, thus taking the next step in system development.

Rough Concept Map. From the ordered lists, Capt McFarren had the authors draw and order their individual lists into networks. Appendix B contains the actual networks. From these initial concept maps, a rough idea of the decision process structure emerges. Figure 3.3 shows a very complex and interwoven process a scheduler must wade through to complete a daily schedule.

Second-cut of Decision Process. Up to this point, the interviews were conducted independently to reduce possible bias. Also, there was no contact between the authors concerning the concept mapping. From this network, Capt McFarren and both authors dissected each concept to arrive at an understandable decision process. It may be noted here that only two subjects were used in the analysis. Appendix B contains the iterations of the decision process.

The major finding was that the whole process was very database intensive. This database took the form of manual grease board tracking schemes for personnel events. In fact, there was a question whether there was any need for a

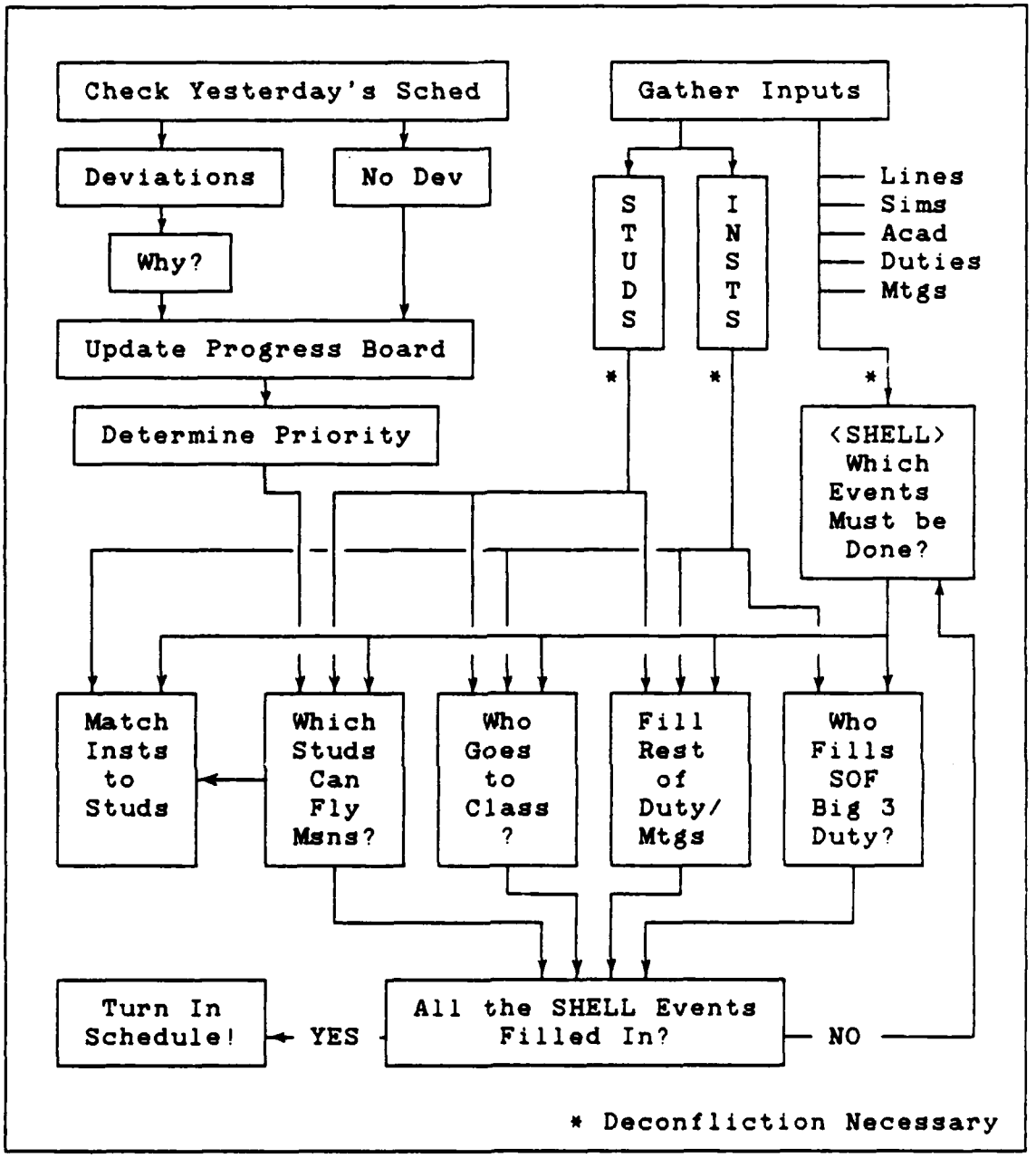


Figure 3.3. Rough Concept Map

DSS because there seemed to be no decisions involved. Further study verified that proper manipulation and presentation of the database was crucial to the matching of the instructors with the students. A DSS does need to include a

good user/machine interface. Furthermore, a DSS should include good dialogue with the user.

Concept Map Formulation. The concept map was now reduced to only the major ideas. Figure 3.4 depicts the actual decision but not all the data-related inputs necessary to achieve that decision. From this simplified concept map, the users and/or builders may identify the kernels (later sections expand on this). With the help of Capt McFarren, the authors traced the steps necessary to make a decision.

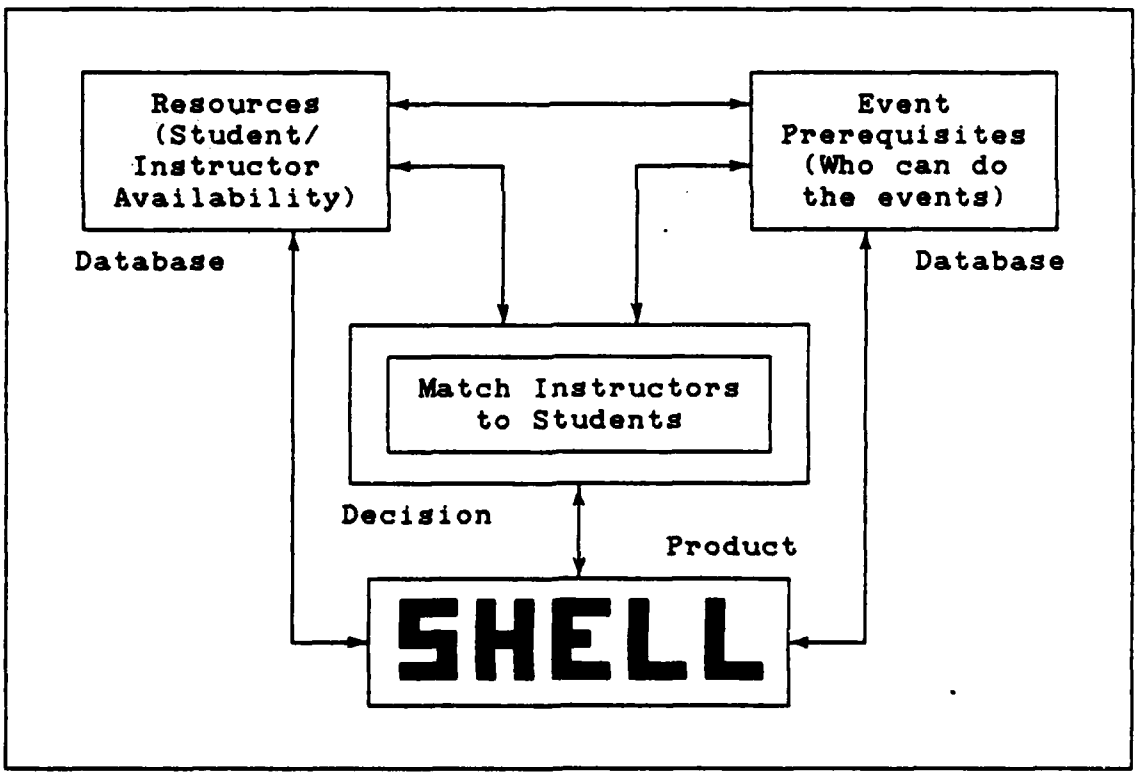


Figure 3.4. Formulated Concept Map

Task Analysis

For this specific DSS, Capt McFarren analyzed the inputs obtained from interviewing the authors. Figure 3.5 depicts the task analysis derived in Appendix C. When given the student grouping/sorting criteria (students must accom-

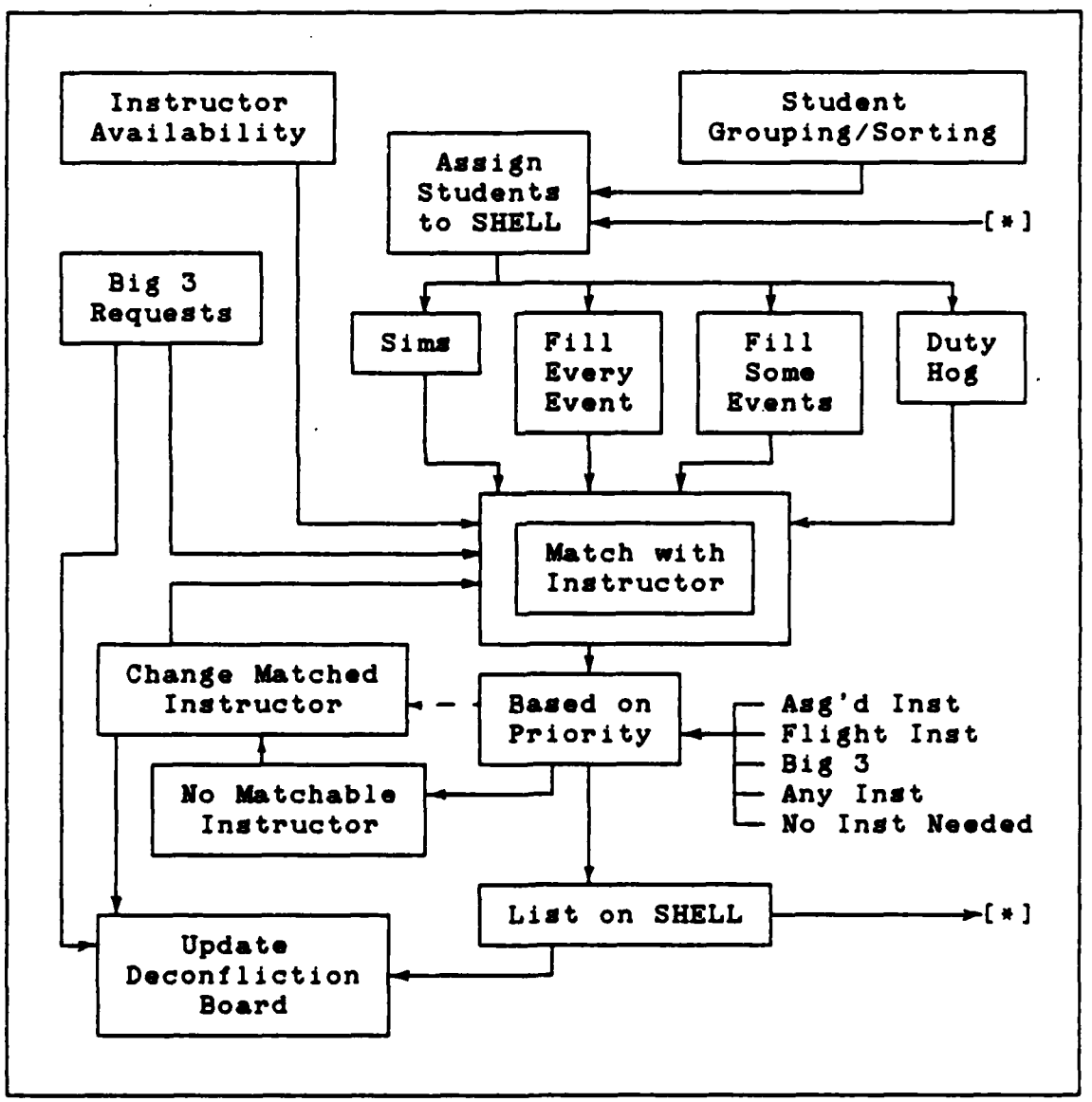


Figure 3.5. Task Analysis

plish certain events) and instructor availability for those certain events, the scheduler must assign students to the Shell and match them with the appropriate instructors. Only when provided with current information may the schedulers accurately accomplish the next day's schedule. Note that the grease board database provides student grouping/sorting and instructor availability. Also, since certain student events are predetermined, their assignment to the shell requires little decision on the scheduler's part. As such, the primary decision emphasis becomes matching the instructors. The following sections discuss this resulting kernel identification.

Because the schedule seldom happens as predicted, the whole decision process may be interrupted at any time. Capt McFarren noted this process of interference in Figure 3.6. The greatest effect this serves is to adjust the databases, depending on the problem, forcing the scheduler to rematch his student/instructor resources. From a scheduling standpoint, the process in Figure 3.6 is normally where most assignment errors occur due to limited time available to update the deconfliction/availability grease boards. This is because interruptions happen near the time the schedule is due. Because every scheduler has his own technique when filling in the possible events and handling interruptions, the builders sought a general procedure to map the decision

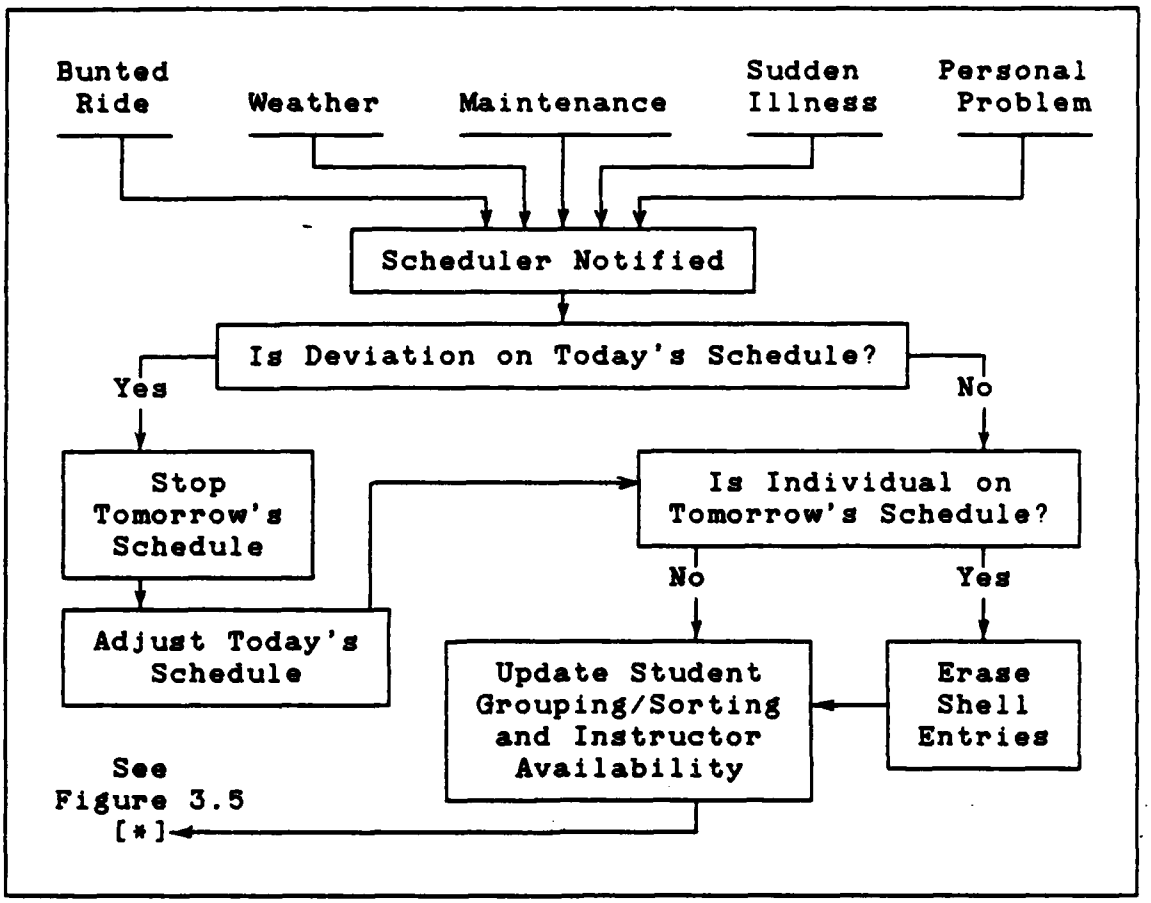


Figure 3.6. Task Analysis Interruptions

process. The experience of the authors verified the accuracy of the decision process as represented by the task analysis.

Data Analysis

This procedure traces actual data throughout the task analysis to ensure compliance with the decision process. It also helps map the data flow to ease construction for DSS

databases. Appendix D contains the database relations found in the tracing procedure.

Feature Chart Development

Table 3.4 shows the information the scheduler needs.

TABLE 3.4

Scheduler Information

A. Tomorrow's schedule.

1. Available students.
2. Available instructors.
3. Priority considerations.
4. Ability to print schedule.

B. Today's schedule.

1. Ability to update events.

C. Schedule inputs.

1. Flying sortie lines from wing
2. Simulator lines from wing.
3. Student.
   a. Availability.
   b. Assigned instructors.
4. Instructor Status.
5. Academic classes and instructors.
6. Duties and meetings.

D. Statistics.

1. Time line by class.
2. Time line by individual student.

E. Course changes.

1. Student syllabus.
2. Class additions/deletions.

This information is represented by a network hierarchy (Figure 3.7). Once the builders establish this network, they may design the storyboard using the feature chart ROMC tools mentioned previously.

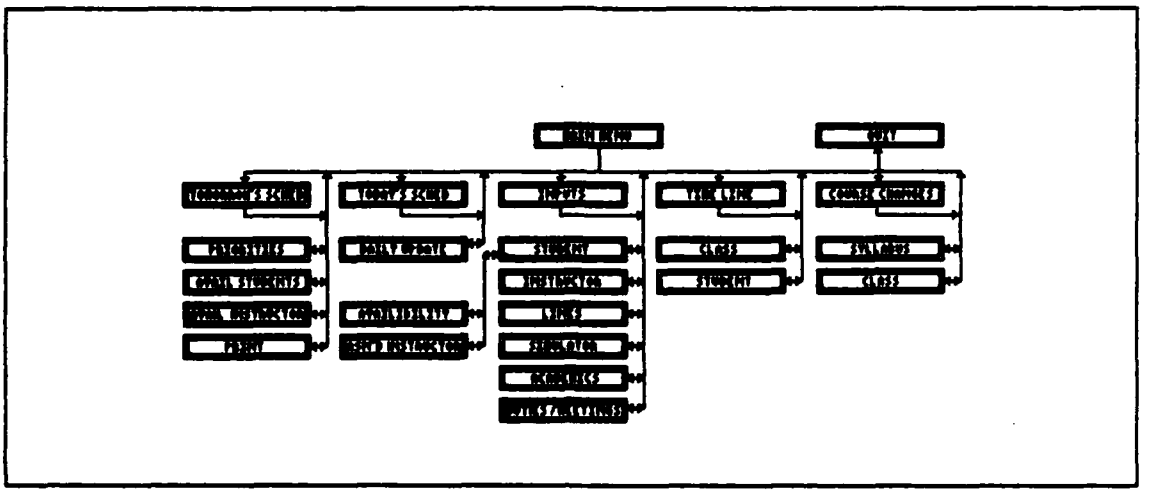


Figure 3.7. Feature Chart Hierarchy

Using the scheduling screen hierarchy depicted in Figure 3.7, the storyboard focuses purely upon the requirements needed by the user. There was no bias toward any commercial software or particular capabilities or limitations. This process reflected the user's needs without regard to any technological restraint. Should a limitation occur after the builders develop the feature chart, they will modify the DSS to include as much as possible within current commercial software bounds.

The evolution of the system via the feature chart concentrates on the overall system and the individual portions. Each category is important because the builders construct

the DSS from these features. Each individual storyboard and its evolution (Appendix F) started from the original storyboard. The initial programming focused on implementation of the storyboard.

Storyboard

A problem facing the storyboard itself is how to place all of the necessary information on the screen without cluttering the presentation. The easiest way is to diagram carefully all of the system components. The builders then arrange and group these parts in a logical manner according to the user's decision process. The builders then order and incorporate these parts into the screen design. Appendix E depicts the incorporation of the feature chart into an actual screen output format following the feature chart hierarchy (Figure 3.7). As previously mentioned, the objective in the storyboard was to create the ideal scheduling screens - what the scheduler needed to see - without technological constraints.

Kernal Identification

The concept of a kernel is a portion of the decision process under consideration. As such, a process may consist of several kernels. For example, to buy a car there are several kernels (concepts) involved. The decisions about what the DM needs or wants, the type of car on the market,

and the DM's financial status would all be possible kernels. Selection of the kernels uses the concept map and task/data analyses to identify the main ideas/concepts present in the decision process.

Determination of the Kernel(s). The concept map shown in Figure 3.4 shows the three central portions of the scheduling decision process:

1) Determination of the available resources
2) Event prerequisites
3) Instructor-to-student matching

The task analysis in Figure 3.5 confirms the three central concepts of resource availability, student event prerequisites, and instructor-to-student matching.

Key Kernel Selection. Thus, given the three kernels from which to select, the determination of the key kernel is only a matter of deciding which of the three is the most important to the user. The concept map (Figure 3.4) merely identifies the kernels without bias. A further analysis of the problem via a task analysis (Figure 3.5) clearly centralizes the importance of the instructor-to-student matching. The concept map provided the greatest indication in that the matching is the only decision kernel in the entire scheduling process. The other two kernels are only database references and, while an important portion of the decision process (as inputs), are less important to the construction of the DSS. Thus, the key kernel is the instructor-to-

student matching and became the point to start construction of the DSS.

Kernel Evaluation

Kernel evaluation may take place in three areas: The relation of the kernel to its environment, validation of the kernel identification, and verification of the key kernel. The builders must evaluate the DSS in each of these three areas to confirm they used the correct kernel for DSS construction. Otherwise, the builders may direct their construction efforts on peripheral kernels.

Environment. The scheduling environment consists of the interaction of wing inputs to the squadron, the squadron's interaction with its personnel, and completion and presentation of a finished schedule back to wing. When considering which kernel(s) to choose, the entire scheduling process hinges on the squadron schedulers and the daily choices they must make. Wing is merely an input source and the recipient of the final product. This essentially places the decision process at the squadron level. Squadron top echelons rely on the scheduling shop to build the schedule. Like wing, the echelons input requests and check the schedule before sending it to wing. This narrows the environment to the scheduling shop and the individual schedulers who construct the actual schedule and amend it should any changes (or additional inputs) occur.

Kernel Selection. Builders (both designers and users) accomplish kernel identification with an understanding of the scheduling decision process. As mentioned in the previous sections, evaluation success takes the form of the accuracy of the process. Also, the concept map and task analysis are of a generic form, accommodating the particular styles of individual schedulers. Figure 3.4 identifies the three kernels using the steps in Chapters II and III:

1. Resources (availability) . . . . Database
2. Event prerequisites (events) . . Database
3. Instructor/student matching . . Decision

Key Kernel. Derived from the concept map and task analysis, selection of the key kernel is valid as well. In tracing both the task and data analyses, the key kernel of instructor-to-student matching becomes the focal decision point. In other words, matching is the primary decision the user makes. Thus, matching is the central concept about which the scheduling process revolves. Therefore, with all three of the kernel evaluation criteria met, construction of the DSS began. Building focused upon matching the instructors to students placed upon the schedule. From there, the builders included resource availability and event prerequisites as the adaptive design/evolution process continued.

Continuing Evaluation. Evaluation must be a continuing process. Feedback must be gathered on the DSS performance to modify the system. Users will have complaints about the DSS. Appendix J details how to evaluate the DSS after it

has been implemented. Listed below is a summary of these suggestions (52:161-166).

Event Logging. Users would write in a notebook, kept next to the DSS, any comments about DSS performance.

Attitude Survey. A questionnaire of multiple-choice, short statement, or open-ended questions should be used throughout the DSS implementation and development.

Rating and Weighing. The methodology involves developing a set of parameters related to the system and effects being evaluated, weighting these parameters in terms of relative importance, and having one or more individuals rate the system on each parameter. Examples of these parameters include schedule creation time, number of scheduling deviations, or percentage of primary instructor correctly assigned.

System Measurement. This method attempts to quantify effects through measurements of the performance of the target system, and therefore it is similar to performance evaluation.

Value Analysis. The approach attempts to quantify subjective value judgments.

Combining Methods. Because of the variety and complexity of the potential effects and because there are problems with all evaluation methods, a combination of methods will probably result in the best evaluation.

Software Selection

Once the Holloman schedulers provided the basic framework for the DSS requirements, the authors identified the programming language characteristics. Tables 3.5 and 3.6 show the desired features and characteristics of a DSS.

Considering the above features, the authors selected five integrated software packages for further evaluation:

1. ABILITY (Version 1.0)
2. LOTUS SYMPHONY (Version 1.2)
3. LOTUS 1-2-3 (Version 2.0)
4. ENABLE (Version 1.15)
5. R:BASE SERIES 5000

TABLE 3.5

Desirable Language Features

- Integrated database management (probably relational)
- User friendliness to nontechnicians
- Both procedural and nonprocedural command structures
- Interactive on-line utilization
- Support of prototyping and adaptive development
- Modest training requirements for end users
- Easy debugging and intelligent default assumptions
- Little or no Complex Code (Cobol, Fortran, etc.)
- Internal documentation generation support
- Understandable code for non-developers

(11:172)

Integrated software packages are software that contain database, spreadsheet, word processing, graphics, and (usually) telecommunications programs. Integrated means that each of the five capabilities can work together to form a powerful product. Table 3.6 shows further selection criteria.

TABLE 3.6

Desirable Language Characteristics

- Low Cost
- Reliability
- Availability
- Compatibility
- Maintainability
- End user orientation
- Programmer productivity
- Hardware/Software operating environment

(11:176-177)

Table 3.7 depicts the software characteristics for each of the five packages under consideration. The authors selected ENABLE for the following reasons: 1) ENABLE comes with the Z-248 microprocessor so the cost is minimal; 2) ENABLE will be available to the training squadrons at

Holloman; 3) The software requires low maintenance (e.g., program modifications/changes) by users and is highly

TABLE 3.7

Integrated Software Comparisons

	COST (#)	AVAILABILITY	COMPATIBILITY	MAINTENANCE	FRIENDLINESS	PRODUCTIVITY	WINDOWS	MOUSE
ABILITY	.99	X	IBM	MED	HIGH	MED	NO	NO
SYMPHONY	695	X	IBM	LOW	MED	HIGH	YES	YES
1-2-3	549	X	IBM	MED	MED	MED	NO	YES
ENABLE	* 78	X	IBM	LOW	HIGH	HIGH	YES	YES
R:BASE	700	X	IBM	LOW	HIGH	HIGH	NO	YES

* Government Price

(15:129)

friendly (does not let the user make mistakes); 4) Enable's help information is exceptional (15:129); and, 5) with pro-

ductivity in mind, the June 24, 1986 issue of PC Magazine wrote:

> The Enable database manager includes strong relational capabilities and a procedural language, and it ranks with some of the best standalone programs. It is well designed, easy to use, generally quite fast... (15:129).

From the above considerations, the authors selected ENABLE for the thesis work. The next step in the process is to identify and precisely define the problem. Chapter IV shows the difference between the planned storyboard and actual DSS that resulted from using ENABLE.

IV. Flight Scheduling DSS

This chapter presents the differences between the planned and actual scheduling DSSs. The authors built the DSS around the kernel system identified in Chapter III. The kernel was identified as matching a student with an instructor. This DSS collects the information necessary to do this matching and presents the scheduler with lists to select a student or instructor. The main effort was to build as much of the actual storyboard as was possible in the time available. The DSS differs from the desired storyboard due to technological limitations and lack of programming time. The DSS consists of three sections:

1. Menus
2. Spreadsheets
3. Databases

Menus

The authors started programming from the storyboard but soon realized that the actual program would have to be different. ENABLE could not exactly create each storyboard screen in every detail, therefore the authors needed to employ a different representation. Some of the menus in the storyboard were not used while others were newly created The following table lists the originally planned menus and compares them to the current DSS menus. The following sec-

tions probe any differences between the planned and current menus with a discussion as to how the discrepancy evolved.

TABLE 4.1

Comparison of Planned and Current DSS Menus

PLANNED DSS MENUS	CURRENT MENU STATUS
Main Menu	Revised
Daily Event Update	Pending
Student Time Line	Pending
Course Changes	Unchanged
Unplanned	Personnel Availability List
Unplanned	Wing Lines Input Prompt
Unplanned	Created Shell Date Prompt

Note: Pending menus indicate the planned DSS menus are not part of the key kernel system and have not been created. Unplanned menus are those menus which were not decision-oriented, but memory aids to prompt the user for various inputs.

Main Menu. The planned main menu (Figure 4.1), differs from the actual menu (Figure 4.2) employed in the DSS due to system software limitations. ENABLE's inability to interact freely between spreadsheets and databases led the authors to deal only with spreadsheets. Although the original main menu would have worked as planned, the authors opted to

```
A.   TOMORROW'S SCHEDULE
B.      Priorities
C.      Student Avail
D.      Instructor Avail
E.      Print
F.   TODAY'S SCHEDULE
G.      Daily Update
     INPUTS
        Students
H.         Availability
I.         Assigned Inst
J.      Instructor
K.      Lines
L.      Simulator
M.      Academic
N.      Duties/Meetings
     TIME LINE
O.      Class TimeLine
P.      Student TimeLine
     COURSE CHANGES
Q.      Syllabus Changes
R.      Class Changes
S.   QUIT
```

Figure 4.1. Planned Main Menu

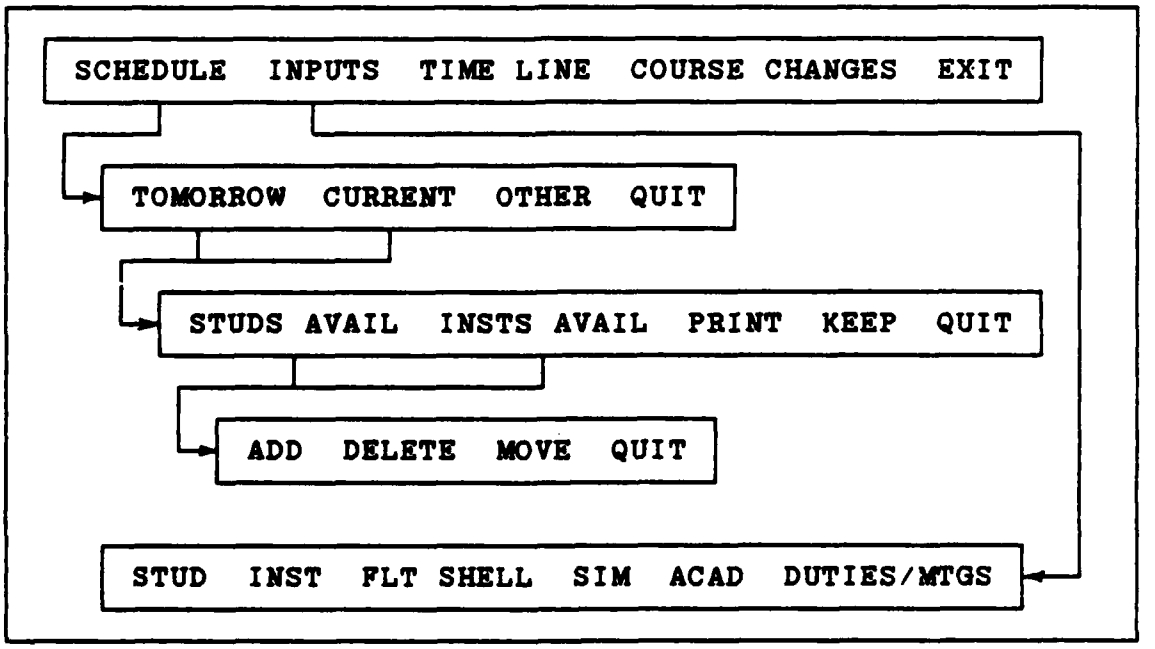


Figure 4.2. Actual Main Menu Hierarchy

remain within the spreadsheet environment. This allowed access to the resident macro commands for simplicity and speed. The user invokes this menu in the same manner as the planned menu.

Daily Event Update. The daily event update menu, which updates the event and prerequisite databases, was not a part of the key kernel system. A lack of programming time did not permit the building of this menu.

Student Time Line. The student time line, which depicts a student's progress relative to his syllabus event flow, was outside the scope of the kernel system. ENABLE's ability to display student progress graphically is excellent. Once a DSS method is developed to track student progress, the display and graphics will assist the user to make decisions. Appendix F shows an example of the graphic capability using a set of fictitious students and time line data points.

Course Changes. The planned course changes menu (Appendix F) remains unchanged. Although not a part of the key kernel system, this menu may be integrated into the DSS at a later date, once a database containing prerequisites is developed.

Personnel Availability List. This menu evolved due to the need to update the personnel availability database for extended (more than one day) periods of time. Critical to the scheduling process itself, this menu allows inputs of

personnel leave, DNIF, and TDY events. The scheduler may change the status of each event when notified.

Wing Lines Input Prompt. This menu, shown in Appendix F, prompts the user to insert the wing lines data diskette into the Z-248's B drive. The authors developed the menu because the Z-248 needs the disk in place to automatically process the wing data. Otherwise, personnel are forced to type the data into the system.

Created Shell Date Prompt. As a follow-on to the wing lines input prompt, once the user creates a new shell he must save it in a date format for use by the kernel system. This menu, depicted in Appendix G, prompts the user to input the shell as a date. It then automatically saves this shell schedule in a kernel-usable file. Each shell schedule's file name is the date the DSS uses to retrieve that file.

Databases

Because of ENABLE's inability to freely interact between spreadsheets and databases, the authors decided to deal strictly with spreadsheets for simplicity, speed, and flexibility. As such, the actual DSS incorporates databases into spreadsheets. The following table compares the planned and current DSS databases.

Student and Instructor Databases. For the reasons mentioned above, the DSS included these needed databases into the Availability spreadsheet (Appendix H).

TABLE 4.2

Comparison of Planned and Current DSS Databases

PLANNED DSS DATABASES	CURRENT DSS DATABASES
Student	Changed to Spreadsheet
Instructor	Changed to Spreadsheet
Syllabus Events	Pending
Inputs/Student/Availability*	Pending
Inputs/Student/Assigned Instructors*	Pending
Inputs/Instructor*	Changed to Spreadsheet
Inputs/Academic*	Changed to Spreadsheet
Inputs/Duties-Meetings*	Changed to Spreadsheet
Course Changes/Syllabus*	Pending
Course Changes/Class*	Pending
Note: Pending databases indicate the planned databases are not part of the key kernel system and have not been created. Asterisked items (*) are database input forms.	

Syllabus Events Database. A lack of programming time did not allow the authors to construct the database. This database, although not a part of the kernel system, will ensure the student is on the proper scheduled syllabus event.

Inputs/Instructor Input Form. Figure 4.3 shows the planned input form for the instructors. Figure 4.4 shows

```
                          434th
                  INPUTS / INSTRUCTOR


INSTRUCTOR NAME _____________          ADD      DELETE

STATUS: RED DOT ___        RCO ___       SOF ___
        FLT CDR ___        BIG 3 ___     FCF ___

AVAIL:  DNIF  _ (Y/N)
        TDY   _ (Y/N)
        LEAVE _ (Y/N)
        OTHER ________
        DATE  _________

DO YOU WISH TO MAKE ANOTHER CHANGE? _ (Y/N)

-------------------------------------------------------------
                        [F1] - HELP
```

Figure 4.3. Planned Instructor Input Form

LETTER OF X'S AND QUALIFICATIONS

IP NAMES		RANK	CP	FLT	WX CAT	EXP	UIPIP	UIPRD	FLTLD	RDIP	RRIP
ALLAG	*	MAJ	IP	B	A	E			X	X	X
ANDEC		MAJ	IW	C		N					X
BECKG	*	LTC	IP	A	A	E	X		X	X	X
BECKW	*	CPT	IP	A	A	E			X	X	X
BOHAM	*	CPT	IP	C	B	E			X	X	X
CASEK	*	MAJ	IP	A	A	E	X	X	X	X	X
DANIJ	*	CPT	IP	C	A	E			X	X	X
DAWSV		MAJ	IP	B	C	N					X
DOELJ	*	CPT	IP	C	A	E			X	X	X
DONAM		CPT	IP	A	C	N					
FRANG	*	LTC	IP	D	A	E			X	X	X
FREDJ	*	CPT	IP	C	B	E			X	X	X
FREIJ		COL	IP	D	B	E			T		X
FUSSJ	*	MAJ	IP	B	A	E			X	X	X
GROSR		CPT	IP	B	B	E					X
HELTC	*	LTC	IP	A	A	E	X		X	X	X
HUMSD	*	CPT	IP	B	A	E	X	X	X	X	X

Figure 4.4. Actual Instructor Database

the actual input form. The planned database is a question-and-answer input to a database. In contrast, the actual database is a listing of all pertinent data for the scheduler. In the actual presentation, the user deals directly with the data (rather than automatically when using the input form) because the spreadsheet format was used.

Inputs/Academic Input Form. Figure 4.6 merges the planned academic input form (Figure 4.5) with the duties/meetings input form. The reason for the integration is the change to spreadsheet format (as previously mentioned).

```
                              434th                    17 OCT 86

                        INPUTS / ACADEMIC

  CLASS       INSTRUCTOR       START TIME      END TIME

  GST-87M     Fenno              0800            1300
  GOR-88B     Heinrichs          0800            1000
  GOR-88B     Heinrichs          1400            1600
  ------      ----------         ----            ----
  ------      ----------         ----            ----
  ------      ----------         ----            ----
  ------      ----------         ----            ----
  ------      ----------         ----            ----
  ------      ----------         ----            ----

  QUIT

------------------------------------------------------------
                          [F1] - HELP
```

Figure 4.5. Planned Academic Input Form

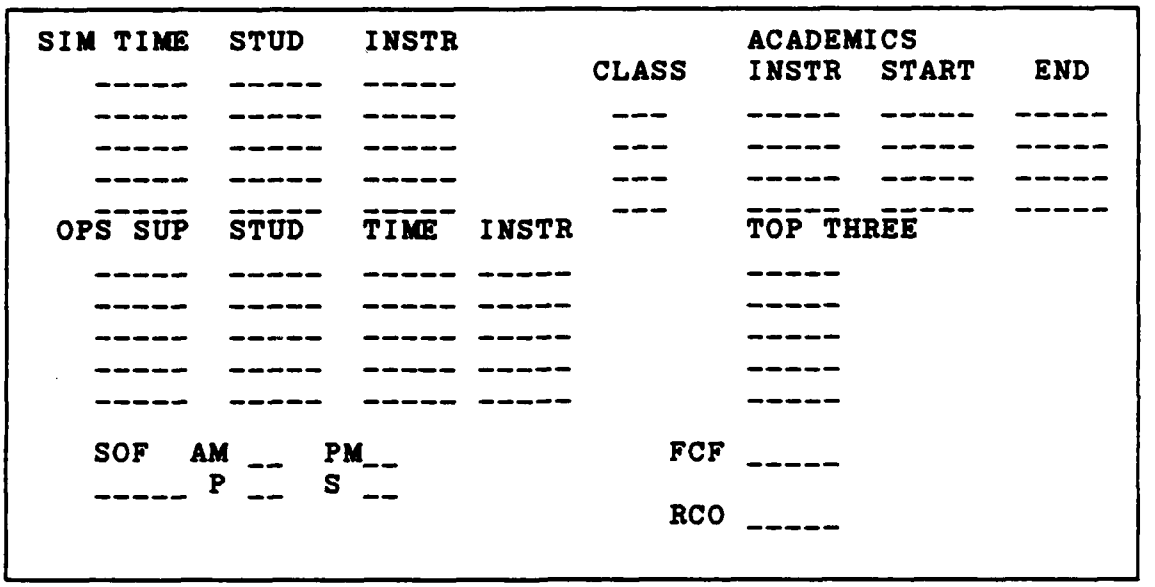


Figure 4.6. Actual Academic/Duties-Meetings Input Form

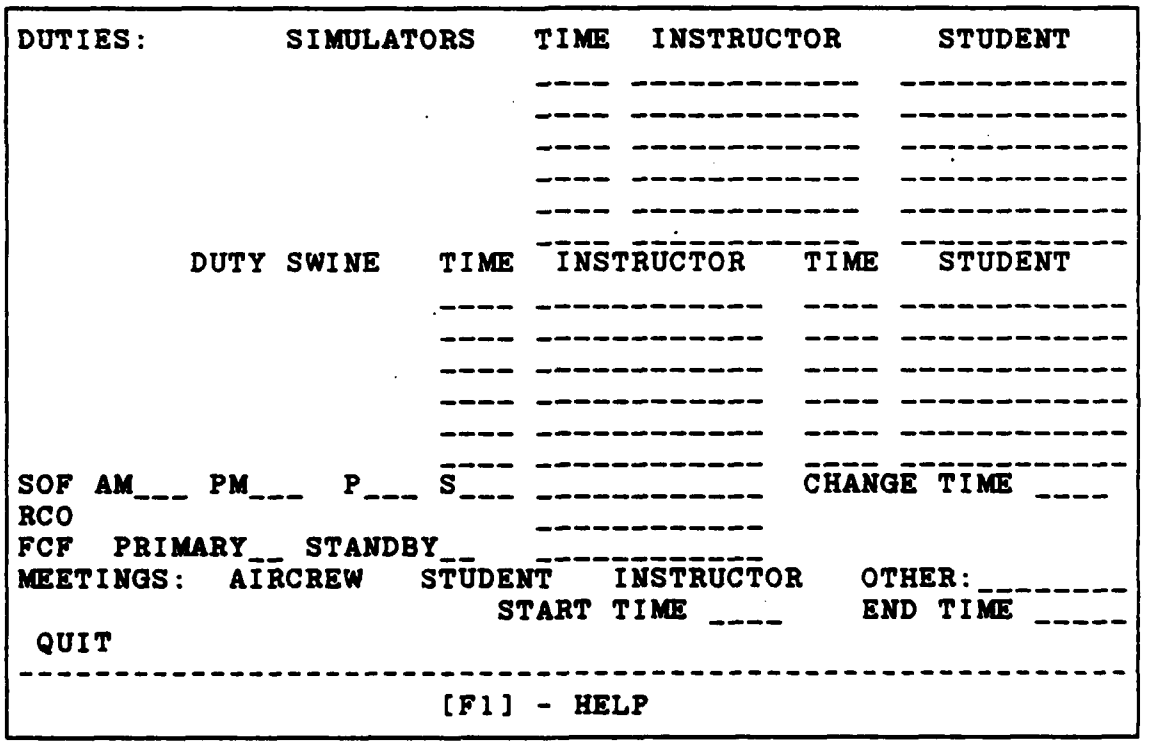


Figure 4.7. Planned Duties-Meetings Input Form

Figure 4.6 is the actual spreadsheet format used in the DSS. Neither the planned nor actual formats automate the input process into the database.

Inputs/Duties-Meetings Input Form. Figure 4.7 above shows the planned shell to schedule duties and meetings.

The next section compares the planned and current uses of the spreadsheet portion of ENABLE.

Spreadsheets

TABLE 4.3

Comparison of Planned and Current DSS Spreadsheets

PLANNED DSS SPREADSHEETS	CURRENT DSS SPREADSHEETS
Tomorrow's Schedule	Unchanged
Today's Schedule	Unchanged
Time Line/Class	Pending
Unplanned	Availability
Unplanned	New Schedule (ZNEWSCH)
Unplanned	Temporary Schedule (ZATEMP)
Note: Pending spreadsheets indicate the planned spreadsheets are not part of the key kernel system and have not been created. Unplanned spreadsheets are those spreadsheets which are needed to make the DSS workings transparent to the user.	

Tomorrow's Schedule. As described in Chapter III, Tomorrow's Schedule is a part of the kernel system. The

screen presentation is currently the same as planned in the storyboard (see Appendix F). The completed capabilities of this schedule are those that are part of the kernel system.

Today's Schedule. Today's Schedule (see Appendix F) is currently the same as planned in the storyboard. Since Today's Schedule is a completed form of Tomorrow's Schedule (see above), the capabilities are the same. The user updates Today's Schedule when a deviation occurs as a record of the completed events.

Time Line/Class. Since student syllabus events are not a part of the key kernel system, the DSS does not display the class time line. Although ENABLE has a very capable graphics software package, it cannot generate the graphs without the data. Presentations will occur once developers integrate the syllabus track database into the DSS.

Instructor Availability. Figure 4.8 shows the actual spreadsheet database for availability of instructors.

ZNEWSCH. The authors created this unplanned spreadsheet to house the coding and formulae necessary for constructing new schedules from the wing line input disk. The storyboard did not account for programming considerations, only final screen presentations. Thus, users needed a method to transfer wing lines data into a usable format.

ZATEMP. This spreadsheet is a temporary completed shell created from ZNEWSCH that is awaiting assignment of a name (date).

D N I F	L E A V E	T D Y	O T H E R
BECKG	HUNSD LINNR	BOHAM FREDJ MINEG MAYRL	DANIJ DAWSV GROSR

	AVAILABILITY				
IP NAMES	DNIF	LEAVE	TDY	OTHER	AVAIL
ALLAG					0
ANDEC					0
BECKG	X				1
BECKJ					0
BECKW					0
BOHAM			X		1
CASEK					0
DANIJ				X	1
DAWSV				X	1
DEAUC					0
DOELJ					0
FRANG					0
FREDJ			X		1
FREIJ					0
GROSR				X	1
HELTC					0
HUFFD					0
HUNSD		X			1
KLINS					0
KOKAA					0
LINNR		X			1
MARVC					0
MAYRL			X		1
MCGRT					0
MILLS					0
MINEG			X		1

Figure 4.8 Actual Instructor Availability Spreadsheet

Chapter V presents the author's conclusions and recommendations. These conclusions will include findings for the scheduling DSS and recommendations for adaptive design in general.

V. Conclusions and Recommendations

Conclusions

This thesis built a Decision Support System (DSS) using off the shelf software (ENABLE) for use in scheduling at the 479th Tactical Training Wing. The DSS is friendly enough to allow non-technically oriented users to use the DSS without learning ENABLE . However, modification of the program will require learning ENABLE. This DSS assists a scheduler in the process of matching instructors to students and deconflicting their schedules. Furthermore, the DSS reads the wing scheduling data file and builds a shell for the duty scheduler in two minutes on a Z-248 microprocessor. The DSS automatically fills in a deconfliction spreadsheet for the scheduler. Finally, at the end of the day, this DSS will help crisis management at the squadron level.

This chapter has two parts. The first part is conclusions and recommendations for the squadron DSS. The second section discusses DSS adaptive design conclusions and recommendations.

Specific Scheduling DSS

Conclusions. This DSS saves a scheduler time over the current situation of maintaining grease boards by using computerized databases and spreadsheets for the manual tracking

process (e.g., this DSS replaces three of the six grease-boards listed in Chapter I). These spreadsheets include the capacity to sort by availability (i.e., personnel missing because of leave, TDY, or DNIF). The DSS also provides a list of available IPs and students from which to select when scheduling. In addition, the DSS automates the deconfliction process.

Due to screen size and resolution limitations, the display cannot present all of the information a scheduler needs. However, this DSS displays up to 60 flights on one screen and the remainder of the information (simulators, Duty Hog, etc.) on another screen. With the touch of one key, the DSS will display any information the scheduler currently has access to.

Programming. The DSS screens evolved from the ROMC memory aid requirement plus the user's need to employ various portions of the DSS at will. The DSS capitalizes on several ENABLE features. The DSS primarily uses the spreadsheet portion for graphic capability and speed. The hierarchical menus used in the scheduling DSS allow easy selection of any desired function. He can also select the MAIN MENU from any screen using the Alt-F10 key. Should the user have any questions about the DSS functions, he may ask the system for help.

The need to include a help function stemmed from our own questions about ENABLE. On-line help and informational

prompts are available while using the DSS The DSS user's manual (Appendix H) provides details about the available help functions Another feature that eases the user/DSS interface is the application of a mouse.

Menu-driven, mouse-operated software is easier for a new user, especially in the spreadsheet application software. This DSS uses mouse hardware and software for greater friendliness and simplicity

Future Work. The authors base the following proposals on their research and extensive scheduling experience This thesis lists the proposals in order of priority

The first proposal is to computerize the monthly instructor duties in a database. When a scheduler creates a shell, the DSS would search the database and include monthly duties on the schedule. The initial shell could include these inputs when created.

Another proposal is to computerize the student training board in a database. The computer could reference the database to find out which event the student was on. Further, the computer could update the database depending on the completed events in the day's schedule. Reference Appendix D for suggested database relations.

In addition to the above, develop a way to check prerequisites against the syllabus flow. The computer could check to see if a student had his prerequisites done accord-

ing to the syllabus. As the scheduler selects a student for an event, the database could recall which event he was on.

Moreover, this thesis proposes adding the ability to deconflict academics, duties/meetings, simulators, etc. to the computerized deconfliction board. The deconfliction board should include all duties that each member must perform the next duty day.

The next proposal is to develop a way to match instructor qualifications to a syllabus ride and student (e.g., ACM qualification). The computer should be able to match an instructor with a student's needs according to what the syllabus requires. This thesis proposes the addition of a model to suggest to the scheduler a match of instructor to student.

A further proposal is develop a dynamic database to contain all of the information required to schedule. For example, if a student fails a ride, the DSS should flag future conflicts/prerequisites using the updated information. This database should include the information suggested above. In addition, it should interface quickly with the display medium and bring up the required or requested information quickly to be transparent to the user.

DSS and Adaptive Design in General

Storyboard, Concept Mapping, and Adaptive Design. The use of storyboarding, concept mapping, and adaptive design

form an extremely powerful methodology for development of DSSs. Storyboarding identifies and defines requirements for a DSS. Concept mapping identifies the key kernel. Task analysis (Figure 3.5), the last part of concept mapping, clearly centralized the importance of the instructor-to-student matching. Adaptive design is an iterative approach to development of the key kernel and implementation of the DSS.

Storyboard. The storyboard is an excellent tool to identify requirements. By defining the problem visually, the storyboard evolves during the early stages of the definition process. As users update their requirements, the storyboard should reflect the changes. The storyboard shows the ideal system, while time, technological, and monetary constraints may limit the actual DSS. The authors started programming from the storyboard but soon realized that the actual program would have to be different.

Adaptive Design. Lack of available thesis time and TDY funds limited the use of adaptive design to the first iteration of development. Lack of current user feedback in the field would have resulted in the development of this DSS in isolation, but fortunately, the builders were also experienced users. Once the builders created a prototype, testing and evaluation by the users at Holloman was necessary to continue adaptive design. Unless co-located, user feedback is hard to get and usually impossible due to only telephone

conversations (no Z-248 or money for TDY was available). A possible methods for information transfer could be by telephone modem or mail a floppy disk.

During programming the storyboard evolution continues, requirements definition evolves for the complete DSS, but the actual programming involves only the key kernel. The key kernel is the essence of the decision process. The kernel evolves once the users evaluate, modify and evaluate it again (see Appendix J for recommendations on evaluation). Adaptive design demands user feedback to grow around the user's needs.

These authors conclude that the best way to do adaptive design is on-site in the users' organizational environment. Immediate feedback would be available to the builders from the most current users. Eventually, with easy-to-use software tools, users will be able to perform adaptive design and apply it to their specific DSS.

Learning ENABLE took a majority of the programming time. To be an effective programmer in ENABLE required approximately a month and a half of learning effort for two programmers. Each aspect of ENABLE required tutorials and training to become familiar with it. The problem with dealing with new technology software is that the form of many capabilities you want included in a program is unclear from the start. No clear-cut solutions exist. Therefore keeping an open mind is essential during this type of unstructured

work. Many questions arose at this point about how the scheduling program was to work, if it would work at all! In retrospect, the best approach would be to learn the software capabilities entirely before making any decisions about how something was to work. Appendix A contains a complete discussion of program development.

The best thing to do in adaptive design is to work on only the kernel portion of your program before starting on the peripherals. Programming efforts should concentrate on the key kernel. The authors included some nice-to-have peripherals that were not necessary according to the concept map development. In building the key kernel, the software influenced adaptive design.

The type of hardware and software a builder is familiar with influences his thinking about DSS program design (e.g., IBM versatility and speed versus Macintosh pull down menus, ease of use, and graphics capability). The type of hardware used must support the rapid prototyping portion of adaptive design.

Rapid prototyping is best done on a dedicated machine (preferably with a hard drive). This thesis effort found that working at home, instead of competing for CPU time at work, resulted in far greater productivity. Most large software programs require frequent shifting of the floppy disks between the drives (e.g., ENABLE requires six disks). Productivity increases by having greater speed and larger

memory within the computer. Teaming up in groups also increases productivity.

Synergism in developing the DSS is present even at the two-person level. Planning, designing, learning, and programming of DSSs are best done in teams of two or more. One person may provide new and different perspectives to others in the team. Furthermore, learning new software is easier in groups of two or more than individually.

Making the DSS workings transparent to the user (without interrupting his decision process) requires a large amount of effort. ENABLE can be made transparent to the user. However, the builder spends the majority of his time making his work transparent, rather than furthering the program's progress. This is not bad because DSS transparency of the DSS structure is necessary for successful implementation and user friendliness. In summary, user access to the data and models must be as easy as possible through effective dialogue.

Software selection may not result in the ideal product for the DSS. For instance, the authors chose ENABLE because that was what the end users would use, not because it was the best product. Shown below are the authors' considerations for choosing software (listed in order of importance):

1. Available Compatible Hardware
2. Software Package Features
3. Cost
4. User Familiarity
5. Ease of Training and Software Use

Recommendations

The most important recommendation is to continue to evolve the kernel DSS through the iterative adaptive design process of building, testing, modification, and so on (for an explanation of adaptive design see Chapter I). Outlined next are recommendations on how to continue the evolution of the DSS.

1. Find a champion
2. Appoint a referee
3. Create a steering group
4. Minimize organizational changes and stresses
5. Train users
6. Documentation
7. Continued support and planning

Champion. Find one person dedicated to the idea of a computerized DSS. Give that person the total responsibility and authority over the development and implementation of the DSS. Allow this person to guide the implementation, evaluation and modification of the system. The authors recommend that when the champion goes PCS the new champion have at least one month of overlap.

Referee. This person, a third party, objective participant, judges which resources or modifications are committed to the project next (see feedback notes from squadron users below)

Steering Group. The steering group should guide the implementation and further development of the Scheduling Program and its use. The authors recommend the steering group include the Ass stant Deputy Commander of Operations,

a wing scheduler, the wing computer manager, all of the chief squadron schedulers, and all of the squadron computer managers. The steering group should meet monthly throughout the implementation phase of the DSS. After the referee deems appropriate, bi-monthly meetings are recommended. Additional meetings are conducted as problems arise.

Minimize Organizational Changes and Stresses. The implementation and use of the scheduling program should not create or delete any jobs existing in the 479th Tactical Training Wing. The aim of the steering group should be to minimize organizational stresses during testing and implementation.

Training of Users. Each user should have a thorough and complete checkout before using the scheduling program. Therefore, each squadron should establish academic classes before the use of the scheduling program. The author recommend the training be done at the squadron level because each squadron will use the program to schedule differently.

Documentation. The champion and each squadron should document feedback from the users. The DSS documents are:

1. Feedback Notes from Squadron Users
2. Hookbook

Continued Support and Planning. The referee will field any complaints/suggestions and decide their merit, presenting his recommendations to the steering group. The scheduling program evolves as more worthwhile suggestions are used.

This continued support and planning is essential to the adaptive design process.

Feedback Notes from Squadron Users. ENABLE includes an excellent word processor. As a user has a problem, he should document it on the spot. The user could then finish scheduling and report the problem to the referee or squadron small computer manager. The champion gathers all feedback and prioritizes the worthwhile changes to the DSS(s). Then, the steering group (with the referee's approval) decides on which change to include in the DSS.

Hookbook. The hookbook contains future plans and ideas for the DSS. The champion documents both his and others' inputs/ideas on 3x5 notecards (Appendix I). The champion should not discard any idea because implementation can not occur immediately. The purpose of the hookbook is not to discard any worthwhile feedback. The champion sorts through a listing of worthwhile ideas from the hookbook to see which idea could be used next. This process of collection in the hookbook and sorting to find the next idea is important for the champion.

Final Recommendation. The authors recommend the continued automation of the scheduling system at Holloman. The enthusiasm and drive present in the 479th TTW will ensure the continued evolution of a DSS. With consideration and care, the computer will make the 479th scheduling system the best in the Tactical Air Force!

BIBLIOGRAPHY

1. Andriole, Stephen J. Class lecture in OPER 652, Decision Support Systems. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, October 1986.

2. Arthur, J. L. and A. Ravindran. "A Multiple Objective Nurse Scheduling Model," AIIE Transactions, 13 (13): 55-60 (March 1981).

3. Berg, Capt. D. H. and Capt. R. G. Nuss. An Improved Missile Combat Crew Scheduling System Using the Simulation Language for Alternative Modeling (SLAM). MS thesis. School of Engineering, Air Force Institute of Technology (AU). Wright-Patterson AFB OH, March 1984.

4. Cody, Ronald P. and Jeffrey K. Smith. Applied Statistics and the SAS Programming Language. New York: Elsevier Science Publishing Co., Inc., 1985.

5. Hines, William W. and Douglas C. Montgomery. Probability and Statistics in Engineering and Management Science (Second Edition). New York: John Wiley & Sons, 1980.

6. Hosios, A. J. and J. M. Rousseau "A Heuristic Scheduling Algorithm," Journal of the Operations Research Society, 31: 749-753 (July 1980).

7. Keen, P. G. W. "Value Analysis: Justifying Decision Support Systems," Management Information System Quarterly, 5: 1-16 (March 1981).

8. Kunzman, Ltc. Chief of Scheduling, 435th Fighter Training Squadron, Holloman AFB, NM. Personal Interview. Holloman AFB NM, 25 Aug 1986.

9. Lee, R. M. "Epistemological Aspects of Knowledge Base Decision Support Systems," Processes and Tools for Decision Support edited by H. G. Sol. North Holland Publishing, 1983.

10. Lembersky, Mark R. and Uli H. Chi. "'Decision Simulators' Speed Implementation and Improve Operations," Interfaces, 14 (4): 1-15 (July-August 1984).

11. Meador, C. L. and Richard A. Mezger. "Selecting an End User Programming Language for DSS Development," Decision Support Systems edited by Ralph H. Sprague,

Jr. and Jugh J. Watson. New Jersey: Prentice-Hall, 1986.

12. McFarren, Michael R. and Clements, Donald W. The Problem Definition Process. DSS Class Term Project, December, 1986.

13. Mortenson, R. E. "Maintenance Planning and Scheduling Using Network Simulations," Winter Simulation Conference Report, Vol 1, 333-340. Atlanta, Georgia, December 1981.

14. Parnell, Maj Gregory S., Assistant Professor, Department of Operational Sciences, Air Force Institute of Technology. Personal Interview. Wright-Patterson AFB OH, 29 August 1986.

15. Petzold, Charles. "Programmable Relational Databases," PC Magazine: 129 (June 24, 1986).

16. Roege, William H. Pilot Scheduling in a Fighter Squadron. MS thesis, Sloan School of Management, Massachusetts Institute of Technology, February 1983 (AD-A126 947).

17. Seagle John P. and Salvatore Belardo. "The Feature Chart: A Tool for Communicating the Analysis for a Decision Support System," Information & Management, 10: 11-19 (January 1986).

18. Sprague, Ralph H. Jr. and Eric D. Carlson. Building Effective Decision Support Systems. New Jersey: Prentice-Hall, Inc., 1982.

19. Turban, Efraim and Paul R. Watkins. "Integrating Expert Systems and Decision Support Systems," Decision Support Systems edited by Ralph H. Sprague, Jr. and Jugh J. Watson. New Jersey: Prentice-Hall, 1986.

20. Valusek, Maj John R., Assistant Professor, Department of Operational Sciences, Air Force Institute of Technology, Personal Interview. Wright-Patterson AFB OH, 12 September 1986.

21. Votipka, Capt David, Wing Computer Manager, 479th Tactical Training Wing, Personal Interview. Holloman AFB, NM., 25 August 1986.

22. Warner, D. M. "Scheduling Nursing Personnel According to Nursing Preferences: A Mathematical Programming Approach," Operations Research, 24 (5): 842-856 (September - October 1976).

23. Yamayee, Z. A. "Maintenance Scheduling, Description, Literature, and Interface with Overall Operations Scheduling." IEEE Transactions on Power Apparatus and System, PAS-101 (8): 2770-2779 (August 1982).

Vita

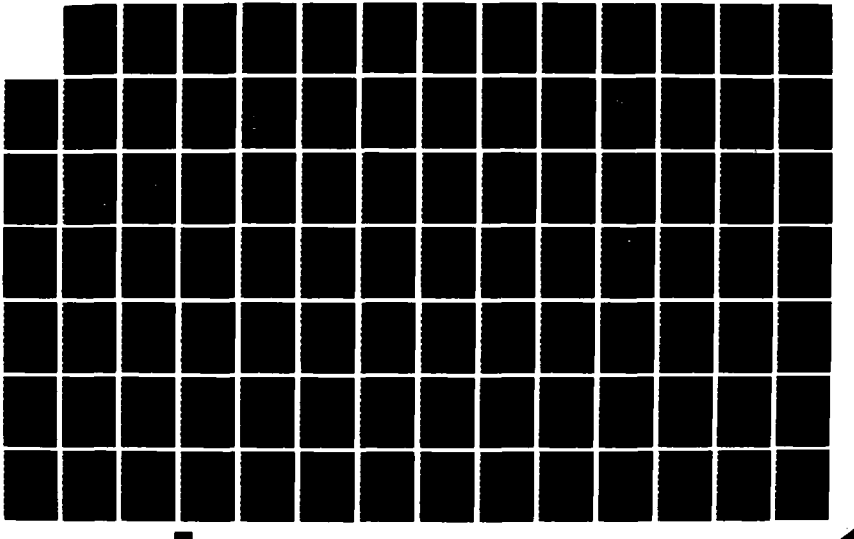
Paul Eugene Trapp was born on 1 August 1955 in Montgomery, Alabama. He graduated from Brooking High School in Brookings, South Dakota in 1973 and attended South Dakota State University, Brookings, South Dakota from which he received the degree of Bachelor of Science in Electrical Engineering in December 1977. Upon graduation, he received a commission in the United States Air Force through the ROTC program. He entered active duty in May 1978 in Undergraduate Pilot Training (UPT) assigned to Vance AFB, Oklahoma. Upon graduation from UPT he attended Lead-In Fighter Training (LIFT) in the AT-38B jet fighter trainer. Upon graduation from LIFT he attended F-4 Replacement Training Unit at Homestead AFB. Following a remote tour at Osan AFB, Korea as an F-4 aircraft commander, he had a follow on tour to Elmendorf AFB, Alaska also flying F-4's. Next assignment was at Holloman AFB, New Mexico where he did daily scheduling for the 434th for one year. Selected for the 465th Academic Squadron at Holloman AFB in March 1984, he served as flight commander to incoming students and Chief of Air-to-Air Academics until entering the School of Engineering, Air Force Institute of Technology, in August of 1985 where he was selected as a member of Who's Who in Colleges and University for 1986 and selected for Tau Beta Pi member.

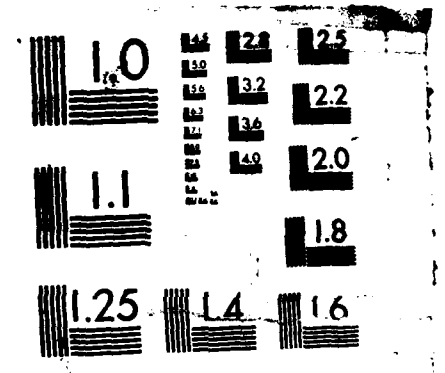
Permanent Address: RR1 Box [illegible] A

Volga [illegible]

AD-A180 265 DESIGN EVOLUTION OF A FIGHTER TRAINING SCHEDULING DECISION SUPPORT SYSTEM(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI 2/3

UNCLASSIFIED J W GRECHANIK ET AL MAR 87 F/G 5/8 NL





Vita

Jeffrey Walter Grechanik was born on 5 November 1956 in Port Lyautey, Morocco. He graduated from Thomas A. Edison High School in Alexandria, Virginia in 1974 and attended the United States Air Force Academy, Colorado, from which he received the degree of Bachelor of Science in Engineering Sciences in May 1978. Upon graduation he received a commission in the United States Air Force. He entered active duty in May 1978 and attended Undergraduate Navigator Training at Mather AFB, California. Upon completion, in March 1979, he attended Replacement Training Unit upgrade into the F-111A aircraft at Mt. Home AFB, Idaho. He then completed an overseas tour at Royal Air Force (RAF) Upper Heyford, United Kingdom flying the F-111E. His next assignment in November, 1979 was to Cannon AFB, New Mexico, flying the F-111D. An Instructor Weapons System Officer at both RAF Upper Heyford and Cannon AFB, his scheduling experience includes six months as a unit operational scheduler and eighteen months as the Chief Scheduler for the Consolidated Training Unit at RAF Upper Heyford. In addition, he served as a unit training scheduler for two years and wing training scheduler for eight months, until entering the School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, in August of 1985.

Permanent Address: 5627 Habersham Way
Alexandria, Virginia 22310

APPENDIX A

Program Development

The authors wrote the program as if they completed each step before going on. A better picture would be an iterative approach, where each step adds to the understanding of the previous steps. The programmer may even go back and revise previous steps. Programming each step is rarely completely done in one step.

The authors quickly completed all tutorials. A review was adequate to get started programming until they met a specific problem. Throughout the development process, the authors used the documentation to help solve specific problems.

ENABLE is a large, slow program unless you have a hard drive. ENABLE requires Six floppy disks to run the program. Using the floppy disks requires constant changing, in and out, of the disk drives during programming. This constant changing of floppy disks was a slow, cumbersome process. Using a hard drive decreased ENABLE run time tremendously and not switching disks is a pleasure. The authors estimated the program turn-around time increased ten-fold. Installation of ENABLE on AFIT's IBM PC-AT made available two places to develop the scheduling program: Work and home. The authors did the major part of the development at home

but if an idea formed at work, the IBM PC-AT was immediately available to work on.

The next progress on development was moving through the documentation page-by-page. The authors explored the Data Base Management (DBM) system using the documentation much like a tutorial. They accomplished all of the examples in the DBM section of the manual until familiar with that specific section. The authors also kept the Quick Reference Guide, a pamphlet summarizing all the commands, handy for any word processing questions. In addition, they placed the ENABLE keyboard overlay on the keyboard.

Then, the authors constructed a database using actual aircrew qualification data. They considered a qualification database essential to the scheduling program. This database construction process is basically a learning process for the software; the authors did little actual programming toward the final product at this stage. The following three stages of development were iterated several times before completion of this learning stage.

1) Database definition was very straightforward. After ½ day programming a good lesson was that a little time spent up front in planning would go a long way. For example, it is a good idea to standardize the format of data the user will enter. By using the same field names and format, the programmer may reduce errors and data transfer between databases will be possible.

2) The Input Form is the screen that the user sees when he inputs data to the database. Since the user would interface directly with this screen, help messages should be available for each input line and on-line for general help (more on help messages below). The Input Form must match with the database definition, so a change to database definition meant a change in the Input Form also. Furthermore, each line could be custom designed for restriction of input. For instance, an input line could contain only a date or numbers or only letters. If the input line received any unauthorized inputs, an error message would result. The error message is custom-designed for each input line.

3) The Database display is also custom-designed. The user may sort and list in any desired order. The computer screen may display whatever is input to the database.

The development and use of help messages became an unexpected aspect of the learning process. The authors could design the input line to be case-sensitive, so JACK is different from Jack. The EXACT planning of what the builder wants the user to input is imperative at this stage. The use of the following four general rules helped programming tremendously:

1. Give onscreen help instructions. Don't make the user search for a key to help, when you can write it on the screen.
2. Have an error message ready in case of error.
3. Have the cursor jump to the next field if necessary.
4. In general, design your help exactly the way you want it BEFORE YOU START. A little planning goes a long way.

At this point the scheduling program was taking on the form of interaction with a database. The problem with dealing with new technology software is that the form of many capabilities you want included in a program is unclear from the start. No clear cut solutions exist. Therefore keeping an open mind is essential during this type of unstructured work. Many questions cropped up at this point about how the scheduling program was to work, if it would work at all! Although in retrospect, the best approach would be to learn the software capabilities entirely before making any decisions about how something was to work.

The next development was copying American Standard Code for Information Interchange (ASCII) into the database. ASCII is a Code that represents each character, number and symbol on a computer screen by an 8-bit binary number. Since the 479th Wing delivered the daily flying sorties times to the squadron on a floppy disk, the scheduling program would need to be able to read that file from the floppy disk. The documentation showed that it was possible to copy ASCII into a database. Upon investigation, ENABLE could copy ASCII from an external file into the database portion.

Then from the database portion into any other part of ENABLE.

The final form of the startup macro does the following process:

1. Reads wing data into database
2. Reads from database into spreadsheet
3. Calculates takeoff and land time
4. Resizes spreadsheet
5. Constructs initial scheduling screen
6. Brings up main menu

The use of macros capability occurred next in the development of the scheduling program. Macros are computer code that execute certain actions in ENABLE automatically. The authors needed to hide the complex portions of the program from the casual user for the scheduling program to be a success. Non-technically oriented personnel should be able to use the scheduling program. So an attempt to have the program process information by itself involved the use of macros. ENABLE excelled in the formation and execution of macros. ENABLE could create a macro while the programmer did the keystrokes. For instance, while the programmer loaded ASCII into the database, the computer was recording the keystrokes. The keystrokes were put into a file that ENABLE could executed at any later time that would, now, automatically load ASCII into the database when the macro was invoked.

An important discovery was the ability to use a mouse in ENABLE. A mouse is a hand-held pointing device for the computer. The discovery came at a time when simplicity was

especially important. Program acceptance and use could depend on the user-friendliness of the scheduling program. The software used to activate a mouse in LOTUS SYMPHONY also worked for ENABLE.

The next development demonstrated the evolutionary development of the main menu in the storyboard. Starting on the main menu was a mistake because of the many iterations the authorse were to perform. The main menu (MM1) initially designed was the one that was storyboarded. The Main Menu covered the whole computer screen and involved a series of hierarchical expanded menus. The next main menu (MM2) covered only the right hand portion of the screen and included all of the possible selections in our program. The authors discarded MM2 as their approach changed from a menu- driven database to a spreadsheet. The discovery that macros would only work inside spreadsheet or database (but not both) shifted the approach to a spreadsheet. The current main menu (MM3) is a series of hierarchical expanded menus at the top of a spreadsheet. The best thing to do in adaptive design is to work on only the kernel portion of your program before starting on the peripherals.

Next, both programmers spent about ½ day going over the storyboard to give form and function to the picture. They considered each storyboard in detail. The authors also gave each item in the storyboard some detail about how the item was to work, given what they learned of the software. For

instance, the programmers defined the QUIT function on each screen to where it would send the user to, once selected. They also considered how each item was to connect with other items in the menu and in other menus. All 21 proposed screens were gone over in detail. A curious thing happened, both programmers realized a lot needed changing to conform with what was possible and realizable. The authors did the storyboard as a first cut. The programmers realized they did not need certain items or could not do them at present. And a few more screens were necessary to complete the program. Therefore, what the programmers could do started to differ from the storyboard at this point.

Just a note here about the storyboard but more detail in Chapter III. The storyboard is the ideal, no technological constraints requirements definition. What the programming could accomplish was quite different. Once the programmers chose the software and started the programming, the screens conformed more to specific software limitations. The storyboard had been an excellent requirements definition tool. Basically, the storyboard had transformed a decision process into computer screens inputs and outputs. Now that the programming had begun, the storyboard was of the form of the ideal program while the program took on the form of what was technologically possible.

At this point, the programmers set up details like colors in all the screens and automatic startup. Colors were

set in the default profile. Also, to make the program transparent to the user, ENABLE will invoke a starting macro that sends the user to the opening menu of the scheduling program instead of the opening menu of ENABLE.

Next, the programmers set up the required window configuration. Shown below is the listing of how the open windows were planned to be in the scheduling program.

1. Tomorrow's schedule
2. Today's schedule
3. Other information as needed

The inability of the macros to work outside the originating functions (e.g., spreadsheet, database, word processing) changed the program considerably. The plan at this point was to use a spreadsheet as the display medium to the user and use a database for the manipulation of data. However, the macros would not work across the two types of functions. For instance, if a macro was invoked in the spreadsheet and moved into another window, say a database, the macro would stop executing. The need for user transparency and ease of use drove the program into a single functional area. The programmers chose the spreadsheet functional area for its graphics capability and speed over the database.

Each programmer specialized at this point. One programmer moved into the file and menu area. The other automated the internal functions in the scheduling spreadsheet itself.

The most pressing question at this point was how to structure the system and, more specifically, the file layout during presentation of the DSS. This task may be broken down into four areas:

1. File structure (what is displayed).
2. File access (macros).
3. File manipulation (Spreadsheet-avail).
4. Menu generation.

The first area was the file structure (what was to be displayed). The initial plan was to have TODAY'S SCHEDULE in the first ENABLE window. TOMORROW'S SCHEDULE would reside in the second window. The third window would contain a dummy macro file which would hold the window in reserve for future schedule creation tasks. The fourth and fifth windows would display instructor and student availability status respectively. Window number six would show student event prerequisites. A reserved space to display schedules other than TODAY'S or TOMORROW'S would be in the seventh window, while additional duties for the last schedule displayed would reside in the last (eighth) window.

From working with the large spreadsheet files in the various windows, the authors discovered a hardware system limitation. ENABLE has the ability to work with only two windows of large spreadsheet files in computers fitted with the 8088-series microprocessor. This limitation is due to the availability of only 640,000 bytes of computer random access memory (RAM), of which ENABLE uses a substantial portion (depending on the size of the files being manipulated).

Because of this limitation, the file structure forced the placement of the availability database in the first window and the desired schedule in the second window. This placement was what the users would need as a minimum to schedule, given the availability of only two windows. Once the programmers set up the file structure, the next task was how to access these files.

File access for this DSS encompasses program start-up and file retrieval. ENABLE provides the ability of using a macro to start up the DSS. The idea was to start ENABLE automatically from computer turn-on and call up the next day's schedule. This would all be transparent to the user, of course. Automatically starting ENABLE was a simple matter of entering a BATCH command within the AUTOEXEC.BAT file resident on the computer. This command also includes the name of the macro file automatically executed once ENABLE starts. Appendix I contains the individual macros. A characteristic of a macro is that it executes until it hands control over to a menu. The menu may contain instructions to continue to another macro, but it can't return to the original macro, unless you wish to start the macro over again. So, the start-up sequence includes several menus intertwined with macros.

The start-up sequence involved much trial-and-error. It began with trying to invoke menus from macros and then link the menus back into other macros. With the syntax mas-

tered, Table A.1 lays out the desired start-up sequence. Once ENABLE has started and the availability spreadsheet resides in the first window, the screen presents the author credits menu for two seconds. Then, the credits menu invokes the system date conversion macro in the availability spreadsheet. This macro uses the current date resident in the computer to determine what day it is. Then, the macro figures out which date would be the next day's schedule. For example, if today was Friday the 13th, the next day would be Monday the 16th. The date retrieval macro takes over after this, creating a new macro which gets the next day's schedule and places it into the second ENABLE window.

Since the schedulers must have the most up-to-date information about their personnel, the DNIF/TDY menu is invoked to query the user as to their status. If users desire a change, the system updates the status of any person(s). Once personnel availability is current, the system transfers the list onto the current scheduling spreadsheet. The user now has all the current information from which to schedule the next day's events. Finally, the system invokes the spreadsheet's main menu. The user is now ready to start scheduling.

The most complex macros involved in the file access functions reside in the availability spreadsheet. These file manipulation macros include system date conversion and personnel availability update capability. The system date

TABLE A.1

DSS Start-up Sequence

SEQUENCE	FILE
1. Start ENABLE from computer turn-on.	AUTOEXEC.BAT
2. Place availability spreadsheet in Window #1.	#{0}.MCM
3. Invoke author credits menu.	F.MNU
4. Invoke system date conversion.	AVAIL.SSF
5. Edit date retrieval macro.	#{1}.MCM
6. Place next day's schedule in Window #2.	#{2}.MCM
7. Invoke DNIF/TDY menu.	D.MNU
8. Update personnel availability.	AVAIL.SSF
9. Transfer current availability to the schedule in Window 2.	#{5}.MCM
10. Go to Window #2 and invoke the main menu of the schedule.	#{5}.MCM

conversion macro, mentioned above, retrieves the date from the internal computer clock and transfers that date into a usable form. The personnel availability macro adjusts the availability spreadsheet so that the user may change the status due to leave, DNIF, TDY, or other circumstances. The macro then sorts through the changes, creating a list of unavailable personnel. The system macros take over to transfer this list to the current schedule.

Following the steps in Table A.1, the start-up macro then generates the schedule's main menu, found on each schedule shell. This menu generation follows the menu hierarchy depicted in Figure 1.1. The macro activates when the user presses the [Shift-F10] and Z keys. Currently, because of the key kernel system, the only active portions are the add, delete, and move functions. For ease of use and because of the repetitive nature of these tasks, the macro activates when the user presses the [Shift-F10] and A keys.

In addition to the schedule spreadsheet menu, five more menus are available. The first menu is the author credits menu. This calls the DSS 'Flight Scheduler' and lists the author's names. The second menu overlays a prompt over the list of unavailable personnel. The displayed prompt queries the user to see if he wishes to change the availability list. If the answer is yes, the menu invokes the above-mentioned personnel availability macro. If not, the menu starts the macro which transfers the list to the current displayed schedule. The third menu prompts the scheduler for changes to the course syllabus tracks or student syllabus assignment. There are no macro commands resident within this menu as it is not a part of the key kernel system. The next menu prompts the user to insert the Wing lines input disk into the computer's A drive. Once done, the menu invokes the macro which transfers the line information to a newly created blank shell. The last menu prompts

the user to enter a name for the new shell. This name will be in a date format (e.g., 20MAR) so the DSS may call up the file at a later time.

Building the menus required explicit knowledge of ENABLE and it's command structure. It took a considerable amount of time to learn how the menus interacted with the macros and vice-versa. Coloring the menus took time as well; the effort involved time-consuming trial-and-error methods because the documentation was not explicit.

At the same time the above external file processing was taking place, the other author was busy building the inner workings of the individual schedules. The internal programming included automating the selection of instructor pilot names and student names. The three processes decided upon were adding, deleting and moving a name from one portion of the schedule to another. Furthermore, the programmers considered completion of an automatic deconfliction sheet. Macros were used to automate all of the above requirements.

The addition macro adds a name to the schedule. Since the deconfliction sheet contained a list of all the instructor and student names, a window opens to show the list to the user. The user selects a name. The macro takes over at this point by doing the following:

1. Checks to see if the time slot is available on the deconfliction sheet.
2. Puts the name at the desired position on the screen if the time slot is free on the deconfliction schedule.
3. Fills out the deconfliction sheet.

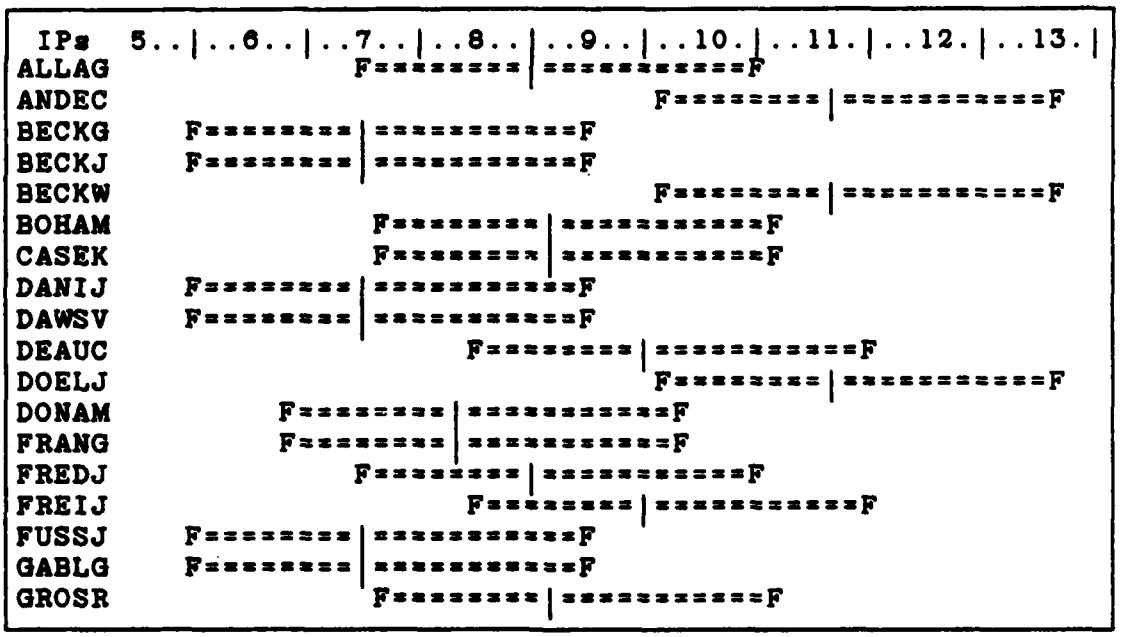


Fig. A.1 Deconfliction Sheet

Additional prompts are added to guide the user through the automatic macro executions.

The deletion of a name from the schedule is a point and press process. The user moves the cursor over the desired name to delete him from the schedule. Upon execution, the deletion macro takes the name off the schedule and deletes that time slot from the deconfliction sheet.

The move macro is similar to the above processes, combining both a deletion and addition process. First, the macro deletes the desired name from the position onscreen and deletes the time from the deconfliction sheet. Second, the process is exactly like the addition macro. The macro adds a name to the desired position, if the time is open on the deconfliction sheet.

APPENDIX B

Concept Map Formulation

Appendix B contains the iterative process of forming the concept map. This process includes the first-cut of the desision process and formation of the main decision list. The decision list is then ordered and placed into a rough concept map. A second-cut of the decision process is done culminating in the actual concept map formulation.

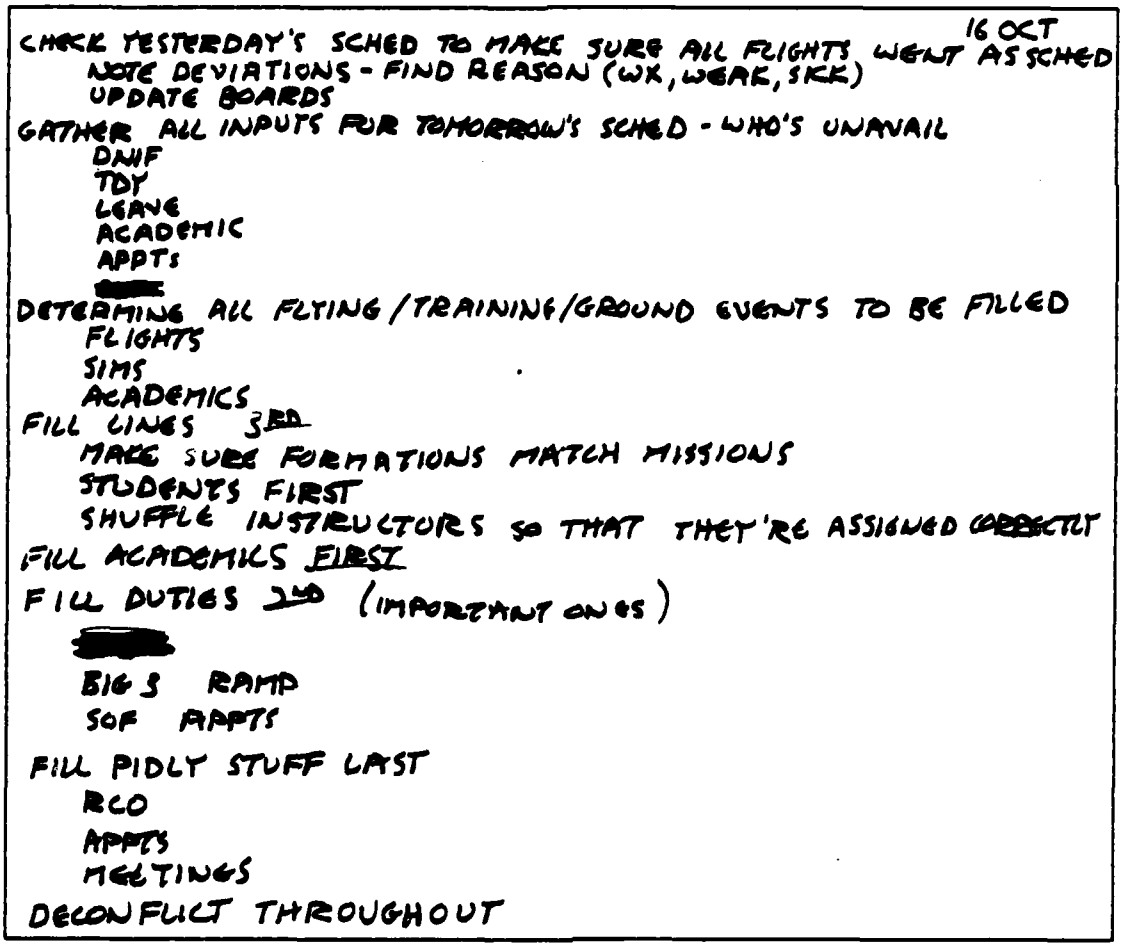


Figure B.1. Capt Grechanik's First-cut of Decision Process

1) FILL IN GREASE BOARD, T.O. TIMES, SIMS, ACADEMICS,
1) DECONFLICT ACADEMICS CLASSWISE MEETINGS, WHICH RIDES STUDENTS ON
2) FILL IN INSTRUCTOR ABSENCES, DNIF, MEETINGS ON DECONFLICTION BOARD
3) FILL IN SET RIDES SUCH AS CHECKRIDES, DO FLYING WITH SQUADRON
4) LOOK AT GRADUATION DATES ON CLASS OR INDIVIDUAL FURTHEST BEHIND - RED DOTS ONLY FLY ONCE
FILL IN STUDENT NAMES AND WHICH RIDE THEY'RE ON
5) LOOK FOR TYPE RIDES - AIR-TO-GND OR AIR-TO-AIR
6) PUT IN STUDENT NAMES WITH A MATCH FROM THE RIDE THEY ARE ON [redacted] COMPARED TO THE [redacted] CONFIGURATION OF THE PLANES AT A SPECIFIC TAKEOFF TIME.
7) FILL IN EXCLUSIVE PERIODS FOR EACH CLASS FIRST USING STEP 6.
8) MATCH IPs TO STUDENT NAMES ACCORDING TO SQUADRON RULES. [redacted] FOR EXAMPLE THE 434th HAS A PRIMARY AND SECONDARY IP ASSIGNED TO THE STUDENT. IF NEITHER THE PRIMARY OR SECONDARY IP IS AVAILABLE THEN MATCH THE STUDENT WITH A IP FROM HIS FLIGHT. [redacted] IF NONE OF HIS FLIGHT IS AVAILABLE THEN ANY IP MUST DO, [redacted]
9) ITERATE STEP 4 THROUGH 8.

RULES & REGULATIONS

SUPPORT REQUIREMENTS

Figure B.2. Capt Trapp's First-cut of Decision Process

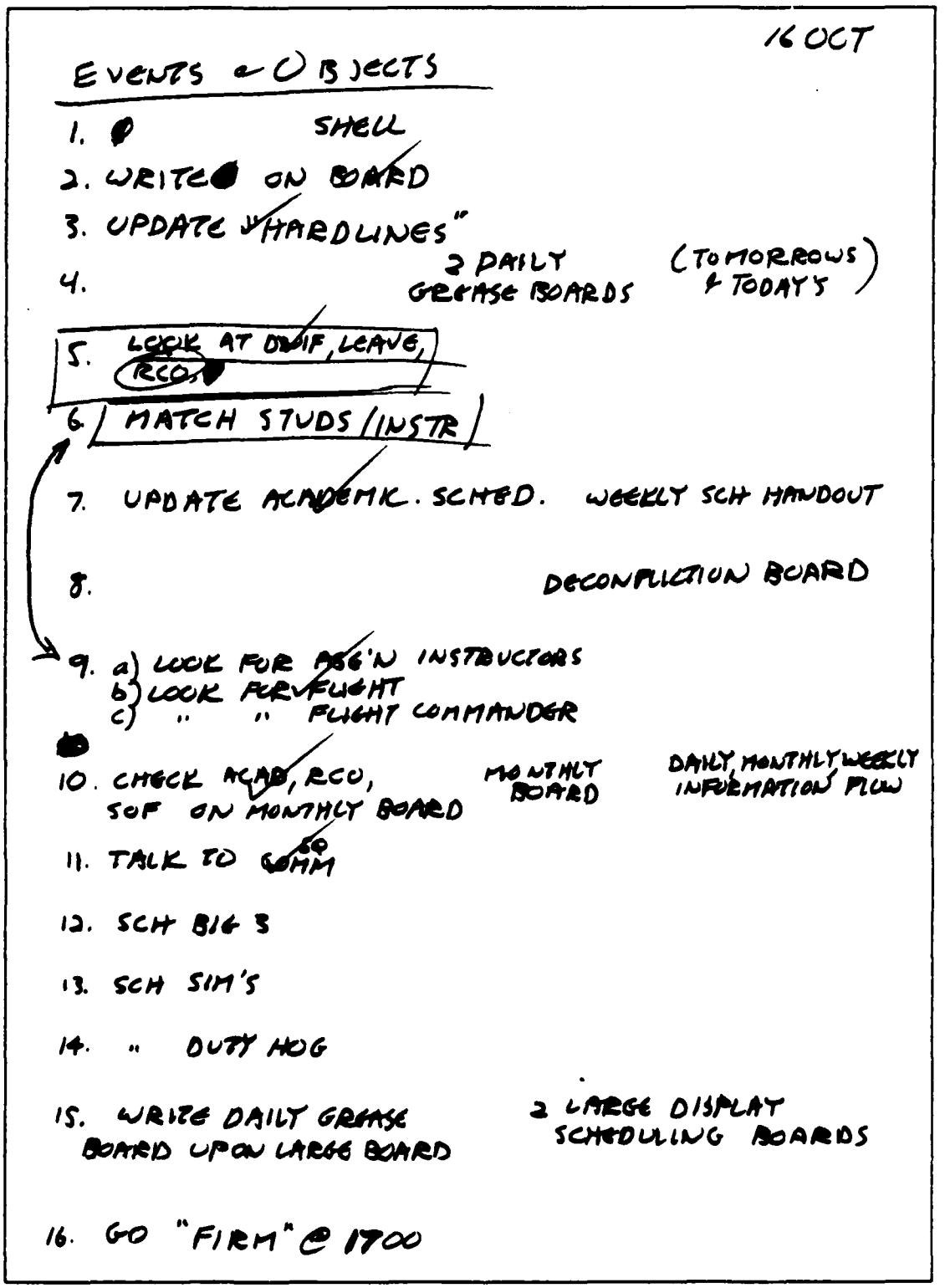


Figure B.3. Main Decision List

1) ENSURE DATA FROM TODAY'S SCHEDULE IS UPDATED ONCE FLYING IS COMPLETE FOR TODAY'S (NIGHT BEFORE OR MORNING)

2) WRITE UP THE SHELL ON TOMORROWS PORTABLE GREASE BOARD

3) " IN HARDLINES

4) WRITE IN DNIFS, LEAVES, RCO, SOF (DUTY HOG)

5) TALK TO (SQ. COMMANDER)(OPS OFF)(ASS'T OPS) (BIG 3)

6) WRITE IN ACADEMIC SCHED (WEEKLY) CLASSES & INSTRUCTORS

7) MATCHING PROCESS

   1) STUDS ⟷ INSTRUCTORS
      a) LOOK FOR ASSIGNED INSTR
      b) LOOK FOR IPs IN FLIGHT MEMBER
      c) LOOK FOR ANYONE

7a) SCHEDULE BIG 3

7b) SCHEDULE SIMS

7c) " DUTY HOG

8) FILL IN DECONFLICTION BOARD

END) WRITE DAILY GREASE BOARD UP ON LARGE BOARD
GO FIRM @ "FIRM TIME"

Figure B.4. Ordered Decision List

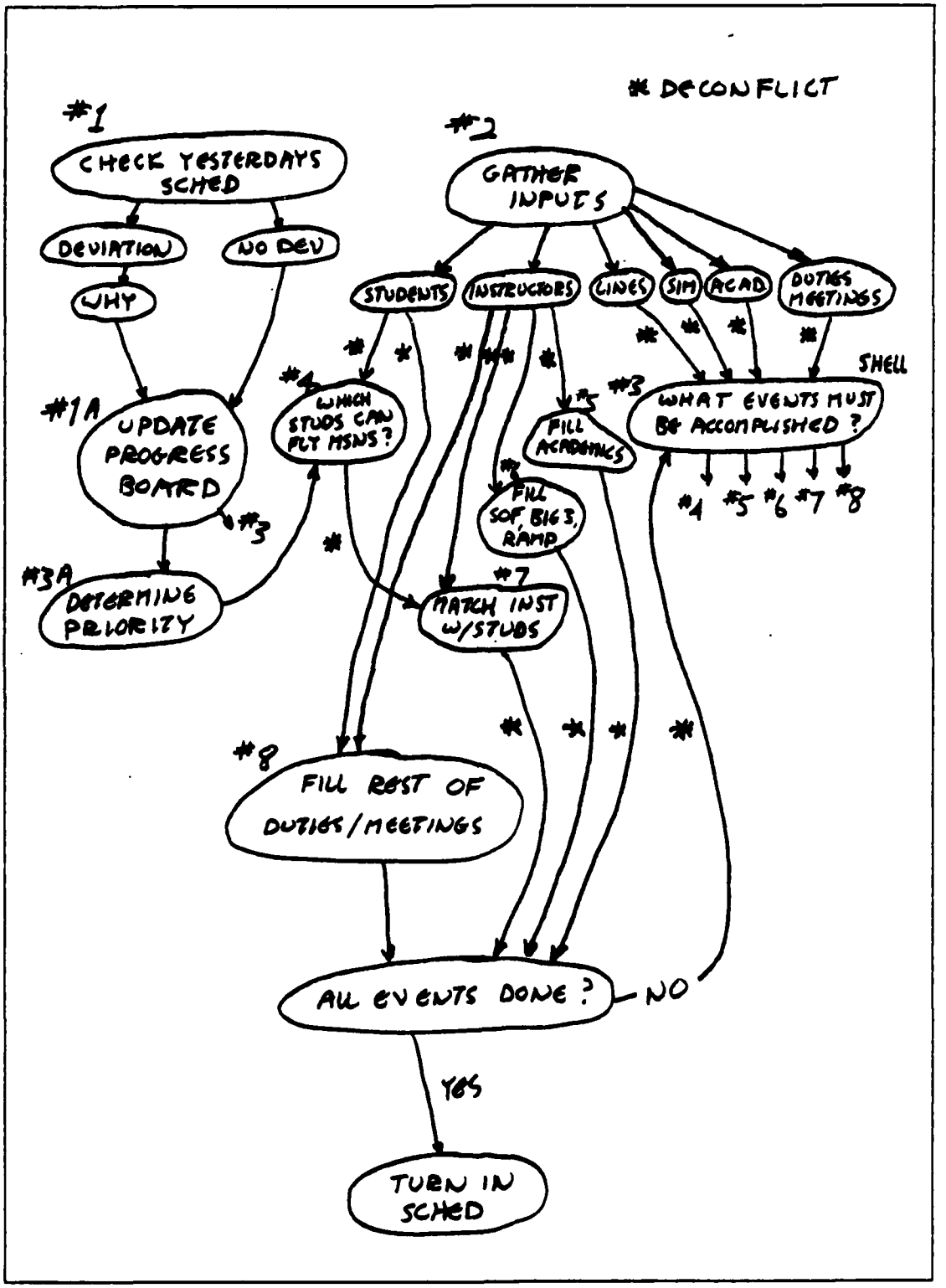


Figure B.5. Capt Grechanik's Rough Concept Map

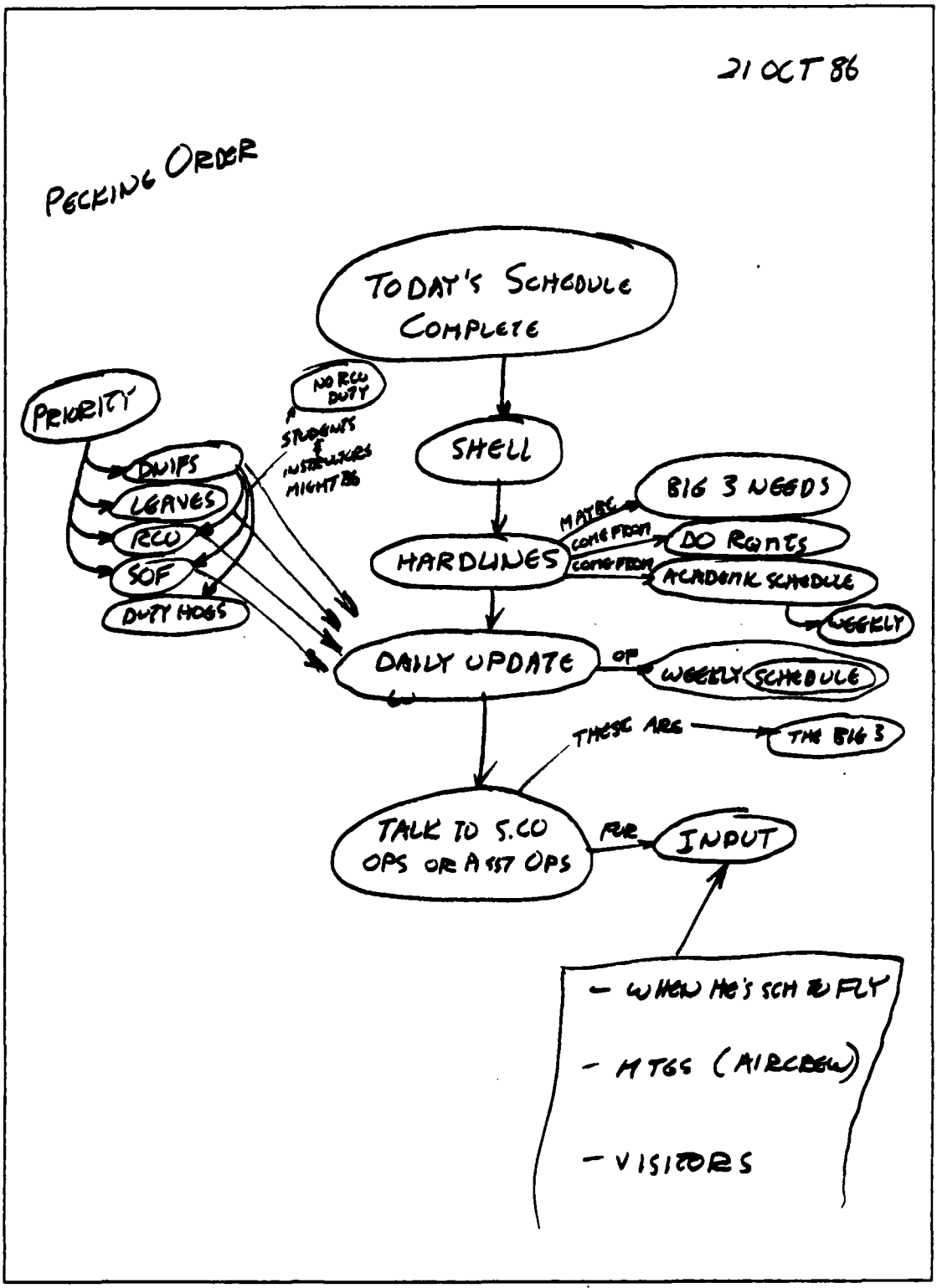


Figure B.6. Capt Trapp's Rough Concept Map

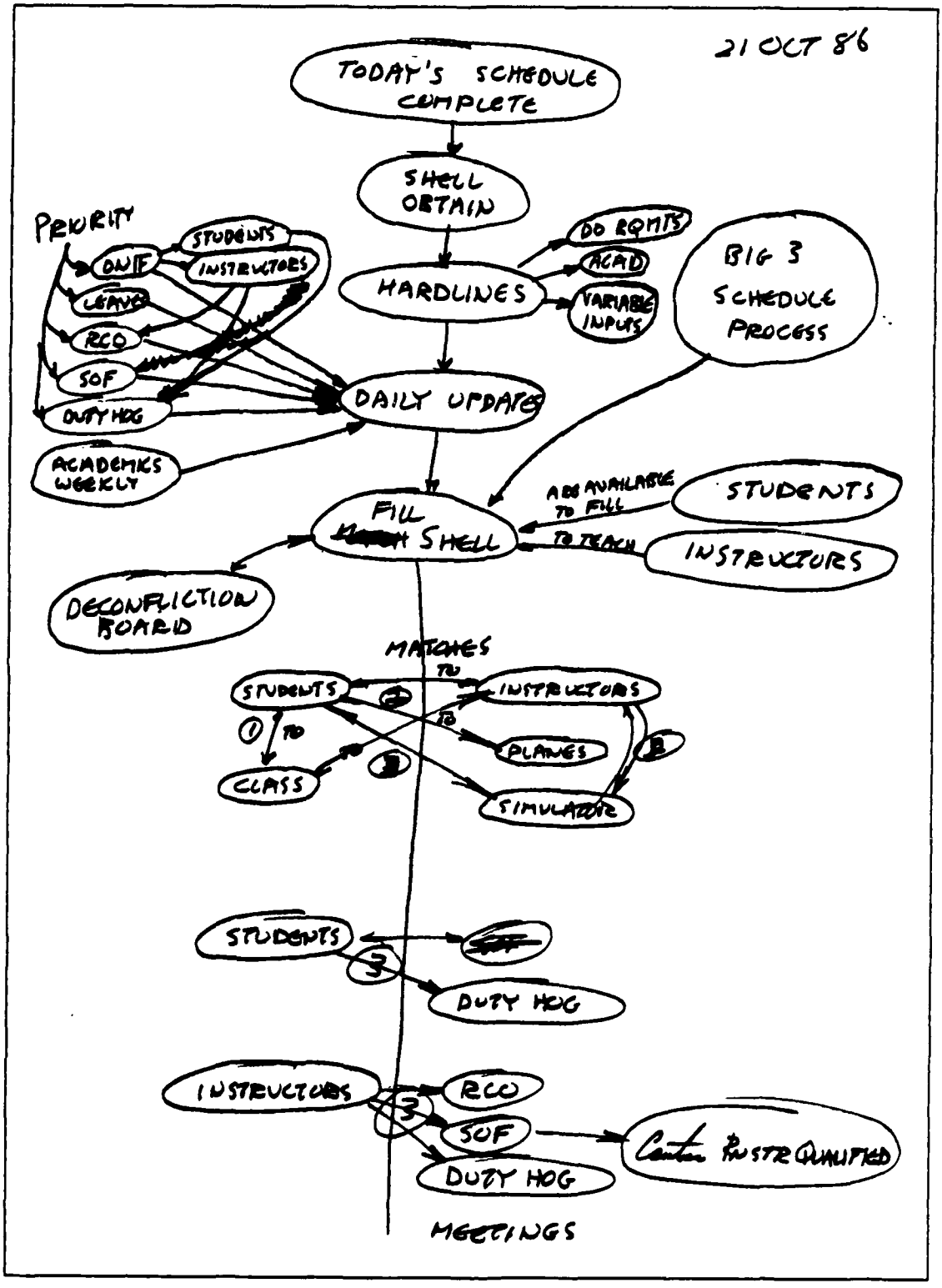


Figure B.7. Second-cut of Decision Process (Iteration #1)

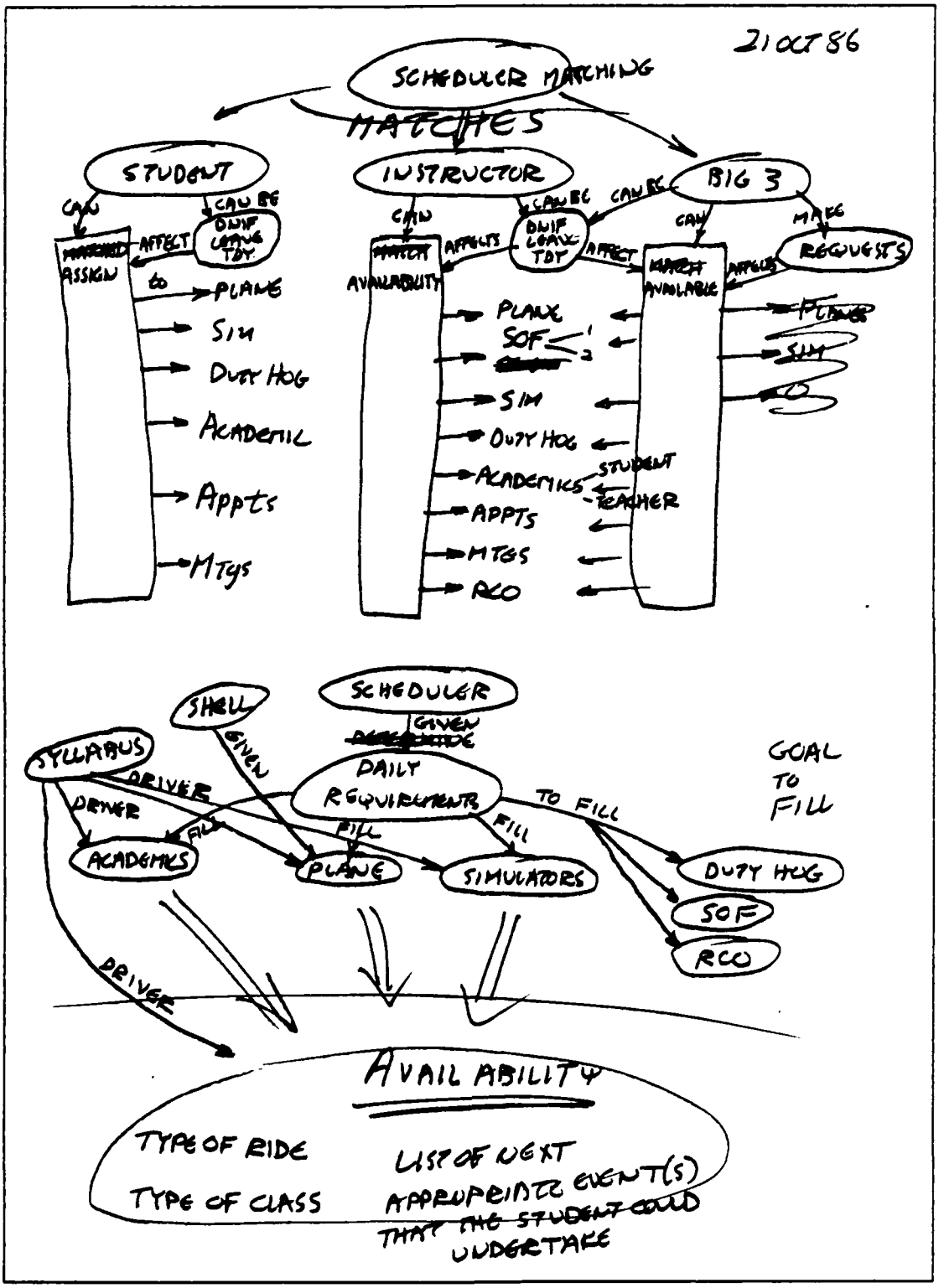


Figure B.8. Second-cut of Decision Process (Iteration #2)

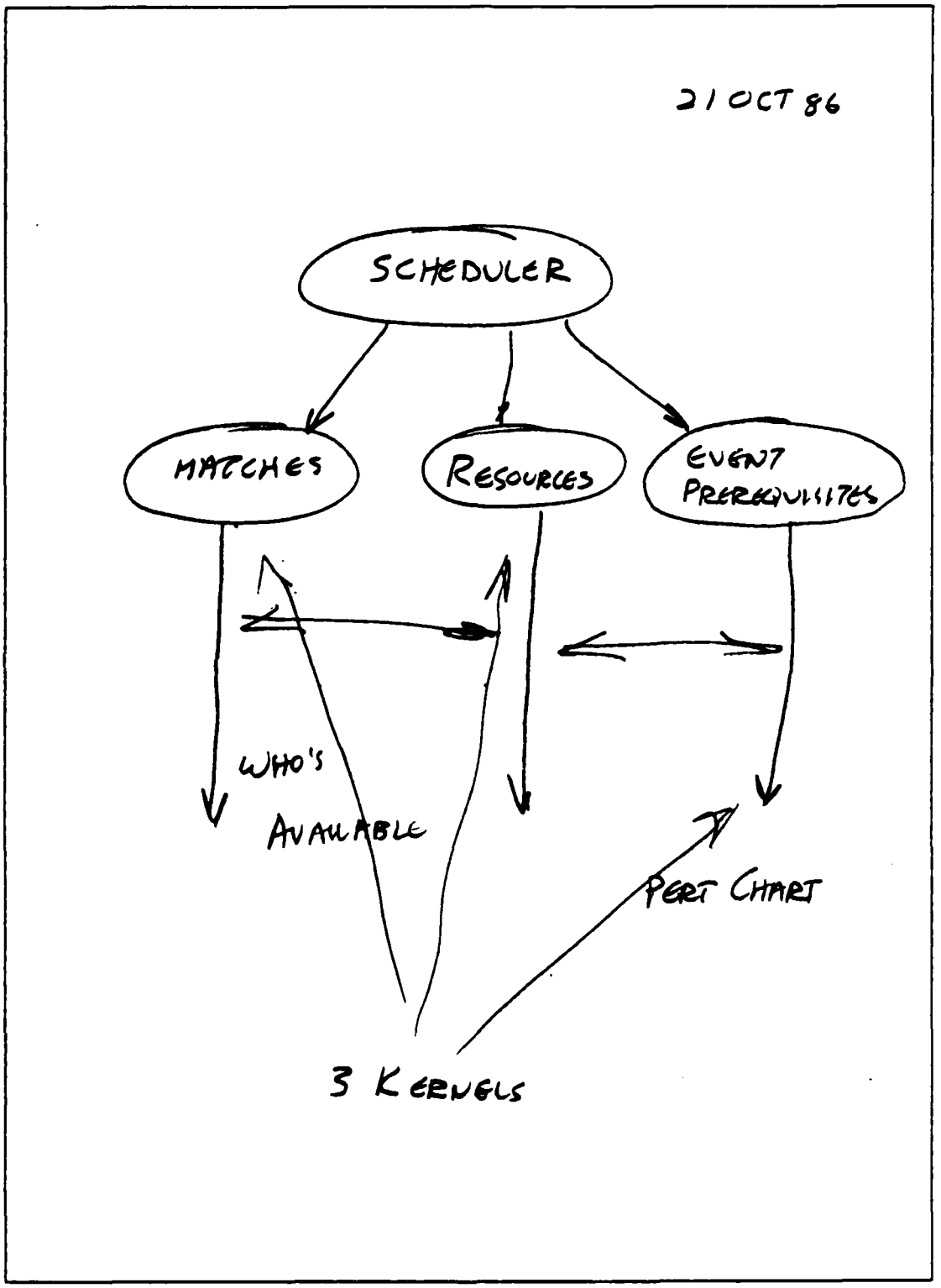


Figure B.9. Concept Map Formulation

APPENDIX C

Task Analysis

Appendix C contains the evolution of the task analysis. It includes a basic scheduling overview as well. This review was for Capt McFarren's benefit as he was unfamiliar with the particulars of aircrew training scheduling. This iterative process of forming the task analysis results in the completed depiction shown in Figure C.5.

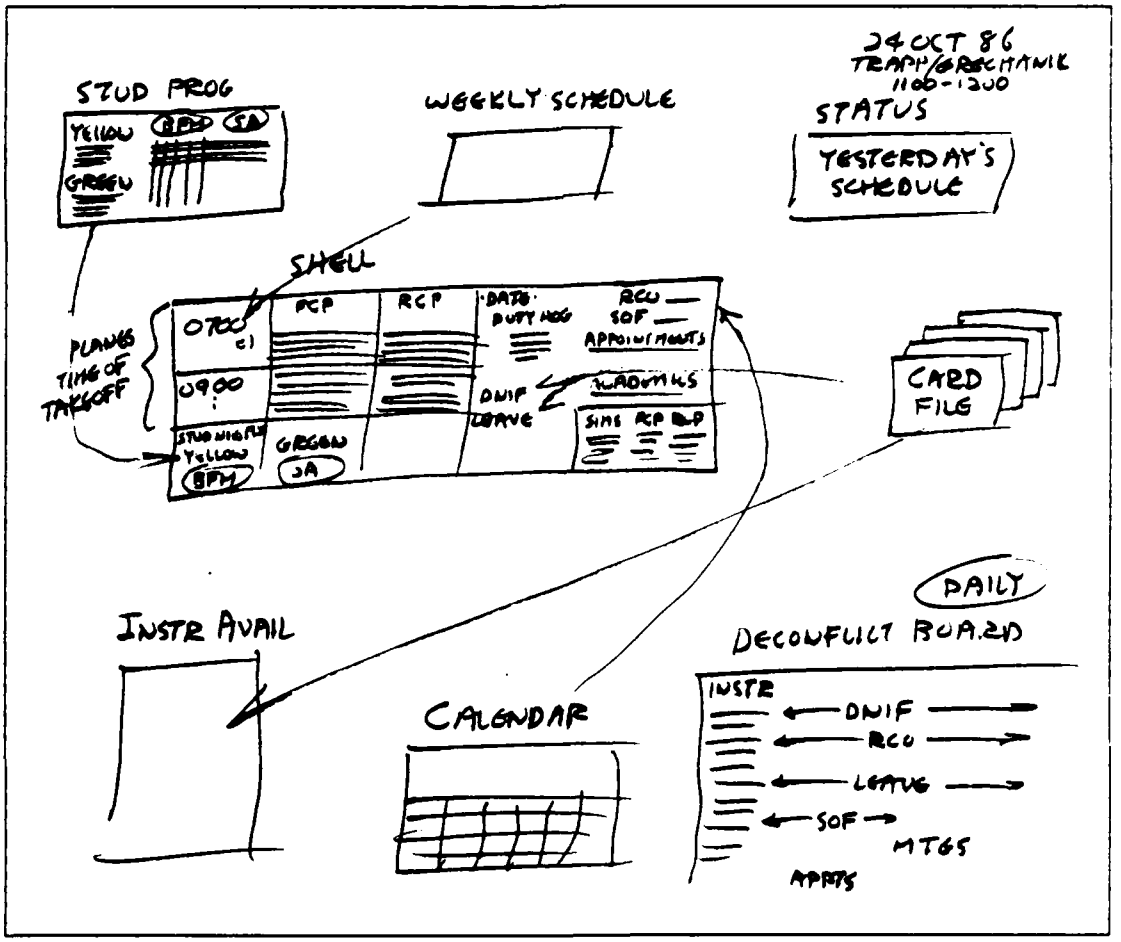


Figure C.1. Scheduler's Input Sources

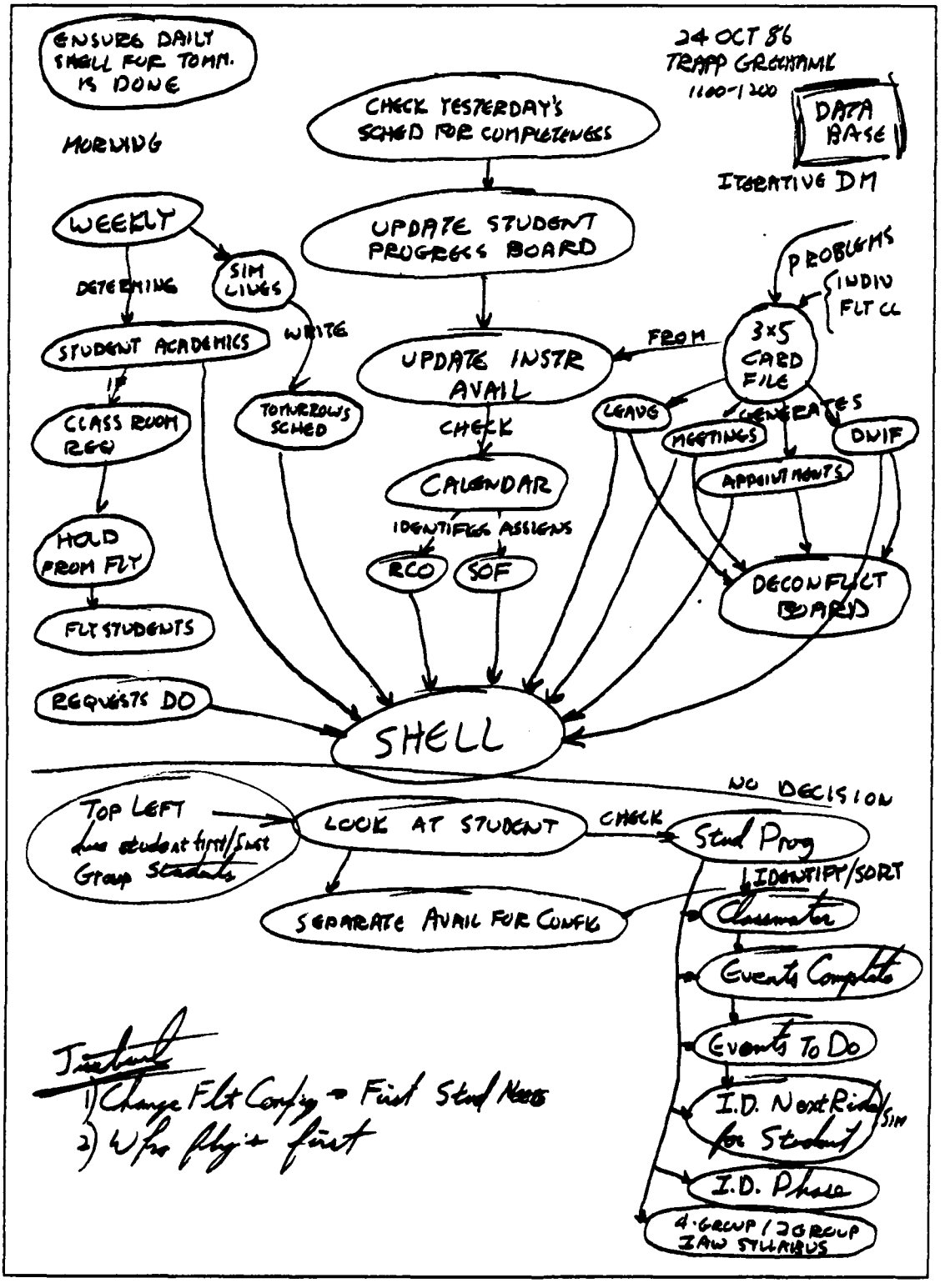


Figure C.2. First Task Analysis Iteration

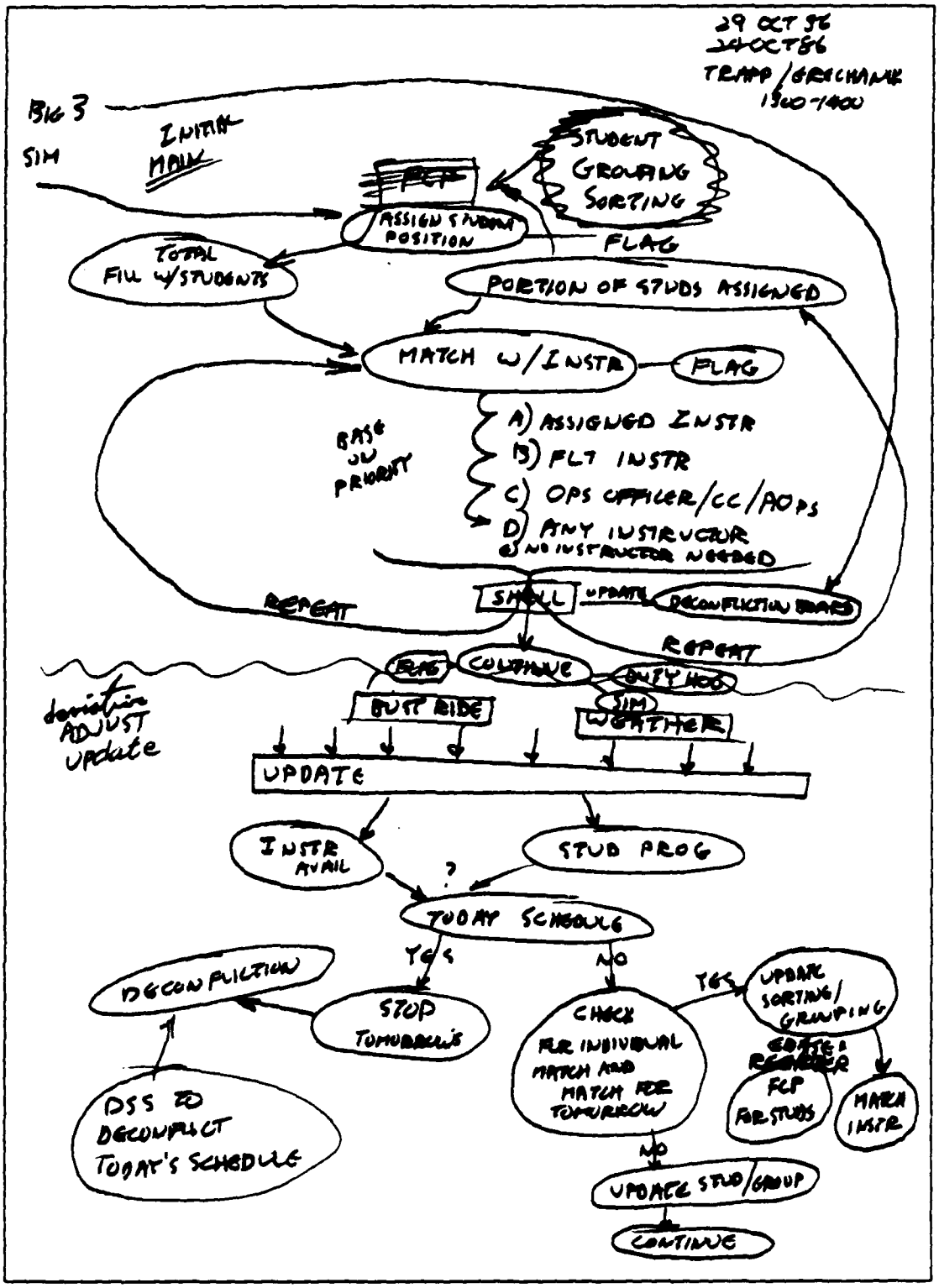


Figure C.3. Second Task Analysis Iteration

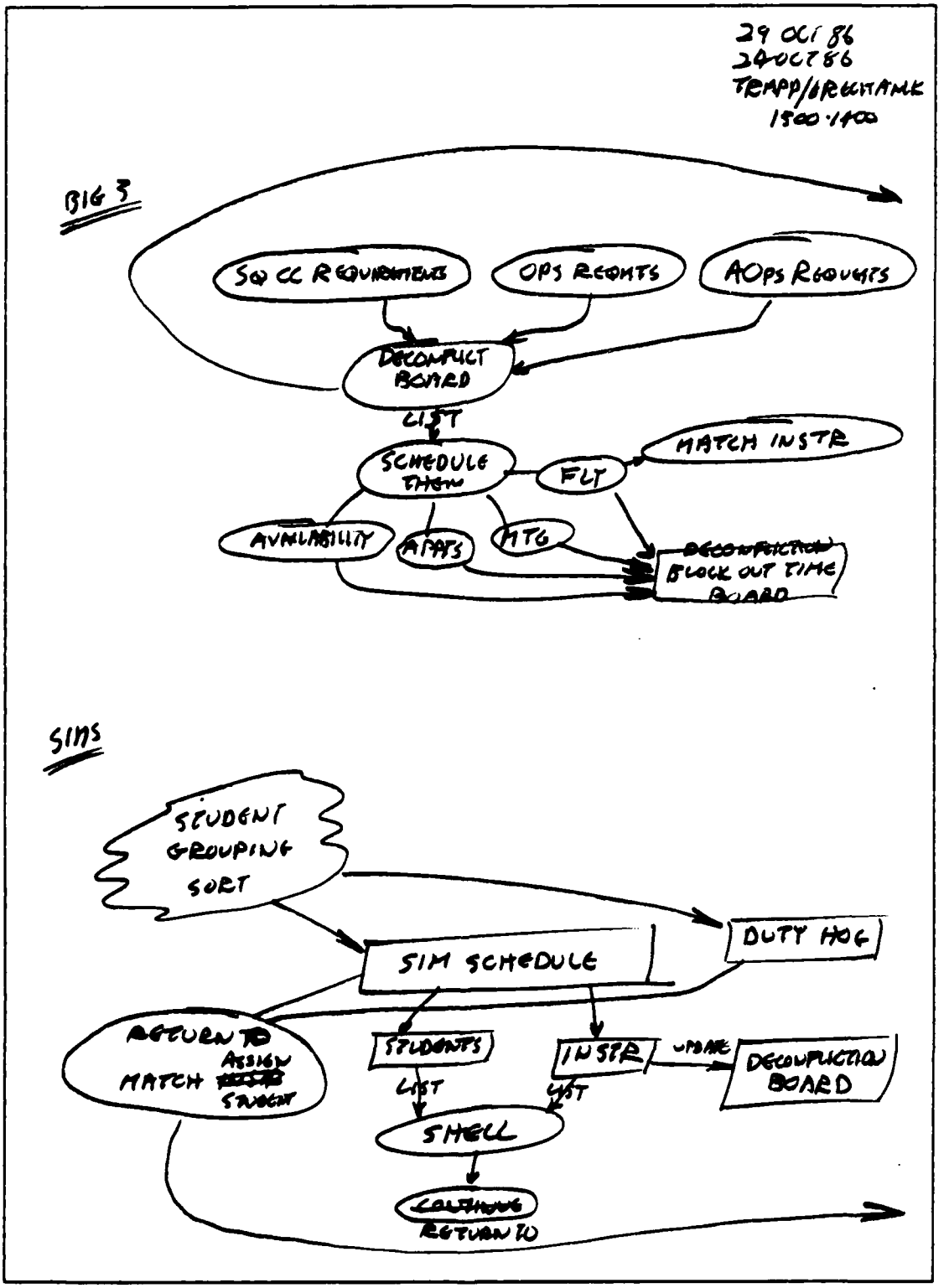


Figure C.4. Third Task Analysis Iteration

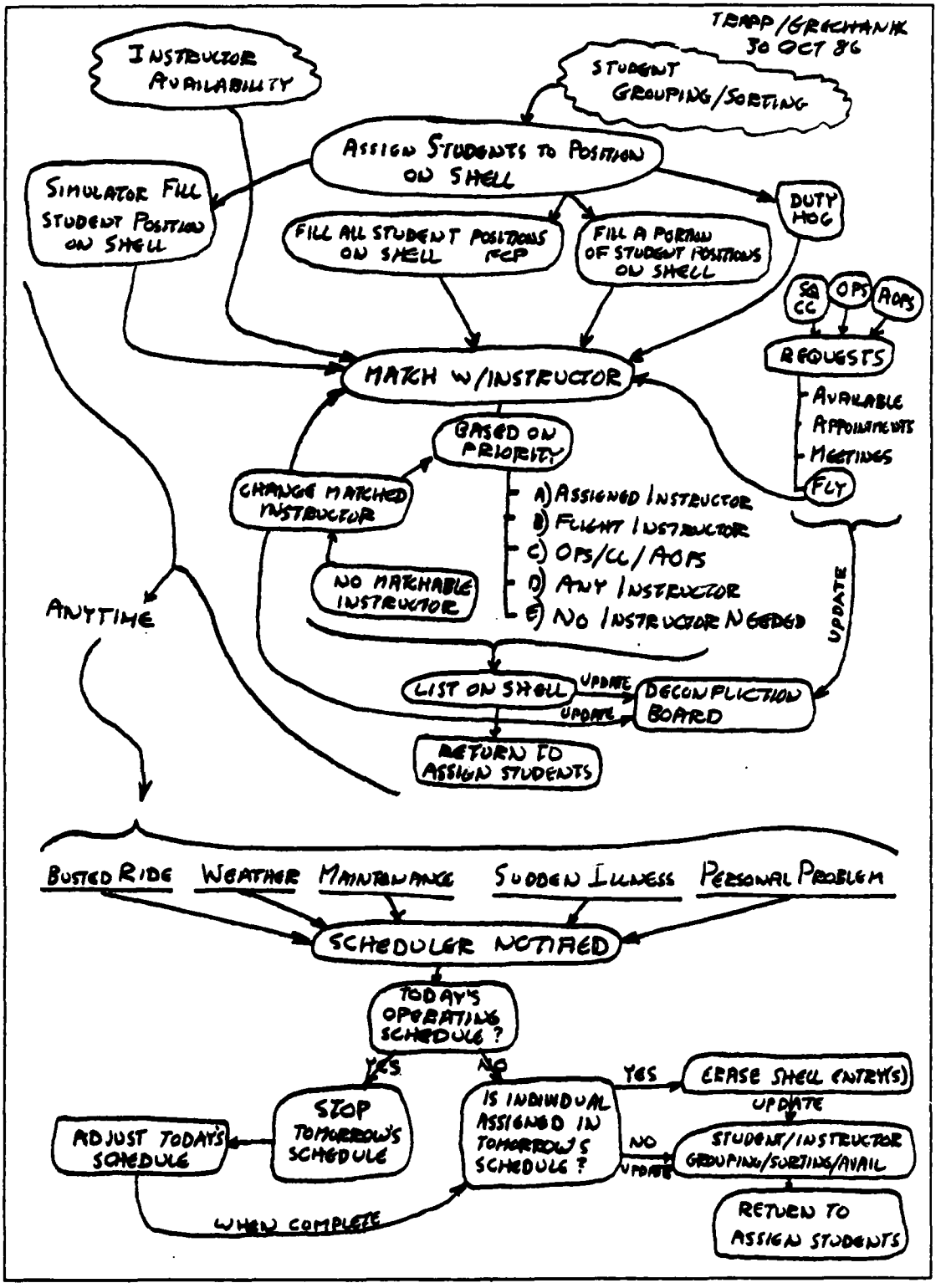


Figure C.5. Completed Task Analysis

APPENDIX D

Data Analysis

Appendix D contains the database relations used for the kernel system. The figure below shows the data variables and the screens under which they fall.

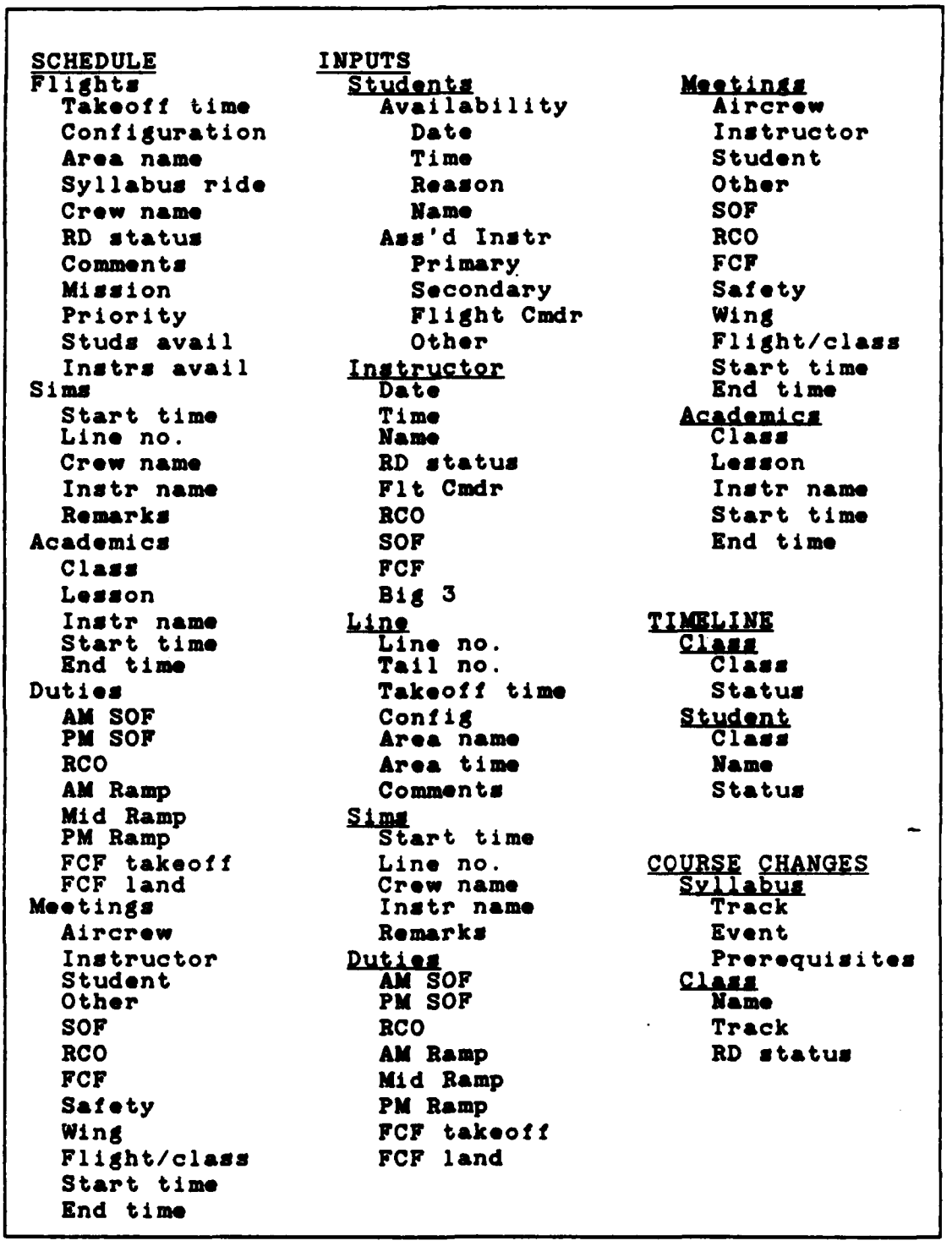


Figure D.1. Data Analysis

APPENDIX E

Feature Chart Evolution

Using the scheduling screen hierarchy depicted in Figure 3-7 and the database relations in Appendix D, the following portion focuses purely upon the requirements needed by the user. This appendix describes the features that each screen must perform. There was no bias towards any commercial software or their particular capabilities or limitations. This process reflected the user's needs without regard to any technological restraint. Should a limitation occur after the feature chart has been developed, the DSS will have to be modified to incorporate as much as possible within current commercial software bounds.

Evolution. The evolution of the system via the feature chart concentrates on the overall system and the individual portions. Each category is important because it is from these features that the DSS will be built. To view the overall picture, the feature chart for the system is discussed first.

System. The system itself must contain several features. First, it must link the data bases to the scheduling process itself so that it becomes interactive. This will ensure the user works with the most current and accurate information. Next, the system must deconflict individuals and

events so that scheduling two people for the same event at the same time (or one person for two events at the same time) does not occur. With these features in mind, the following sections focus on each portion of the system.

Main Menu. Referring to the screen hierarchy (Fig. 3-7), the first feature addressed was the main menu. An important characteristic of this menu was to be able to access it from any place in the DSS. Once activated, it should take the user to any other portion of the system.

Tomorrow's Schedule. The next item considered was Tomorrow's Schedule. The primary feature required was to include a full day's flying events upon a single screen - a limitation which currently existed. This screen must contain space for takeoff time, crew names, aircraft configuration, area name, mission type, and comments for each sortie. Also, pop-up windows displaying who was available to fly was necessary. These windows would display prioritized lists from which the scheduler would choose as well as an option to change the priority used to create the lists. These features should resemble the grease boards currently in use.

Today's Schedule. The features mentioned above would be incorporated into Today's Schedule. The presence of this schedule was a convenience to the user, as he often must change it when daily deviations occur. It would also eliminate the need of having to load the schedule from the DSS files each time a change was needed.

Today's Schedule/Daily Event Update. This screen would query the user as to whether or not Today's Schedule events went as planned. It would then, based upon his response, update the data base. The screen must be able to discern the type of event, who was scheduled, and the reason for the deviation.

Inputs/Student/Availability. This screen must be capable of adding or deleting student events which affect his scheduling availability. These events include DNIF, TDY, and leave and the applicable day(s). Also, the screen must have the capability to enter appointments or meetings which may span hours or days.

Inputs/Student/Assigned Instructors. This screen must be able to input the instructors a student is assigned to fly with. It should reference the student's class and primary instructor. In addition, the screen should have space for up to five extra assigned instructors. This screen will primarily be an input to the data base used to prioritize instructor matching to students.

Inputs/Instructor. This screen is essentially the same as Inputs/Student/Availability, but with an additional feature. The extra feature must be able to input changes to an instructor's qualification data base. These qualifications would include red dot, RCO, SOF, flight commander, Big 3, and functional check flight.

Inputs/Academic. In the same manner as the previous screen, the primary feature would be to update or change an academic meeting. It should have the option to change the availability of either a class of students, instructor, or both. The screen should reference the student class, date and time of the academic meeting, and the instructor assigned to teach.

Inputs/Duties-Meetings. This screen must display the numerous additional duties and meetings which occur each day. Also, there must be space to enter instructor or student names, depending on the type of duty. Much like the schedule screens described above, this screen should have the pop-up windows displaying choices of qualified and available individuals.

Time Line/Class. This screen must be able to graphically depict the time line for a student class. The presentation should be a bar-type graph for ease of analysis showing the status to be above or below the zero (even) level. The screen should also contain a numerical representation for individual classes as well as the date of the last update.

Time Line/Student. This screen must be able to graphically depict the time line for each student in a class. The presentation should be a bar-type graph for ease of analysis showing the status to be above or below the zero (even) level. The screen should also contain a numerical represen-

tation for individual students, a total time-line status for the class, and the date of the last update.

Course Changes/Syllabus. This presentation must list the specific syllabus track under revision. Also, it must list each track event and the prerequisites needed to accomplish that event. There should be space enough for at least eight prerequisites for each event.

Course Changes/Class. This screen must be able to add, delete, or change the members of a student class. The presentation should display the students with their assigned syllabus track and an indication of red dot status. There should be a date showing the last time an update happened.

APPENDIX F

Storyboard

The actual DSS computer screen output format (storyboard) must support the basic premise of user process flexibility. In other words, it must be able to assist users with varied experience and individual techniques, yet not be cumbersome to interpret or manipulate. At the same time the DSS must be powerful enough to support any decision sequences the user may require. The latter requisite is best accomplished using an iterative approach over a period of time with the user employing the DSS and relying on the structure of the kernel to assist him. As such, the authors base the screen output format on the decision process and kernel identified in the previous chapter.

The authors use the ROMC user-builder interface technique to develop the storyboard. ROMC is the acronym for Representation, Operations, Memory aids, and Control Mechanisms. Each of these tools enables the designer to translate user requirements into DSS components. This structure allows the flexibility necessary to satisfy both the user's needs and the builder's design requirements. The storyboard needs these tools to build effective output screens.

Representation, the first tool, depicts the actual screen presentation needed by the user. This data representation must

satisfy the user's needs in a clear and concise manner. The order of the representations must be logical as well, conforming to the user's decision or thought process sequence. This last thought is especially important because the DSS should not distract the user as he thinks, rather it should ease his decisions by presenting him the next piece of information when needed. Should the user require closer scrutinization of a particular piece of data or focus in on a present screen portion, manipulation of the representation may be necessary.

The second tool, operations, enables the user to manipulate the representations on the screen to suit his individual technique. This may take the form of actual data manipulation (e.g., sorting), scale change (e.g., viewing a larger or smaller portion of the screen), or adding/deleting various amounts of data for clarity or interpretation. Of course, the user accomplishes these operations when needed so as not to distract his thought process.

The next tool used in designing the storyboard is memory aids. These helpful reminders guide the user through his decision process with the use of icons, windows, highlighted (colored) information, or various flags. Memory aids remind or warn the user of certain decisions at specified times.

Finally, control mechanisms are interwoven throughout the entire kernel, enabling any user to skip tedious or familiar processes. Control mechanisms ease movement to any part of the

kernel. Control mechanisms may take the form of selectable menus or predefined function keys.

A problem facing the storyboard itself is how to place all of the necessary information on the screen without cluttering the presentation. The easiest way to present the information is to diagram all of the kernel components carefully. The authors arranged and grouped these components in a logical manner according to the user's decision process. The authors then ordered these componenets into the screen design.

Table F.1 show the information the scheduler needs to make a schedule:

TABLE F.1

Needed Scheduler Information

- Tomorrow's schedule.
  - Available students.
  - Available instructors.
  - Priority considerations.
  - Ability to print schedule.
- Today's schedule.
  - Ability to update events.
- Schedule inputs.
  - Flying sortie lines from wing
  - Simulator lines from wing.
  - Student.
    - Availability.
    - Assigned instructors.
  - Instructor Status.
  - Academic classes and instructors.
  - Duties and meetings.
- Statistics.
  - Time line by class.
  - Time line by individual student.
- Course changes.
  - Student syllabus.
  - Class additions/deletions.

This information is best presented in a network hierarchical fashion (Figure 3.7). Once this network is established, the authors may design the storyboard using the ROMC tools previously mentioned.

Using the scheduling screen hierarchy depicted in Figure 3.7, the following portion will focus purely upon the requirements needed by the user. There will be no bias towards any commercial software or their particular capabilities or limitations. This is to reflect the user's needs without regard to any technological restraint. After storyboard development, the DSS will have to be modified to include the kernel within current commercial software bounds.

MAIN MENU

Representation: This presentation lists all of the screens in the scheduling program hierarchy. The screen is compressed to the right of the page to allow overlaying of any other screen without hiding information. The main sub-headings are shown in capitals.

Operations: The user employs the up and down arrows to select one of the categories. Hitting [ENTER] will display the selection. As users gain more experience, the letters along the left side of the menu will activate the desired selection.

Memory Aids: As the user scrolls through the selections with the up and down arrow, a one line prompt appears at the bottom of the screen explaining where the cursor rests.

The major categories in the scheduling program hierarchy are capitalized. Further subheadings are indented and not all capitals to further break out the list to the user.

Control Mechanisms: Pressing function key [F1] brings up online help on the screen. Pressing [F1] again gets rid of it.

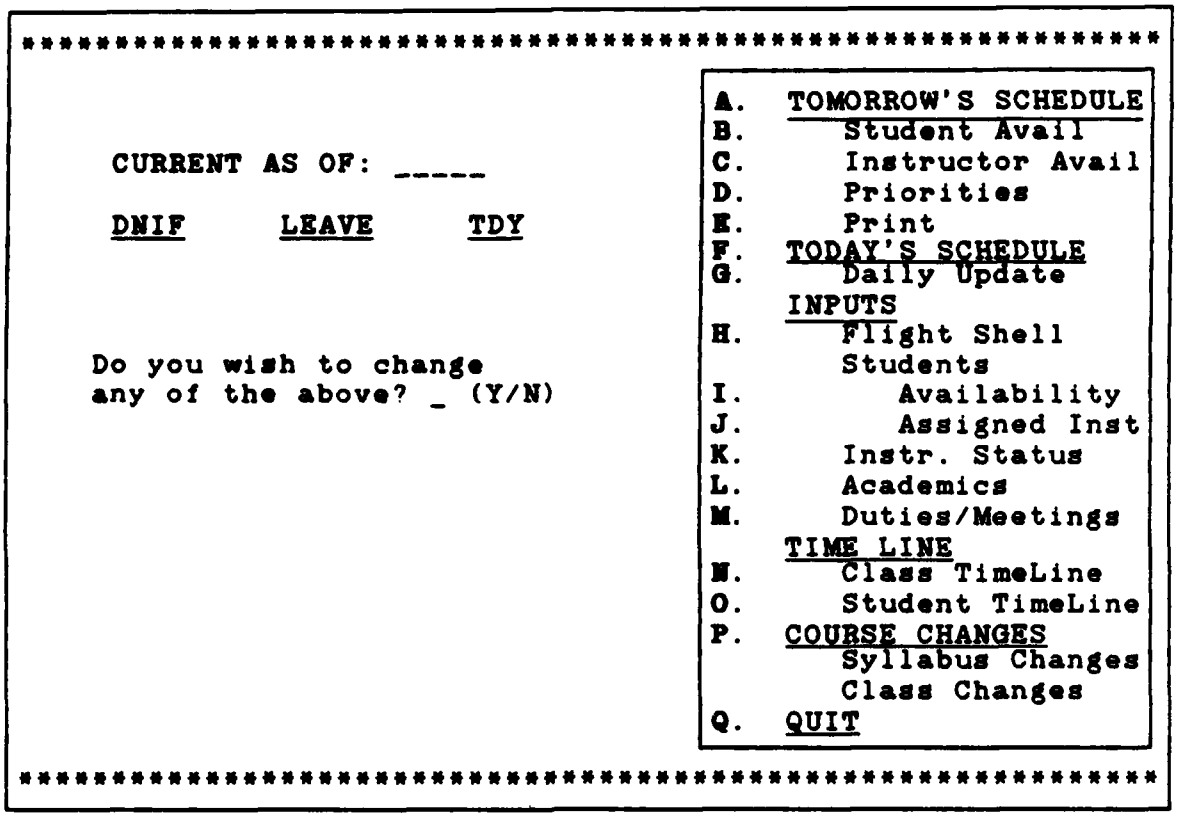


Figure F.1. Main Menu

TOMORROW'S SCHEDULE

Representation: This presentation depicts all of the flying sorties on one screen for ease of scheduling. The lower windows will overlay the screen trying to hide as little information as possible.

Operations: The user positions the cursor, via arrow keys, to certain items of the schedule (e.g., on a FCP space). Upon selecting STUDS AVAIL, a prioritized list of students will appear on the right side of the screen. The user then selects a student name. Upon pressing [ENTER], the student's name will be placed under the cursor. Similar operations occur upon selecting INSTRUCTORS AVAIL and PRIORITIES. PRINT will print out a hard copy of the schedule. QUIT will exit the scheduling program after saving the work.

Memory Aids: As each option is selected, a one line explanation appears directly below. For instance, if STUDS AVAIL is selected, a message stating 'A PRIORITIZED LIST OF STUDENTS' will appear.

Control Mechanisms: The arrow keys will move the cursor to the next field. The information may be entered manually by positioning the cursor where desired and typing it in. Typing in new information is activated either by the [ENTER] key or selecting one of the fields described in Operations. The [F1] key brings up HELP instructions for this screen. The [F2] key brings up the MAIN MENU.

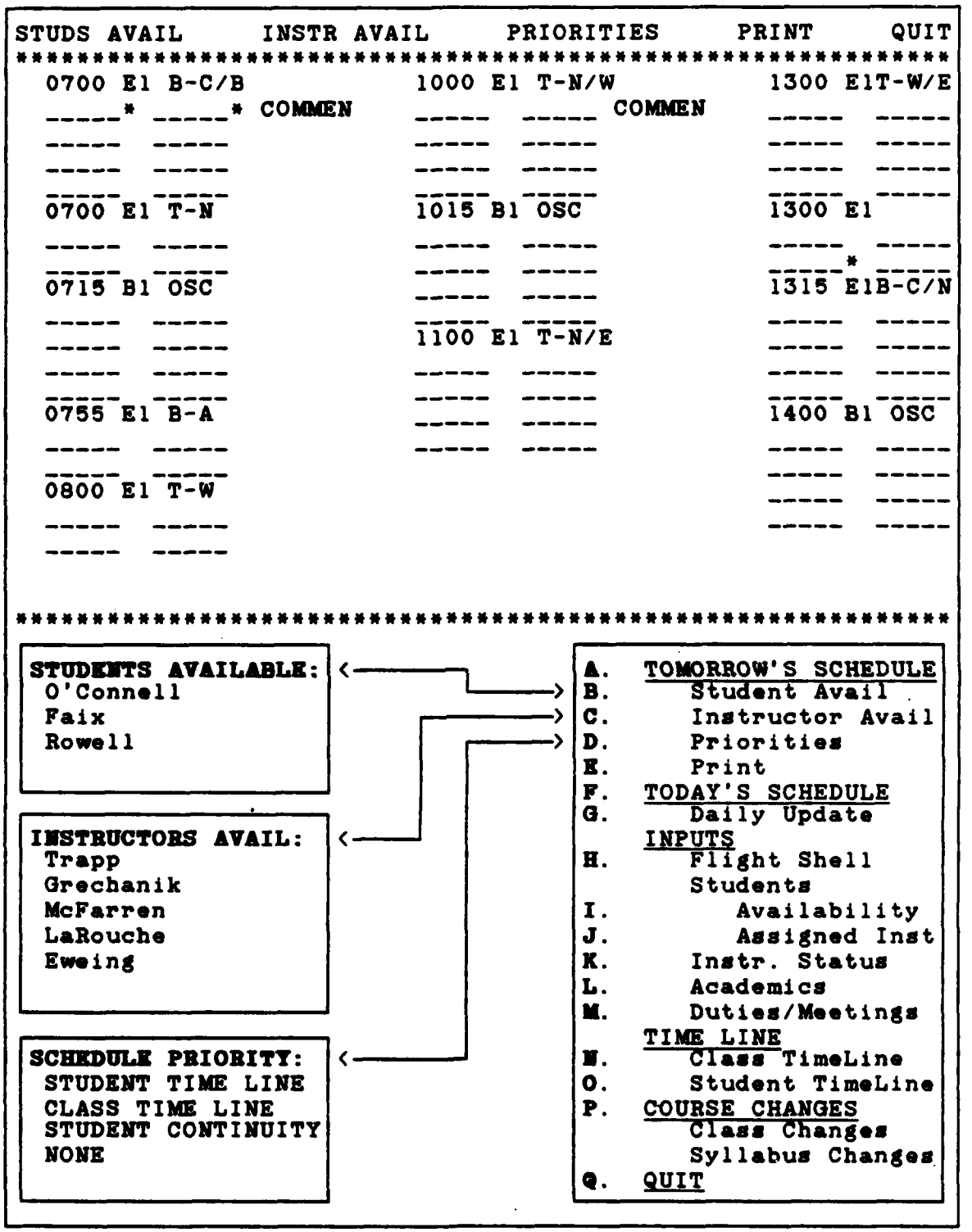


Figure F.2. Tomorrow's Schedule

TODAY'S SCHEDULE

Representation: This display is much the same as that of TOMORROW'S SCHEDULE. The type ride will have been filled in as the schedule was completed. Names will be present in the underlined spaces. TODAY'S SCHEDULE will be a copy of TOMORROW'S SCHEDULE that was worked on the day before.

Operations: The UPDATE option allows the scheduler to annotate any deviations that occurred on TODAY'S SCHEDULE. This option also lets the user research any past schedule to find out what was scheduled. PRINT allows a hard copy of whatever is on the schedule at that time to be printed. The QUIT option saves his work and takes the user back to the MAIN MENU. Function key [F1] brings up instructions on how to use the schedule screen.

Memory Aids: As each option is selected, a one line explanation appears directly below. For instance, if UPDATE is selected, a message stating 'Update TODAY'S SCHEDULE' will appear.

Control Mechanisms: The arrow keys will move the cursor about the screen to each field. The user will be able to enter information in two ways. First, the information may be entered manually by positioning the cursor where desired and typing it in. Typing in new information is activated when hitting the [ENTER] key. Second, the user may select one of the fields described in Operations. The [F1] key brings up HELP instructions for this screen. The [F2] key brings up the MAIN MENU.

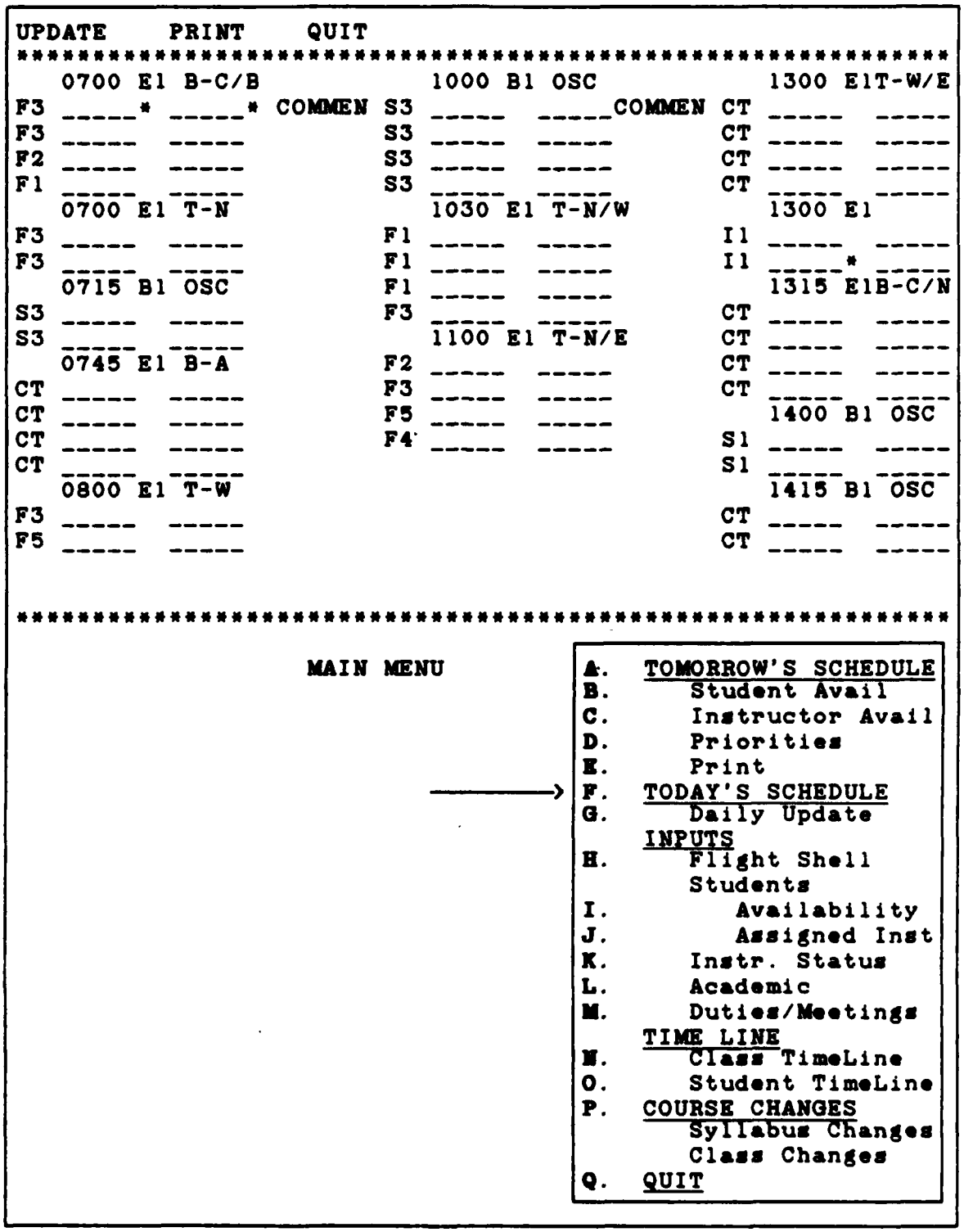


Figure F.3. Today's Schedule

TODAY'S SCHEDULE / DAILY EVENT UPDATE

Representation: The presentation will be a series of questions and fillins. The (Y/N) questions will have default values so the user may simply accept the values proposed by the program.

Operations: The user may select either a sortie, simulator, student, or classroom activity, and give the reason for the deviation. He will then be queried as to the accuracy of his inputs. The screen will further prompt him for additional deviation occurrences.

Memory Aids: The user is always prompted as to the accuracy of his inputs and if he wants to quit or continue.

Control Mechanisms: The arrow keys allow movement about the screen. Operations upon the representation consist of filling in the appropriate entry and hitting the [ENTER] key. Answering NO to the 'DO YOU WISH TO MAKE ANOTHER CHANGE' question will exit to TODAY'S SCHEDULE. The [F1] key brings up HELP instructions for this screen. The [F2] key brings up the MAIN MENU.

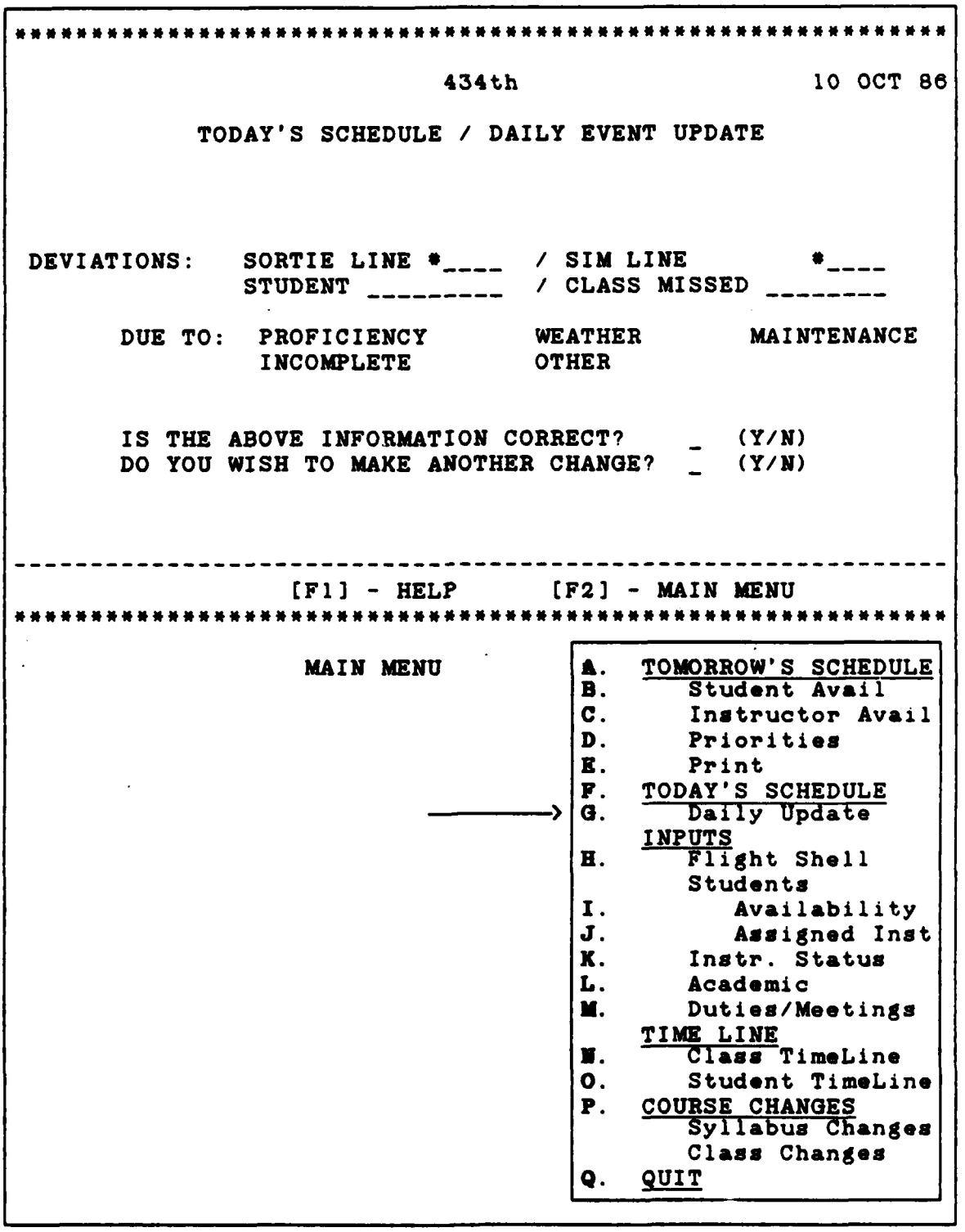


Figure F.4. Daily Event Update

WING LINES INPUT PROMPT

Representation: The presentation will be a prompt to insert the disk containing the daily wing line data.

Operations: The user is queried for the correct disk.

Memory Aids: The user is prompted for his inputs and his next action.

Control Mechanisms: No control mechanisms are used for this screen.

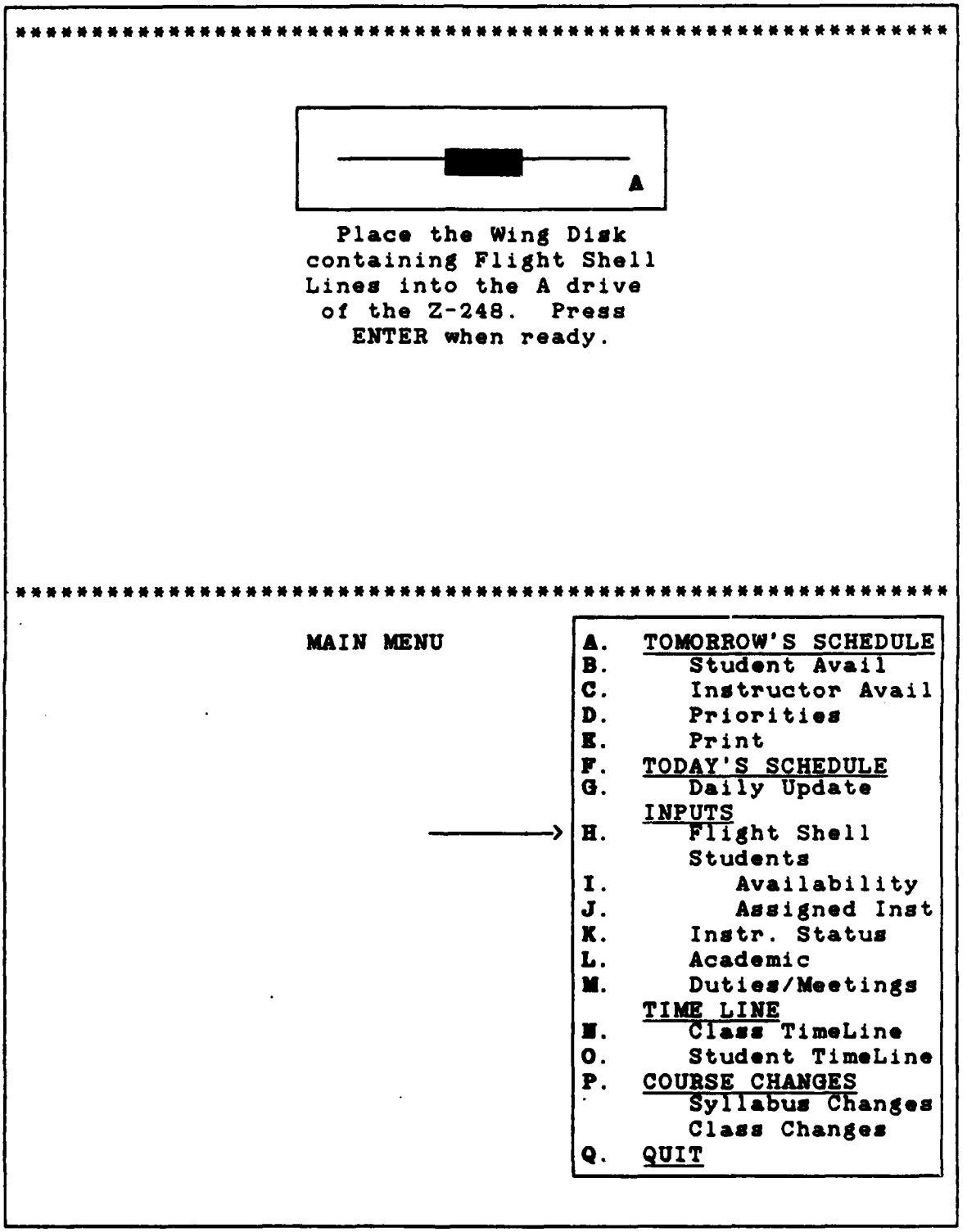


Figure F.5. Wing Lines Input Prompt

INPUTS / STUDENT / AVAILABILITY

Representation: The student's name and whether the event will be added or deleted comprises the initial input line on the screen. A vertical list to include DNIF, TDY, LEAVE, and OTHER event type follows with start/end dates. Another OTHER event line is included with start and end times for a specific date. The QUIT option allows the user to return to the MAIN MENU.

Operations: The user inputs the students name and whether to add or delete his name from one of the availability categories.

Memory Aids: The user is shown most available categories on the screen. Addition or deletion and student name are prompted for.

Control Mechanisms: The arrow keys will move the cursor along the available choices while the [ENTER] key implements the selection (for the ADD, DELETE, DNIF, TDY, and LEAVE entries only). The rest of the entries must be input before using the [ENTER] key. The user is prompted for another change. If yes, the information is input and the screen cleared for another input. If no, the information is input and the user is returned to the main menu. The [F1] key brings up HELP instructions for this screen. The [F2] function keys displays the MAIN MENU.

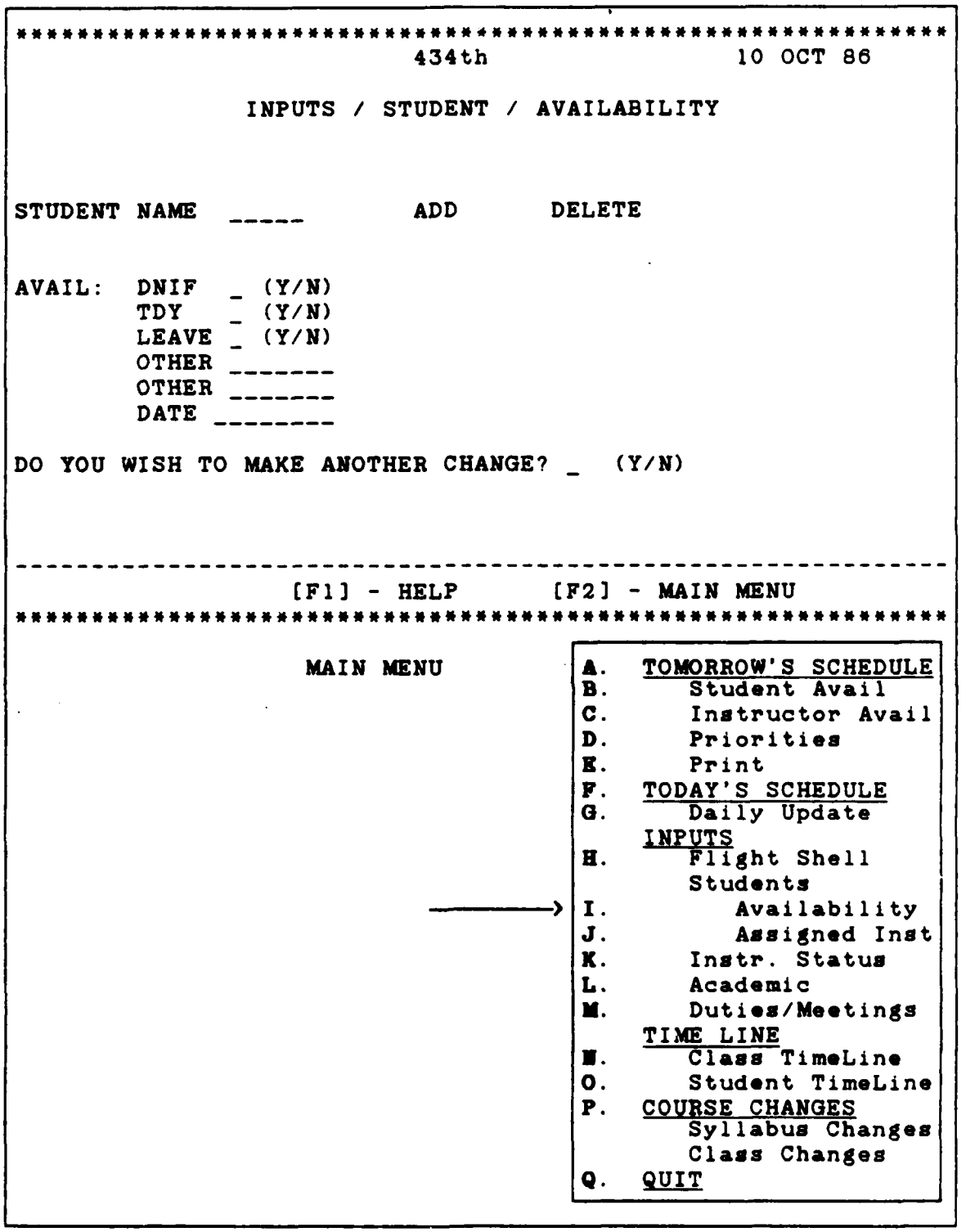


Figure F.6. Student Availability

INPUTS / STUDENT / ASSIGNED INSTRUCTORS

Representation: The student's name is listed by class. To the right is the primary and secondary assigned instructor's name. Space for additional instructors is provided in the adjacent columns. The QUIT option allows the user to return to the MAIN MENU.

Operations: The user will input the primary and secondary assigned instructors. The user is then prompted for another input with 'DO YOU WISH TO MAKE ANOTHER INPUT?'. If yes, the information is saved and the screen cleared for another input. If no, the information is input and the user is returned to the main menu.

Memory Aids: The headings are listed directly above the input slots. One line information prompts will be displayed at the bottom of the screen when the cursor rests on specific locations. When the cursor rests below the PRIMARY slot the prompt 'Enter the students Primary Instructor' will be at the bottom of the screen. [F1] - HELP is listed at the bottom of the screen.

Control Mechanisms: The [F1] key brings up HELP instructions for this screen.

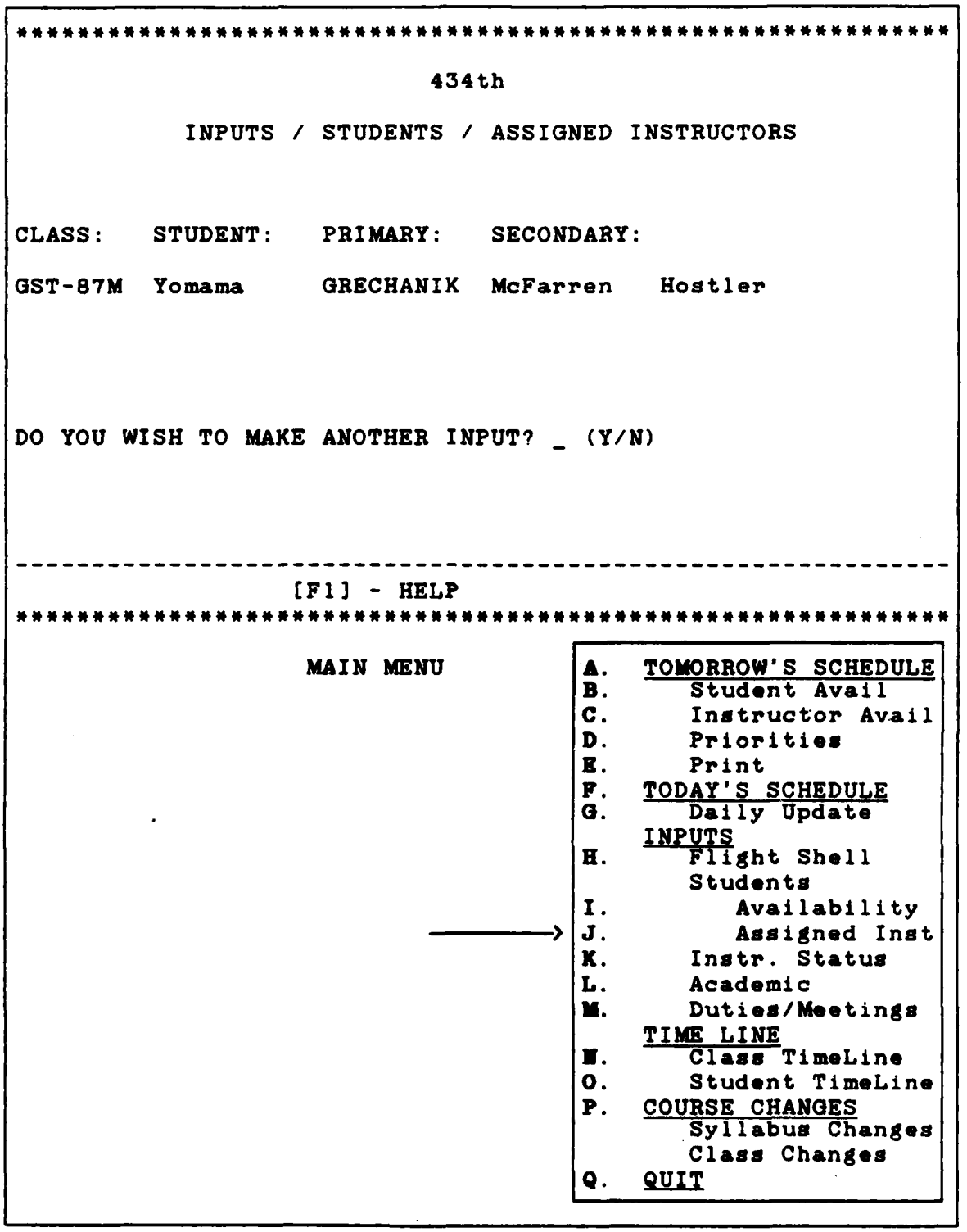


Figure F.7. Student Assigned Instructor

INPUTS / INSTRUCTOR

Representation: The instructor's name and addition or deletion is the initial input line on the screen. The user is prompted for a STATUS or AVAIL input. A vertical list to include DNIF, TDY, LEAVE, and OTHER event type follows with start/end dates. Another OTHER event line is included with start and end times for a specific date. The lower prompt allows the user to continue or return to the MAIN MENU.

Operations: This screen changes the status or availability of a instructor in the database. The Instructors name may be added or his status may be changed. The user is then prompted for another input with 'DO YOU WISH TO MAKE ANOTHER CHANGE?'. If yes, the information is saved and the screen cleared for another input. If no, the information is input and the user is returned to the main menu.

Memory Aids: One line of information, specific to the cursor location, will be displayed as the cursor is moved from slot to slot.

Control Mechanisms: The [F1] function key brings up HELP instructions for this screen.

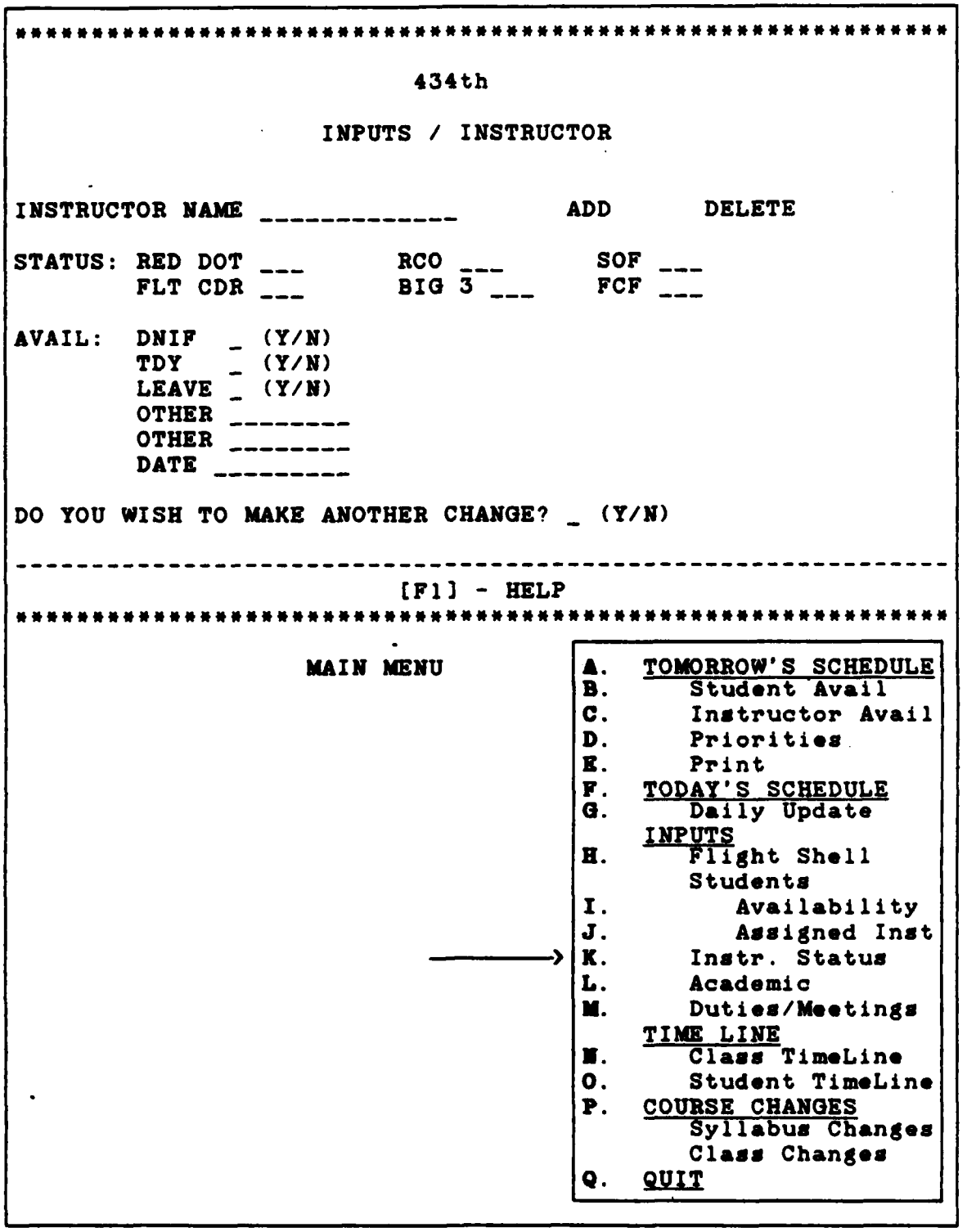


Figure F.8. Instructor Status

INPUTS / ACADEMIC

Representation: The data will be entered in vertical columns titled using the shown information headings. The QUIT option allows the user to return to the MAIN MENU.

Operations: The data will be used in a database to deconflict the academic instructors and student classes

Memory Aids: The columns are titled appropriately to assist the user.

Control Mechanisms: The arrow keys will move the cursor along the available choices. The entries must be typed before using the [ENTER] key. The [F1] key brings up HELP instructions for this screen.

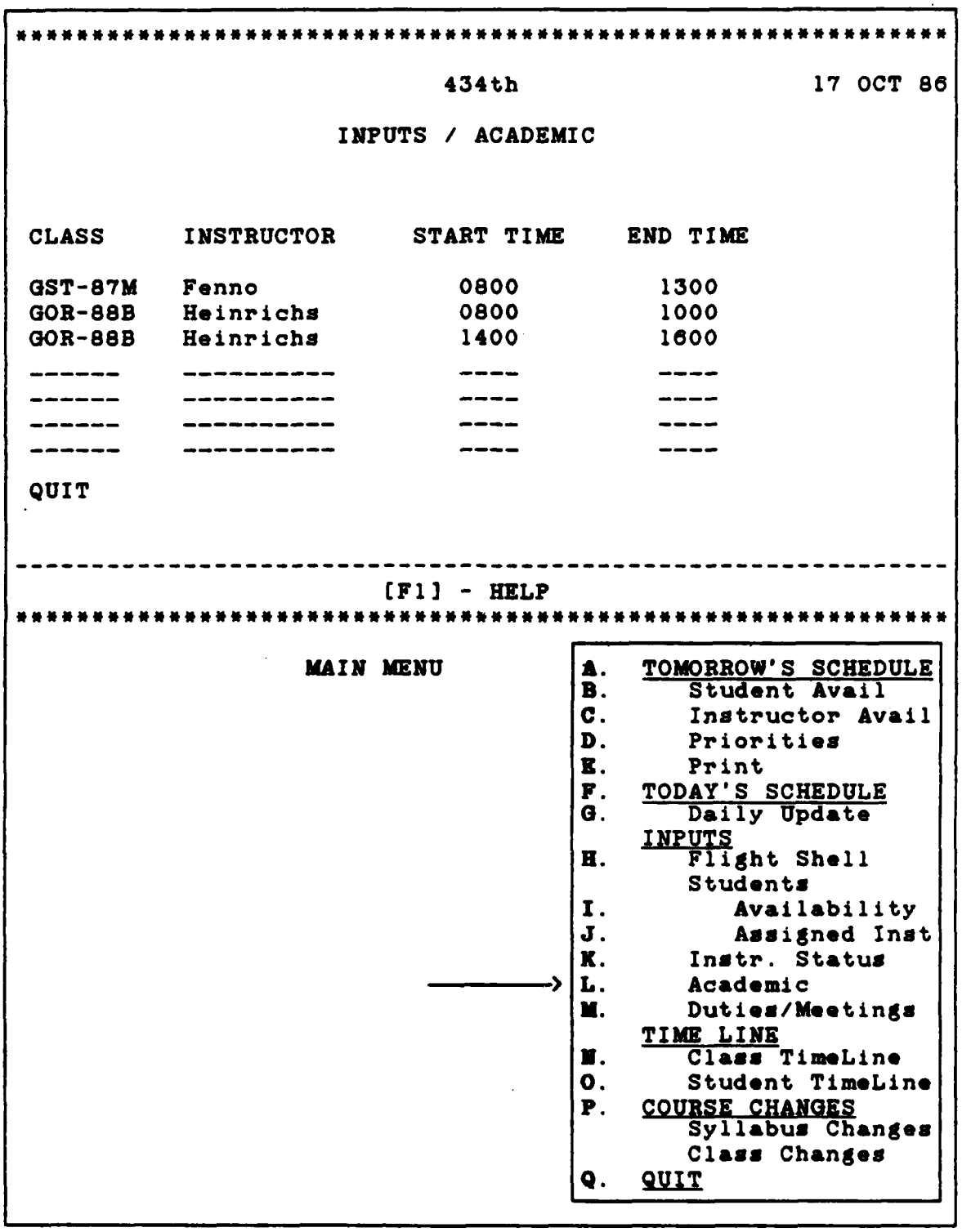


Figure F.9. Academic Inputs

INPUTS / DUTIES-MEETINGS

Representation: DUTIES will be the first category to include inputs for SOF, RCO, RAMP supervisor, functional check flights (FCFs), "BIG 3," and DUTY SWINE entries. SOF has space for AM or PM duty as well as Primary or Secondary slots. RCO and DUTY SWINE have START/END times. All entries have spaces for INSTRUCTOR names. In addition, STUDENT name slots are provided for DUTY SWINE. MEETINGS comprise the rest of the screen. The type of meeting (AIRCREW, STUDENT, INSTRUCTOR, OTHER, SOF RCO, RAMP, FCF, or SAFETY is selected first. The managerial level is selected next WING, SQUADRON, FLIGHT, or CLASS). Finally, the START/END times are noted. The QUIT option allows the user to return to the MAIN MENU.

Operations: As the cursor is placed upon a STUDENT or INSTRUCTOR slot, a window with a list of available choices for the particular duty will appear.

Memory Aids: The labels are listed vertically to stand out to the user.

Control Mechanisms: The arrow keys will move the cursor along the available choices while the [ENTER] key carries out the selection for all except START/END times, STUDENT/INSTRUCTOR names, and OTHER. The rest of the entries must be typed before using the [ENTER] key. The [F1] function key brings up HELP instructions for this screen.

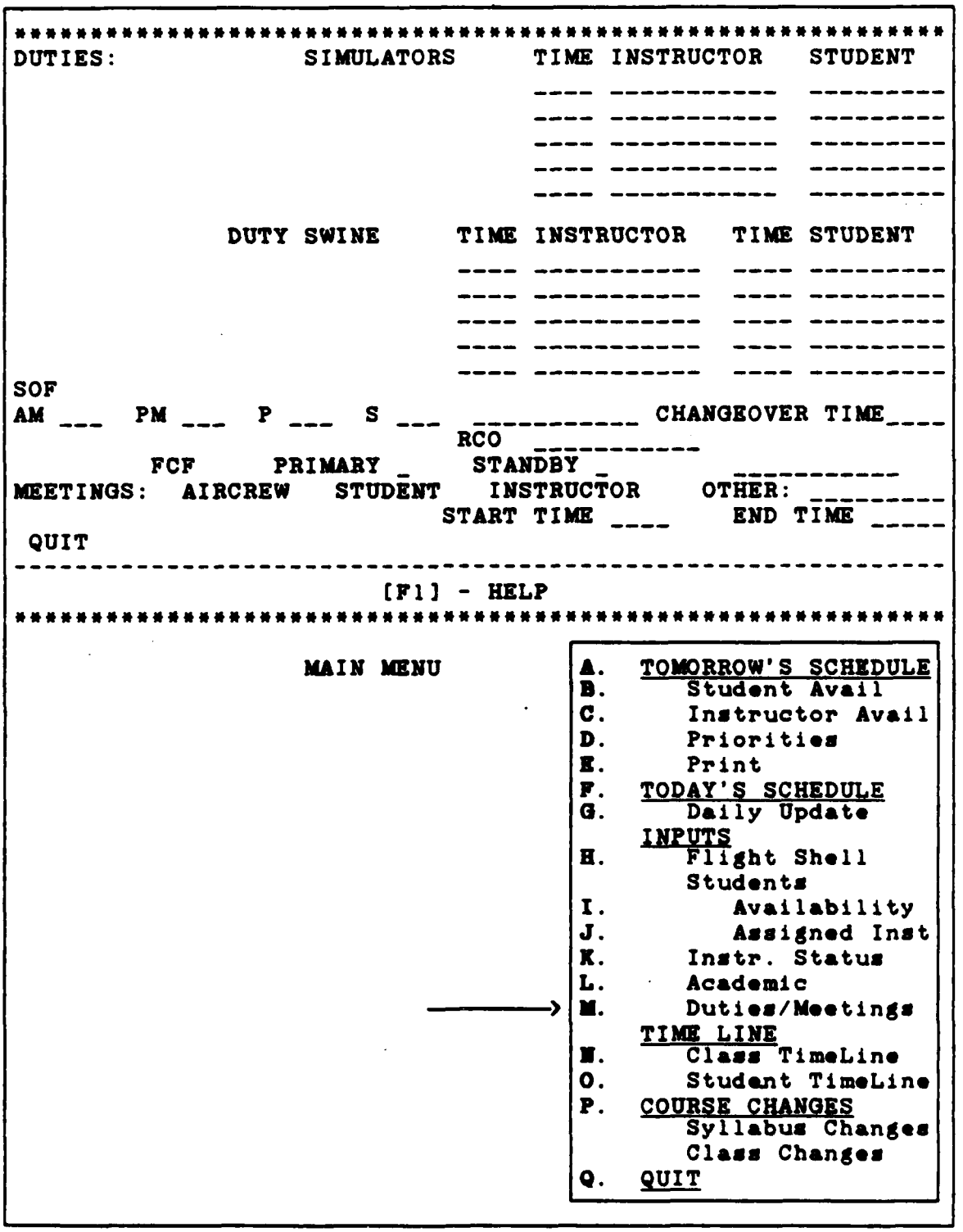


Figure F.10. Duties/Meetings

TIME LINE / CLASS

Representation: This presentation will graphically show all of the student classes' timelines. The E will denote the class that is right on schedule while the horizontal bars will depict classes that are ahead or behind their timeline.

Operations: The graph will be displayed when the user selects 'Class Timeline' from the MAIN MENU.

Memory Aids: The class names are depicted along the left side of the graph.

Control Mechanisms: The [F1] function key brings up HELP instructions for this screen.

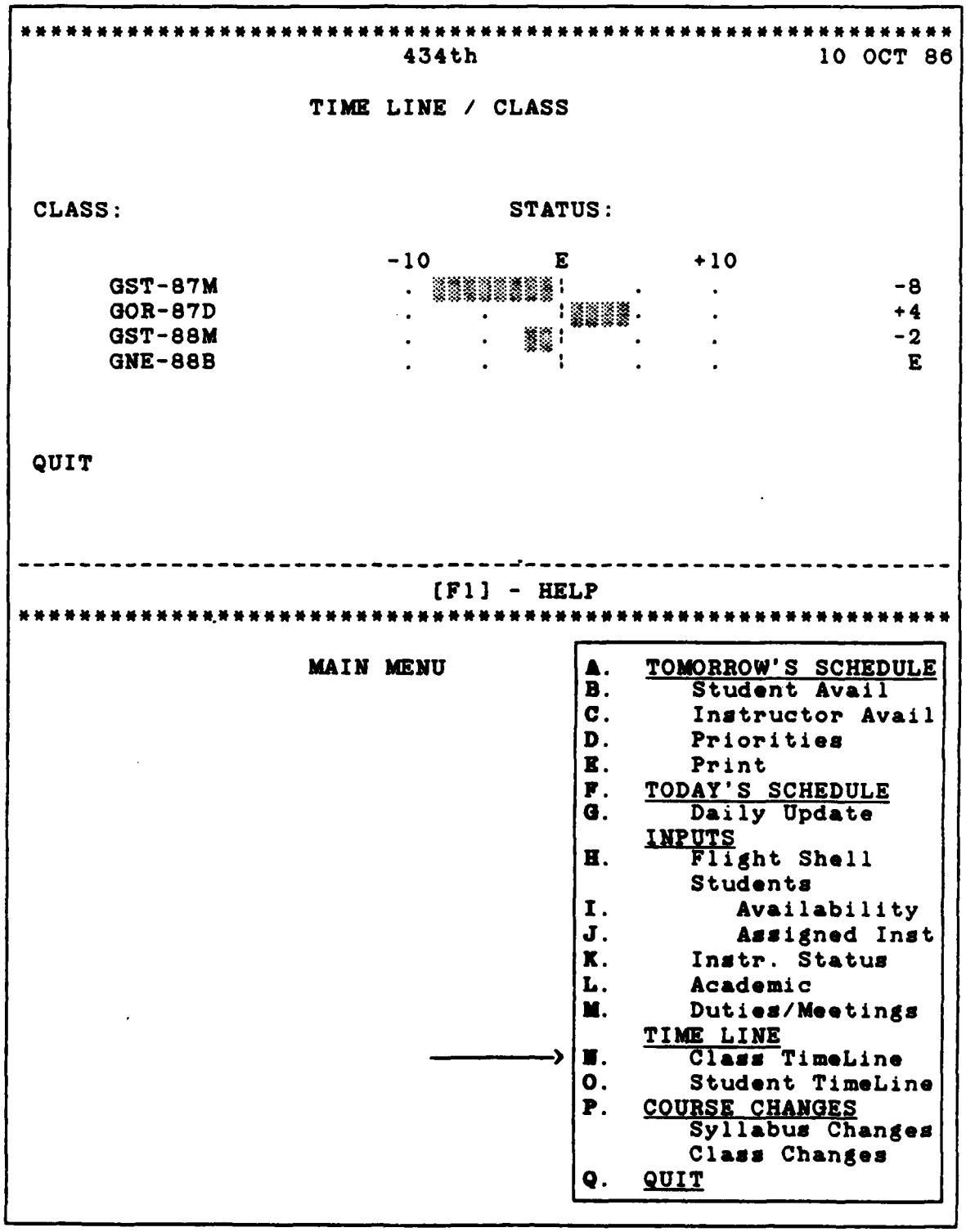


Figure F.11. Class Timeline

TIME LINE / STUDENT

Representation: This screen will overlay the main menu and select the student class the user wants to display.

Operations: Selection of a class will display the individual students and their time line progress.

Memory Aids: A one line information prompt will appear just below the Select Class.

Control Mechanisms: The [F1] key brings up HELP instructions for this screen.

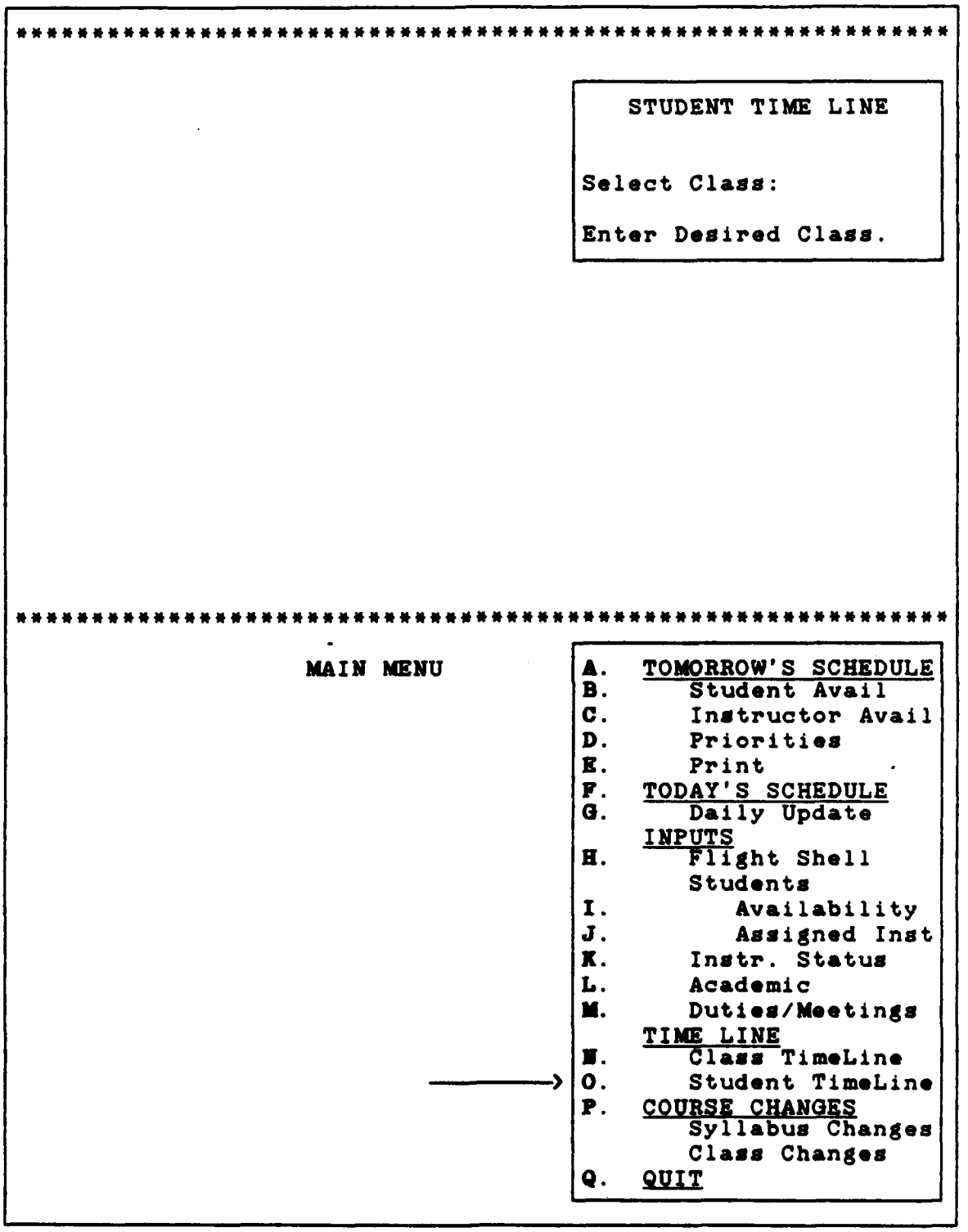


Figure F.12. Student Timeline

COURSE CHANGES

Representation: This screen will prompt the user for:

1) Syllabus Change and the Track or

2) Class Change, the Class and the Action.

Operations: Selecting either of the above options moves the user to one of the next two screens. Typing in the CLASS name will retrieve an old class or create a new one with that designation. Further selection moves the user to the appropriate screen.

Memory Aids: The only possible inputs will be listed on screen (e.g., (A,B,C)).

Control Mechanisms: One line information prompts will appear when the cursor is on a specific slot. The [F1] key brings up HELP instructions for this screen.

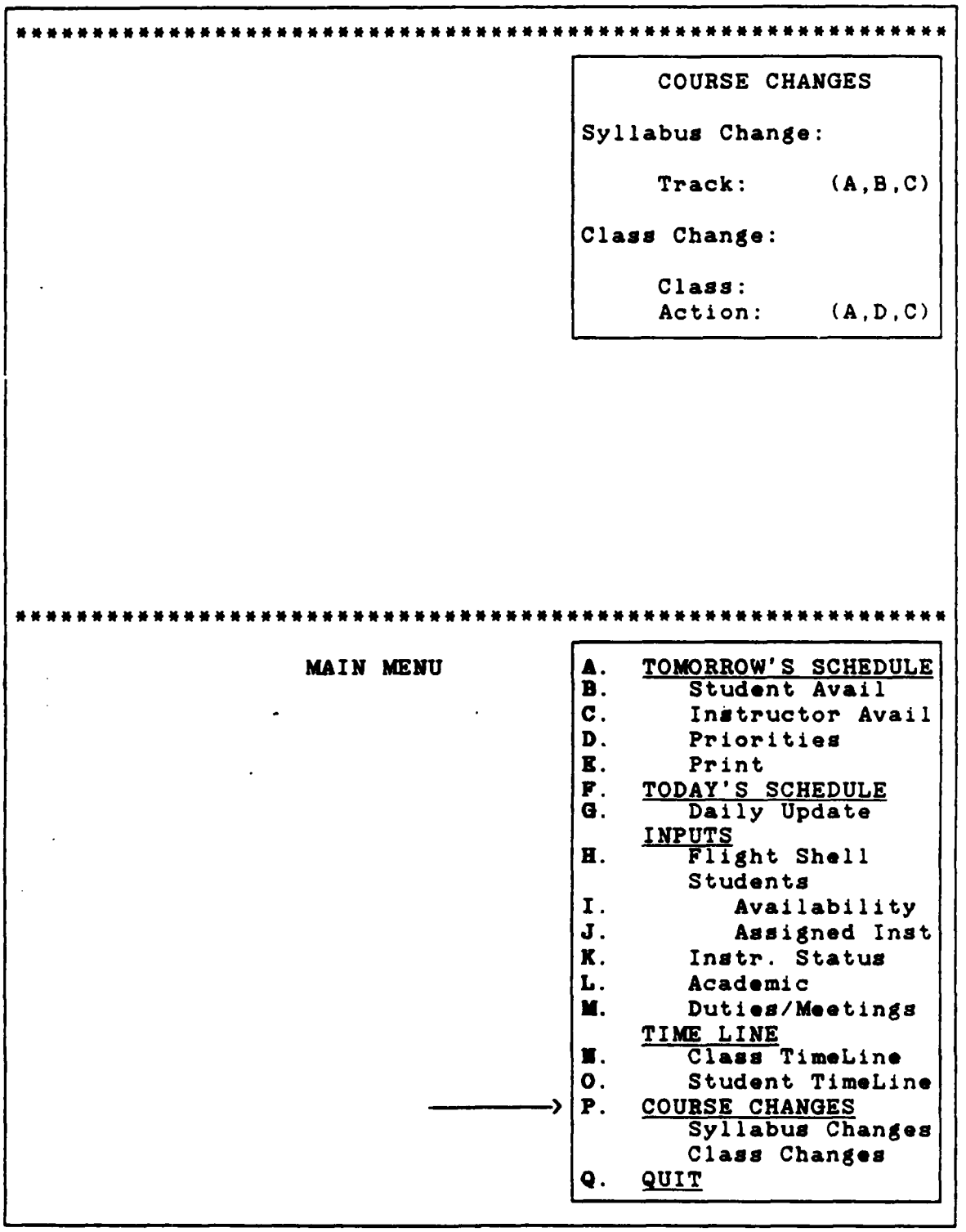


Figure F.13. Course Change Menu

COURSE CHANGES / SYLLABUS

Representation: A column of specific events lies on the left portion of the screen. Beside each event is space to enter a number of prerequisites needed to accomplish that event. In addition, a QUIT choice is included to enable the user to exit to the MAIN MENU if desired.

Operations: This screen will allow changes to the syllabus database.

Memory Aids: The syllabus track that the user has selected will be displayed on the screen. The events will be listed as they appear in the syllabus.

Control Mechanisms: The arrow keys will move the cursor along the available choices while the [ENTER] key implements the typed in selection. The [F1] key brings up HELP instructions for this screen.

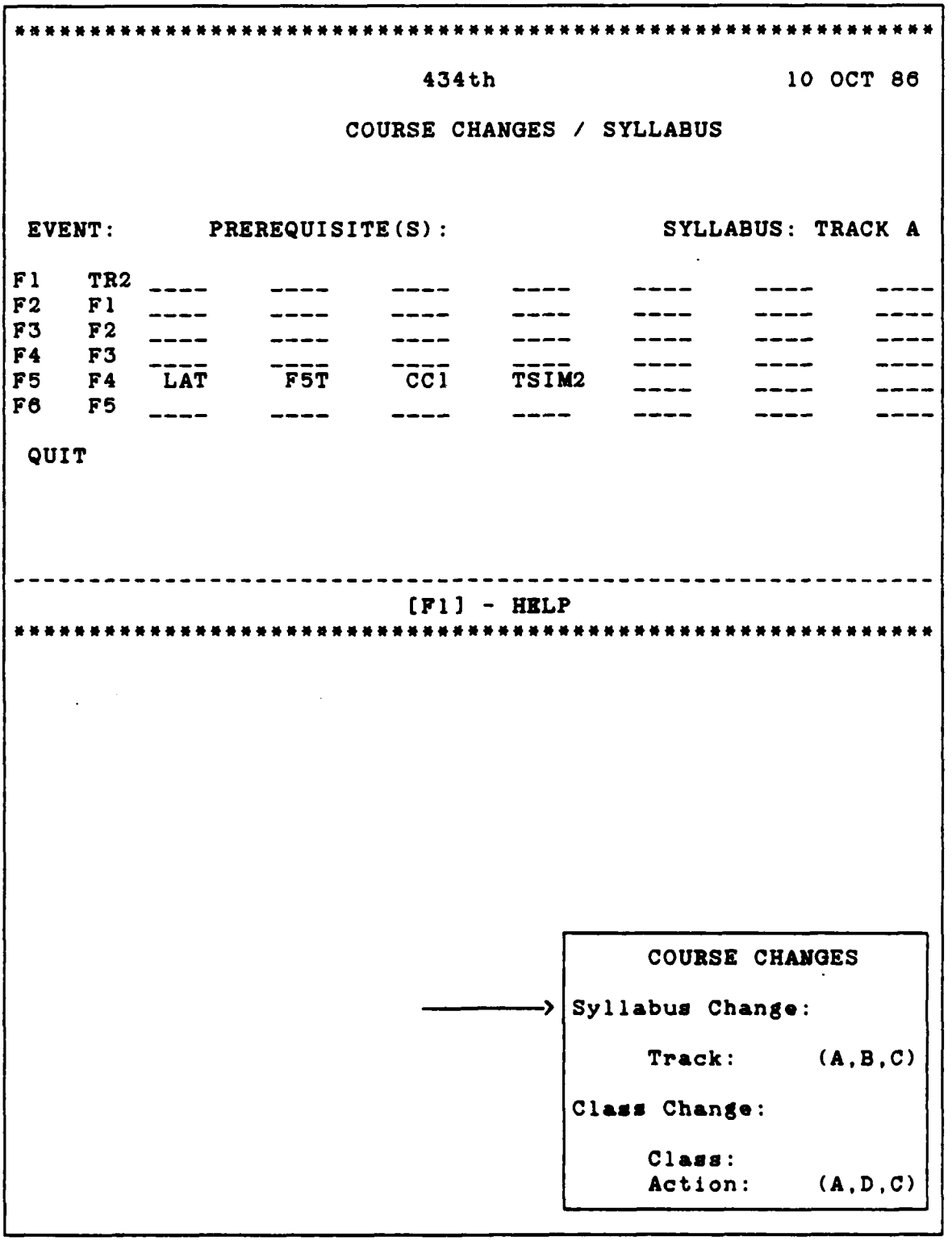


Figure F.14. Syllabus Changes

COURSE CHANGES / CLASS

Representation: A column of specific students lie on the left portion of the screen. Beside each student is his specific syllabus TRACK. Immediately adjacent is a note to reflect his red dot status, if any. In addition, a QUIT choice is included to enable the user to exit to the MAIN MENU if desired.

Operations: A pop-up window appears in the lower right of the screen to enable the user to input changes or create a new class of students.

Memory Aids: The class that the user is changing is displayed at the right side of the screen.

Control Mechanisms: The up and down arrow keys will move the cursor along the available choices while the [ENTER] key implements the typed-in selection. The [F1] key brings up HELP instructions for this screen.

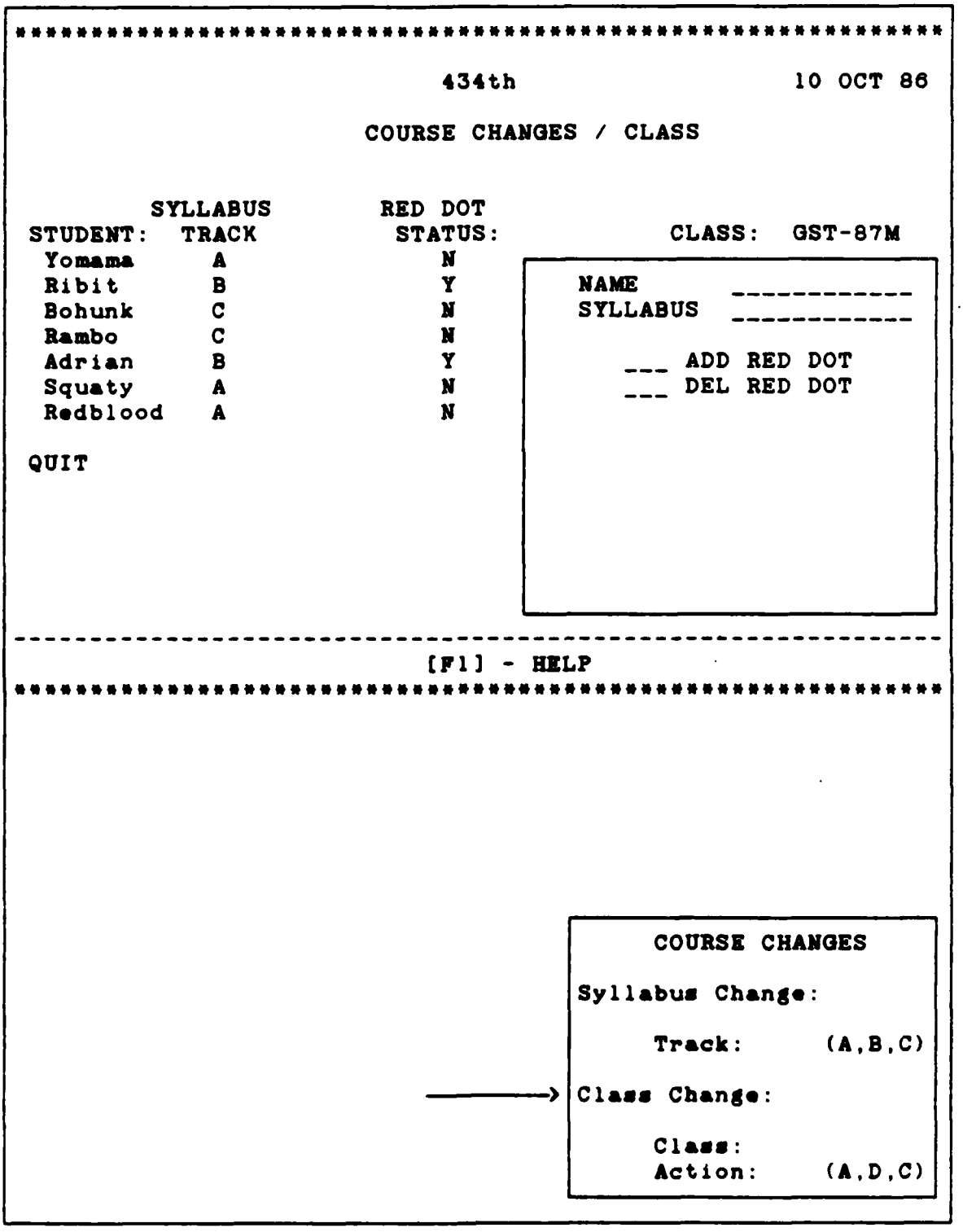


Figure F.15. Course Changes

APPENDIX G

Program User's Manual

Start-up.

This manual describes what the user will actually see on the computer screen and how he should use the screens. When the computer is turned on, the resident macros initially start the Flight Scheduler (FS) program. The DNIF/TDY change prompt (Figure G.1) appears above the latest list of personnel who are unavailable (or will be unavailable) for scheduling. This menu asks the user if the people listed in the various categories

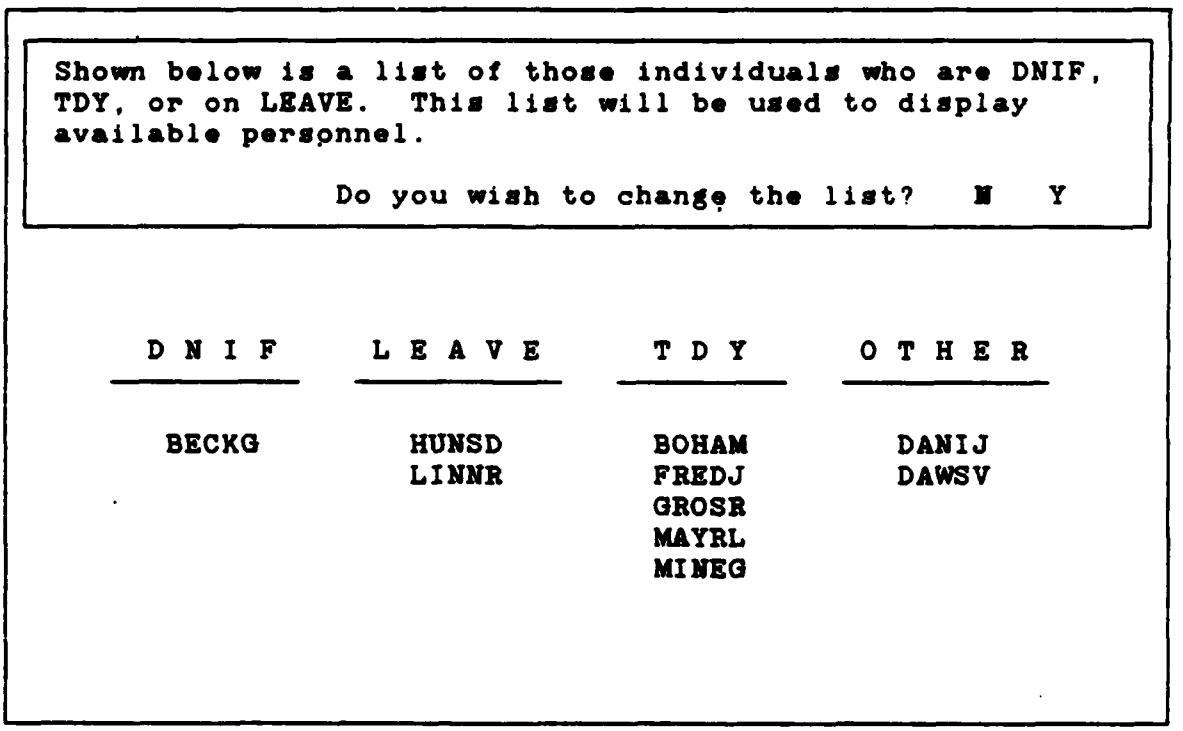


Figure G.1. DNIF/TDY Change Prompt

should be changed. The red letters, N (for no) and Y (for yes), appear at the end of the prompt with N highlighted. If the user does not wish to change the availability list, he presses the [ENTER] key, and the program proceeds. If he wants to change the listing, he presses the left or right arrow keys. The Y is now highlighted. To start the list changes, the user presses the [ENTER] key and the program continues.

If the user wanted to change the availability list, the spreadsheet in Figure G.2 appears. The user may now move freely about the screen using the arrow keys. To add an X to

WARNING:

DO NOT WRITE IN THE AVAIL COLUMN AS THIS DESTROYS THE FORMULAS NEEDED TO CREATE THE LISTING!

the list, the user positions the cursor across from the desired name and under the desired column. He then types an X and presses the [ENTER] key. To remove an X, the user simply presses the space bar followed by the [ENTER] key. Once the user makes all the changes, he simultaneously presses the [Shift] and [F9] keys, then the [S] key. This action causes the program to create an updated availability list.

> **Note:** If more than one X lies beside a crew name (e.g., DNIF and OTHER), the computer places the name under the right-most category (i.e., OTHER).

After the user updates the availability list, the program transfers the updated list to the next day's schedule. The main menu then appears at the top of the spreadsheet.

AVAILABILITY

IP NAMES	DNIF	LEAVE	TDY	OTHER	AVAIL
ALLAG					0
ANDEC					0
BECKG	X				1
BECKJ					0
BECKW					0
BOHAM			X		1
CASEK					0
DANIJ				X	1
DAWSV				X	1
DEAUC					0
DOELJ					0
DONAM					0
FRANG					0
FREDJ			X		1
FREIJ					0
FUSSJ					0
GABLG					0
GROSR				X	1
HELTC					0
HUFFD					0
HUNSD		X			1
KLINS					0
KOKAA					0
LINNR		X			1
MARVC					0
MAYRL			X		1
MCGRT					0
MILLS					0
MINEG			X		1

Press [Shift-F9] then [S] when finished.

Figure G.2. Availability Screen

Scheduling Screen Menu

The main scheduling screen menu is activated by pressing the [Shift] and [F9] keys simultaneously followed by the [Z]

key. The menu appears at the top of the spreadsheet (Figure G.3). The scheduler uses the right and left arrow keys to move the cursor from SCHEDULE to INPUTS, TIME LINE, etc. When the cursor rests on one of these options, a brief outline of what that option does appears on the line below. For example, SCHEDULING will take you either to Tomorrow's, Current, or Other schedule. When the user wants to schedule, he presses the [Enter] key or the [S] key. This takes the user to the next subset of menus (Figure G.4). The EXIT option takes the user out of the DSS.

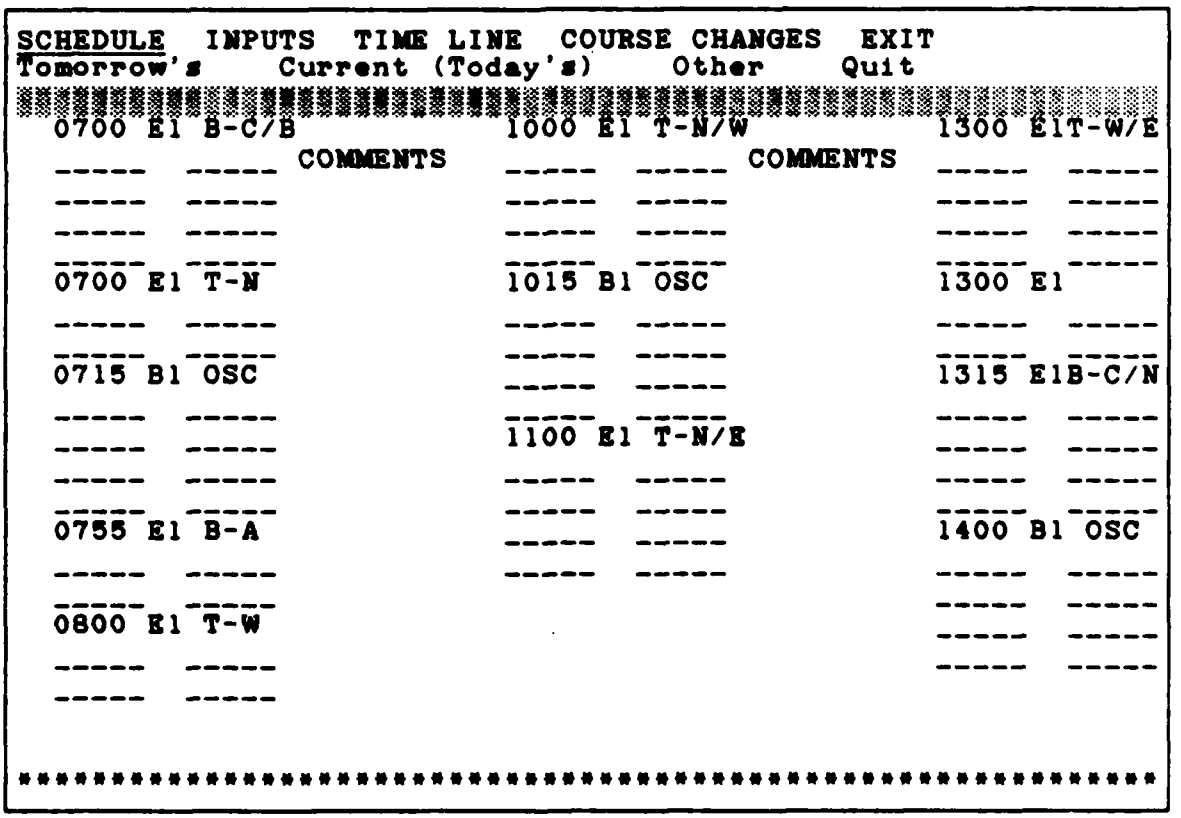


Figure G.3. Main Scheduling Screen

Figure G.4 depicts the DSS menu hierarchy. Until the system is fully developed, the only options available at this time are those with connecting arrows. Since the created schedules contain the same macro functions, they act alike. Thus, TOMORROW and the CURRENT SCHEDULEs both perform the same functions of STUDS AVAIL, INSTS AVAIL, etc. In turn, STUDS AVAIL and INSTS AVAIL both ADD, DELETE, MOVE, or QUIT.

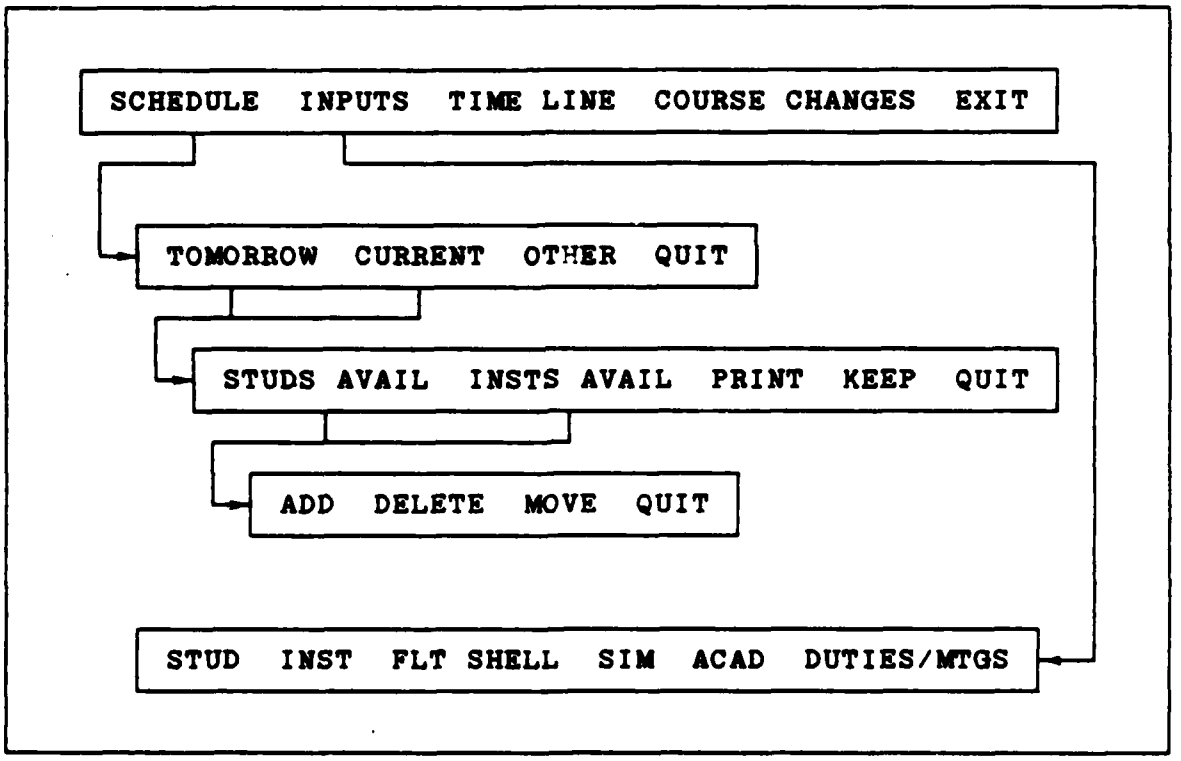


Figure G.4. Actual Main Menu Hierarchy

SCHEDULING

The scheduling menus are active only in the flight shell portion of the spreadsheet and not in the additional duty or deconfliction areas. The placement of personnel is as follows:

To start scheduling, the user has only to:

1. Press [Shift-F9], then [Z] to bring up the menu.
2. Position cursor over SCHEDULE using right or left arrow keys.
3. Press [Enter] to get to the next menu.
4. Position cursor over TOMORROW or CURRENT using right or left arrow keys.
5. Press [Enter] to get to the next menu.

- or -

1. Press [Shift-F9], then [Z] to bring up the menu.
2. Press [S] (Schedule) then [T] (Tomorrow) or [C] (Current).

In addition to the schedule shell in Figure G.3, the [PgDn] key takes the user to the additional duty screen (Figure G.5). The [PgUp] key returns him to the flight shell.

```
SIM TIME   STUD    INSTR                    ACADEMICS
                                  CLASS    INSTR   START    END
  -----    -----   -----           ---     -----   -----   -----
  -----    -----   -----           ---     -----   -----   -----
  -----    -----   -----           ---     -----   -----   -----
  -----    -----   -----           ---     -----   -----   -----
  -----    -----   -----           ---     -----   -----   -----
  -----    -----   -----           ---     -----   -----   -----
OPS SUP    STUD    TIME   INSTR             TOP THREE
  -----    -----   -----  -----              -----
  -----    -----   -----  -----              -----
  -----    -----   -----  -----              -----
  -----    -----   -----  -----              -----
  -----    -----   -----  -----              -----
  -----    -----   -----  -----              -----
  SOF   AM __  PM__                   FCF  -----
  -----  P __   S __
                                      RCO  -----
```

Figure G.5. Additional Duty Screen

Filling in the Schedule

When the user wants to fill in the schedule, he selects from a list of STUDentS AVAILable or INSTructorS AVAILable. Each category has ADD, DELETE, and MOVE options. Figure G.6 shows the screen after the user selects SCHEDULE/TOMORROW/STUDS AVAIL/ADD (or [Shift-F9], [A], [A]). A list of available students appears at the right of the screen. Using the up/down arrow keys, the user positions the cursor over the desired name and presses the [Enter] key. The program prompts him to position the cursor over the desired space and press the [Enter] key. The program fills in the flight time on the deconfliction

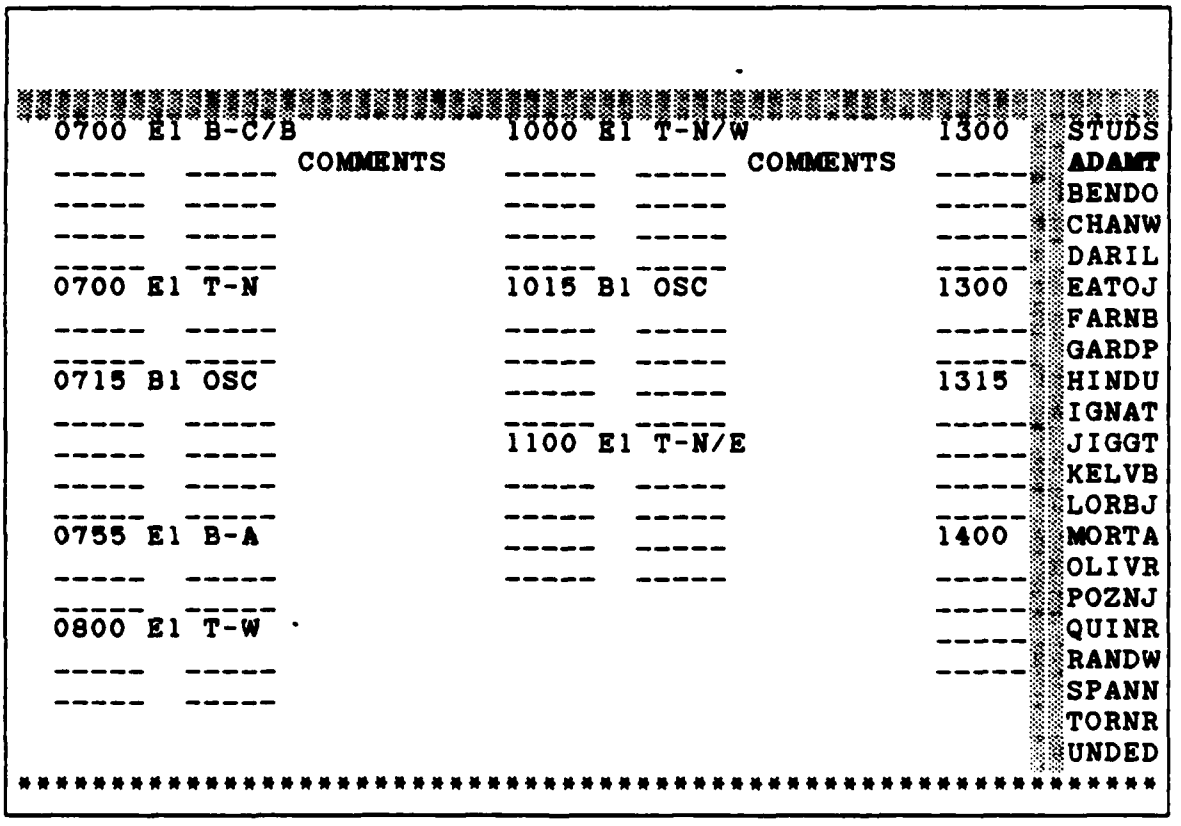


Figure G.6. Student Add Screen

spreadsheet (Figure G.7) and then places the name in the selected flight position. If there is a conflict, the program will not enter the name on the shell. The DELETE function works in reverse. The user presses [Shift-F9], [A], then [D]. He then places the cursor over the name he wants removed from the shell and presses the [Enter] key. The name is removed from the shell and the flight from the deconfliction sheet. The MOVE function activates when the user presses [Shift-F9], [A], then [M]. The program uses the delete and add functions to reposition the crew name in addition to revising the deconfliction board. The program automatically provides prompts to the user in case he forgets what to do. The QUIT option moves the user to select between STUDS and INSTS AVAIL. In summary, the sequences of keys to press to fill the next day's schedule are:

If in the next day's schedule:

To start scheduling students:
[Shift-F9] [Z] [S] [T] [S]

Add Student	[Shift-F9] [Z] [S] [T] [S] [A]
Delete Student	[Shift-F9] [Z] [S] [T] [S] [D]
Move Student	[Shift-F9] [Z] [S] [T] [S] [M]

or

Add Student	[Shift-F9] [A] [A]
Delete Student	[Shift-F9] [A] [D]
Move Student	[Shift-F9] [A] [M]

To start scheduling instructors:
[Shift-F9] [Z] [S] [T] [I]

Add Instructor	[Shift-F9] [Z] [S] [T] [I] [A]
Delete Instructor	[Shift-F9] [Z] [S] [T] [I] [D]
Move Instructor	[Shift-F9] [Z] [S] [T] [I] [M]

or

Add Instructor	[Shift-F9] [A] [A]
Delete Instructor	[Shift-F9] [A] [D]
Move Instructor	[Shift-F9] [A] [M]

Deconfliction Sheet

The deconfliction sheet allows the user to visually deconflict flight information. The user may call up the screen by pressing the [Tab] key. He may move within the sheet using the arrow keys. To return to the main shell, the user must press the [Home] key.

Figure G.7 depicts the deconfliction sheet. Takeoff times are indicated by the '|' symbol (e.g., ALLAG has an 0830 takeoff) with time included for 90 minute brief, 60 minute flight,

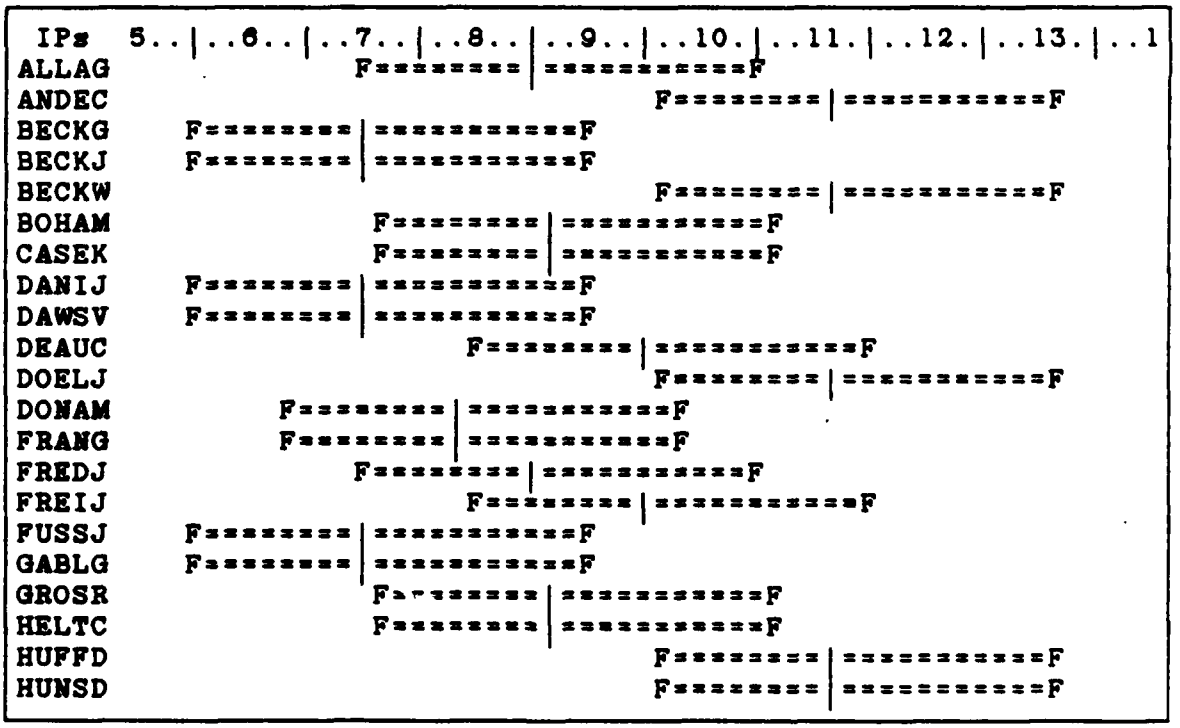


Figure G.7. Deconfliction Screen

and 60 minute debrief. The increments are accurate to the nearest ten-minute interval. If the user wants to add information other than flights, he may type the information in on the applicable line. The program searches the time block for characters and does not enter the name on the shell if a conflict exists.

Shell Inputs

As the program is constructed presently, it reads data from a file called 434EDIT.ASC. This file is the edited ASCII version of the wing lines disk. The file (Figure G.8) has all the commas removed so the program may read the data properly. The data must be in this form and named 434EDIT.ASC for the program to accept the data. With this format complete, the user may create the shell.

```
401  0  510  610  1  520  550 SA B-1 OSC
402  0  510  610  1  520  550 SA B-1 OSC
403  0  510  610  1  520  550 SA B-1 OSC
404  0  510  610  1  520  550 SA B-1 OSC
405  0  555  655  1  565  605 B  E-1 BK-C
406  0  555  655  1  565  605 B  E-1 BK-C
407  0  570  670  1  580  610 SA B-1 OSC
408  0  570  670  1  580  610 SA B-1 OSC
409  0  570  670  1  580  610 SA B-1 OSC
410  0  570  670  1  580  610 SA B-1 OSC
411  0  690  790  1  700  730 SA B-1 OSC
412  0  690  790  1  700  730 SA B-1 OSC
413  0  690  790  1  700  730 SA B-1 OSC
414  0  690  790  1  700  730 SA B-1 OSC
415  0  735  835  1  745  785 B  E-1 BK-B
416  0  735  835  1  745  785 B  E-1 BK-B
```

Figure G.8. Wing Lines Input Data File (434EDIT.ASC)

Creating a New Shell

Once the proper 434EDIT.ASC file has been created (see previous section), the user may select the flight shell input portion of the menu. He presses the [Shift-F9] and [Z] keys to invoke the main menu and selects INPUTS followed by FLT SHELL.

The proper key sequence is:

[Shift-F9] [Z] [I] [F]

The wing lines input prompt is displayed (Figure G.9) with instructions on what to do next. After the user presses the [Enter] key, the data is transformed and written to a new file.

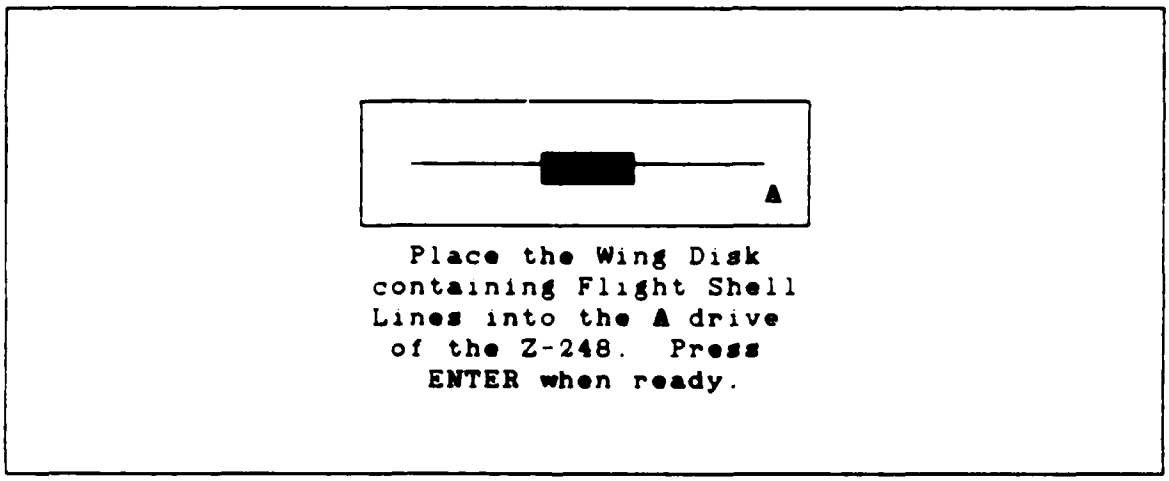


Figure G.9. Wing Lines Input Prompt

The system now prompts the user to name the file (Figure G.10). The file name is the date for which the data is to be used. In other words, if the data is for the third day of April, the file name will be 3APR.

NOTE: You must save this created shell under the date of it's use with the date format DMMM or DDMMM (e.g., 3JAN or 28FEB).

Enter the date of the new schedule: _____

Figure G.10. Shell Creation Prompt

Saving Work

It is advisable to periodically save the spreadsheet while scheduling. This prevents loss of information due to operator, program, or system errors. There are several methods by which to save the schedule:

Press [F10] [S] [A] [Enter] [R] or
Press [/] [S] [A] [Enter] [R] or
Press [Alt-F10] [Enter] [R]

If it Doesn't Work

The program is designed to handle most error situations. If for some reason the program halts or refuses to go any farther, it is not advisable for the user to take it out on the computer. Follow these steps:

1. STOP!
2. Write down what happened that led up to the malfunction (i.e., which keys were pressed)
3. Quit the program - the macros write to many areas of the spreadsheets and files so saving these errors may destroy future access to the schedule

APPENDIX H

Program Code and Documentation

The authors used the ENABLE macro language in the following code. Modification of Flight Scheduler (FS) requires a thorough understanding of ENABLE capabilities. The explanations of the code in this appendix will be adequate to understand how FS works

MACROS

```
.. $0 MCM
.. This is the AUTOEXEC startup macro
..
.. Bypass ENABLE startup menu
..
(End)
..
.. Call AVAILability spreadsheet up
..
usrAVAIL SSF~
(F2)kz1~
..
.. Invoke F (credits) menu
..
(F10)
F
..
.. Pause 2 seconds before continuing
..
(2X)(Pause)
..
.. Enter data to bypass credits menu
..
1~
..
.. END of $0 MCM
```

```
.. $1.MCM
;; This is the continuation of
;; the start-up macro, $0.MCM,
;; invoked by F.MNU.
;;
;; Invoke system date conversion
;; in AVAIL.SSF.
;;
(!F9)
y
;;
;; Edit date retrieval macro.
;;
(F9)
wo~~mrm2~~
;;
;;       Erase macro screen.
;;
(F9)
(Del)
(End)
(6X)~
(Home)
..
..       Write new date macro.
..
usr(Home)
(Down)
..
..       Copy next day's date in macro.
..
(&F5)
1(Down)
~
(Down)
 SSF(Down)
(4X)(Left)
..
..       Copy [RTN] to end of macro.
..
(&F5)
1(2X)(Down)
~
..
;; Save date retreival macro.
;;
(&F10)
a(F10)
qy
;;
;; Retreive tomorrow's
;; schedule to window #2.
```

```
;;
{F9}
wo{&F9}
2{F9}
wlg
;;
;; Invoke DNIF/TDY menu.
;;
{F2}kz1~
{^F10}
d
;;
;; END of #1.MCM


;; #2.MCM
;; This macro is modified to
;; reflect the current date
usr16DEC.SSF~


;; #3.MCM
;; This macro is a bland designed
;; to reserve Window #3 for future
;; copy/input operations
;;
;; END of #3.MCM


;; #4.MCM
;; This macro reads ASCII data from
;; file 434EDIT.ASC and transfers it
;; to ZATEMP spreadsheet via a
;; database file 434.DBF.
;;
;; Save file and go to main menu.
;;
{F9}
s{Home}
~
ry
;;
;; Close window and select ZNEWSCH.SSF.
;;
{F9}
wcymfa{End}
;;
;; Copy ZNEWSCH.SSF to ZATEMP.SSF.
;;
```

```
c{2X}~
ZATEMP.SSF~
y{Esc}
;;
;; Copy ASCII file 434EDIT.ASC
;; to database file 434.DBF.
;;
udic434.$BF~
fa434EDIT.ASC{2X}{Down}
u434.CTF~
{!F9}
c~
{Esc}
;;
;; Create field names for 434.DBF.
;; d434.DBF~
{PgDn}
~
{2X}{Down}
LINE,ZERO,ONE,TAKEOFFTIM,LANDTIME,AREATIME1,
AREATIME2,CONFIG,AREA,TYPERIDE~
;;
;; Open window containing ZATEMP.SSF.
;;
{&Home}
usrZATEMP.SSF~
;;
;; Copy data from 434.DBF starting
;; at ZATEMP.SSF cell $f60.
;;
/d{Up}
~
{F7}
{59X}{Down}
{F7}
{&F5}
~
osf60~
;;
;; Go to ZATEMP.SSF cell b1 and
;; copy formula thru cell b61.
;;
/mod1{F10}
q{Home}
{Right}
/wc~
b2..b61~
;;
;; Shift ZATEMP.SSF data in
;; sn- and so-columns one
;; column to the right.
;;
```

```
/wmsn60..so120~
so60{2X}~
;;
;; Center-justify so-column data.
;;
/wracso60..so120~
;;
;; Copy configuration substring
;; conversion formula to cell
;; sn59 and invoke spreadsheet
;; f-macro to convert load
;; configurations and create
;; scheduling flight shell.
;;
{F2}sn59~
@substr(sm59,1,1)&('1')~
/wcsn59{2X}~
sn60..sn120~
{!F9}
f
;;
;; Copy additional duty shell
;; below flight shell.
;;
/wced1..eu20~
a21{2X}~
;;
;; Invoke G.MNU to query user
;; to name the new ZATEMP.SSF
;; scheduling shell.
;;
{^F10}
g
;;
;; END of $4.MCM



;; $5.MCM
;; This is the continuation of
;; the start-up macro, $0.MCM &
;; $1.MCM, invoked by D.MNU.
;;
;; Transfer personnel availability
;; list to schedule in window #2.
;;
{F2}la1~
{F9}
w2g/ccw11a1..mz59~
la1~
;;
;; Copy Available instructors
```

```
;; to cell t2 in window #2.
;;
(F8)
mb4..mb59~
t2(2X)~
(Home)
;;
;; Activate scheduling main menu.
;;
(!F9)
z
;;
;; END of #5.MCM
```

THE FOLLOWING IS FROM AVAIL.SSF

LETTER OF X'S AND QUALIFICATIONS

IP NAMES	RANK	CP	FLT	WX CAT	EXP	UIPIP	UIPRD	FLTLD	RDIP	RRIP
ALLAG •	MAJ	IP	B	A	E			X	X	X
ANDEC	MAJ	IW	C		N					X
BECKG •	LTC	IP	A	A	E	X		X	X	X
BECKJ •	CPT	IP	B	B	E			X	X	X
BECKW •	CPT	IP	A	A	E			X	X	X
BOHAM •	CPT	IP	C	B	E			X	X	X
CASEK •	MAJ	IP	A	A	E	X	X	X	X	X
DANIJ •	CPT	IP	C	A	E			X	X	X
DAWSV	MAJ	IP	B	C	N					X
DEAUC •	CPT	IP	A	A	E			X	X	X
DOELJ •	CPT	IP	C	A	E			X	X	X
DONAM	CPT	IP	A	C	N					
FRANG •	LTC	IP	D	A	E			X	X	X
FREDJ •	CPT	IP	C	B	E			X	X	X
FREIJ	COL	IP	D	B	E			T		X
FUSSJ •	MAJ	IP	B	A	E			X	X	X
GABLG •	CPT	IP	C	A	E			X	X	X

AVAILABILITY				AVAIL COLUMN FORMULA
DNIF	LEAVE	TDY	OTHER	AVAIL	
				0	@COUNT(MW4..MZ4)
				0	@COUNT(MW5..MZ5)
				0	@COUNT(MW6..MZ6)
X				1	@COUNT(MW7..MZ7)
				0	@COUNT(MW8..MZ8)
				0	@COUNT(MW9..MZ9)
				0	@COUNT(MW10..MZ10)
				0	@COUNT(MW11..MZ11)
		X		1	@COUNT(MW12..MZ12)
				0	@COUNT(MW13..MZ13)
				0	@COUNT(MW14..MZ14)
				0	@COUNT(MW15..MZ15)
				0	@COUNT(MW16..MZ16)
				0	@COUNT(MW17..MZ17)
		X		1	@COUNT(MW18..MZ18)
				0	@COUNT(MW19..MZ19)
				0	@COUNT(MW20..MZ20)

```
{\S}
Makes columns of DNIF, TDY, & LEAVE people
************************************************************


----------------------------------------Determine Red Dot IPs

/xcnf4~

--------------------------------Erase old DNIF/TDY/LEAVE list

/wtc(F2)la9~/wre.(4x)(Right)(PgDn)~

----------------------------Make range name DNIF in cell la9

/wrncDNIF~~(Right)

----------------------------Make range name TDY in cell lc9

/wrncTDY~~(Right)

---------------------------Make range name OTHER in cell ld9

/wrncOTHER~~

--------------------------------Go to cell na3 (Availability)

(F2)na3~
```

```
-----------------------------------------------Go down a cell
(Down)
-----------------------------------------Call that cell AVAIL
/wrncAVAIL~~
--------------------------If end of list, call D.MNU and end
/xi(AVAIL=9)~(F2)kz1~(^F10)d/xq
-----------------------------------If AVAIL=0, go to cell nj11
/xi(AVAIL<1)~/xgnj11~
-----------------------Go to marked cell in row-call it TEST
(End)(Left)/wrncTEST~~
--------------------------Go to column heading-call it CAUSE
(3x)(PgUp)(2x)(Down)/wrncCAUSE~~
---------------------------If CAUSE is DNIF, go to cell nj21
/xi(CAUSE='DNIF')~/xgnj21~
--------------------------If CAUSE is LEAVE, go to cell nj27
/xi(CAUSE='LEAVE')~/xgnj27~
---------------------------If CAUSE is TDYF, go to cell nj33
/xi(CAUSE='TDY')~/xgnj33~
--------------------------If CAUSE is OTHER, go to cell nj39
/xi(CAUSE='OTHER')~/xgnj39~
-------------------------------------Go to cell called TEST
(F2)TEST~
-----------------------------------------Find name-call it D
(22x)(Left)/wrncD~~
-------------------------Go to cell below DNIF-call it DNIF
(F2)DNIF~(Down)/wrncDNIF~~
```

```
------------------------------Copy name called D to DNIF cell
{F8}D~DNIF~
-------------------------------------Go to cell called TEST
{F2}TEST~
-------------------------------Go back to availability column
{End}{Right}/xgnjll~
-------------------------------------Go to cell called TEST
{F2}TEST~
-----------------------------------------Find name-call it L
{23x}{Left}/wrncL~~
-------------------------Gotc cell below LEAVE-call it LEAVE
{F2}LEAVE~{Down}/wrncLEAVE~~
-----------------------------Copy name called L to LEAVE cell
{F8}L~LEAVE~
-------------------------------------Go to cell called TEST
{F2}TEST~
-------------------------------Go back to availability column
{End}{Right}/xgnjll~
-------------------------------------Go to cell called TEST
{F2}TEST~
-----------------------------------------Find name-call it T
{24x}{Left}/wrncT~~
----------------------------Go to cell below TDY-call it TDY
{F2}TDY~{Down}/wrncTDY~~
-----------------------------Copy name called T to TDY cell
{F8}T~TDY~
```

```
----------------------------------------Go to cell called TEST
{F2}TEST~
------------------------------Go back to availability column
{End}{Right}/xgnj11~
----------------------------------------Go to cell called TEST
{F2}TEST~
-----------------------------------------Find name-call it 0
{25x}{Left}/wrnc0~~
--------------------------Goto cell below OTHER-call it OTHER
{F2}OTHER~{Down}/wrncOTHER~~
----------------------------Copy name called 0 to OTHER cell
{F8}0~OTHER~
----------------------------------------Go to cell called TEST
{F2}TEST~
------------------------------Go back to availability column
{Right}/xgnj11~


{\Y}
Converts date to '1JAN' format.
================================================================


----------------------------------------Write 0 in cell no49
{F2}no49~0~
--------------------------If today isn't Sat, go to cell no8
/xi(@MOD(@TODAY,7)<>6)~/xgno8~
----------------------------Go to cell no43 - write formula
{F2}no43~@TODAY+2
```

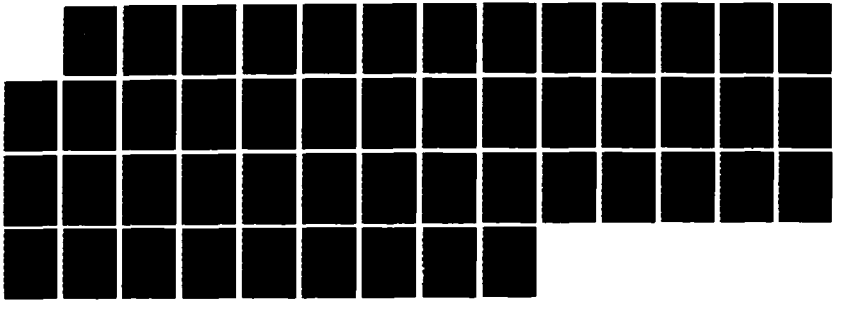
```
------------------------------Go down a cell - write formula
{Down}@TODAY+3~/xgno17~
-------------------------If today isn't Fri, go to cell no11
/xi(@MOD(@TODAY,7)<>5)~/xgno11~
------------------------------Go to cell no43 - write formula
{F2}no43~@TODAY
------------------------------Go down a cell - write formula
{Down}@TODAY+3~/xgno17~
-------------------------If today isn't Sun, go to cell no14
/xi(@MOD(@TODAY,7)<>0)~/xgno14~
------------------------------Go to cell no43 - write formula
{F2}no43~@TODAY+1
----------------------------Down a cell-write formula-Go no17
{Down}@TODAY+2~/xgno17~
------------------------------Go to cell no43 - write formula
{F2}no43~@TODAY
-------------------------Down a cell-write formula-Goto no17
{Down}@TODAY+1~/xgno17~
-----------------------------------------Write 1 in cell no49
{F2}no49~1~
----------------------------If 1 then write JAN in cell no45
/xi(@MONTH(no43)=1)~{F2}no45~JAN~
----------------------------If 2 then write FEB in cell no45
/xi(@MONTH(no43)=2)~{F2}no45~FEB~
----------------------------If 3 then write MAR in cell no45
/xi(@MONTH(no43)=3)~{F2}no45~MAR~
```

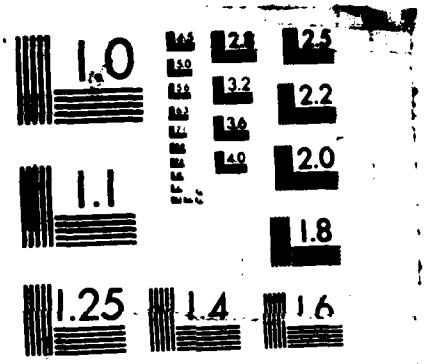
```
----------------------------If 4 then write APR in cell no45
/xi(@MONTH(no43)=4)~(F2)no45~APR~
----------------------------If 5 then write MAY in cell no45
/xi(@MONTH(no43)=5)~(F2)no45~MAY~
----------------------------If 6 then write JUN in cell no45
/xi(@MONTH(no43)=6)~(F2)no45~JUN~
----------------------------If 7 then write JUL in cell no45
/xi(@MONTH(no43)=7)~(F2)no45~JUL~
----------------------------If 8 then write AUG in cell no45
/xi(@MONTH(no43)=8)~(F2)no45~AUG~
----------------------------If 9 then write SEP in cell no45
/xi(@MONTH(no43)=9)~(F2)no45~SEP~
---------- ----------------If 10 then write OCT in cell no45
/xi(@MONTH(no43)=11)~(F2)no45~OCT~
----------------------------If 11 then write NOV in cell no45
/xi(@MONTH(no43)=11)~(F2)no45~NOV~
----------------------------If 12 then write DEC in cell no45
 xi(@MONTH(no43)=12)~(F2)no45~DEC~
-------------------------------If cell no49=1 go to cell no3~
/xi(no49=1)~/xgno37~
-------------------------------------------------Go to cell no46
(F2)no46~
----------------------------If two digits in day, go to no34
/xi(@LEN(@STRING(@DAY(no43)))=2)~/xgno34~
-----------------------------------Write day + month  (5MAR)
@SUBSTR(@STRING(@DAY(no43)),1,1)&no45~
```

```
-----------------------------------------------Go to cell no35
 \gno35~
-----------------------------------Write day + month  (10MAR)
@SUBSTR(@STRING(@DAY(no43)),1,2)&no45~
---------------------------Go to cell no44-copy to cell no43
{F2}no44~/wc~{Up}~
-----------------------------------------------Go to cell no16
 \gno16~
-----------------------------------------------Go to cell no47
{F2}no47~
----------------------------If two digits in day, go to no41
 \i{@LEN(@STRING(@DAY(no43)))=2}~/xgno41~
-----------------------------------Write day + month  (5MAR)
@SUBSTR(@STRING(@DAY(no43)),1,1)&no45~
-----------------------------------------------Go to cell no42
 \gno42~
-----------------------------------Write day + month  (10MAR)
@SUBSTR(@STRING(@DAY(no43)),1,2)&no45~
---------------------------------Go up one cell - quit macro
{Up}/xq
---------------------------------------**Today's System Date
@TODAY
-----------------------------------**Next Day's System Date
@TODAY+1
-------------------------------------------------**System Month
DEC
```

AD-A180 265 DESIGN EVOLUTION OF A FIGHTER TRAINING SCHEDULING DECISION SUPPORT SYSTEM(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI J W GRECHANIK ET AL MAR 87 3/3

UNCLASSIFIED F/G 5/8 NL





```
----------------------------------------**Today's Schedule Date

15DEC

----------------------------------**Next Day's Schedule Date

16DEC

--------------------------------------------**Carriage Return

~

-------------------------------------------------**Test Flag

1


{\T}
Process DNIF status change
============================================================


----------------Go to cell ma1, then cell mb4 and set titles

{F2}ma1~{F2}mb4~/wtb

--------------------------Go to cell ne4, then 8 spaces left

{F2}ne4~{8x}{Left}

-------------------------------------------------Quit macro

/xq


ZATEMP.SSF


{\A}
This macro brings up
Scheduling menu (A,D,M,Q)
=========================

/xi(ra49="S")~/xmra65~/xq

/xmrg65~/xq
```

```
(MESS1)

{\B} IPs
Searchs for takeoff time and fills in deconfliction board
and select an IP Name and put in on the schedule
==============================================

----------------------------Instruct User to position cursor

{F8}MESS1~a20~~{?}/wrea20~~

-------------------------------------------------Message One

USE ARROW KEYS OR MOUSE TO POSITION

CURSOR WHERE IP IS DESIRED (THEN ENTER)

------------------------Names Cell START and Moves Up a Cell

/wrncSTART~~{up}

-----------------------------------------Names cell TOTIME

/wrncTOTIME~~

----If length of TOTIME is 5 then move the pointer up & loop

/xi(@LEN(TOTIME)=5)~{Up}/xgre10

-----------Check first alphanumeric, if letter move two left

/xi(@ALPHA(@SUBSTR(TOTIME,1,1)))=1~{2x}{Left}/wrncTOTIME~~

----------------------Position the screen to open the window

{F2}c1~{16x}{Right}

-----------Open Window and Display IP Names, Pause for input

/mtsv{Home}{F6}/mtsu{Down}{?}/wrncNAME~~

------------------------------Search a list of takeoff times

/xcka1~

--------------------Indicate the range that would be filled

/wrncBEGIN~.{11x}{Right}~
```

```
-------Check to see if there is already some event scheduled
/xi(@COUNT(BEGIN)>0)~/xgre22
-----------If no conflict exists, copy the name to the shell
/wcNAME~START~
----------Go to the deconfliction sheet and fill in a flight
{F2}BEGIN~F={Right}=={Right}=={Right}
=={Right}=|{Right}=={Right}
=={Right}=={Right}=={Right}=={Right}=F~
--------------Close the small window and redisplay the shell
/mtsc{Home}{F2}START~
-----------------------------------------------------The end
/xq


{\C} Students
Searchs for takeoff time and fills in deconfliction schedule
and select a student name and put in on the schedule
==========================================================


------------------------Names cell start and moves up a cell
/wrncSTART~~{up}
-------------------------------------------Names cell totime
/wrncTOTIME~~
--------If length of totime = 5 move the pointer up and loop
/xi(@LEN(TOTIME)=5)~{Up}/xgrn4
-----------Check first alphanumeric, if letter move two left
/xi(@ALPHA(@SUBSTR(TOTIME,1,1)))=1~{2x}{Left}/wrncTOTIME~~
----------------------Position the screen to open the window
{F2}bg1~{17x}{Right}
```

```
--------Open window , display IP Names, pause for user input
/mtsv{Home}{F6}/mtsu{Down}{?}/wrncNAME~~
------------------Search totimes & find amount to move right
/xcka1~
----------indicate range that is filled if flier is assigned
/wrncBEGIN~.{11x}{Right}~
--------------------------------Check if there is a conflict
/xi(@COUNT(BEGIN)>0)~/xgrn17
--------------No conflict exists, copy the name to the shell
/wcNAME~START~
--------------Go to deconfliction sheet and fill in a flight
{F2}BEGIN~F={Right}=={Right}=={Right}=={Right}
=|{Right}=={Right}=={Right}=={Right}
=={Right}=={Right}=F~
--------------Close the small window and redisplay the shell
/mtsc{Home}{F2}START~
-----------------------------------------------------The end
/xq


{\D}
Searchs for takeoff time, deletes name from main schedule
and deletes time slot from the deconfliction sheet.
==========================================


---------------------------Names cell NA and moves up a cell
/wrncNA~~{up}
-------------------------------------------Names cell TOTIME
/wrncTOTIME~~
```

```
-------------If totime = 5 then move the pointer up and loop
/xi(@len(TOTIME)=5)~{up}/xgrd28
-----------Check first alphanumeric, if letter move two left
/xi(@ALPHA(@SUBSTR(TOTIME,1,1)))=1~{2x}{Left}/wrncTOTIME~
-----------Searches for the name in NA and send cursor to it
/xckb1~
------------------Gives the found name the range name 'NAME'
/wrncNAME~~
-----------------------Searches for a totime and moves right
/xcka1~
-----------------------------------Erase the range indicated
/wre.{11x}{Right}~
---------------------Go to starting place and put in '_____'
{Home}{F2}NA~_____~
-----------------------------------------------------The end
/xq


{\E}
Searchs for totime, deletes name from main schedule and time
slot from the deconfliction board
==========================================================


--------------------------Names cell NA and moves up a cell
/wrncNA~~{up}
-----------------------------------------Names cell totime
/wrncTOTIME~~
--------------If TOTIME length is 5 move pointer up and loop
/xi(@len(TOTIME)=5)~{up}/xgre40
```

```
---------Check first alphanumeric, if a letter move two left
/xi(@ALPHA(@SUBSTR(TOTIME,1,1)))=1~{2x}{Left}/wrncTOTIME~~
--------------Searches for name in NA and send cursor to it
/xckc1~
-----------------------------------------Name the name NAME
/wrncNAME~~
----------------------Searches for a totime and moves right
/xcka1~
-----------------------------------Erase the range indicated
/wre.{11x}{Right}~
--------------------Go to starting place and put in '_____'
{Home}{F2}NA~_____~
-----------------------------------------------------The end
/xq


{\F}
This macro forms the basic
shell using 2hrs 30min turn times.
==================================


----------------------Go to so 121 and write 999, go to so60
{F2}so121~999~{F2}so60~
-----------Name ZA and if cell below equals zero, go to ra13
/wrncZA~.{Down}~/xi(@COUNT(ZA)=0)~/xgra13~
---------------------------------------Center cell contents
{F4}{Down}/xgra11~
----------------------------Copy information starting in c2
/wcsn60..so120~c2~~{Home}{Right}{Down}
```

```
------------------------------Move the first line into row 1
/wm.{2x}{Right}~{Up}~{Up}
----------------------------------Name the first line GOTIME
/wrncGOTIME~~
-----------------------------Name the first takeoff time FTO
{Up}/wrncFTO~~{Down}
-----------------------------------Write in the blank spaces
_____{Right} {Right}_____{2x}{Left}{Down}
---------------------------------------Name the next cell XA
/wrncXA~~
---------------------------If XA equals FTO then go to ra17
/xi(XA=FTO)~/xgra17~
----------------------If XA is less than FTO then go to ra27
/xi(XA<FTO)~/xgra27~
--------------If time between flights > 2.5 hours go to ra25
/xi((GOTIME+231)>XA)~/xgra25~
---------------Move the column of takeoff times to the right
/wm.{3X}{PgDn}{2x}{Right}~
------------------------------Move six to the right and down
{End}{Up}{End}{Up}{6x}{Right}{Down}~
------------------------------Move six to the right and down
{End}{Up}{End}{Up}{6x}{Right}{Down}/xgra14~
--------------------If XA is greater than FTO then move down
/xi(XA>FTO)~{Down}
----------------------Move the column of takeoff times down
/wm.{3x}{PgDn}{2x}{Right}~{Down}~/xgra16~
```

```
----------------------Erase any extra takeoff times and quit
/wre.{3x}{PgDn}{2x}{Right}~{Home}/xq


{\G}
Moves an instructor name from one slot to another
=================================================

---------Prompts user to position cursor over the IP to move
{F8}MESS2~a20~~{?}/wrea20~~
---------------------------names cell na and moves up a cell
/wrncNA~~{up}
----------names cell totime
/wrncTOTIME~~
--if length of totime is 5 then move the pointer up and loop
/xi(@len(TOTIME)=5)~{up}/xgsfil
-----------check first alphanumeric, if letter move two left
/xi(@ALPHA(@SUBSTR(TOTIME,1,1)))=1~{2x}{Left}/wrncTOTIME~~
----------------------position the screen to open the window
{F2}c1~{16x}{Right}
--------------------open the window and display the IP Names
/mtsv{Home}{F6}/mtsu
---------------searches for name in 'NA' in instructor list
/xckb1~
-------remember it by calling it 'name'
/wrncNAME~~
-------------------------searches for totime and moves right
/xcka1~
```

```
-----------------------------------erase the range indicated
/wre.{11x}{Right}~
-------Prompts user to position cursor where the IP is to go
{F8}MESS3~a20~~{F6}{?}{F9}{Del}ba20~~
-------------------------names cell start and moves up a cell
/wrncSTART~~{up}
-------------------------------------------names cell totime
/wrncTOTIME~~
-----------if length of totime is 5 move pointer up and loop
/xi(@LEN(TOTIME)=5)~{Up}/xgsf22
-----------check first alphanumeric, if letter move two left
/xi(@ALPHA(@SUBSTR(TOTIME,1,1)))=1~{2x}{Left}/wrncTOTIME~~
------------go to NAME and move the appropriate amount right
{F6}{F2}NAME~/xcka1~
--------------------------indicate range that would be filled
/wrncBEGIN~.{11x}{Right}~
------------------------------check if there's some conflict
/xi(@COUNT(BEGIN)>0)~/xgsf33
------------no conflict exists, move the ips name to 'START'
{F2}START~/wcNA~~{F2}NA~_____~
----------go to the deconfliction sheet and fill in a flight
{F2}BEGIN~F={Right}=={Right}=={Right}
=={Right}=|{Right}=={Right}
=={Right}=={Right}=={Right}
=={Right}=F~
```

```
--------------------close the little window and go to {Home}

{F6}{F9}wx{Home}

-----------------------------------------------------the end

/xq


  *SUB*
{\X} This macro returns to instructor availability menu.
=======================================================

{F9}wx{Home}{F2}j9~/xmra50~



{\Z}
This forms the user input menu.
===============================

/xmra50~/xq


SCHEDULE     INPUTS     TIME LINE     COURSE CHANGES     EXIT
/xmra55~     /xmrj55~   /xmrt55~      /xmra50~          /xmrh50~


TOMORROW     CURRENT                  OTHER     QUIT
/xmra60~     /xi(ra49=0)~/xmra60      /xmra55~  /xmra50~
             /xi(ra49=1)~{F9}w2g


STUDS AVAIL                      INSTRUCTORS AVAIL
/wrncPATH~~{F2}ra49~S~           /wrncPATH~~{F2}ra49~I~
{Home}{F2}PATH~/xmra65~          {Home}{F2}PATH~/xmrg65~


PRINT       KEEP                   QUIT
/xmra60~    {F9}se~r/xmra60~       /xmra55~


Student
ADD         DELETE      MOVE       QUIT
{!F9}c/xq   {!F9}e/xq   /xmra65~   /xmra60~


NO          YES
/xmra50~    {&F9}q/xq


STUDENT     INSTRUCTOR  FLIGHT SHELL  SIMULATOR
```

```
/xmrj55~    {&F9}6/xq    {^F10}e/xq     /xmrj55~

ACADEMIC    DUTIES/MEETINGS
/xmrj55~    /xmrj55~


Instructor
ADD          DELETE       MOVE       QUIT
{!F9}b/xq    {!F9}d/xq    /xmra65~   /xmra60~
```

COMPARISON MACROS

==========

KA -- SEARCHES FOR A TAKEOFF TIME AND THE APPROPIATE AMOUNT TO MOVE RIGHT

KB -- SEARCHES FOR AN INSTRUCTOR NAME AND SENDS THE CURSOR TO IT IN THE DECONFLICTION SCHEDULE

KC -- SEARCHES FOR A STUDENT NAME AND SENDS THE CURSOR TO IT IN THE DECONFLICTION SCHEDULE

KA

SEARCHES FOR A TAKEOFF TIME AND THE APPROPIATE AMOUNT TO MOVE RIGHT

==========

```
/xiTOTIME=600~{2x}{Right}/xr
/xiTOTIME=605~{2x}{Right}/xr
/xiTOTIME=610~{2x}{Right}/xr
/xiTOTIME=615~{2x}{Right}/xr
/xiTOTIME=620~{2x}{Right}/xr
/xiTOTIME=625~{2x}{Right}/xr
/xiTOTIME=630~{2x}{Right}/xr
/xiTOTIME=635~{2x}{Right}/xr
/xiTOTIME=640~{2x}{Right}/xr
/xiTOTIME=645~{3x}{Right}/xr
/xiTOTIME=650~{3x}{Right}/xr
/xiTOTIME=655~{3x}{Right}/xr
/xiTOTIME=700~{3x}{Right}/xr
/xiTOTIME=705~{4x}{Right}/xr
/xiTOTIME=710~{4x}{Right}/xr
/xiTOTIME=715~{4x}{Right}/xr
/xiTOTIME=720~{4x}{Right}/xr
/xiTOTIME=725~{5x}{Right}/xr
/xiTOTIME=730~{5x}{Right}/xr
/xiTOTIME=735~{5x}{Right}/xr
/xiTOTIME=740~{5x}{Right}/xr
/xiTOTIME=745~{6x}{Right}/xr
/xiTOTIME=750~{6x}{Right}/xr
/xiTOTIME=755~{6x}{Right}/xr
/xiTOTIME=800~{6x}{Right}/xr
/xiTOTIME=805~{7x}{Right}/xr
```

```
/xiTOTIME=1230~{20x}{Right}/xr
/xiTOTIME=1235~{20x}{Right}/xr
/xiTOTIME=1240~{20x}{Right}/xr
/xiTOTIME=1245~{21x}{Right}/xr
/xiTOTIME=1250~{21x}{Right}/xr
/xiTOTIME=1255~{21x}{Right}/xr
/xiTOTIME=1300~{21x}{Right}/xr
/xiTOTIME=1305~{22x}{Right}/xr
/xiTOTIME=1310~{22x}{Right}/xr
/xiTOTIME=1315~{22x}{Right}/xr
/xiTOTIME=1320~{22x}{Right}/xr
/xiTOTIME=1325~{23x}{Right}/xr
/xiTOTIME=1330~{23x}{Right}/xr
/xiTOTIME=1335~{23x}{Right}/xr
/xiTOTIME=1340~{23x}{Right}/xr
/xiTOTIME=1345~{24x}{Right}/xr
/xiTOTIME=1350~{24x}{Right}/xr
/xiTOTIME=1355~{24x}{Right}/xr
/xiTOTIME=1400~{24x}{Right}/xr
/xiTOTIME=1405~{25x}{Right}/xr
/xiTOTIME=1410~{25x}{Right}/xr
/xiTOTIME=1415~{25x}{Right}/xr
/xiTOTIME=1420~{25x}{Right}/xr
/xiTOTIME=1425~{26x}{Right}/xr
/xiTOTIME=1430~{26x}{Right}/xr
/xiTOTIME=1435~{26x}{Right}/xr
/xiTOTIME=1440~{26x}{Right}/xr
/xiTOTIME=1445~{27x}{Right}/xr
/xiTOTIME=1450~{27x}{Right}/xr
/xiTOTIME=1455~{27x}{Right}/xr
/xiTOTIME=1500~{27x}{Right}/xr
/xiTOTIME=1505~{28x}{Right}/xr
/xiTOTIME=1510~{28x}{Right}/xr
/xiTOTIME=1515~{28x}{Right}/xr
/xiTOTIME=1520~{28x}{Right}/xr
/xiTOTIME=1525~{29x}{Right}/xr
/xiTOTIME=1530~{29x}{Right}/xr
/xiTOTIME=1535~{29x}{Right}/xr
/xiTOTIME=1540~{29x}{Right}/xr
/xiTOTIME=1545~{30x}{Right}/xr
/xiTOTIME=1550~{30x}{Right}/xr
/xiTOTIME=1555~{30x}{Right}/xr
/xiTOTIME=1600~{30x}{Right}/xr
/xiTOTIME=1605~{31x}{Right}/xr
/xiTOTIME=1610~{31x}{Right}/xr
/xiTOTIME=1615~{31x}{Right}/xr
/xiTOTIME=1620~{31x}{Right}/xr
/xiTOTIME=1625~{32x}{Right}/xr
   /xr
```

KB
SEARCHES FOR AN INSTRUCTOR NAME AND SENDS THE CURSOR TO IT IN THE DECONFLICTION SCHEDULE
==========

```
/xina=s2~{F2}s2~/xr
/xina=s3~{F2}s3~/xr
/xina=s4~{F2}s4~/xr
/xina=s5~{F2}s5~/xr
/xina=s6~{F2}s6~/xr
/xina=s7~{F2}s7~/xr
/xina=s8~{F2}s8~/xr
/xina=s9~{F2}s9~/xr
/xina=s10~{F2}s10~/xr
/xina=s11~{F2}s11~/xr
/xina=s12~{F2}s12~/xr
/xina=s13~{F2}s13~/xr
/xina=s14~{F2}s14~/xr
/xina=s15~{F2}s15~/xr
/xina=s16~{F2}s16~/xr
/xina=s17~{F2}s17~/xr
/xina=s18~{F2}s18~/xr
/xina=s19~{F2}s19~/xr
/xina=s20~{F2}s20~/xr
/xina=s21~{F2}s21~/xr
/xina=s22~{F2}s22~/xr
/xina=s23~{F2}s23~/xr
/xina=s24~{F2}s24~/xr
/xina=s25~{F2}s25~/xr
/xina=s26~{F2}s26~/xr
/xina=s27~{F2}s27~/xr
/xina=s28~{F2}s28~/xr
/xina=s29~{F2}s29~/xr
/xina=s30~{F2}s30~/xr
/xina=s31~{F2}s31~/xr
/xina=s32~{F2}s32~/xr
/xina=s33~{F2}s33~/xr
/xina=s34~{F2}s34~/xr
/xina=s35~{F2}s35~/xr
/xina=s36~{F2}s36~/xr
/xina=s37~{F2}s37~/xr
/xina=s38~{F2}s38~/xr
/xina=s39~{F2}s39~/xr
/xina=s40~{F2}s40~/xr
/xina=s41~{F2}s41~/xr
/xina=s42~{F2}s42~/xr
/xina=s43~{F2}s43~/xr
/xina=s44~{F2}s44~/xr
/xina=s45~{F2}s45~/xr
/xina=s46~{F2}s46~/xr
/xina=s47~{F2}s47~/xr
/xina=s48~{F2}s48~/xr
```

```
/xina=s49~{F2}s49~/xr
/xina=s50~{F2}s50~/xr
/xina=s51~{F2}s51~/xr
/xina=s52~{F2}s52~/xr
/xina=s53~{F2}s53~/xr
/xina=s54~{F2}s54~/xr
/xina=s55~{F2}s55~/xr
/xina=s56~{F2}s56~/xr
/xina=s57~{F2}s57~/xr
/xina=s58~{F2}s58~/xr
/xina=s59~{F2}s59~/xr
/xina=s60~{F2}s60~/xr
```

KC

SEARCHES FOR A STUDENT NAME AND SENDS THE CURSOR TO IT IN THE DECONFLICTION SCHEDULE

==========

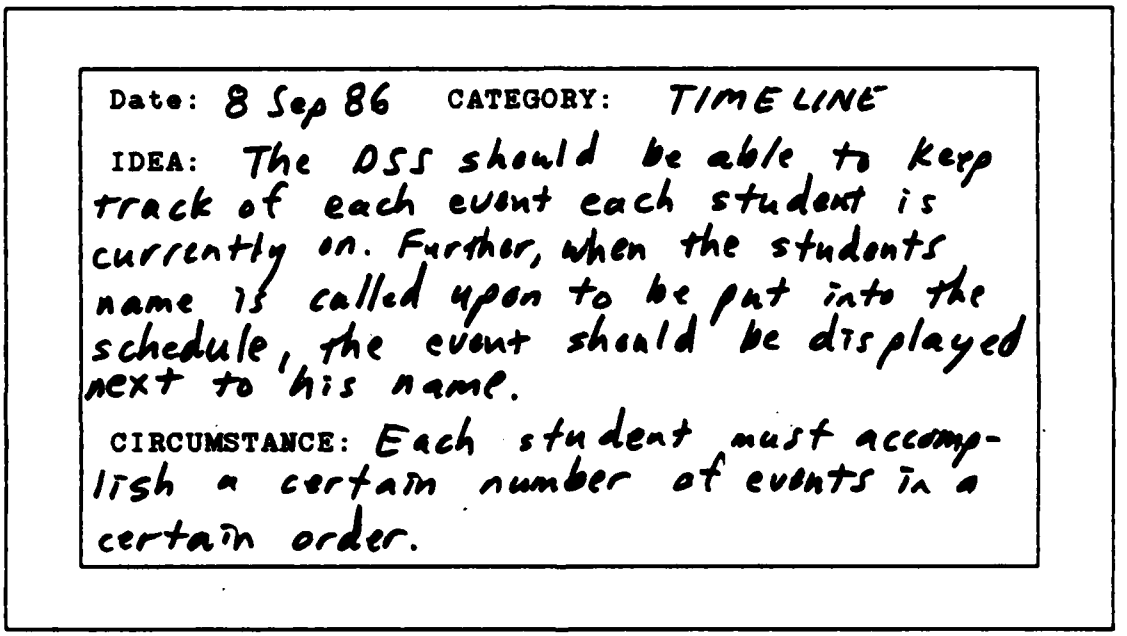
```
/xina=bx2~{F2}bx2~/xr
/xina=bx3~{F2}bx3~/xr
/xina=bx4~{F2}bx4~/xr
/xina=bx5~{F2}bx5~/xr
/xina=bx6~{F2}bx6~/xr
/xina=bx7~{F2}bx7~/xr
/xina=bx8~{F2}bx8~/xr
/xina=bx9~{F2}bx9~/xr
/xina=bx10~{F2}bx10~/xr
/xina=bx11~{F2}bx11~/xr
/xina=bx12~{F2}bx12~/xr
/xina=bx13~{F2}bx13~/xr
/xina=bx14~{F2}bx14~/xr
/xina=bx15~{F2}bx15~/xr
/xina=bx16~{F2}bx16~/xr
/xina=bx17~{F2}bx17~/xr
/xina=bx18~{F2}bx18~/xr
/xina=bx19~{F2}bx19~/xr
/xina=bx20~{F2}bx20~/xr
/xina=bx21~{F2}bx21~/xr
/xina=bx22~{F2}bx22~/xr
/xina=bx23~{F2}bx23~/xr
/xina=bx24~{F2}bx24~/xr
/xina=bx25~{F2}bx25~/xr
/xina=bx26~{F2}bx26~/xr
/xina=bx27~{F2}bx27~/xr
/xina=bx28~{F2}bx28~/xr
/xina=bx29~{F2}bx29~/xr
/xina=bx30~{F2}bx30~/xr
/xina=bx31~{F2}bx31~/xr
/xina=bx32~{F2}bx32~/xr
/xina=bx33~{F2}bx33~/xr
/xina=bx34~{F2}bx34~/xr
```

```
/xina=bx35~{F2}bx35~/xr
/xina=bx36~{F2}bx36~/xr
/xina=bx37~{F2}bx37~/xr
/xina=bx38~{F2}bx38~/xr
/xina=bx39~{F2}bx39~/xr
/xina=bx40~{F2}bx40~/xr
/xina=bx41~{F2}bx41~/xr
/xina=bx42~{F2}bx42~/xr
/xina=bx43~{F2}bx43~/xr
/xina=bx44~{F2}bx44~/xr
/xina=bx45~{F2}bx45~/xr
/xina=bx46~{F2}bx46~/xr
/xina=bx47~{F2}bx47~/xr
/xina=bx48~{F2}bx48~/xr
/xina=bx49~{F2}bx49~/xr
/xina=bx50~{F2}bx50~/xr
/xina=bx51~{F2}bx51~/xr
/xina=bx52~{F2}bx52~/xr
/xina=bx53~{F2}bx53~/xr
/xina=bx54~{F2}bx54~/xr
/xina=bx55~{F2}bx55~/xr
/xina=bx56~{F2}bx56~/xr
/xina=bx57~{F2}bx57~/xr
/xina=bx58~{F2}bx58~/xr
/xina=bx59~{F2}bx59~/xr
/xina=bx60~{F2}bx60~/xr


EMBEDDED FORMULA'S
CONVERTS THE WING DATA INTO THE PROPER FORMAT (0830)
==================================================

@int(SI59/60)*100+(@mod(SI59,60))
@int(SI60/60)*100+(@mod(SI60,60))
@int(SI61/60)*100+(@mod(SI61,60))
@int(SI62/60)*100+(@mod(SI62,60))
@int(SI63/60)*100+(@mod(SI63,60))
@int(SI64/60)*100+(@mod(SI64,60))
@int(SI65/60)*100+(@mod(SI65,60))
@int(SI66/60)*100+(@mod(SI66,60))
@int(SI67/60)*100+(@mod(SI67,60))
@int(SI68/60)*100+(@mod(SI68,60))
@int(SI69/60)*100+(@mod(SI69,60))
@int(SI70/60)*100+(@mod(SI70,60))
@int(SI71/60)*100+(@mod(SI71,60))
@int(SI72/60)*100+(@mod(SI72,60))
@int(SI73/60)*100+(@mod(SI73,60))
@int(SI74/60)*100+(@mod(SI74,60))
@int(SI75/60)*100+(@mod(SI75,60))
@int(SI76/60)*100+(@mod(SI76,60))
@int(SI77/60)*100+(@mod(SI77,60))
@int(SI78/60)*100+(@mod(SI78,60))
```

APPENDIX I

HOOKBOOK

The hookbook is an extension of a daily diary incorporating both ideas for future applications and future work required to evolve the DSS. The authors recommend the format shown below. It must contain four essential elements.



(20)

Figure I.1. Notecard Example

The elements of the hookbook necessary to document the idea evolution process are:

1. Date of the idea.
2. Category of the idea.
3. Idea.
4. Circumstance(s) under which the idea came about.

The date and category help the developer or user classify the various thoughts. This classification may help direct

future research/modification efforts or areas of emphasis. The idea itself is important, but so is the circumstances under which the idea evolved. This situation brings about an understanding of the environment out of which the user documented his concept.

25 Aug 86 PRIORITY

Idea: Be able to sort by student priority. These priorities would include scheduling by student continuity, such as the case where a student(s) falls behind his training schedule due to weather or maintenance. Thus, the student farthest behind would be scheduled first. Circumstance: TDY to Holloman AFB confirmed the user need - our experience in scheduling taught us this feature was necessary.

31 Aug 86 PRIORITY

Idea: A parallel thought would schedule an entire class by continuity in the same manner as an individual student. Circumstance: Our experience in scheduling taught us this feature was necessary. Not only must the scheduler monitor a specific student, questions from management are about the status of a particular class.

31 Aug 86 PRIORITY

Idea: In addition, be able to schedule each student with their assigned instructor(s). This prioritized matching would assign a student's primary instructor first, followed by his secondary instructor, flight commander, operations officer, or any other qualified instructor (as a last resort). Circumstance: Our experience in scheduling taught us this feature was necessary. Safety-of-flight considerations always influenced the need for an instructor to fly with a particular student, whenever possible, for better rapport and instructional technique.

2 Sep 86 AVAILABILITY

Idea: Incorporate the ability to sort by availability (i.e., there is always some personnel missing for Leave, TDY or DNIF) and bring up a list of only available IPs and Students to select from when scheduling. The procedure currently used is selecting from a list of all of the IPs and Students in the squadron. Circumstance: Our experience in scheduling taught us this feature was necessary. Often, personnel were TDY or on leave and the schedulers were not informed as to the status change. Supervisors who knew of the orders usually told the schedulers the change of status as they were reviewing the schedule for errors (right before the schedule was due to Wing).

2 Sep 86 QUALIFICATIONS

Idea: Same as above, except for sorting by qualifications. For example, some students are specially monitored (i.e., put on Red Dot status). Red Dot students can only fly with Red Dot qualified IPs. The DSS should be able to flag the user about a mismatched Red Dot student/instructor relationship. Circumstance: Our experience in scheduling taught us this feature was necessary. The instructor/student matchup depends on the qualifications of the IPs and the characteristics of the student.

4 Sep 86 DECONFLICTION

Idea: The DSS should be able to deconflict each squadron member's individual schedule. As the squadron scheduler constructs tomorrow's schedule, he fills out a separate Deconfliction Board. Perhaps deconflict graphically by making a timeline across the top of the screen and, as the squadron scheduler schedules each event, the DSS would display XXXXXXXXXX's under that time block for that indi-vidual. Circumstance: One of the most difficult tasks in scheduling was relying on the deconfliction board during an end-of-day crisis. In most cases it was kept up-to-date. The times the boards were not current were the times sorties and training events were not accomplished due to scheduling a person for different events at the same time.

8 Sep 86 TIME LINE

Idea: The DSS should be able to keep track of which event each student is currently on. Moreover, when the student's name is called upon to be put into the schedule, the event that specific student is on should be displayed next to his name. Circumstance: Each student must accomplish a certain number of events in a certain order. Squadrons do not have the luxury of extra flights, simulators, or make-up classes due to scheduling errors while tracking a student's events.

8 Sep 86 PREREQUISITES

Idea: The DSS should be able to check for completion of prerequisites for each and every event. Circumstance: The syllabus is kept near the scheduling grease board and is consulted for completion of prerequisites prior to sched-uling a student for a ride. A student must accomplish certain prerequisites for each event so that flight safety is not compromised (e.g., the student flies a low-altitude mission without having had the proper academic training).

18 Sep 86 SYSTEM

Idea: Use a database to store, manipulate,and sort information for use in a spreadsheet schedule. Circumstance: Through literature search and interviews with experts on various software, we discovered the flexibility of a spreadsheet combined with the sorting/manipulating power of a database is an ideal solution to a scheduling DSS problem.

27 Sep 86 FLIGHT SHELL

Idea: Copy the Wing-supplied Flight Shell into a Database. Circumstance: Ltc Kunzman at Holloman AFB mentioned he was working on a program to convert the raw data, currently in an unusable mixed-format form supplied by Wing, for use on the Z-100 computer.

13 Oct 86 **REPRESENTATION**

Idea: Design a MAIN MENU that will represent graphically all of our screens. The user should be able to select any screen from any position within the hierarchy. And select the MAIN MENU from any screen, for instance, use the function key F2 to bring up the MAIN MENU in any application. Perhaps have the MM show up in a window to the side of the current screen so as to hide as little information as possible. Circumstance: Storyboard evaluation generated the idea based upon the ROMC memory aid requirement plus the user's needed ability to employ various portions of the DSS at will.

20 Oct 86 **CONTROL**

Idea: Be able to reprogram function keys (F1,...,F10) to give the user and programmer greater flexibility. Circumstance: Storyboard design of ROMC control of different functions depended upon programming the function keys. A DOS window to an external assembly language program is the only method to accomplish redefining function keys known (beyond knowledge scope) to the authors at this time.

23 Oct 86 **MEMORY AIDS**

Idea: On-line Help should be available in any screen. Circumstance: The need to incorporate a Help function stemmed from our own questions about software and, most certainly, a new user's initial questions about a program that would be written.

29 Oct 86 MEMORY AIDS

Idea: Informational prompts would be nice. Circumstance: Natural curiosity about what a program is doing and what the user should do next arose from our own experience in working with and evaluating various software for use in the DSS.

4 Nov 86 SYSTEM

Idea: Tomorrow's schedule and Today's schedule will be input forms to a data base. Sorting and categorizing will be accomplished and displayed to a user via a report form. Circumstance: Discovery that ENABLE is very limited in it's ability to interface a database quickly and simply with a spreadsheet schedule.

10 Nov 86 CONTROL

Idea: Make the DSS much more user friendly by enabling the user to use mouse to select commands and move around the screen. Circumstance: Our experience in using menu-driven, mouse-actuated software in spreadsheet application software convinced us as to the ease, practicality, and versatility of a mouse, especially to a new user.

14 Nov 86 SYSTEM

Idea: Create the entire scheduling program in a spreadsheet application. Circumstance: Discovery that ENABLE database, while very powerful in the data sorting and manipulating process, could not support a dynamic input form necessary for scheduling. Sorting, while slower in spreadsheet, takes about the same time as a database by the time the spreadsheet/database interface takes place. In addition, coding within a spreadsheet is much simpler and far more flexible than from within a database.

20 Nov 86 SYSTEM

Idea: Make a mouse menu to allow use with ENABLE. Circumstance: Discovery that a LOTUS SYMPHONY menu loaded into ENABLE allowed the mouse to function, although not all the menu functions worked in ENABLE.

23 Nov 86 EVALUATION

Idea: Use 'SIDEKICK' for an evaluation tool. In each application of our DSS have a 'SIDEKICK' file that tracks user comments on the DSS performance. The usual evaluation criteria should be used:

a. Ease of input.
   - Help calls.
   - Mistakes.
b. Changes in decision.
c. Service.
   - Time on the system.
   - # of iterations.
d. Productivity.
   - Time/motion study.
   - Delphi method.

Circumstance: A method needs to be found to document user comments for feedback analysis and design evolution. Sidekick does work within ENABLE, yet it redefines the [TAB] key function and, in certain cases, inhibits macro operation.

30 Nov 86 DECONFLICTION

Idea: The DSS should be able to track 'CREW REST'. Crew rest is a 12 hour period that must be observed from the end of an individuals last event to the first event of the next day. The DSS should be able to inform the user if 'CREW REST' is violated. Circumstance: Flying regulations reuire a certain period of rest between flying activities. Working the deconfliction problem with the spreadsheet reminded us of this fact.

2 Dec 86 MEMORY AID

Idea: 'Idiot-proof' the program. The DSS should be able to find erroneous inputs and flag the user. Circumstance: Past experience with programs and ROMC incorporation of memory-aid flags define this requirement.

15 Dec 86 EVALUATION

Idea: Use ENABLE word processing windows to track user comments for the design evolution process. Circumstance: Discovery that Sidekick would, during a thesis progress demonstration, adversely affect the advanced programming applications currently residing within the existing code.

30 Jan 87 IMPLEMENTATION

Idea: Create an INSTALL program to move the scheduling program and its applicable files into an INSTALL-generated directory. Circumstance: Noticing, through installing several word processing programs, the ease by which the programs were set up requiring a minimum of user interaction.

1 Feb 87 IMPLEMENTATION

Idea: Create a batch file to enter ENABLE, find out the name of the next day's schedule, exit ENABLE, test to see if the files exist, then re-enter ENABLE and prompt the user to create the scheduling spreadsheet file. Circumstance: Presently, the program crashes (stops) if the next day's schedule is not present when called by the DSS.

APPENDIX J

Evaluation Methodology

Evaluation must be a continuing process. DSS evaluation should begin before the technical phases (analysis, design, development, testing, and installation) and continue beyond the life of the DSS. If evaluation is done at the end of the life of the DSS, it is likely not to be done at all, and if done it is likely to be a 'confirmation' rather than a useful source of information (18:158).

What to Measure. Figure J.1 shows the measures that can be used to evaluate the impact of the scheduling DSS. The measures are divided into four categories (18:59).

1. Productivity measures are used to evaluate the impact of the DSS on decisions.
2. Process measures are used to evaluate the impact of the DSS on decision making.
3. Perception measures are used to evaluate the impact of the DSS on decision makers.
4. Product measures are used to evaluate the technical merit of the DSS.

How to Measure. The evaluation will be described in terms of two systems: (1) an initiation system (the DSS)

PRODUCTIVITY MEASURES

1. Time to reach a decision
2. Cost of making a decision
3. Results of the decision
4. Cost of implementing the decision

PROCESS MEASURES

1. Number of alternatives examined
2. Number of analyses done
3. Number of participants in the decision making
4. Time horizon of the decision
5. Amount of data used
6. Time spent in each phase of decision making
7. Time lines of the decision

PERCEPTION MEASURES

1. Control of the decision-making process
2. Usefulness of the DSS
3. Ease of use
4. Understanding the problem
5. Ease of 'selling' the decision
6. Conviction that the decision is correct

PRODUCT MEASURES

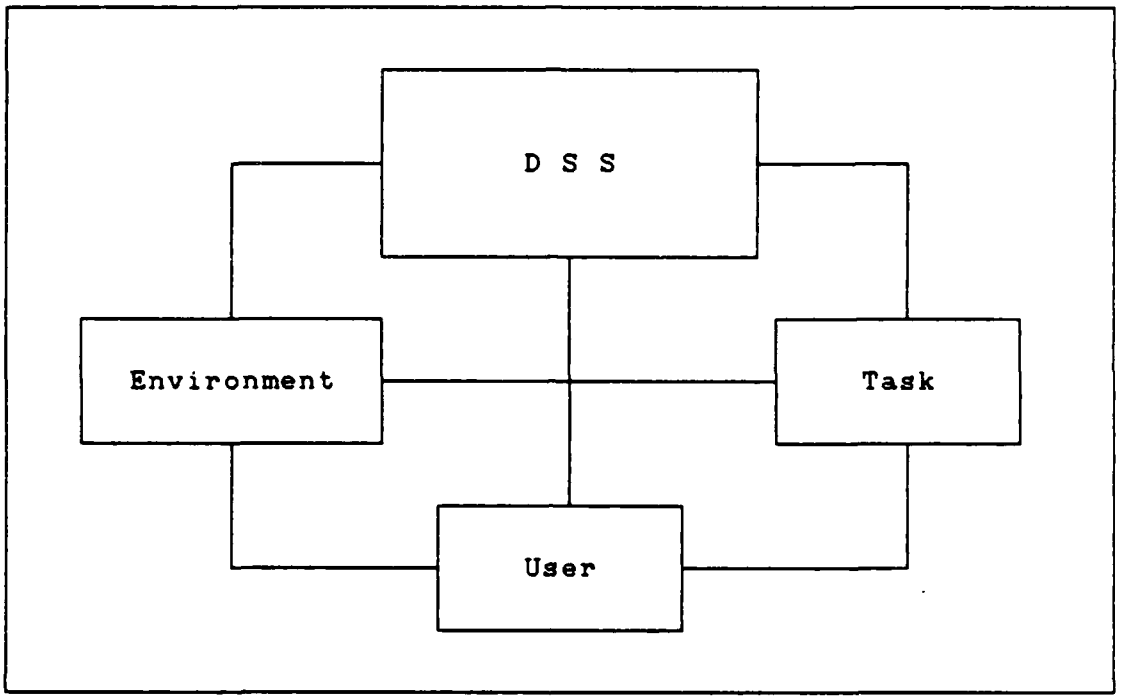
1. Response time
2. Availability
3. Mean time to failure
4. Development costs
5. Operation costs
6. Maintenance costs
7. Education costs

(18:160)

Figure J.1. Measures for DSS Evaluation

whose impact is to be evaluated, and (2) a target system (the decision, the organization, the product) on which impact is to be measured. Each evaluation will describe the expected impact of an initiating system on a target system (18:159).

The initiating and target systems can be all or parts of four basic systems shown below in Figure J.2.



(Sprague:28)

Figure J.2 The Decision-Making System

Initiating and target systems, for evaluation, can be various inter and intra combinations of elements of the four basic systems. The initiating system will usually be all or specific parts of the DSS.

Although the decision support system is treated as a black box in Figure J.2, it it important to recall that the overall system is the decision-making system. The decision-making system consists of a manager/user who uses a decision support system to confront a task in an organizational environment.

Five methods are selected for use to measure and evaluate the impact of the scheduling DSS. These methods do not represent the only possible evaluation schemes but they do cover a wide range from quantitative to opinion oriented methods. The following paragraphs briefly describe each method, list its apparent advantages and disadvantages, and present examples of its use (18:161).

Event Logging. In an event logging evaluation, events that might indicate DSS impact are recorded. An example is keeping a log of customer complaints before and after implementation of a customer service DSS. Event logging is the least structured of the methods to be discussed, and it has the least well defined set of techniques. The methodology for event logging is a combination of historiography and journalism. The basic technique is the recording of events that relate to the effects being investigated. The events may be actions, opinions, newspaper articles, items from memos, dates, and so on. Judgment is required in selecting the events to be recorded, and the set of possible relevant events is not predefinable. The recording may be on a continuous or a before/after basis. The evaluation is simply an analysis of the recorded events.

Event logging is most useful when quantitative measures cannot be used, when a time series of effects is of interest, and when multiple effects are being considered. Event logging can also provide valuable insight for designers and

users of similar systems, and can be a useful background for other evaluations. Event logging has a wide range of applicability, is relatively simple to perform, does not require special data collection techniques, and can be performed on a continuing basis. However, it usually results in large volumes of data which may be difficult to interpret, and it is difficult to guarantee the completeness and accuracy of the data.

Attitude Survey. The attitude survey is a method that attempts to measure opinions through a questionnaire administered to a set of individuals. The questionnaire may be mailed out or administered via interviews. Questions cover a wide variety of attitudes, and usually provide multiple-choice, short statement, or open-ended answer formats. There is a large body of psychological and behavioral research which can be used in designing and analyzing the questionnaires. For example, question questionnaires have been developed for measuring perception of working environment, motivation, and interpersonal contacts. For best results the questionnaires are administered at various intervals during the development and use of the DSS and the results are compared. There are four major problems that may arise in attitude surveys (18:163).

1. In developing a questionnaire to measure attitudes, one may be trying to quantify that which is not quantifiable and the subsequent analysis may be misleading.

2. Questionnaire respondents may interpret questions differently than intended, and answers to some questions may influence answers to others. Thus unintended impacts may be measured, or a single impact may be interpreted as more than one.
3. Administering questionnaires may be an inconvenience to individuals and expensive (in terms of hours lost by respondents and hours spent by interviewers).
4. The method does not identify the causes of any measured changes.

Because of these problems it is best to use questionnaires and analysis techniques validated by others or by pretests, and if possible, to facilitate data collection by sampling. An attitude survey is most useful when the target system consists primarily of people and when perception measures are being used.

Rating and Weighing. Rating and weighting is a highly structured method for a composite numerical evaluation. The methodology involves developing a set of parameters related to the system and effects being evaluated, weighting these parameters in terms of relative importance, and having one or more individuals rate the system on each parameter. For example, department managers could be asked to rate a DSS on accuracy, timeliness, and usability of displays using a scale of 1 to 10, and to assign a relative importance (weight) to each of the three factors. The summation of the ratings times the corresponding weights gives an evaluation score. The scores from each individual may be averaged or totaled.

The rating and weighting method may involve several evaluations and a comparison, or a single evaluation. However, because ratings, and sometimes the weights, are ordinal numbers (i.e., they can be ranked, but their differences are meaningless), the summation of ratings times weights is undefined theoretically. Moreover, in comparing two scores, only if all rating for one system are less than those for the other, and the same evaluators rate each system, do the scores necessarily give an indisputable direction of difference. Therefore, in analyzing results of rating and weighting, the evaluator must remember that the numbers are not precise and may not accurate. If the ratings and weights are reliable, this method is the easiest to interpret. If not, the method will be misleading (18:164).

System Measurement. This method attempts to quantify effects through measurements of the performance of the target system, and therefore it is similar to performance evaluation. In fact, if the target system is the DSS, the evaluation should use product measures for the DSS. Measuring the time to locate a suspect before and after implementation of a DSS for a police department is an example of a system measurement evaluation. The measurements may be collected automatically by DSS or other sensors, through questionnaires, interviews, or observations, or extracted from documents. The method of evaluation is usually a before/after comparison of measurements. The analysis of the measure-

ments often utilize statistical techniques, the data collection is straightforward and often employs sampling, and the results are usually easy to interpret. Unfortunately, the method has a narrow range of applicability because it requires that the affected elements of the target system have performance characteristics that can be quantified. However, if this requirement is satisfied, system measurement is the most reliable evaluation method (18:164).

Value Analysis. Peter Keen recently proposed an evaluation technique which he called value analysis (7:1-16). The approach is similar to cost/benefit analysis, with three important differences. First, the emphasis is on benefits first and costs second. This emphasis is based on the assumption that it is the benefits that are of primary interest to decision makers and that the calculations of cost/benefit ratios are not necessary if the benefits meet some threshold and the costs are within some acceptable limit. Second, the method attempts to reduce risk by requiring prototyping as part of the evaluation method. The assumption here is that the prototype is a low-risk, relatively inexpensive way to obtain relatively accurate evaluation data. Third, the method evaluates DSS as an R & D effort rather than a capital investment. An R & D evaluation tends to encourage innovation rather than return on investment.

Keen describes value analysis as a series of four steps.

1. Establish the operational list of benefits that the DSS must achieve to be acceptable.
2. Establish the maximum cost that one is willing to pay to achieve the benefits.
3. Develop a prototype DSS.
4. Assess the benefits and the costs.

The advantages of the value analysis approach are that it it simple and integrated with an installation approach (prototyping). The main disadvantage is that it is a limited form of evaluation, which may not include all the measures that are relevant. Value analysis also is a much less rigorous method than either the cost/benefit or system analysis techniques. Although not reported in the DSS literature, value analysis seems very close to the intuitive approach that many managers use to evaluate DSS (18:166). Figure J.3 summarizes the different methods.

Combining Methods. The choice of evaluation method will depend on the DSS and the impacts being investigated, and on the evaluator and the environment in which the evaluation is performed. Because of the variety and complexity of the potential effects and because there are problems with all evaluation methods, a combination of methods will probably result in the best evaluation. The usefulness of the

MODEL	EVENT LOGGING	ATTITUDE SURVEY	RATING AND WEIGHTING	SYSTEM MEASUREMENT	VALUE ANALYSIS
Objective	To log events relating to impact on services	To determine impact on attitudes on services	To determine impact through ratings	To test the hypothesis of no difference in services	To determine whether or not to continue to use the DSS
Measures	Events relating to services	Questions on services	Ratings	Times, quantities maybe others	Dollar value of services, DSS costs
Treatments Experiment Units	Before and after DSS Services	Before and after DSS People	Before and after DSS Services	Before and after DSS Services	Prototype DSS Cost and benefit items
Plan for Assigning treatments to units	Use same before and after (i.e., block on services	Block on services	Block on services and evaluators	Block on services	Treatment; assign to units
Plan for selecting units	Judgemental selection	Random selection	Judgemental selection	Either all services or randon selection	Judgemental selection
Analysis criteria and techniques	Qualitative comparison of logged events	Chi-square comparison of response frequencies	Compare overall scores (sums of) rating times weight	Wilcoxon signed ranks comparison for each measure	Are benefits obtained within cost threshold

Figure J.3. Comparison of Five Methods for DSS Evaluation

event log in explaining changes in attitudes has already been mentioned. An attitude survey measures difference impacts. System measurement analysis uses data collection techniques and therefore can be combined easily. Regardless of which combination of methods is chosen, the breadth that the combination provides to the evaluation is worth the effort.

An Example. As an example of the use of the model for DSS evaluations, consider the design of a statistical evaluation of the impact of a DSS on the preparation of a flight schedule (summarized in Figure J.4). The objective is to test the hypothesis that the DSS does not affect schedule preparation time. The measure is the time (person-hours) to prepare the plan, the treatments are the planning procedures before and after implementation of the DSS, and the experimental units are the schedulers that must prepare flight schedules. Suppose that it is decided to use the system measurement method and to measure preparation times under each treatment. A random sample of 10 out of 30 flight plans is selected. The analysis criterion is a comparison of mean plan preparation times, with a significance level of 0.05 (95 percent confidence of accepting the evaluation hypothesis if it is true). The plan for assigning treatments to experimental units is to block on schedulers (i.e., to measure preparation times by the same scheduler before and after system implementation).

METHOD:	System measurement
OBJECTIVE:	Test the hypothesis that the DSS does not affect scheduling time
MEASURE:	Schedule preparation time
TREATMENTS:	Schedule preparation time before and after the DSS is implemented
EXPERIMENTAL UNITS:	Schedulers in the organization
PLAN FOR ASSIGNING TREATMENT TO UNITS:	Measure all units for each treatment
SAMPLE SELECTION PLAN:	Random sample of 10
ANALYSIS CRITERIA AND TECHNIQUES:	95 percent chance of accepting hypothesis if true using paired test (or Wilcoxon signed test)

DATA (HYPOTHETICAL)

SCHEDULER	BEFORE (PERSON-HOURS)	AFTER (PERSON-HOURS)
1	11	9
2	12.5	8
3	11	8.5
4	12	8
5	10.5	9
6	11	7.5
7	12.5	8.5
8	12	9
9	11.5	6
10	12	8

ANALYSIS RESULTS	
PAIRED t TEST:	(t= -9) Reject hypothesis (at 0.05 level)
WILCOXON TEST:	(T= 0) Reject hypothesis (at 0.05 level)
CONCLUSION	There is sufficient proof that the DSS had significant impact on preparation for the flight schedule.

Figure J.4 Syllabus Student Training Flow

Assuming that the plan preparation times are normally distributed, with the same variance for each treatment, the analysis technique would be a paired test (with nine degrees of freedom) of the difference in times (see Cody:91-94 for a discussion of this test). If the assumptions cannot be made, the Wilcoxon signed ranks test could be used for the analysis (Hines:311).

APPENDIX K

Glossary

Air Combat Maneuvers (ACM)

Air Combat Maneuvers are three ship aircraft syllabus sorties that practice two-ship offensive and defensive moves against a single defender. Only those students that are going on to the F-15 aircraft receive these rides. The two-ship fighting unit is stressed as the target jet is attacked and defended against.

Aircraft Configuration

The aircraft configuration refers to the ordnance that maintenance puts on the aircraft. The aircraft configuration may be six bombs on one sortie and clean (no bombs) on the next. The configuration must match the type of mission that the aircraft is to fly.

ASCII

The American Standard Code for Information Interchange (ASCII) is a seven-bit information code used by many computers to represent letters, numbers, and special characters. Computer microprocessors interpret this code and perform commands based on the coded instructions. All user inputs to ASCII-based computers are translated into the seven-bit code, processed, and displayed as alphanumeric

characters on an output device (e.g., computer screen or printer).

Basic Fighter Maneuvers (BFM)

Basic Fighter Maneuvers are two-ship aircraft sorties that practice basic offensive and defensive maneuvers against another airborne target. Early rides limit the maneuverability of the opponent to allow the student to practice against a predictable target. Later rides have an instructor fly against the student in predefined scenarios.

Check Ride

An aircraft sortie used to check the instructor's performance level and proficiency. Two types of check rides are given. The instrument check evaluates the Instructor Pilots instrument flying ability. The qualification check tests tests the IP's instructional talents.

Crew Rest

The time span from an aircrew's last official duty to the time of his first scheduled event the next day. This time must not be less than twelve hours. Normally, it is the time an aircrew leaves the squadron until he shows up the next day.

Decision Support System (DSS)

A Decision Support System is a system that assists in making decisions. A DSS is usually computer-based with access to a data base that will contain information required by a user.

Duty Not Involving Flying (DNIF)

Duty Not Involving Flying is a category of pilots that does not allow the pilot to fly aboard aircraft. DNIF occurs mainly due to medical reasons. Any pilot that cannot participate in active flying must be classified as DNIF. In essence, DNIF includes all those pilots that are not physically ready to perform flying duties.

Duty Officer (Duty Hog)

The Duty Officer sits at the front desk of the squadron to ensure that the flying operations run smoothly. The Duty Hog posts takeoff times and coordinates with maintenance to get tail numbers of scheduled operational aircraft. In the case of a ground aborted aircraft, the Duty Hog gets a different aircraft for the crew and coordinates new area (i.e., range or low-level) time.

Duty Scheduler

The Duty Scheduler is an IP that generates the next day's schedule. He coordinates with maintenance, operations, and students to deconflict the next day's schedule. The duty

scheduler is responsible for completing the schedule with any last-minute changes. He must follow any and all syllabus rules for scheduling student rides. He must ensure that the student is assigned the proper event and has met all of the applicable prerequisites.

Event

An event is an occurrence that is scheduled by the training syllabus at Holloman. An event could be a jet training flight, an academic lecture, or a briefing. Events are scheduled at specific times with specific requirements and prerequisites. .

Flight Lead

The flight lead is the designated commander of the group of planes in his formation. The flight lead briefs the mission 1.5 hours prior to takeoff. In the briefing, the flight lead covers ground operations, takeoff, departure, mission elements to accomplish, patterns and landings. During the sortie the flight lead is responsible for the actions of all flight members.

Floppy Disk

A floppy disk (disk, for short) is a thin, encased magnetic disk upon which computers store and retrieve bits of information. Much like a cassette tape, disks are highly

portable and easy to use. Microcomputers normally use 5¼ inch floppys able to store 360,000 bytes of information.

Formation (Form)

Formation rides are aircraft syllabus that increase proficiency in basic multiple-aircraft formation skills and single-ship advanced handling. On later rides, the student practices basic four-ship formation and two-ship tactical formation.

Greaseboard

A greaseboard is a smooth, transparent surface upon which users write with wax or crayon-type markers. This board usually overlays some sort of spreadsheet, and is easily erased. This latter feature is important because users normally use greaseboards to record information which rapidly changes.

Hard Drive

Also known as a hard disk, a hard drive is a magnetic disk capable of storing and retrieving tens of millions of bytes of information. This capability greatly enhances a computer's ability to store and run various programs. Most hard drives are inside the computer as actual 'hard'ware, hence the name.

Instructor Pilot (IP)

An Instructor Pilot is a pilot that is qualified to teach students how to fly air combat and surface attack (see Glossary) . All IPs must have at least 400 hours in a fighter aircraft. The IPs must go through 150 days of training with other IPs in an upgrade course before being allowed to fly with students. On training missions IPs instruct students from the back seat of the AT-38B aircraft. IPs brief and lead, and then after the flight, critiques the student's performance.

Kernel

A kernel is the central or essential part of a concept. Applied to a DSS (see below), kernels are the central parts or ideas which make up the entire system. For example, take the act of withdrawing money from a bank. The kernels may include choosing which bank branch to go to or drawing the money from which account (e.g., checking or savings). Another kernel may be deciding where to draw the money from (e.g., teller, drive-through window, or money machine) or how much money to withdraw. All of these concepts are central to the act of withdrawing money from a bank.

Key Kernel

The key kernel is the fundamental, or most important, kernel about which the other kernels center. The prospective

system user usually selects this kernel because he knows what is important to him. Using the above bank example: If a person wants only to use his savings account, then this may be the key kernel. For instance, he may have only one savings account at a certain branch which has no drive-through window or money machine.

Key Kernel System

The key kernel system is the DSS (see below) built about the key kernel. Consider the bank example mentioned above. The key kernel system would assist the user by providing the information necessary about his accounts. This DSS would be expand at a later time to include the rest of the kernels.

Lead In Fighter Training (LIFT)

Lead In Fighter Training is training given to students who are going to fly fighter aircraft in the United States Air Force. When the student graduates from Undergraduate Pilot Training (UPT) and gets assigned to fighter aircraft, he comes to Holloman for upgrade training. LIFT teaches the student how to correctly drop bombs and fly air combat correctly in a plane he already knows how to fly. The AT-38B used at Holloman is a slightly modified version of the T-38 jet trainer.

Letter of X's

The letter of X's is a document that contains all of the qualifications of all a squadron's IPs. All of the IPs names are contained in the letter. Beside each name is the qualifications of that specific IP. For instance an IP may be an RCO, SOF, and a red dot qualified IP.

Line

A line of scheduling is a part of the schedule that consists of one plane, one takeoff time, and often one IPs name and one students name. The most often context is taken as a group of lines. For example, the schedule consists of 39 lines of flying, meaning there are 39 aircraft sorties with a certain aircraft configuration to be filled.

Macros

A macro is a small computer program containing code which tells the computer how to perform various tasks. Macros normally contain specialized code peculiar to a computer or a larger computer program (software). The computer users usually write these special programs.

Mouse

A mouse is a hand-activated peripheral computer device which moves the computer screen cursor in response to it's own movement. The mouse contains one or more keys used to

initiate various computer commands and functions. It is most useful in spreadsheet, word processing, and graphics applications.

Navigation (Nav)

Navigation rides are aircraft syllabus sorties that practice visual low-level procedures. The student flies the aircraft at 500 to 1000 feet above the ground on a predetermined course over the ground. Later rides practice low altitude formation maneuvering in a two-ship sortie.

Range Control Officer (RCO)

Range Control Officer is a qualified IP who is responsible for the safe operation of a controlled bombing range. In addition to monitoring range procedures, the RCO ensures that the weather conditions are appropriate for dropping bombs safely. Every airplane must have radio contact with the RCO to drop bombs on that bombing range. The RCO also supervises several personnel for range maintenance.

Red Dot

Red Dot is a status that refers to students and instructor pilots. A red dot student is a student that needs extra attention due to unusual circumstances or proficiency. Perhaps the student has had a long layoff from his last flight. The operations officer of each squadron screens the incoming

students for any unusual circumstances and assigns that particular student a 'red dot' status. The only instructor pilots to fly with a red dot student are red dot instructor pilots. Only the more experienced instructor pilots acquire red dot status. The idea is to match up experienced instructor pilots with less proficient students for safety and instructional reasons.

Scheduling Shell

The Shell is a list of takeoff times, area times, number of aircraft, and configurations that constitute the starting point of the daily schedule. The Wing Scheduling shop deconflicts takeoff and area times among the four squadrons at Holloman and publishes the shell weekly. The schedulers at each of the four squadrons use the shell as an initial input to the daily scheduling process

Storyboard

A storyboard is a phrase coined by cartoonists to denote a single picture in a series of animated pictures. This single picture was an important to the animation in that it reflected a main concept about the story. Consider, for example, a cartoon showing Mickey Mouse at an amusement park. The first storyboard may show him at the gate counter buying a ticket. The following ones may show him riding the ferris wheel, merry-go-round, and side show in succession.

In this way, the storyboard shows each important step in the animation. Applied to a DSS (see below), the storyboard shows each important computer screen necessary to complete the entire system.

Supervisor of Flying (SOF)

The SOF is a qualified IP who is responsible for the flying operations of all the AT-38s at Holloman AFB. He has the authority to cancel flights due to weather or other circumstances. He also is there to assist any aircraft should it experience an airborne emergency. The SOF also resolves any conflicts that are happening during flying operations.

Surface Attack (SA)

Surface Attack rides are four-ship aircraft syllabus sorties to practice basic range, bombing, and strafing procedures. The student drops practice bombs on a controlled range during this phase of training.

Timeline

The timeline is a measure of each individual's or class's currency according to the syllabus. Each event is listed in the syllabus on the day it should occur. If all events occur when they should then the timeline stays at zero. However, if the student or class falls behind for some reason then the timeline will reflect the difference between the

number that should have occurred and the number that have already occurred. For instance, if the student or class has accomplished 34 events by the 23rd day and the syllabus required 37 events by that time, the timeline reflects a -3. The -3 indicates the student or class is 3 events behind the syllabus schedule.

Transition (TR)

Transition rides are single aircraft syllabus that get the student familiar with the aircraft again. The student learns handling/aerodynamic characteristics of the AT-38B and is introduced to the local flying area. The student also practices takeoffs and landings at Holloman AFB.

END

6-87

DTIC