N96-12950

EXPERT SYSTEM FOR SCHEDULING SIMULATION LAB SESSIONS

By Chet Lund *Lockheed Engineering & Sciences Company*
2400 NASA Road One MC/C07
Houston, TX 77058

ABSTRACT. Implementation and results of an expert system used for scheduling session requests for the Systems Engineering Simulator (SES) laboratory at the NASA Lyndon B. Johnson Space Center (JSC) are discussed. Weekly session requests are received from astronaut crew trainers, procedures developers, engineering assessment personnel, software developers, and various others who wish to access the computers, scene generators, and other simulation equipment available to them in the SES lab. The expert system under discussion is comprised of a data acquisition portion - two Pascal programs run on a personal computer - and a CLIPS program installed on a minicomputer. A brief introduction to the SES lab and its scheduling background is given. A general overview of the system is provided, followed by a detailed description of the constraint-reduction process and of the scheduler itself. Results from a ten-week trial period using this approach are discussed. Finally, a summary of this expert system's strengths and shortcomings are provided.

INTRODUCTION

The Systems Engineering Simulator (SES) lab at the NASA Lyndon B. Johnson Space Center (JSC) provides the real-time engineering simulation capability needed to support various aspects of the Space Shuttle and the Space Station Programs. The SES has been used as a design and analysis tool throughout the Space Shuttle Program.

Early in the Space Shuttle Program the SES was used to conduct conceptual design studies concerned with Orbiter handling qualities, displays and controls, and orbital operations. As the Shuttle Program advanced, the SES provided a testbed in which flight software requirements (mainly guidance, navigation, and control) could be evaluated. The SES was also used extensively in supporting the design of the Remote Manipulator System (RMS). In 1984 the Manned Maneuvering Unit (MMU) was added to the SES. It has provided on-line support during several Space Shuttle missions, most notably the Solar Maximum repair mission.

More recently, the SES developed the Orbiter/Space Station docking simulation. To develop the capability, reasonably sophisticated mathematical models of the Space Station were installed in the simulation. Mass properties, docking port geometry, RMS grapple fixture geometry, aerodynamics, attitude control system, reaction control system (RCS), and visual models are included in the mathematical models. Additionally, a complex Orbiter-to-Space Station Thruster plume impingement model was developed and installed. The plume impingement model produces reasonably accurate forces and moments on the Space Station that would result from any of the Orbiter's 38 primary RCS thruster exhaust plumes impinging on the Space

Station's surfaces during an Orbiter approach.

These are just some of the many functions that the SES has played a role in, and will continue to serve in, throughout the Space Shuttle and Space Station Programs. Interested readers may find a more detailed description of the SES lab and its functions in [1].

SES Lab Equipment

The SES lab is a large complex consisting of dedicated computers, crew stations, computer-generated imagery visual systems, and graphics systems. Minicomputers provide interfaces to the crew stations, host the graphics systems which generate cockpit displays and real-time displays for test evaluators, and also provide the data recording function for the simulations. The mathematical models are also stored here. A large mainframe computer hosts the Space Shuttle entry and landing simulation and is used in conjunction with the Shuttle forward crew station (or forward cockpit).

The SES crew stations include the aforementioned forward cockpit, the Shuttle aft crew station (aft cockpit), a MMU crew station, and a Space Station crew station (cupola). All stations include flight-like displays provided by electronic scene generators so as to make a simulation session as realistic as possible to the participants. The crew stations are arranged in separate enclosures to facilitate parallel simulations.

Approximately 15 lab equipment pieces - i.e., computers (and the math models), crew stations, scene generators, etc. - are available to the lab users.

Where An Expert System Comes In

In earlier times and with a smaller lab, the SES lab manager generated the weekly schedule manually and fairly easily. However, the lab has grown over the years and so has the level of complexity, causing management to consider automating this task.

Some examples of this complexity: Two parallel simulations may proceed during a scheduled session - one on the "A-Side" and one on the "B-Side" - as long as the equipment that each person has requested is mutually exclusive of the other's hardware needs.

Furthermore, an increased workload in SES activities has recently forced the lab to expand its working hours. Altogether, there are 76 schedulable sessions in a week - ([5 days/week* 3 shifts/day * 2 sessions/shift * 2 parallel simulations/session] + [2 days/week * 2 shifts/day * 2 sessions/shift * 2 simulations/session]).

On the average, between 60-75 session requests are submitted each week. Those who need the Aft Cockpit and/or the MMU for their simulations must run on the A-Side. Others who can accomplish their tasks without these equipment pieces can usually run on the B-Side. On infrequent occasions a requestor will ask for both sides simultaneously.

Another factor considered is the relative priority of each project. Certain recurring events such as astronaut crew training are given a high priority. Priorities of other projects such as conceptual design studies or software development work change weekly according to each project's due date. The lab manager must be fully aware of each project's status so as to make the most effective usage of the lab's resources.

Also, the time slots requested are considered whenever possible. There are those who would rather not work third shifts and/or weekends. An attempt is made to accommodate these requests when feasible. Projects also dictate that work must be completed on/before a given date, thereby making some sessions useless to the requestor.

Taking all these factors into consideration when scheduling is a monumental task for the SES lab manager, particularly when scheduling is only one of the many functions that this individual is responsible for. Human errors can and do appear occasionally. The scheduler can inadvertently assign a lab equipment to two people simultaneously, or some hardware that is unavailable or down for repair might get assigned. Some projects cannot run opposite others. Because of the dynamic nature of the job, last-minute changes can cause a completed schedule to be entirely revamped.

In summary, scheduling relies heavily upon human knowledge and experience. But humans are prone to make mistakes as well as subjective judgments. And because the job is very demanding, human scheduling experts are hard to come by and retain. It is for these reasons that an attempt has been made to automate the scheduling process.

OVERVIEW OF THE SYSTEM

The system was developed to mimic the actual process used in generating a weekly schedule. The weekly requests are first reviewed for completeness and accuracy. Requests containing noticeably incorrect or inconsistent data are corrected or resolved by the lab manager. He also assigns a relative priority to each request based upon his knowledge of the various projects' upcoming due dates or the relative importance of the requested session. A data entry specialist then keys the information from the request into a PC-based Pascal program, using both the mouse and the keyboard interfaces. The graphics/mouse interface is vital to this aspect of the system in that, with over 70 data fields associated with each request, the time spent on the data entry phase has been cut in half (versus using a keyboard interface only).

After the requests have been entered and saved to disk, a second Pascal program is called to update the availability statuses of the various equipment found in the lab. For example, any equipment scheduled for preventative maintenance during a session can be marked as being "unavailable" for that session.

From this second program (and assuming that both of the above tasks have been completed, resulting in a request file and an equipment configuration file), one can then initiate that portion of the expert system that looks for "compatible" pairs of session requests - i.e., those pairs of users who can run simulations in parallel because the equipment requested by each is mutually exclusive of the other person's (and they have both specified a given time slot as being "acceptable").

When two compatible requests are found, they are further constrained by checking the Equipment Configuration File for equipment availability during a given time slot. Should at least one equipment requested be found unavailable, this compatible pair is no longer considered as a candidate for that time slot. This process continues exhaustively until all compatible pairs have been considered for the time slots they deemed desirable.

Those pairs having passed this constraining test are written to a file in CLIPS deffacts format. This will serve as an input file to a CLIPS program (the third and final one in the expert system), which does the actual assigning of compatible pairs to sessions, by priority. If a compatible pair cannot be found for a given session, then that time slot will be assigned to just one person who has the highest remaining priority of those tasks being scheduled. Before completing, this CLIPS program writes a schedule to a disk file, which is then printed out and reviewed by the manager. He has the final decision of whether to use any or all portions of it.

DETAILED DESCRIPTION OF SYSTEM

Start of the Scheduling Process

The first constraint check compares a requestor's list of equipment against the Equipment Configuration File for all schedulable sessions. If a person has requested an equipment that is not available for a given session, that requestor is not considered as a candidate for that session. But assuming that his/her requested equipment are all available, this single user is written to the CLIPS file (in the event that no pair can be found for this slot), and the next constraint check is made - comparing that person's equipment requests against the next person's in the linked list data structure.

User 1's list of requested equipment is compared against User 2's list. The check made is that of a Boolean Exclusive-Or function. That is, if User 1 has requested Equipment X and so has User 2, then these two users are no longer considered compatible. This might be referred to as a "hard" constraint. Now, there also exists a case of a "soft" constraint, and it has to do with a user requesting one or more of the three scene generators (referred to as the ESG2, the POLY, and the CT6). Let us briefly look at this issue before continuing on with the scheduling process.

"Soft" Constraints

There are situations where a user needs a specific scene generator, in effect saying: "I've got to have the (ESG2/POLY/CT6) scene generator, or else I can't do my job." One reason for this is that not all scene generators are capable of generating the desired scene for a simulation session. This again would be considered a hard constraint.

But then there are occasions where any one of the three scene generators is acceptable to the requestor. "I don't care which one you assign to me, just as long as I get one." This would be considered a "soft" constraint. Listed below are the different possibilities that must be considered when verifying a soft constraint between two users.

(Requesting the same generator)

	User 1	User 2
Case 1	NEEDS	NEEDS
Case 2	NEEDS	WANTS
Case 3	WANTS	NEEDS
Case 4	WANTS	WANTS

Case 1 is the "hard" constraint example. If both requestors say they "need" it, then these two are considered incompatible. Cases 2, 3, 4, where "wants" is one of the choices specified, are examples of "soft" constraints and require further investigation.

Consider the following example: User 1 and User 2 match up compatibly on all equipment, excepting the scene generators. Assume all three scene generators are available. User 1 "needs" ESG2 and POLY. User 2 "wants" either the ESG2 or the POLY, but just one of the two is sufficient. In this case, User 1 and User 2 would be incompatible because if User 1 needs them, User 2 would be "locked out."

What if User 1 "needs" ESG2 and POLY, and User 2 "wants" POLY or CT6? Now, they would be considered compatible, because User 1 can be assigned his/her equipment, and User 2 can be assigned the CT6 scene generator.

As long as ONE of the scene generators not "needed" by User i is available and deemed as "wanted" by User j, then Users i and j are compatible, and this soft constraint is resolved. Similarly, for the case where both users "want" a scene generator and at least one of the two has requested TWO or more scene generators, then the soft constraint is resolved (our implicit rule is to assign just ONE scene generator if the requestor specifies "wants" and not "needs").

Cases 2, 3, and 4 above can be expressed in Boolean Algebra terminology. Using the following notation for these Boolean variables:

A_1 = ESG2 Requested by User 1	$\sim A_1$ = ESG2 Not Requested by User 1
A_2 = POLY Requested by User 1	$\sim A_2$ = POLY Not Requested by User 1
A_3 = CT6 Requested by User 1	$\sim A_3$ = CT6 Not Requested by User 1
B_1 = ESG2 Requested by User 2	$\sim B_1$ = ESG2 Not Requested by User 2
B_2 = POLY Requested by User 2	$\sim B_2$ = POLY Not Requested by User 2
B_3 = CT6 Requested by User 2	$\sim B_3$ = CT6 Not Requested by User 2

Compatible : Boolean;

Case 2: User 1 "needs" and User 2 "wants". Then -

$$\text{Compatible} := (\sim A_1 \& B_1) \text{ OR } (\sim A_2 \& B_2) \text{ OR } (\sim A_3 \& B_3)$$

or, to generalize:

$$\textbf{Compatible} := \textbf{OR(i, i=1,N)} \ \{ \sim A_i \& B_i \}$$

As long as "Compatible" evaluates to TRUE, User 1 and User 2 are compatible on this soft constraint.

Case 3: User 1 "wants" and User 2 "needs". Then -

$$\text{Compatible} := (A_1 \& \sim B_1) \text{ OR } (A_2 \& \sim B_2) \text{ OR } (A_3 \& \sim B_3)$$

or, to generalize:

$$\textbf{Compatible} := \textbf{OR(i, i=1,N)} \ \{ A_i \& \sim B_i \}$$

Case 4: User 1 "wants" and User 2 "wants". Then -

$$\begin{aligned}\text{Compatible} := {} & (\sim A_1 \& B_1) \text{ OR } (\sim A_2 \& B_2) \text{ OR } (\sim A_3 \& B_3) \text{ OR} \\ & (A_1 \& \sim B_1) \text{ OR } (A_2 \& \sim B_2) \text{ OR } (A_3 \& \sim B_3) \text{ OR} \\ & (A_1 \& B_2) \text{ OR } (A_1 \& B_3) \text{ OR } (A_2 \& B_1) \text{ OR} \\ & (A_2 \& B_3) \text{ OR } (A_3 \& B_1) \text{ OR } (A_3 \& B_2)\end{aligned}$$

or, to generalize:

$$\begin{aligned}\textbf{Compatible} := {} & [\ \textbf{OR (i i=1,N)} \ \{ \sim A_i \& B_i \} \ ] \ \textbf{OR} \\ & [\ \textbf{OR (i i=1,N)} \ \{ A_i \& \sim B_i \} \ ] \ \textbf{OR} \\ & [\ \textbf{OR (i,j i=1,N j=1,N i.NE.j)} \ \{ A_i \& B_j \ ]\end{aligned}$$

Back to the Scheduling Process

Assuming that User 1 and User 2 have passed the first two constraint checks, the last constraint check made in this program determines that if either User 1 or 2 has requested an equipment, the Equipment Configuration File is checked to see if the equipment is available for this session. If it is, then User 1 and User 2 (with their associated priorities and the session number) are written as a "compatible-pair" entry to a CLIPS-formatted def-facts file. This file will be the input file to the third and final (CLIPS) program in the expert system.

This entire constraint-reduction process is repeated - that is, User 1 is compared with User 3, User 1 with User 4, and so forth - until all combinations have been exhausted.

Schedule Compatible Pairs for Available Sessions

This third and final program is written in CLIPS, as mentioned earlier. The "deffacts" file created by Program 2 is opened/read. Also, the Request File created by Program 1 is read in; it contains the auxiliary request-related information - such as requestor's name, phone number, activity description, etc. - that is used for listing out the people scheduled for the various sessions.

The program schedules sessions in order from the most desirable (first shift Monday through Friday) to the least desirable (third shift). Two deffacts, shown below, are used here. Deffact "next-session" contains the next session number to be scheduled, where 1 = Session 1 on Monday, 2 = Session 1 on Tuesday, 8 = Session 2 on Monday, etc. Deffact "sessions_left" is a list structure showing those remaining sessions to be scheduled, in the order specified. After a session has been scheduled, the "next-session" fact is modified to contain the left-most number from the "sessions_left" fact. Then, "sessions_left" is also changed to remove a session number from its list once it has been "moved" to "next-session."

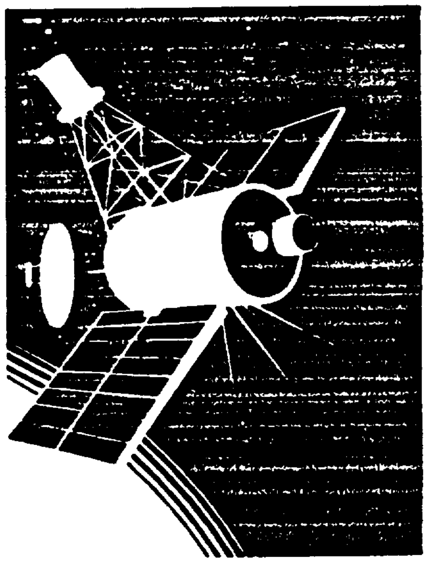
When the final value (0) in "sessions_left" is encountered, the program halts. Note that third shift on weekends (numbers 34, 35, 41, and 42) have been omitted from "sessions_left" because these time slots are currently not used.

```
(next-session 1 Monday)
(sessions_left 2  3  4  5  8  9  10  11  12  15  16  17  18  19
               6  7  13 14 22 23 24  25  26  20  21  27  28  29
               30 31 32 33 36 37 38  39  40  0)
```

The general searching order is to:

- ✦ find a compatible pair where both have the current highest priority,
- ✦ find a pair where one of the two has the highest priority,
- ✦ find just one person (leaving the other slot open for anyone who can use it) having the current highest priority, and
- ✦ leave the slot open because no one remaining had specified this session as an acceptable choice.

Also factored into these searching rules is a check to see if either one or both of the current pair being scrutinized were assigned to the last session as well. The reasons behind this are twofold: Those requesting multiple sessions will have a tendency toward wanting to work consistent hours that week (instead of first shift today, third shift tomorrow, etc), and second, this scheme tends to not schedule a multiple session requestor twice on any given day with a gap between sessions (first and third session, for example). A gap would require lab participants to work a non-contiguous eight-hour day.



RESULTS

This system was run for a ten-week trial period. The criteria used for comparison was the number of requests assigned versus the total number requested that week. Shown below are the results.

WEEK	NUMBER ASSIGNED	NUMBER REQUESTED	PERCENTAGE ASSIGNED
Week 1	45	55	81.8
Week 2	52	59	88.1
Week 3	54	59	91.5
Week 4	53	58	91.4
Week 5	47	54	87.0
Week 6	48	54	88.9
Week 7	49	65	75.4
Week 8	59	76	77.6
Week 9	55	66	83.3
Week 10	56	71	78.9

These results are consistent with those obtained by manual scheduling before making forced adjustments. That is, a high percentage of the requests can be satisfied by assigning those with the highest priority to the slots they deemed acceptable. But to fit in the remaining requests, the lab manager must force-assign people to slots they did not specify, or he may assign slots to requestors if they can forgo the use of equipment that is unavailable during that session.

WHAT WAS LEARNED

The approach taken towards the scheduling task had its strong points and its shortcomings. One positive aspect was that the high-priority requests were almost always scheduled, leaving the lower-priority requests to be assigned manually by the lab manager. Another was that a multiple session requestor would often be assigned contiguous sessions as designed. And seldom did a project request get assigned non-contiguous slots within the same day.

A negative point is that a user who requested sessions for two or more DIFFERENT projects that week was often assigned non-contiguous slots within a given day (no check was made to see if the same person was assigned to an earlier session that day). Also, the program found only one schedule. Perhaps better schedules could have been generated to fit in more requests, had some factor of randomness and a looping mechanism been introduced into the program.

Another very influential aspect that became self-evident during the project was the importance of getting requestors to abide by the request submission deadline. Unfortunately, some people at times would not know what their workload for the following week was until the request deadline had passed. Hence, their requests often came in late - typically up until four hours before a completed schedule was to be reviewed by NASA officials. With manual scheduling, one could make certain allowances to accommodate the late entries. However, four hours leaves very little time for the CLIPS program to execute on a minicomputer, particularly with 20 or more interactive users logged in at the time.

SUMMARY

Because of the aforementioned problems, the CLIPS scheduler was eventually replaced by a FORTRAN program on a mainframe to utilize its CPU speed. Most of the problems encountered with the

CLIPS version have been addressed successfully in the new one. The names of users requesting time for different projects are now checked so non-contiguous slots within a day are not assigned to any user. Subject to the above criteria, compatible pairs are randomly selected and assigned to a schedule slot. A completed schedule is then evaluated according to several grading factors, and the 10 schedules with the highest scores are always saved (and later printed at a specified timeout period). The lab manager now has a choice of which schedule to use as a starting base.

One method of circumventing the late submission problem has worked with limited success. "Dummy" requests with the same priority and with the same typical equipment requested by those expected latecomers are entered to serve as place-holders. This allows the scheduler to be started up with more lead time than previously permitted, thus yielding higher-quality schedules.

Because of the constantly changing requirements brought on by new projects, it is felt that it would be difficult, at best, to program in all the constraint checks that are needed. The best that one can expect from the scheduler output is that it is just a starting base that will still require at least some human manipulation to satisfy the constraints associated with that week's requests and to force-fit in any requests that the scheduler cannot handle.

REFERENCES

[1] St. John, R. H., Moorman, G. J., and Brown, B. W., "Real-Time Simulation for Space Stations", PROCEEDINGS OF THE IEEE, Vol. 75, No. 3, March 1987.