KCP--613-4336

DE91 005254

Distribution Category UC-705

# ISAPS - INTELLIGENT SCHEDULING AND PLANNING SYSTEM

M. S. King

Published August 1990

Manuscript submitted to Design Productivity Institute Conference, Rolla, Missouri. Conference held at Honolulu, Hawaii, February 5, 1991.

WM

Technical Communications
Kansas City Division

Allied-Signal Aerospace Company

Allied Signal

# ISAPS – Intelligent Scheduling And Planning System

M.S. King, W.C. Rutherford, J.V. Grice, Allied-Signal Inc., Kansas City Division*
K.L. Kessel, M. Orel, IntelliCorp, Inc.**

## ABSTRACT

ISAPS is a scheduling and planning tool for shop floor personnel working in a Flexible Manufacturing System (FMS) environment. The ISAP system has two integrated components: the Predictive Scheduler (PS) and the Reactive Scheduler (RS). These components work cooperatively to satisfy the four goals of the ISAP system, which are: G1) meet production due dates, G2) maximize machining center utilization, G3) minimize cutting tool migration, and G4) minimize product flow time. The PS is used to establish schedules for new production requirements. The RS is used to adjust the schedules produced by the PS for unforeseen events that occur during production operations. The PS and RS subsystems have been developed using IntelliCorp's Knowledge Engineering Environment (KEE), an expert system development shell, and Common LISP. Software Quality Assurance (SQA) techniques have been incorporated throughout the development effort to assure the ISAP system meets the manufacturing goals and end user requirements.

## INTRODUCTION

The Kansas City Division of Allied-Signal Inc. is in the process of implementing a Flexible Manufacturing System (FMS) which will produce discrete, machined electrical component housings. The FMS represents the first fully automated machining system to be installed at the Kansas City Division (KCD). The system includes six four-axis horizontal machining centers, two coordinate measuring machines (CMM), one automatic wash station, two automated guided vehicles (AGV), four work in process pallet pools, and five manual stations for loading and unloading parts, building fixtures, and reviewing product that fails inspection (see Figure 1). The system is controlled by five computers networked together using Ethernet/DECnet. The FMS is scheduled to begin operation in December 1991.

The current production scheduling system for the shop floor is largely manual input, labor intensive, and without timely feedback. Decisions about product mix, priority of jobs, and the allocation of finite manufacturing resources are all made with limited knowledge of the impact on production schedules, resource utilization, and overall operating efficiency. In the environment of an integrated, highly interdependent manufacturing system such as the FMS, the volume of information and knowledge required to schedule resources has the potential to quickly overwhelm the individual responsible for decision making.

The ISAP system is being developed to provide scheduling and planning personnel an integrated tool to assist them with the development of production plans, performing "what-if" scenarios off-line, and to shorten response and recovery time from unforeseen events. The work reported in this paper covers the development of the prototype ISAP system, the architecture and technologies employed in the system, the constraints and heuristics considered, how schedules are generated, the user interface, and the current status of the system.

---

* Operated for the United States Department of Energy under Contract Number DE-AC04-76-DP00613

** Under a Subcontract with Allied-Signal Inc.

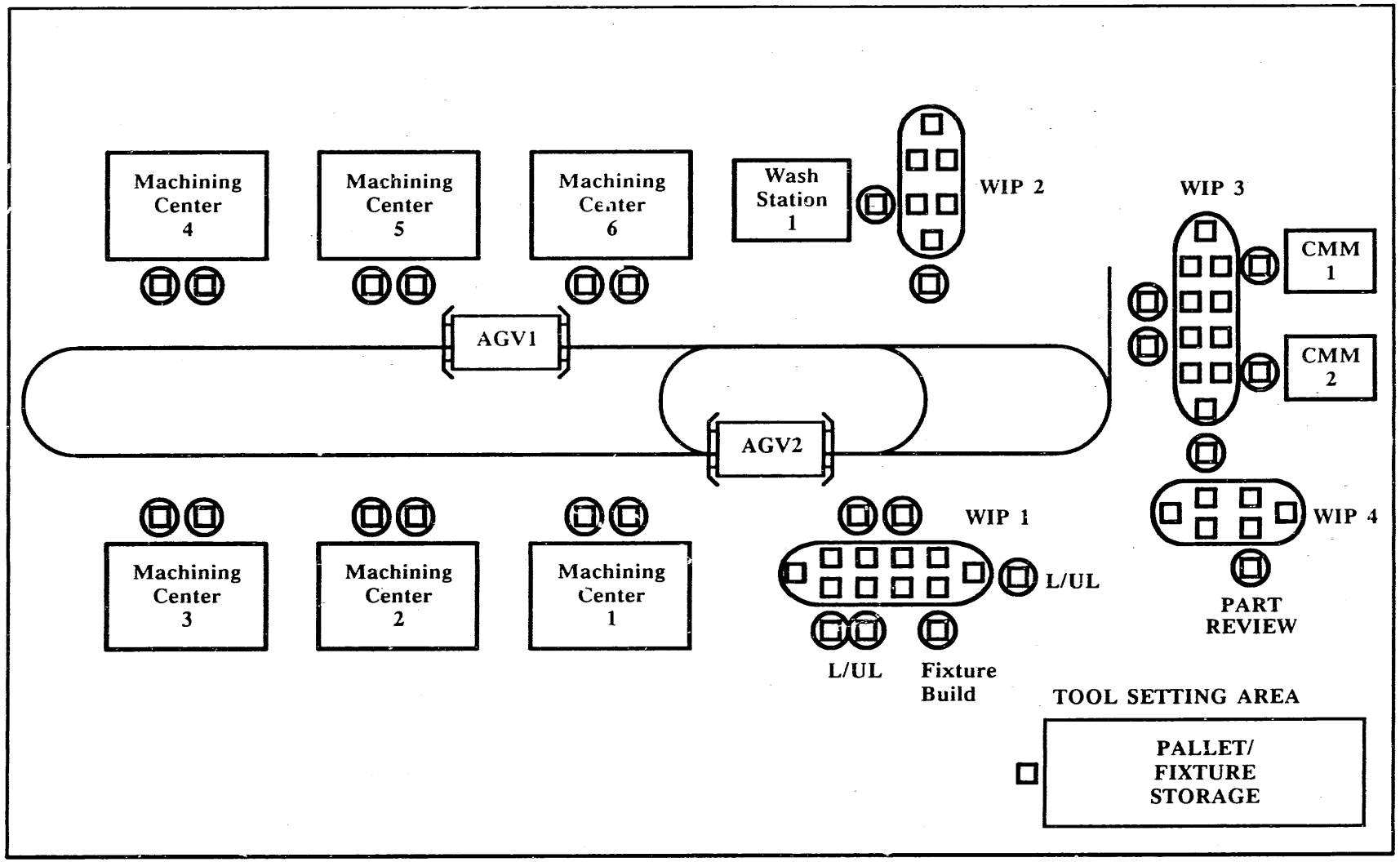


Fig. 1 FMS Schematic Diagram

## THE DEVELOPMENT PROCESS

The system development methodology being used is a hybrid of the classical waterfall or cascade lifecycle approach plus a rapid prototyping technique typically found in expert system development environments. SQA requirements (design reviews, audits, code walkthroughs) are an integral part of the development process.

A project team approach was selected to manage the development activities. The team members and their roles are as follows:

KCD Personnel

Project Manager - Provides technical guidance for the timely completion of the project and is responsible for the development of a system that meets the needs of the end users;

Technical Consultant - Knowledgeable of operational issues related to the FMS (a simulation and programming expert);

End Users - Provide scheduling and planning expertise and assist with the design of the user interface.

SQA Administrator - An independent agent whose function is to assure that the system being developed meets the requirements set forth in the Scope of Work for ISAPS. Also leads the review and audit meetings.

IntelliCorp Personnel

Project Manager - Provides guidance for the timely completion of the project and is responsible for the user interface, documentation, and user training;

Knowledge Engineer - Responsible for knowledge acquisition, coding, testing the system, and technical training;

Technical Consultant - Reviews the project goals and assists with the selection of technologies that lend themselves to meeting the project objectives.

The development process used for ISAPS contains seven steps. The steps are:

1. System Requirements Analysis,
2. Functional Specification Preparation,
3. Detailed System Design,
4. Coding,
5. Unit and Integration Testing,
6. Acceptance Testing,
7. Technology Transfer and User Training.

Rapid prototyping occurs throughout the process to explore solution techniques and to keep the end users involved as the system is being developed.

## THE ISAP SYSTEM ARCHITECTURE

The scheduling and planning system is composed of two major components: the Predictive Scheduler (PS) and the Reactive Scheduler (RS). These components are accessible through a single user interface that is graphical in nature with mouseable icons and

pop up menus. The PS and RS are enhanced versions of the Scheduler Booster Module, an IntelliCorp, Inc. product that runs on top of KEE. The primary method for viewing the schedules produced by ISAPS is through the ActiveGantt Booster Module, also an IntelliCorp product. Figure 2 illustrates the relationship of the various components of ISAPS.

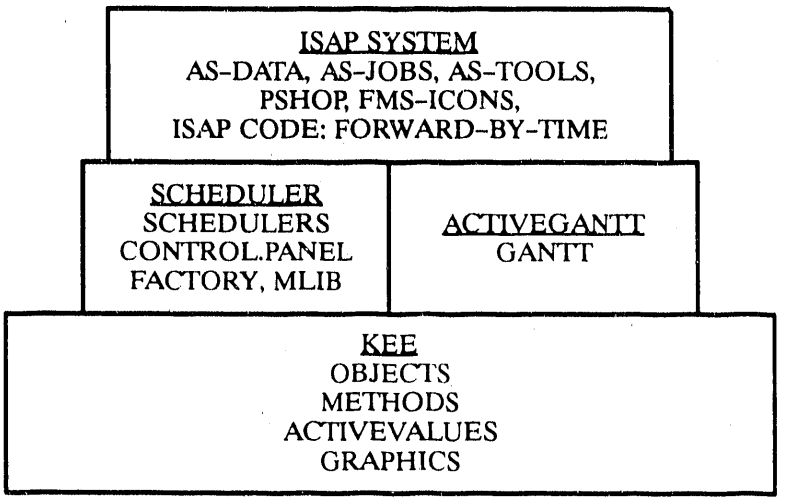


Fig. 2 Components of the ISAP System

The PS is used to establish schedules for new production requirements on a variable planning horizon. The RS is used to adjust the schedules produced by the PS for unforeseen events that occur during production operations, such as equipment failures, changing priorities, and product mix. A common model of the FMS is employed by the PS and RS which defines the basic system configuration and availability of resources to be considered for scheduling.

The data required to create the FMS model knowledge bases (KBs) are supplied from external files, as are the data for the jobs to be scheduled, the tooling, and fixtures to be used for the planning and scheduling session. Any of this data may be entered by the users directly and any loaded knowledge base may be modified through the user interface. The strategy of designing the ISAP system to be data driven from external files was chosen early in the project to provide maximum flexibility during development and to separate code from data for ease of maintenance.

There are ten KBs used in the ISAP system (refer to Figure 2). The Scheduler Booster Module works with the KBs SCHEDULERS, CONTROL.PANEL, FACTORY, and MLIB. The SCHEDULERS KB contains scheduling strategies, heuristics, reporting capabilities, and modifications specific to the ISAP system. The CONTROL.PANEL KB contains information to build the user interface to the ISAP system. The FACTORY KB contains the factory objects, including fixtures, jobs, operations, tools, and other FMS resources. The MLIB KB is a working copy of FACTORY. The ActiveGantt Booster Module uses the GANTT KB to create the Gantt chart of the proposed schedule for the user.

The remaining KBs are specific to the ISAP system. AS-DATA is the static or baseline KB which represents the unaltered state of the FMS at the beginning of a scheduling session. PSHOP is the working copy of AS-DATA used during scheduling. The AS-JOBS KB contains information about the jobs to be scheduled and the AS-TOOLS KB contains cutting tool information. The FMS-ICONS KB contains graphical information which is used to create a two dimensional view of the FMS called FMS.VIEW. FMS.VIEW is available through the user interface and is an alternative means of viewing schedule information.

## CONSTRAINTS AND HEURISTICS

The ISAP system is being developed to achieve four goals: G1) meet production due dates, G2) maximize machining center utilization, G3) minimize cutting tool migration, and G4) minimize product flow time. The design of the ISAP system needed to account for a number of physical limitations or constraints imposed by the FMS design to accomplish the stated goals.

### Constraints

The first constraint the scheduling system has to consider is the number of machining centers and their availability. Machines in the FMS will have routine preventative maintenance performed. Machining centers and CMMs will be off-line at regular intervals for program testing and process prove in. These two conditions reduce the availability of the machines, which impacts goals G1, G2, and G4. ISAPS considers downtimes during the scheduling process.

The second constraint is the space available on each machining center for cutting tools. Cutting tools to be used in the FMS have been grouped into toolkits, with each toolkit capable of producing a finite number of part operations. All tools for all part operations cannot reside in the FMS at the same time. As the product mix changes, machining centers must be retooled or sit idle until the product mix changes again, making them available for scheduling. This constraint impacts G2 if the machine sits idle waiting for jobs, G3 if the toolkits are swapped out frequently, and potentially G1 and G4 due to reduced availability of the machining centers.

The third constraint deals with fixtures and pallets. For each part operation to be scheduled, only one fixture will be available. In addition, the maximum number of pallets available to hold fixtures is less than the number of fixtures required to perform all part operations. This constraint implies that as the product mix changes, fixture/pallet combinations will have to be broken down and rebuilt, impacting G1, G2, and G4 by reduced availability of the required fixtures.

The preceding discussion points out that the four goals of the ISAP system are in conflict with one another. Therefore, a strategy needed to be developed that would address meeting as many of the goals G1 through G4, while violating as few of the constraints, as possible. In the remainder of this section, the heuristics used by the ISAP system will be presented. In the following section, the method by which the heuristics are applied to generate a schedule that satisfies the goals of ISAP will be presented.

### Heuristics

Three heuristics have been identified as the most likely candidates to meet the goals of the ISAP system. The heuristics are MINDOWN (MINimum DOWNtime), SLACK (job SLACK time), and PRIORITY.

The MINDOWN heuristic finds the job that will cause the minimum downtime or minimum time a machine has to wait for the job. This translates roughly into finding the job with the earliest start time for a given machine.

The SLACK heuristic chooses jobs with the least amount of slack time available. Slack time in this context is the difference between a job's due date and the maximum of the time a fixture will be available to run that job or the time that the job's last operation was completed. If a job has low slack time, it will tend to be selected earlier for scheduling.

The PRIORITY heuristic is simply a numeric value between 1 and 100. All jobs default priority is 50. Hot jobs and those that are behind schedule will eventually have their priority increased so they are chosen sooner for scheduling.

The default order of precedence for the three heuristics is MINDOWN, SLACK, then PRIORITY. This means that if two jobs are available for scheduling that create the

same downtime for a machine, the one with the least slack time is selected by the SLACK heuristic. If the jobs have identical slack times, then the job with the highest priority is selected. If a tie still remains, the first job considered is selected. It should be noted that as soon as a distinction has been made between two jobs, the application of the heuristics stops. The default ordering of the heuristics can be changed through the user interface.

## SCHEDULE GENERATION

The discussion that follows depicts how ISAPS generates a schedule for the FMS by applying the heuristics described above and other LISP based functions. The FORWARD-BY-TIME strategy is the basis for the Predictive Scheduler in ISAPS. Figure 3 should be referred to for visualization of the process flow.

Jobs are sequences of part operations in the FMS. This means that if a part makes three passes through the FMS, then one of those passes is a job. ISAPS uses a different definition of job. For ISAPS, a job is all of the operations done to one part. Thus, all three passes of a part through the FMS is called a job. Each operation included in a job (Loads, Machining, Unload) is then referred to as a part operation, operation, or activity.

FORWARD-BY-TIME first prioritizes the jobs by calling PRIORITIZE-JOBS with a list of the jobs to be scheduled. PRIORITIZE-JOBS first creates job slack times and then function JOB-SORTS sorts the jobs based on the heuristics: MINDOWN, SLACK, and PRIORITY. A list of prioritized jobs is returned.

The next significant operation is to obtain a prioritized list of all FMS machines. Prioritizing machines is done with the function, PRIORITIZE-MACHINES. PRIORITIZE-MACHINES sorts the machines based on their time availabilities. Machines that are available earliest are given highest priority.

At this point, FORWARD-BY-TIME has a list of prioritized jobs and prioritized machines. It then selects the highest priority machine and attempts to assign a job to that machine. To assign a job to the highest priority machine, the function SELECT-BEST-ALLIED-JOB is called.

SELECT-BEST-ALLIED-JOB starts by obtaining a job, or jobs, that can be performed on the machine. GET-DOABLE-JOBS selects jobs according to whether the machine has the proper tooling to do the job and whether the job will be waiting for the machine. If the machine does not have the correct tooling or must wait for the job, then the job is included as a doable job only if there are no other jobs that were found that cause less waiting. GET-DOABLE-JOBS returns a list of doable jobs, which may be an empty list (no doable jobs found).

SELECT-BEST-ALLIED-JOB then enters a conditional statement that first asks if no doable jobs could be found. If none were found, then the function FIND-BEST-JOB is called which finds doable jobs given a toolkit change is made.

FIND-BEST-JOB initially calculates the amount of time it will take for the machine to become available after it has had a toolkit change. This time represents the new machine availability time. FIND-BEST-JOB then calls the function BEST-JOB-FOR-MACHINE which attempts to find a new best job based on the new machine availability time.

BEST-JOB-FOR-MACHINE loops through the jobs attempting to find a doable job with a start time that is later than the new machine availability time. To find a doable job, all machines with tools required for that job must be found. A conditional statement is entered that asks if there were any machines that can do that job. The two alternatives are

1. There are no machines with tools that can do that job. A second conditional statement is entered. It determines if there are any jobs that are available before the

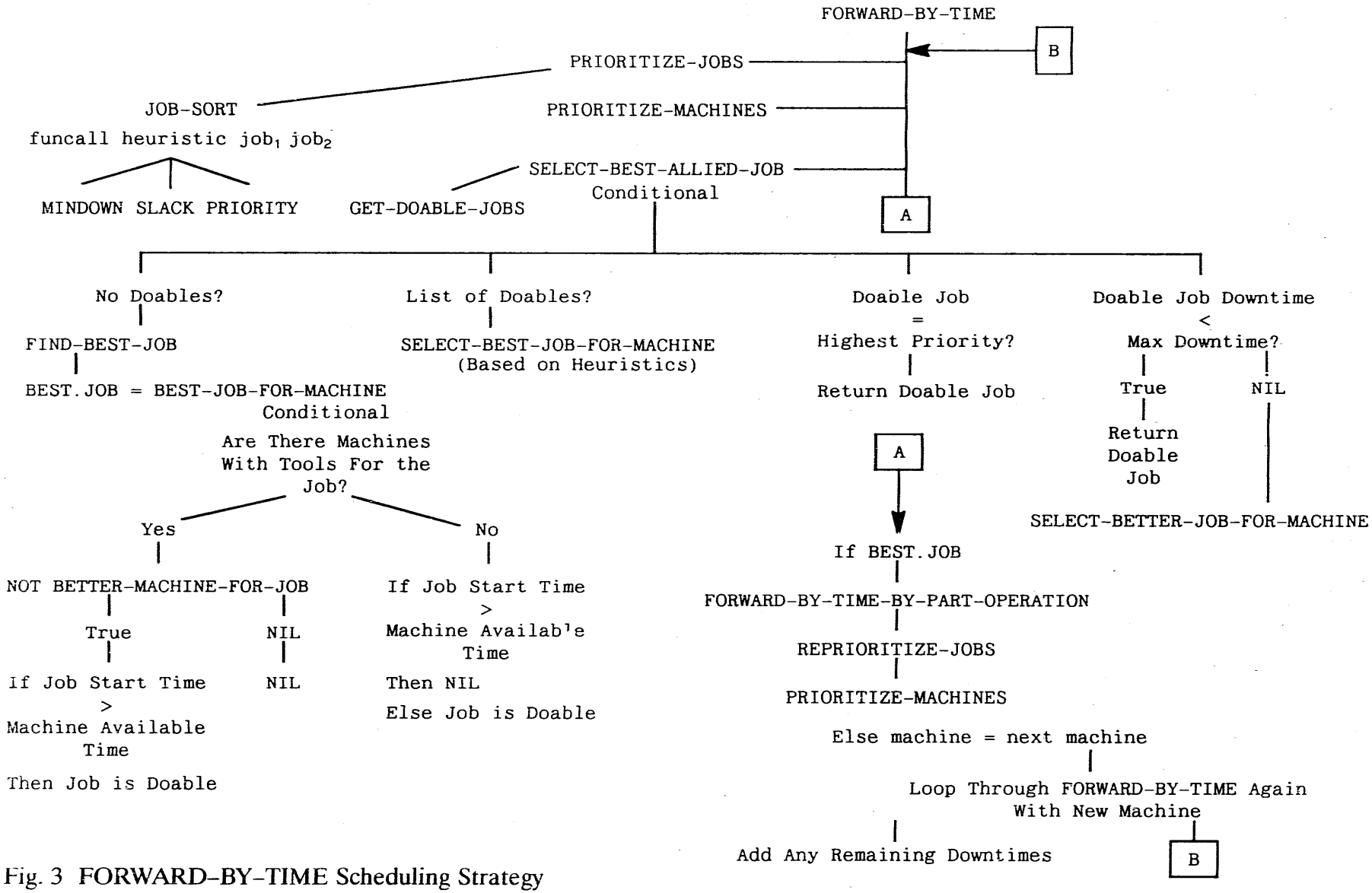


Fig. 3 FORWARD-BY-TIME Scheduling Strategy

new machine availability time. If there are such jobs, then these jobs are added to the list of doable jobs, and a toolkit change is required. If there are no jobs that can be started prior to the new machine availability time, then the first job encountered that can be started after the new machine availability time is returned.

2. There are one or more machines that can do that job. The function asks if any of those machines could do the job sooner than the machine with the new availability time. If there is no better machine to do that job, and if the job does not have to wait to be performed on the machine with the new availability time, then the job becomes a doable job.

The function BEST-JOB-FOR-MACHINE then enters a final conditional statement (not shown). If there are one or no doable jobs, then BEST-JOB-FOR-MACHINE returns with either that job or an empty list. If the function has found several doable jobs, then the function SELECT-BEST-JOB-FOR-MACHINE is called with the machine and the list of doable jobs.

SELECT-BEST-JOB-FOR-MACHINE selects the best job out of the doable jobs based on all but the highest priority heuristic. This approach is used because the highest priority heuristic will already have been used to prioritize the jobs. The next priority heuristic is then invoked to break ties among the doable jobs.

Returning from BEST-JOB-FOR-MACHINE assigns the doable job to the variable BEST.JOB in function FIND-BEST-JOB. If BEST.JOB has a non-nil value, then the machine in question is retooled with the toolkit that can perform that BEST.JOB and also perform the greatest number of different jobs.

BEST.JOB is then returned to SELECT-BEST-ALLIED-JOB, which subsequently returns to FORWARD-BY-TIME.

The next possibility that the function SELECT-BEST-ALLIED-JOB deals with is if there originally were doable jobs for the highest priority machine. In this instance, SELECT-BEST-JOB-FOR-MACHINE would have been called directly. This selects the BEST.JOB based on the remaining heuristics and would return that job to FORWARD-BY-TIME.

The next conditional statement alternative within SELECT-BEST-ALLIED-JOB is to ask if the doable job is the highest priority job. If the doable job equals the highest priority job, then this job is returned to FORWARD-BY-TIME.

The last alternative of the conditional statement asks if the doable job's start time causes more machine downtime than is acceptable (currently 30 minutes). If not, then the doable job is returned to FORWARD-BY-TIME; otherwise, the job causes too much downtime, and the function SELECT-BETTER-JOB-FOR-MACHINE is called.

SELECT-BETTER-JOB-FOR-MACHINE obtains (1) the doable job's start time minus the maximum downtime. Since the job's start time is known to exceed the maximum downtime allowed, this will return a positive number. The function then finds (2) the time at which the machine would become available after a toolkit change. If (1) is less than (2), then this means that changing the machine's toolkit will take more time than waiting for the downtime to run the doable job. ISAP will select the doable job, even though it exceeds the allowable downtime to avoid a toolkit change. If (2) is less than (1), then it will take less time to do a toolkit change than it will to wait past the downtime to run the doable job. In this case, the function BEST-JOB-FOR-MACHINE is called.

BEST-JOB-FOR-MACHINE proceeds as it did when called from FIND-BEST-JOB, with the exception that it does not consider jobs whose start time is greater than (1). If BEST-JOB-FOR-MACHINE returns a job, then the machine is retooled and that job is selected. Note that the selected job will be the highest priority job out of the job's list.

When SELECT-BETTER-JOB-FOR-MACHINE returns, SELECT-BEST-ALLIED-JOB also returns. As a result, the variable BEST.JOB, in the function FORWARD-BY-TIME, has an assigned value. Note that the value could be nil. The function FORWARD-BY-TIME then enters a conditional with two options (the lower A connector in Figure 3). The first option asks if BEST.JOB is non-nil. If it is, then FORWARD-BY-TIME-BY-PART-OPERATION is called. FORWARD-BY-TIME-BY-PART-OPERATION schedules the job on the selected machine. This function will be covered below.

After the job is scheduled by FORWARD-BY-TIME-BY-PART-OPERATION, the jobs and machines are reprioritized to take this information into account. For the jobs, the function REPRIORITIZE-JOBS is called. For the machines, PRIORITIZE-MACHINES is called again.

REPRIORITIZE-JOBS updates the jobs information with the scheduling of the BEST.JOB. If all instances of BEST.JOB are finished, it is removed from the jobs list. If it is not finished, then the job's slack time is recalculated and the jobs are reordered according to the function JOB-SORT.

The second option in the FORWARD-BY-TIME conditional is invoked if BEST.JOB is nil. It sets the machines to the remaining machines and begins the loop again (the upper B connector in Figure 3), selecting the first machine on the new list of machines. In other words, if a job could not be found for the first machine on the list, then the second machine on the list is selected and a job is found for the second machine. This loop continues until there are no more machines, or until all the jobs are scheduled.

The only remaining task for FORWARD-BY-TIME is to schedule remaining downtimes. Remaining downtimes are those downtimes that were not scheduled as part of the call to FORWARD-BY-TIME-BY-PART-OPERATION. Remaining downtimes will exist when all of the jobs were scheduled prior to a downtime's start time. Downtimes that begin after all of the jobs end will not have been scheduled. Thus, those downtimes will be scheduled after all of the jobs have been scheduled. For each machine, any remaining downtimes are added to the schedule. FORWARD-BY-TIME then returns.

The preceding discussion was a step by step walk through of how the Predictive Scheduler in ISAPS generates a schedule for the FMS. The functional description for FORWARD-BY-TIME-BY-PART-OPERATION was deferred during the description of the scheduling process to avoid confusion. The following paragraphs provide a brief review of the functions required to create the schedule once a BEST.JOB has been found.

The function FORWARD-BY-TIME-BY-PART-OPERATION, called from FORWARD-BY-TIME, receives the BEST.JOB and the selected machine. The function obtains the fixture required for BEST.JOB's next operation and assigns a pallet to that fixture if the fixture does not already have a pallet.

If the fixture requires a pallet, then the first operation that is scheduled is a build. An activity is created that represents the build operation. An activity is represented as a Common LISP defstruct. The activity is then scheduled by a call to the function, SCHEDULE-ACTIVITY. After the build is scheduled, a call is made to the function, FORWARD-BY-FMS-PASS.

FORWARD-BY-FMS-PASS takes the BEST.JOB, the selected machine, and the fixture as input. The function has a conditional statement with two options. It asks if the operation to be performed next is a load operation. If it is, then the load is scheduled; otherwise the alternative operation is scheduled (this can occur during reactive scheduling when a partially completed job must be rescheduled). All operations are scheduled with a call to SCHEDULE-ACTIVITY. Scheduled jobs are updated so that their completeness is known.

The function SCHEDULE-ACTIVITY requires all of the information needed to schedule an activity. SCHEDULE-ACTIVITY finds the time that the fixture is available, the time that the activity will start, and the time that it will end. It then takes all of this information, and the other information needed to schedule an activity (e.g., job, activity, fixture, etc.), and calls one of three functions with this information. Which function SCHEDULE-ACTIVITY calls depends on whether the activity to be scheduled is a machining center activity, and whether a BEST.HOLE must be found for a non-machining center activity. A BEST.HOLE is an open interval between to scheduled activities that may be used to place the current activity being scheduled. If the activity is a machining center operation, then the function FORWARD-ASSIGN-JOB-TO-MACHINE is called. If the activity is a non-machining center operation, and a BEST.HOLE is to be found, then the function FORWARD-ASSIGN-JOB-TO-SPECIAL-MACHINE is called. Otherwise, the function FORWARD-ASSIGN-SPECIAL-ACTIVITY is called.

The behavior of FORWARD-ASSIGN-JOB-TO-MACHINE and FORWARD-ASSIGN-SPECIAL-ACTIVITY is similar. They insert the activity's start and end times, fixture, machine, and so on, into the activity's structure (LISP defstruct). The information contained in the structure can then be used to build the Gantt chart using ActiveGantt.

Machining center downtimes are also inserted into the schedule by FORWARD-ASSIGN-JOB-TO-MACHINE.

## USER INTERFACE

The user interface for the ISAP system is graphical in nature, using mouseable buttons, icons, and pop up menus. The relationship of the user interface to the rest of ISAPS is shown in Figure 4.

The primary output of ISAPS is the Gantt chart display. It shows the machines scheduled vertically and the times they are scheduled horizontally. The blocks of time scheduled for each machine represents an instance of a job operation. Selecting a block using the mouse provides information related to the processing of the job, such as job number, start time, completion time, fixture number, and duration.

Reports available to the user include detailed information about jobs, machines, operations, fixtures, and their associated schedules. Machine utilization is tracked by ISAPS for scheduling sessions and is also available through the user interface. All reports may also be written to files for later review.

A second interface is available to the System Manager for ISAPS. Direct manipulation of the KBs and associated LISP code may be accomplished using KEE commands. The end users will not have the capability to permanently change the static FMS model information. Only the System Manager will be allowed to make such changes. This feature has been developed to assure the integrity of the system and to provide tighter configuration control over the baseline system.

## CURRENT SYSTEM STATUS

The prototype for the ISAP system Predictive Scheduler has been completed. All of the functionality described above has been implemented. The detailed system design for the Reactive Scheduler is near completion, with a working test version currently under review.

The ISAP system is being developed on a Symbolics 3620 workstation using KEE Version 3.1. The delivery platform for the full prototype system is an Apollo DN4500 workstation running Domain/OS SR10.2 and using KEE Version 3.1. The development of the additional scheduling functions used Domain/CommonLISP Version 4.0.

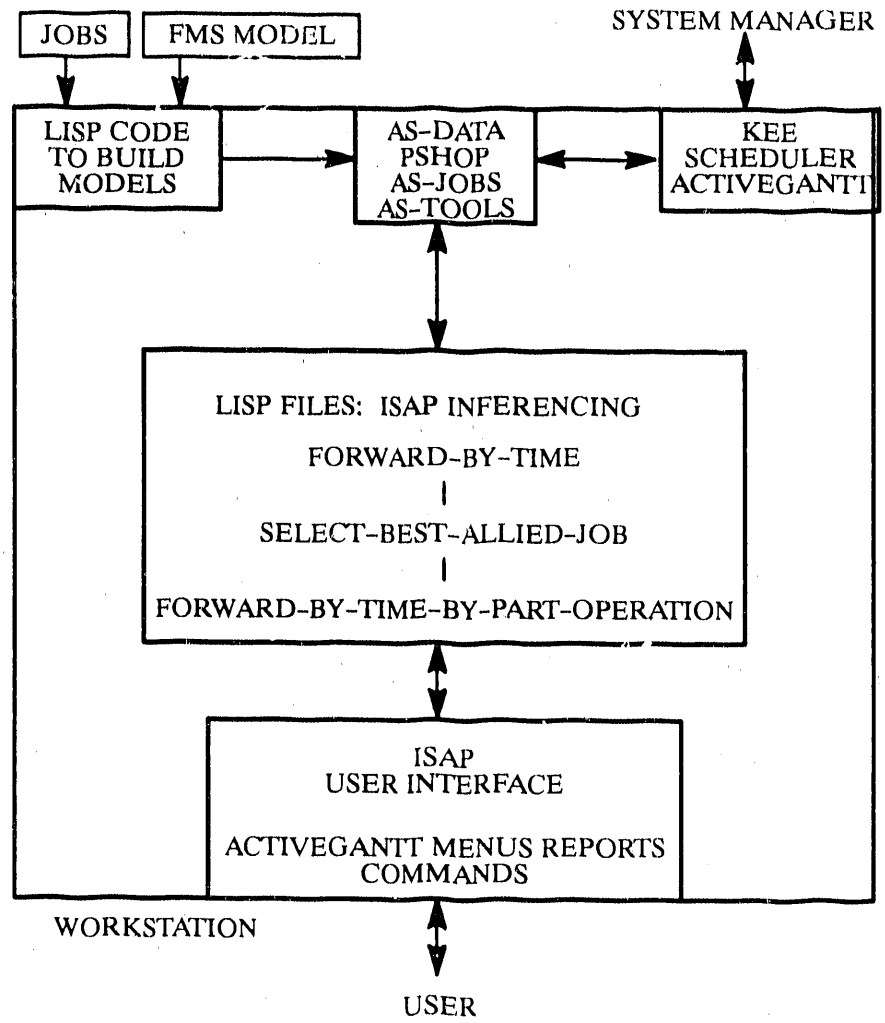


Fig. 4 ISAPS User Interface

ACKNOWLEDGMENTS

IntelliCorp® and KEE® are registered trademarks of IntelliCorp, Inc.

ActiveGantt™, Scheduler™, and Knowledge Engineering Environment™ are trademarks of IntelliCorp, Inc.

Apollo™, Domain/OS™, and Domain/CommonLISP™ are trademarks of Apollo Computers Inc.

Symbolics™ is a trademark of Symbolics Inc.

REFERENCES


1. KEE Software Development System Core Reference Manual, Version 3.1, IntelliCorp, Inc. Mountain View, CA, July, 1990.
2. Scheduler Booster Module: Reference and User Manual, Scheduler Version 1.0, IntelliCorp, Inc. Mountain View, CA, 1989.
3. ActiveGantt Booster Module: User and Reference Manual, ActiveGantt Version 1.0, IntelliCorp, Inc. Mountain View, CA, 1989.
4. Domain/CommonLISP User's Guide, Domain/CommonLISP Version 4.0, Hewlett-Packard Co., April 1990.
5. Software Quality Assurance Handbook, Software Quality Assurance Group, Allied-Signal Aerospace Company, Kansas City Division, Document Number KCP-613-4118, June 1989.

# END

# DATE FILMED

01 / 15 / 91