

ISA:
A PROTOTYPE INTELLIGENT SCHEDULING ASSISTANT
FOR
DEFENSE ACQUISITION MANAGEMENT

Jerry L. Moran
Program Executive Office, Networks
Fort Monmouth, New Jersey 07703

Wade H. Shaw Jr.
Department of Electrical & Computer Engineering
Air Force Institute of Technology
Dayton, Ohio 45433

ABSTRACT

This paper presents an application of artificial intelligence to defense acquisition management. The goal is to demonstrate how knowledge-based methods could improve scheduling of defense acquisition programs. The objectives are first, to determine the kind of knowledge needed to tailor schedules and second, to develop a framework for using that knowledge to generate tailored schedules. Scheduling of defense acquisition programs is a difficult problem for which expert systems are an appropriate solution methodology. This research identified 35 characteristics of a defense acquisition program which affect the applicability, duration, and relationships of tasks required to go from receipt of a Program Management Directive to contract award. It extends the model network concept used in the Aeronautical Systems Division (ASD) and the Air Force Acquisition Logistic Command (AFALC) of the United States Air Force. ISA, a prototype Intelligent Scheduling Assistant, successfully shows how knowledge used to tailor program schedules can be captured in a rule-based system. ISA uses the values of acquisition program characteristics to generate tailored schedules. The concept is applicable to any project scheduling problem based on network models.

INTRODUCTION

The average defense acquisition program experiences cost growth of 30 percent to 100 percent while program deliveries slip 30 percent. Cost growth and schedule slippage are due primarily to instability in establishing requirements, planning, and budgeting for weapon systems (Gansler, 1986:1). Scheduling can reduce instability in a program. Scheduling is a process of deciding what work needs to be done, who will be responsible for the work, when the work should be done, and what order the work should be done in. Network schedules provide graphical representations of plans which can be used to indicate progress, identify problems, and aid communications (Woffinden, 1987).

Despite its advantages, scheduling is difficult because no two defense acquisition programs are the same. This variability makes it difficult to decide what work is needed, who should be responsible, how long it should take, and what order it should be done in. Experience is often the best guide to developing a schedule. However, due to the high rotation rate among both military and civilian personnel, many people assigned to defense acquisition programs are new and inexperienced (Gansler, 1986:9).

Model networks attempt to capture the expertise of experienced program personnel. Model networks must be tailored to specific programs and is a process of deciding which tasks apply, specifying who is responsible for each task, estimating how long each task takes, and making appropriate changes in the order of the tasks.

The Problem

Generalized model networks are difficult to use because they are too general and lack sufficient tailoring guidance. Model networks include every conceivable task that may occur in the program schedule. Some of these tasks are mutually exclusive and may not occur in the same network. Suggested task durations may vary widely. Unfortunately, model networks

lack sufficient guidance to help inexperienced personnel make tailoring decisions. Thus, users often find tailoring model networks to be a difficult and time-consuming process.

The goal of this paper is to demonstrate how knowledge-based methods could improve scheduling of defense acquisition programs. The objectives are first, to determine the kind of knowledge needed to tailor schedules and second, to develop a framework for using that knowledge to generate tailored schedules. This paper presents a demonstration prototype for generating tailored schedules for awarding defense contracts.

Approach

An existing model network consisting of 48 tasks and covering 28 months was used as a starting point for this research. The development effort was divided into the six distinct phases typical of many knowledge-based system developments. These phases were problem assessment, knowledge acquisition, prototype design, tool selection, prototype development, and prototype evaluation.

The problem assessment phase was used to acquire a thorough understanding of the scheduling and tailoring processes. A literature review, focusing on project scheduling, was used to gain a better understanding of scheduling techniques. Program management personnel, defense contractors, and individuals with scheduling knowledge were interviewed to gain their insights on the use of network schedules and model networks. The data collected was used to assess the potential for developing an expert system for generating tailored schedules.

The knowledge acquisition phase was used to determine the specific knowledge needed to make tailoring decisions. Program management personnel were interviewed to identify the kinds of data, knowledge, and procedures used to tailor tasks in the 48-task model network.

The prototype design phase was used to establish system requirements and design the demonstration prototype. The design was approached from a functional point of view and considered twelve principles for the design of interactive computer systems.

The tool selection phase was used to select tools for the demonstration prototype. Tool selection was based on availability, power, sophistication, support facilities, reliability, maintainability, and tool features.

The prototype development phase was used to develop the demonstration prototype. The prototype implements that subset of the system requirements deemed necessary to demonstrate the feasibility of an operational system. The prototype was developed using an incremental approach.

The prototype evaluation phase was used to evaluate the demonstration prototype. The prototype was evaluated by testing its ability to generate tailored schedules. The prototype was also evaluated by potential users.

SCHEDULING TECHNIQUES

Schedules are typically represented using bar charts or activity networks. The most popular form of bar chart is the Gantt chart. Activity networks are used in the Critical Path Method (CPM) and the Program Evaluation and Review Technique (PERT). A wide variety of computer-based scheduling tools

which use bar charts and/or activity networks have been developed. Model networks are used by ASD and others to augment PERT-based schedule development. Researchers in artificial intelligence have investigated techniques for developing knowledge-based plans and schedules.

Computer-based Scheduling Tools

Computer programs for project management were first developed in the 1950s during the U.S. Navy's Polaris project. Commercial marketing of mainframe software for CPM applications, used primarily to manage large projects in construction, started in the 1960s. The first commercial personal computer (PC) version of project management software, the Harvard Project Manager, was introduced in 1983. More than 100 PC-based project management programs, ranging from \$35 to almost \$8000, are commercially available today (Fersko-Weiss, 1987:153,166). The Air Force Acquisition Logistics Command (AFALC) developed the Computer Supported Network Analysis System (CSNAS) as a PERT/CPM-based interactive scheduling system for program management. CSNAS currently has more than 100 users, is written in FORTRAN, and is available in mainframe (HP3000 and VAX) and PC (IBM-compatible and Zenith) versions (Clark, 1987).

The Defense Systems Management College (DSMC) is developing the Program Manager's Support System (PMSS) to support the decision-making process of the PM. Designed for Zenith and IBM-compatible PCs, PMSS uses decision support systems technology. PMSS contains modules that develop Gantt charts, develop PERT networks, perform risk analysis, and do milestone management (Scanlon and Schutt, 1987).

All computer-based scheduling tools require the user to determine and input activities, durations and relationships for a project. The software determines timing and produces a graphical representation of the project schedule based on these inputs. Thus, the user must make all scheduling and tailoring decisions independent of the project management software used.

Model Networks

CSNAS contains a database of model networks for weapons system acquisition programs (AFALC, 1987:234). Various organizations in ASD have developed their own model networks for internal use. A model network typically consists of a CSNAS data file, a PERT network, and a set of reference sheets. The CSNAS data file, containing the schedule information, can be copied and tailored to a specific program. The PERT network graphically portrays the generic schedule. The reference sheets provide brief descriptions (duration, participants, references and other information) of each task.

The Deputy for Reconnaissance/Strike and Electronic Warfare (ASD/RW) is developing a model network for long term programs and training new program managers in ASD/RW. The model network, split into two phases, attempts to capture the corporate experiences of ASD/RW. The Phase I network, completed in August 1985, covers 48 activities and 28 months from receipt of a Program Management Directive (PMD) to contract award. The Phase II network, undergoing validation, covers 76 activities and 8 years from contract award to Program Management Responsibility Transfer (PMRT) (Zornes, 1987).

The Deputy for Aeronautical Equipment (ASD/AE) used model networks to develop the Program Office Internal Networking and Tracking System (POINTS). Derived from a historical review of 32 programs in ASD/AE, POINTS contains model networks for use by program and functional managers in ASD/AE (ASD/AE, 1987).

Model networks include every conceivable task that may occur for a given defense acquisition program. Some of these tasks are mutually exclusive and may not occur in the same network. Some task durations vary up to a year between pessimistic and optimistic estimates. Unfortunately, model networks offer little tailoring guidance to the user. Thus, model networks are difficult and time-consuming for the inexperienced person to use, and they have failed to gain acceptance by their intended users.

Knowledge-based Scheduling Techniques

Scheduling is a difficult, knowledge-intensive problem for two main reasons. First, many potential schedules may be developed using different combinations of tasks, durations, and relationships. Second, the evaluation of the quality of a schedule is hard to assess (Fox, B.R. and Kempf, 1985:487-488). Early AI research focused on developing general techniques for solving problems. The General Problem Solver (GPS) uses states, differences between states, and operators for changing states to represent problems. The key to solving a problem using GPS is to select operators which reduce the differences between the current state and the goal state. The Stanford Research Institute implemented GPS in STRIPS, a system for generating plans as a sequence of operators (Winston, 1977:131,137-143).

Sacerdoti used procedural networks to show that plans could be created that allow operations to execute in parallel (Sacerdoti, 1975:206-208). His approach, implemented in Nets of Action Hierarchies (NOAH), recursively expands a goal step into a network of sub-steps which achieve the goal. Procedural networks are similar in structure to PERT networks. Procedural networks have been enhanced for a number of knowledge-based planners/schedulers. NONLIN (Non-linear planner) was developed by Tate as an aid for constructing project networks (Tate, 1977). Wilkins and Robinson developed SIPE (System for Interactive Planning and Execution) as a domain-independent interactive planner (Wilkins and Robinson, 1981). PLANNER'S WORKBENCH, developed by a group headed by Hayes-Roth, is an aid for re-planning (Hayes-Roth and others, 1981). DEVISER, a general-purpose planner/scheduler built by Vere, provides a capability for adding time constraints to a procedural network (Vere, 1983).

Fox used constraint-directed search in ISIS (Intelligent Scheduling and Information System) to construct job-shop schedules (Fox, Mark S., 1983). He discovered that human schedulers spend 80-90% of their time trying to determine what constraints apply to schedule generation. ISIS was designed to develop schedules that satisfy as many constraints as possible in near real-time.

Many other knowledge-based planning/scheduling techniques have been investigated. Hayes-Roth simulated errand-planning using a blackboard approach (Hayes-Roth and others, 1979). Fukumori generated train schedules using range-constriction search (Fukumori, 1980). Gazdik combined fuzzy sets and graph theory in FNET to handle uncertainty associated with scheduling (Gazdik, 1983). Sage used multiple criteria decision making to solve scheduling problems (Sage, 1984). Newman and Kempf used opportunistic scheduling to develop schedules for a robot that tends machines in a manufacturing plant (Newman and Kempf, 1985).

This paper uses knowledge-based techniques to extend the model network concept. The prototype Intelligent Scheduling Assistant (ISA) generates tailored schedules using program knowledge and task knowledge. The results are exported to CSNAS for actual schedule computation and network drawings.

PROBLEM ASSESSMENT

An experience survey was conducted by interviewing seventeen people with various degrees of experience and responsibility for scheduling. Four upper-level and middle-level managers in ASD, two program managers in ASD/RW, three schedulers in ASD, two people who worked on ASD/RW's Phase I model network, three contractors who provide scheduling support to ASD/RW, and three people outside of ASD were interviewed. A summary of the survey results is available in Moran, 1988.

Scheduling in ASD/RW

ASD/RW is a matrix organization made up of systems directorates and functional area directorates. Functional area personnel support programs in the systems directorates. One person may support multiple programs. Program management teams, consisting of a PM and functional area representatives, are formed early in a program. Some programs simply use suspense calendars to keep track of events. Others use milestone charts to schedule the program. The teams that use network schedules usually tailor schedules from similar programs or build a schedule from scratch. Some programs hire contractors to do scheduling.

When modifying the Phase I model network, the program management team analyzed network tasks, durations, and relationships. The analysis was done individually or in team meetings. Tasks were reviewed to determine which, if any, did not apply to the program. New tasks were added to the network during the review process. Task durations were reviewed to determine if changes were needed. Task relationships were examined for changes which would maximize parallel execution of tasks. The schedule was modified to reflect all changes, and the review process iterated until a general consensus was reached.

Scheduling Problems

Four factors contributed to scheduling problems in ASD/RW. First, most programs lack people who are experienced in developing schedules. Second, schedules are used improperly. Third, programs operate with limited resources and manpower. Fourth, many defense acquisition programs are in a constant state of flux.

Schedule development and maintenance is a time-consuming and manpower-intensive process. It takes weeks of coordination effort to develop a realistic program schedule. Considerably more effort is needed to manage and maintain a schedule. Regardless of the cause, unrealistic program schedules lead to increased program costs and unforeseen schedule delays.

Expert System Potential

A conventional computer program manipulates data using algorithms. A knowledge-based system is a computer program that manipulates knowledge using heuristics. An expert system is a knowledge-based system which performs at the level of an expert in a specific problem domain (Waterman, 1986:24-30).

Expert systems could reduce problems of inexperience by capturing the corporate knowledge and experiences of ASD/RW. Expert systems could reduce unintentional misuse of schedules by generating realistic schedules based on program knowledge and task knowledge. Expert systems could be used to suggest alternatives for resolving schedule problems. Expert systems could reduce resource requirements for schedule development by reducing the demand on experienced personnel within ASD/RW. Expert systems could facilitate schedule changes caused by changes in requirements. Expert systems could improve the quality and consistency of schedule development.

Waterman suggests that expert systems should be considered only when expert systems are possible, appropriate, and justified (Waterman, 1986:127). Similarly, Prerau offers more than fifty factors to consider when evaluating potential expert system applications (Prerau, 1985:26-30). The factors suggested by both authors boil down to evaluating the feasibility, suitability and desirability of building an expert system. Thus, the potential for using expert systems to improve model networks is evaluated in terms of feasibility, suitability and desirability.

Feasibility

Expertise is not specifically available for tailoring the Phase I model network. However, a wealth of knowledge and experience for selecting tasks, durations, and relationships is available from experienced program managers and functional area personnel. Regulations, policy letters, and operating instructions provide another source of knowledge. The Phase I network itself contains a great deal of knowledge about required tasks, durations and relationships.

Suitability

All solutions are appropriate for scheduling defense acquisitions. Although schedules may change quite often, scheduling knowledge such as regulatory requirements changes much less often. Thus, defense acquisition provides a relatively stable domain. Scheduling requires symbolic manipulation and uses heuristics to achieve solutions. It takes years of experience to develop proficiency in scheduling. The ability to document decisions and provide explanations is desirable.

An expert system is appropriate for implementation in ASD/RW. The expert system could be phased in over time as knowledge bases are developed for each functional area. Non-optimal solutions are acceptable. Indeed, since requirements and funding are likely to change, the generation of an optimal schedule is not needed. The greatest difficulty for implementation lies in the testing of the expert system. Because a unique correct solution for any program does not exist, the reasonableness of any solution is a subjective judgment. The subjectiveness of the solution makes explanation capabilities very important when the solution is reviewed for acceptability.

Desirability

A need for the expert system exists. ASD/RW is responsible for more than 30 defense acquisition programs and lacks sufficient numbers of experienced schedulers to cover all of these programs. They also lack sufficient funding to contract out scheduling for all of the programs. Thus, an expert system would help leverage the corporate experience in ASD/RW.

An expert system could provide a variety of benefits. It could save the time of experienced personnel and reduce coordination efforts. It could provide a degree of consistency in schedule generation. It could provide a record of the decision process that could be used when reviewing the schedule for endorsement.

KNOWLEDGE ENGINEERING

The problem assessment phase revealed that the Phase I network had not been accepted by its intended users. Only two cases were discovered where the Phase I network had been used. Experts in tailoring the Phase I network do not exist. Thus, on-site observation and problem analysis could not be used to obtain knowledge about tailoring the Phase I network.

Despite the problems, the approach described for using the Phase I network seemed reasonable. Members of the project team would analyze each task individually to determine its applicability, duration, and relationship to other tasks in the networks. Modifications were made where deemed appropriate. A team approach was necessary because no single individual knew enough about the entire acquisition process to tailor the complete network. Thus, ISA proved to be a multiple expert problem.

Knowledge Engineering Interviews

People were selected for knowledge engineering interviews in three ways. First, people interviewed during the problem assessment phase were asked to recommend individuals who were knowledgeable in different functional areas. Second, some knowledge engineering interviews led to referrals to other people who were deemed to be better qualified in some aspect of the problem. Third, functional area supervisors were contacted for assistance with tasks which were not adequately covered by referrals.

Knowledge engineering interviews focused on each task in isolation. Individuals were asked for criteria which could be used to determine task applicability, duration and relationships. First, the individual being interviewed was asked under what circumstances a task would not be applicable. Once applicability criteria were established, the individual was asked how reasonable durations would be estimated. Finally, individuals were asked to comment on the task relationships defined in the Phase I network.

Knowledge Engineering Results

Knowledge engineering interviews focused on the 48 tasks contained in the Phase I network. One of these tasks was dismissed as inappropriate, three were combined with other tasks, and eight new tasks were identified. Most of the additions were the result of expanding single tasks into two or more tasks. A complete listing of the tasks is contained in Moran, 1988.

Although some individuals were able to clearly define criteria for making tailoring decisions, most found it difficult. This is a major obstacle to the development of an expert system because the quality of the schedule is dependent on the quality of the knowledge in the system. However, the purpose of this paper was to demonstrate how knowledge-based techniques could be used to construct ISA. Since the technique for

generating the schedule is not dependent on specific program and task knowledge, the quality of the knowledge collected was not critical.

Thirty-five program characteristics were defined for use in tailoring various tasks. Thirteen of these criteria affect the applicability of twenty-three tasks. For example, follow-on programs do not require a New Start Review. The remainder of the criteria (and some of the previous thirteen) affect task durations.

PROTOTYPE DESIGN

This section discusses system requirements, prototype design, and tool selections. System requirements are defined for an operational Intelligent Scheduling Assistant (ISA). The prototype design covers that subset of the system requirements deemed necessary to demonstrate the feasibility of the concept. Tools were selected based on system requirements and the prototype design.

The design goal for the Intelligent Scheduling Assistant was to improve the PM's ability to develop, maintain, and analyze program schedules. A conceptual view of an ISA is provided in Figure 1.

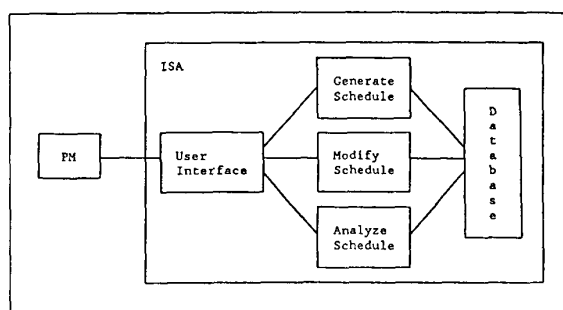


Figure 1. Conceptual View of ISA

The PM is the intended user of the system. The user interface allows the PM to direct schedule development, modification and analysis. Schedule generation builds a schedule by selecting tasks, durations, and relationships. Schedule modification changes tasks, durations, and relationships in an existing schedule. Schedule analysis computes schedule timing and suggests alternatives for resolving schedule conflicts. Separate knowledge bases could be used for schedule generation, schedule modification, and schedule analysis. The database maintains information for defense acquisition programs. A complete discussion of design principles for the prototype is given in Moran, 1988.

Prototype Design

The purpose of the prototype is to demonstrate the feasibility of an Intelligent Scheduling Assistant (ISA) by providing a user interface; I/O facilities; and methods for generating and modifying schedules. Actual schedule computation and drawing is provided by interfacing with existing project management software. ISA uses a database to maintain program information. A table of schedule data and a table of program characteristics are associated with each program. Schedule data consists of tasks (number, duration, description, start date, complete date, etc.) and relationships (links) between tasks. Program characteristics are parameters (dollar values, type of program, etc.) that are used to select tasks and compute task durations. All schedule development and modifications are done using working copies of the program information associated with the schedule.

ISA uses menus and explanatory prompts to guide the user through the scheduling process. Menu choices are limited to six options and color is used to enhance system appeal. Default responses (where applicable) are offered and acceptable responses are listed. ISA checks for errors and notifies the user of invalid responses.

ISA provides I/O utilities to load and save program information. The save utility allows the user to save program information and provides an option to export the data to CSNAS. The load utility makes working copies of program information and provides an option to import data from CSNAS.

ISA generates schedules using a rule set and the values of relevant program characteristics. ISA prompts the user for the values of program characteristics needed to generate a schedule. ISA uses rules to select tasks to be included in the program schedule based on the values of the program characteristics. The rules add the task to the schedule table, compute the task duration, and establish task links.

ISA modifies schedules based on changes to the values of program characteristics. The user changes program characteristic values as desired. Then ISA builds a new schedule using the new values and the procedure described above. ISA allows the user to modify existing schedules or create alternate schedules using this approach.

Tool Selection

Guru was selected as the tool for building the demonstration prototype. Waterman recommends evaluating expert-system building tools in terms of power and sophistication; support facilities; reliability; maintainability; and tool features when choosing a tool to use (Waterman, 1986:143). Guru's integrated package of knowledge processing capabilities (expert systems, database management, screen management, programming language, and text processing) clearly provide sufficient power and sophistication for development of ISA. Guru's support facilities (menu interface, interactive environment, help facilities, and trace facilities) speed the development and maintenance of ISA. Guru's reliability and maintainability are satisfactory. Guru's features are sufficient for ISA. It provides a forward-chaining inference engine to reason from program characteristics to schedule decisions, a database to maintain information for defense acquisition programs, and a programming language to implement the user interface.

CSNAS was selected to compute schedule timing and create schedule drawings. CSNAS allows data exchange using data files, is the primary scheduling tool used in ASD, and was used for the Phase I network. Guru and CSNAS are available in PC and VAX versions. Thus, use of these software tools makes the migration of the system to the AMS system possible if desired.

PROTOTYPE DEVELOPMENT AND EVALUATION

The prototype was implemented in a top-down fashion using Guru's structured programming, screen management, database management, and expert system facilities. Structured programming tools were used to build the framework of the prototype. Screen management facilities were used to enhance the appearance of prompts for user commands and data input. Database facilities were used to maintain defense acquisition program information. The CSNAS import/export facility was built using a combination of programming tools and database management. Expert system facilities were used to build two knowledge bases one to prompt the user for the values of program characteristics and one to generate the schedule.

Data Structures

Guru's database facilities are used to manage information for defense acquisition programs. Figure 2 illustrates the structure of the database. The prototype uses a program table to keep track of acquisition programs. A set of three tables is associated with each program. One table contains the program characteristic values.

The other two tables (one for tasks and one for links) contain schedule data. A separate set of working tables is used to load copies of program information. All data manipulation is performed using the copies in the working tables. Changes are saved from the working tables back to the appropriate program tables. The transfer table supports transfer of program information to and from ASCII data files. All data is stored in tables as strings. A full discussion of the table construction is given in Moran, 1988.

User Interface

A menu-driven interface was implemented to guide the user through the process of developing, modifying, and analyzing schedules. Prompts are used to explain system options and

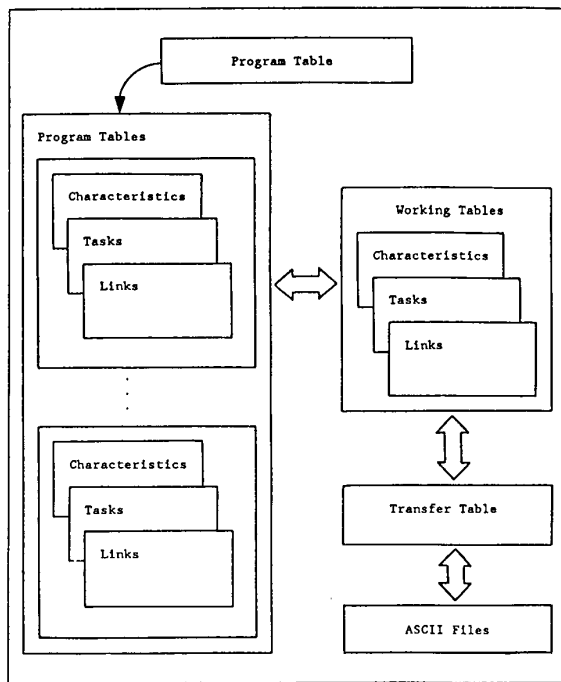


Figure 2. Database Structure

request data input. Color is used to improve the appeal of the prototype. All menus contain a maximum of six choices (including exit), and the user selects an option by typing a single letter. Equivalent operations (such as Show Data) in different menus use the same command. The prototype evaluates all inputs and advises the user of invalid selections.

CSNAS Interface

The prototype interfaces with CSNAS through an ASCII data file. This requires the user to export data to the ASCII file and exit to the operating system. Then CSNAS is started, and the ASCII file is loaded into CSNAS for schedule computations, reports, plot generation, etc. If desired, the user can save any changes made by CSNAS. The prototype can import the changed data for further manipulation such as performing schedule analysis. Although this process is undesirable for an operational system, it is sufficient to demonstrate that the system can interface with other software packages using data files.

Input of Program Characteristic Values

Rules are used prompt the user for the values of relevant program characteristics. The user is asked to confirm or change values of program characteristics whenever the source of the value is "DEFAULT". The source of the value is changed to "USER" upon user acceptance of the default or input of a new value. Rules are used to change the values of characteristics which are irrelevant based on user inputs for related characteristics. For example, the existence of a security classification guide is marked "NA" with a source of "RULE NOCLASS" when the user states that the program is unclassified.

Schedule Generation

The centerpiece of the prototype is a rule set for generating a schedule based upon program characteristic values. The flow of data between the working tables and the rule sets is depicted in Figure 3. Rule set "Get Characteristics" prompts the user for the program characteristic values and stores the responses in the working characteristics table. Rule set "Build Schedule" uses

the stored values to determine task applicability, durations, and relationships. Data for applicable tasks is stored in the working task and link tables.

The "Build Schedule" rule set builds the schedule from scratch. It begins with the start task and continues to add relevant tasks until the network is completed. The program characteristic values are used to determine if a task is needed and to calculate the duration of the tasks. The rules make links between the task being added and all of its immediate predecessors (tasks which must be completed before the new task begins).

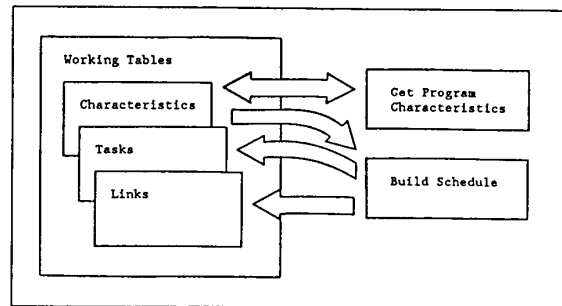


Figure 3. Data Flow Between Working Tables and Rule Sets

The prototype contains rules for 52 tasks. Up to 23 of these tasks may not apply to (be included in) a schedule for a specific program. Some tasks are mutually exclusive. For example, Justification and Approval Activities and Source Selection activities will never occur in the same network. Many tasks are included or omitted as a set. For example, five tasks are related to Source Selection. These complex inter-relationships make it difficult to determine the exact number of potential schedules which could be generated. A conservative estimate is that the prototype could generate up to 8000 different networks considering only the presence or absence of tasks. More than 10,000 different schedules could be generated when taking into account the permutations created by different task durations.

The prototype creates the start task and works forward to the finish task. New tasks are added when all of their predecessors have been created. This process continues until a complete schedule is generated. This mimics one way that a person might build a schedule. One advantage of this approach is that the first task is always 10, the last task is always the largest task number, and the magnitude of the task number gives an indication of where the task occurs within the topology of the network.

The key parts of a rule are the premise clause (IF statements) and the action clause (THEN statements). When the conditions in the premise clause are true, Guru executes the statements in the action clause. The premise of rules for tasks which will always occur are written so that the rule will always fire (execute THEN statements) at the appropriate time during schedule development. The premise of rules for tasks which may not occur are written so that the rule will fire only when the task applies to (should be included in) the schedule being generated. This process is best illustrated by considering an example rule in the prototype.

Figure 4 shows the rule for a task which may not apply to a schedule for a specific acquisition program. Variable "vthreat" contains the value for the program characteristic "SYSTEM THREAT". If a threat does not exist, this rule will never execute, and the "THREAT INPUT" task will not be included in the network. If a threat does exist, the rule will not fire until variable "tfpmd" is KNOWN (given a value). Variable "tfpmd" is used to store the task number for the task "FINAL PMD". Thus, this rule fires after the "FINAL PMD" task has been created only if a system threat exists. If the premise is true, a new task is created and given the next task number. The duration, description, and OPR are changed as shown, and

variable "tthreat" is assigned the task number for "THREAT INPUT".

```

IF: vtthreat = "YES" and KNOWN(tfpmd)
THEN: PERFORM ADDTASK
      CURRTASK.DUR = " 23"
      CURRTASK.DESCR = "THREAT INPUT"
      CURRTASK.OPR = "PM & FTD"
      tthreat = CURRTASK.ID
      ATTACH 1 TO CURRLINK
      CURRLINK.PRED = tfpmd
      CURRLINK.SUCC = ttthreat

```

Figure 4. Rule for "THREAT INPUT"

The final three lines create the link between "THREAT INPUT" and "FINAL PMD" by adding a new record to the link table (CURRLINK) and changing the values of the predecessor and successor fields. Note that this approach only makes links to predecessors when the rule is executed. The rule only contains knowledge about what the task depends on. It contains no knowledge about what follows the task. Other rules presenting more complex aspects of the network model are discussed in Moran, 1988. Each task in the network is handled by a single rule. The premise of the rule is written to create the task at the appropriate point in schedule development. Tasks which always exist have premises which are always true at the appropriate point in schedule development. Tasks which may not occur have premises that are true only when program characteristic values indicate that the task should be included in the network. The actions of the rule create the task, calculate task duration using program characteristic values, and make links to all immediate predecessors.

This process makes each rule relatively independent of the other rules. The only dependencies are those created by task links. Thus, all task data (duration, description, OPR, schedule dates, etc.) can be changed within the action clause of a rule without affecting other rules. When adding or deleting rules, the developer/maintainer need only concern himself with the task involved and its immediate predecessors and successors. If a task is deleted (the rule for the task removed from the rule set), the developer/maintainer must insure that its immediate predecessors still have successors and its immediate successors still have predecessors. When adding tasks (creating a new rule), the developer/maintainer must create the premise so that the task will be created at the appropriate point in the network. The action must create the task, calculate task duration, and make links to preceding tasks. Succeeding tasks must be modified to insure that proper links are made to the new task.

Schedule Modification

A rudimentary facility for modifying an existing schedule by changing program characteristic values was implemented. The user can create a new schedule (or simply modify the existing one) by changing program characteristic values. This lets the user generate multiple versions of a program schedule by changing key program characteristic values. Thus, alternative approaches can be evaluated by generating a schedule for each alternative. For example, schedules might be generated for both sole source and competitive acquisitions. The two schedules could then be compared to see which is better given the risk involved and other appropriate factors.

Evaluation

The prototype accepts more than six trillion possible combinations of program characteristic values and can generate thousands of schedules. Testing of all possible inputs is impossible. Thus, the system was evaluated in two ways. First, tests were run to increase the confidence that the system would generate a schedule regardless of the combination of program characteristic values. Then the prototype was demonstrated to potential developers and users of an operational system. Each person who witnessed a demonstration was asked to evaluate the prototype by answering a questionnaire.

Prototype Testing

The system was implemented using modular components. Each component/collection of components was tested as it was built. The user interface was tested (using stubs for each option) by selecting every option available. The CSNAS interface was tested by exporting and importing data files. Exported data files were loaded into CSNAS and tested for compatibility.

The rule set for input of program characteristic values was

tested by using it to prompt for program characteristic values and reviewing the stored data to see that it matched the inputs. Testing the rule set for schedule generation proved to be more difficult. Thirteen program characteristics (containing 49 choices) directly affect the existence of 23 tasks. The remaining program characteristics affect the duration of tasks. Initial efforts were spent on generating a schedule based on default program characteristic values. Then, program characteristic values which affected task applicability were changed and new schedules generated. Finally, random input of program characteristic values was used.

Approximately twenty schedules were generated using different program characteristic values. When the prototype failed to generate a complete schedule, Guru's trace facility was used to locate the problem. Proper schedules were not generated for two reasons. Some tasks were incorrectly omitted due to an incorrect premise statement for the rule. These errors were corrected by modifying the premise. Some tasks were missing a predecessor and/or a successor. These errors were corrected by modifying the actions of appropriate rules to insure that links were established. Most problems were located and fixed within an hour.

The rule set for schedule generation was also evaluated by manually reviewing each rule. Rules for tasks that always occur were reviewed to insure that their premises were always true at the proper point in schedule development. Rules for tasks that may not occur were reviewed to insure that they would execute only when the task should be included in the network. Finally, cross-checks were made between rules to insure that proper links would be made to create a complete network regardless of which tasks were used.

User Evaluation

The prototype was demonstrated to 18 people. Schedules were generated, modified, and exported to CSNAS to demonstrate system operation. Plots of five schedules generated by the system were shown to the evaluators following system operation. Everyone was asked to complete an evaluation questionnaire after the demonstration was completed. Figure 5 summarizes the responses to the first four questions.

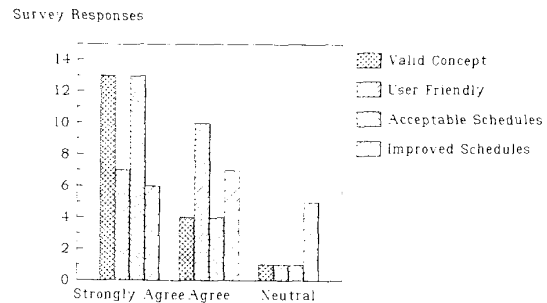


Figure 5. Summary of Primary Responses

Evaluations were completed by people from the Aeronautical Systems Division (ASD), the Air Force Acquisitions Logistics Command (AFALC), and The Applied Science Corporation, Incorporated (TASC). TASC provides scheduling support to ASD. Sixteen of the eighteen evaluators (88 percent) had used network scheduling more than three times, had used automated project management tools (primarily CSNAS), and had used model networks. 100 percent of the respondents believe that network scheduling has merit. The prototype received no negative comments. Table 1 cross-tabulates validity of the concept with the degree of schedule improvement.

CONCLUSIONS

The prototype successfully demonstrates the feasibility of building ISA. User evaluations were consistently favorable. 95% of the evaluators recommended development of an operational system or further research. Although the prototype's user interface was limited to menus and a few verbose

		Improvement is Significant		
		Strongly Agree	Agree	Neutral
Concept	Strongly Agree	61%	11%	0%
	Agree	5%	11%	5%
	Neutral	5%	0%	0%

Table 1. Cross-tabulation of Concept and Improvement

prompts, 95% of the evaluators rated the potential for user-friendly operation as good to excellent. 65% rated the quality of schedules generated by the prototype as reasonable. 95% rated the prototype as an improvement to model networks.

The lack of a consensus on the quality of schedules generated by the prototype is derived from two factors. First, most of the people who evaluated the system did not examine the schedules closely. Second, schedule quality is difficult to quantify. However, schedules generated are meant to be recommendations which must be reviewed and approved by members of the program management team. Therefore, the absence of negative evaluations is significant considering the limited knowledge of the prototype and the intended use of the results.

Cross-tabulation of concept and improvement shows strong support for both metrics. More than 65% of the evaluators recommended development of an operational system and agreed that the prototype was much better than existing model networks.

The concept is translatable to other network schedules using the procedures outlined in this paper. The first step is to identify all tasks which may be included in a given schedule. Each task is analyzed to determine when it applies, how durations are estimated, and what relationships exist. A single rule can be created to handle each task. The premise of the rule is written so that the task is created at the appropriate time during schedule development. The actions of the rule compute task durations and establish links to immediate predecessors. This process can be used to generate schedules without prior knowledge of the final network topology. Furthermore, only the predecessors of tasks need to be defined.

Schedule generators would be easy to build for an industry like construction where tasks, durations, and relationships are well-defined. The quality of the generated schedules would probably be very high under such circumstances. Schedule generators can be built for other model networks and where no previous work has been done to generate schedules. The time required to build the initial prototype would range from one to two months for a network with fifty tasks.

Interfaces can be written to export schedule data to other software programs. The same technique used to create the CSNAS data file can be used to export data in an appropriate format for database management systems, other project management tools, or other software program which allow data interchange.

Schedules generated using this approach are only as good as the knowledge contained in the system. All generated schedules should remain subject to review by members of the program management team. The knowledge in the system must be continually reviewed and refined. The time to reach maturity is unknown.

RECOMMENDATIONS FOR FURTHER RESEARCH

The prototype was designed to demonstrate the feasibility of generating schedules using a knowledge-based approach. However, schedule generation is only one part of the scheduling problem. A significant amount of effort is needed to maintain schedules. Thus, methods for modifying schedules by adding, deleting, or modifying tasks are essential. Analysis of schedules is another key element to successful use of

network schedules. Thus, research into resolving schedule conflicts using knowledge-based techniques is needed. Adding lines of reasoning to the current system would be valuable to people who review schedules generated by the prototype. Finally, automated procedures for generating rules for ISA would be beneficial.

Modifying Tasks

Users will quickly become disenchanted with any computer program which does not allow them to easily modify schedules which are generated. Methods for adding, deleting, or modifying tasks in the schedule are essential to successful implementation of an operational ISA. Although such changes can be made using CSNAS or other project management tools, this is not an acceptable solution in the long term.

The user should be allowed to add any task which is deemed relevant to his program. These may be tasks that should ultimately be added to the rule base for schedule generation, or they may be tasks that are peculiar to the program in question. The user should be allowed to delete tasks which are deemed irrelevant to the program. The rule base for schedule generation may require modification to account for the cause of the deletion, or the deletion might be peculiar to the program. The user should be able to modify task data such as description, duration, and schedule dates. The user should also be able to restructure the network.

Analyzing Schedules

System requirements include computation of schedule timing and resolution of conflicts. The ability to compute schedule timing without exiting the system is a needed improvement. An ability to recommend alternatives to resolve schedule conflicts would dramatically increase the usefulness of the system. Alternatives for resolving schedule conflicts could be generated using a rule set. Consider the case where the schedule fails to meet its target date for completion. The system could examine the network to determine which tasks lie on the critical path. Rules could be used to generate recommendations for reducing the schedule. The recommendation would be accompanied by an explanation which included an assessment of the risk to the program if the change is made.

Providing Reasoning

A record of the lines of reasoning used to generate a schedule would be useful to people who review the schedule. The reviewer could refer to the line of reasoning whenever he does not understand why a task was included (or omitted), how a duration was estimated, or how relationships were established. If the reviewer disagreed with the line of reasoning, then he would contact the person responsible for the schedule to recommend changes.

Automating Rule Generation

The rules contained in the Intelligent Scheduling Assistant are fairly simple to construct. Automating rule generation would facilitate the development of rule sets for other model networks. The system could prompt the user for all possible tasks and knowledge needed to decide the applicability, duration, and links for each task. The system could use this knowledge to generate rules for schedule development.

BIBLIOGRAPHY

- AFALC. CSNAS User Guide. Air Force Acquisition Logistic Command, Wright-Patterson AFB, OH, 15 July 1987.
- ASD/AE. POINTS Schedule and Planning Handbook. Deputy for Aeronautical Equipment, Wright-Patterson AFB, OH, 15 April 1987.
- Clark, Albert L. CSNAS Training Course. Air Force Acquisition Logistic Command, Wright-Patterson AFB, OH, 11-12 August 1987.
- Fersko-Weiss, Henry. "Master Plans: Project Management Software," PC Magazine, 6/16: 153-209 (September 29, 1987).
- Fox, B. R. and K. G. Kempf. "Complexity, Uncertainty and Opportunistic Scheduling," Proceedings of the Second Conference on Artificial Intelligence Applications, 487-492. Washington, D.C.: IEEE Computer Society Press, 1985.

- Fox, Mark S. Constraint-Directed Search: A Case Study of Job-Shop Scheduling. PhD dissertation. Carnegie-Mellon University, Pittsburg, PA, December 1983, (AD-A138-307).
- Fukumori, Koji. Fundamental Scheme for Train Scheduling (Application of Range-Constriction Search). Contract N00014-75-C-0643. Report AI-M-596. Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, September 1980 (AD-A095-522).
- Gansler, Jacques S. "Defense program instability: causes, costs and cures," *Defense Management Journal*, 22: 3-11 (Second Quarter 1986).
- Gazdik, Igor. "Fuzzy-Network Planning FNET," *IEEE Transactions on Reliability*, R-32,3:304-313 (August 1983).
- Hayes-Roth, Barbara and others. PLANNER'S WORKBENCH: A Computer Aid to Replanning. Report RAND/P-6688. Rand Corporation, Santa Monica, CA, October 1981 (AD-A113-331).
- Hayes-Roth, Barbara and others. Modeling Planning as an Incremental, Opportunistic Process. Contract N00014-78-C-0039. Report RAND/N-1178-ONR. Rand Corporation, Santa Monica, CA, June 1979 (AD-A071-035).
- Moran, Jerry L. "ISA: A Prototype Intelligent Scheduling Assistant for Defense Acquisition Management," Masters Thesis, Department of Electrical and Computer Engineering, Air Force Institute of Technology, March 1988.
- MDBS. GURU Reference Manual Volume 1. Micro Data Base Management Systems, Lafayette, Indiana, May 1987.
- Newman, P. A. and K. G. Kempf. "Opportunistic Scheduling for Robotic Machine Tending," *IEEE 1985 Second Conference on Artificial Intelligence Applications*, 168-175. Washington, D.C.: IEEE Computer Society Press, 1985.
- Prerau, David S. "Selection of an Appropriate Domain for an Expert System," *The AI Magazine*, VI/2:26-30 (Summer 1985).
- Sacerdoti, Earl D. "The Nonlinear Nature of Plans," *Advance Papers of the Fourth International Joint Conference on Artificial Intelligence*. 206-214. Los Altos, CA: William Kaufmann, Inc. September 1975.
- Sage, Andrew P. and Chelsea C. White, III. "ADRIANE: A Knowledge-Based Interactive System for Planning and Decision Support," *IEEE Transactions on Systems, Man, and Cybernetics*. SMC-14,1:35-47 (January/February 1984).
- Scanlon, Kathryn M. and Harold J. Schutt. The Program Manager's Support System (PMSS) An Executive Overview and Systems Description. Defense Systems Management College, Fort Belvoir, VA, 1 January 1987.
- Tate, A., "Generating Project Networks," *Proceedings of the 5th International Conference on Artificial Intelligence*, 888-893. Cambridge, MA: IEEE Press, August 1977.
- Vere, Steven A. "Planning in Time: Windows and Durations for Activities and Goals," *IEEE Transactions on Pattern Analysis and Machine-Intelligence*, PAMI-5,3:246-267 (May 1983).
- Waterman, Donald A. A Guide to Expert Systems. Reading, MA: Addison-Wesley Publishing Company, 1986.
- Wilkins, David E. and Ann E. Robinson. An Interactive Planning System. Contract F49620-79-C-0188, N00014-80-C-0300. Report TN-245. Artificial Intelligence Center, SRI International, Menlo Park, CA, July 1981 (AD-A103-019).
- Winston, Patrick Henry. Artificial Intelligence. Reading, MA: Addison-Wesley, 1977.
- Woffinden, Duard S. EENG 543, Software Project Management. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, November 1987.
- Zornes, Scotty E. Program Analyst. Personal interview. Aeronautical Systems Division, Air Force Systems Command, Wright-Patterson AFB, OH, 25 August 1987.

ABOUT THE AUTHORS

Jerry L. Moran is received the degree of Bachelor of Science in Electrical Engineering from Kansas State University in May 1979. A recipient of the George C. Marshall award and Top 5 Percent Fellowship award, he was commissioned in the Regular Army in May 1979 through the ROTC program. He served three years as a platoon leader, company executive officer, and battalion intelligence officer with the 12th Engineer Battalion in West Germany. Following attendance at the Teleprocessing Operations Officer Course at the Air Force Institute of Technology (AFIT), he was assigned to the United States Army Electronic Proving Ground (USAEPG), Fort Huachuca, Arizona. Besides serving as a Software and Evaluation Test Officer and Company Commander, he helped establish an artificial intelligence capability at USAEPG. Captain Moran returned to AFIT to complete his MS degree in Electrical Engineering in March, 1988 and is currently with the Program Executive Office for Networks at Fort Monmouth, New Jersey.

Wade H. Shaw, Jr. is an Assistant Professor of Electrical Engineering at the Air Force Institute of Technology in Dayton, Ohio. He completed his Ph.D. in Engineering Management at Clemson University and has published numerous research papers on a variety of subjects including Systems Modeling, Decision Support Systems, Simulation, Software Engineering and Project Management. Dr. Shaw is a Senior Member of the Institute of Electrical and Electronics Engineers and a member of the Institute of Industrial Engineers, the Decision Sciences Institute, The Institute of Management Science, the Operations Research Society of America and the Society for Computer Simulation. He is a registered Professional Engineer in Ohio and South Carolina and has been selected to the rank of Major in the US Army Signal Corps.