SIMULATION OF MISSION TIMELINES FOR THE SPACELAB PROGRAM

J. Steven Roberts
Michael A. Fague
Claude D. Pegden

ABSTRACT

This paper traces the development of the Special Purpose Simulation Language (SPSL) for timeline analysis. The SPSL language includes several important features which substantially reduce the modeling effort for timeline analysis compared to general purpose languages such as GPSS.

INTRODUCTION

Part of the process of planning new space programs at the National Aeronautics and Space Administration (NASA) is to study the operations which take place on the ground, before and after the planned spaceflight, to ensure that when the program becomes a reality the ground operations will function in an efficient and cost-effective manner. A typical program study is the Spacelab project. Spacelab is a joint project of NASA and the European Space Agency, and is a major element of the Space Shuttle system. The concept is of a modular, reconfigurable and reusable payload, assembled for each mission from an inventory of hardware components. The long-range planning problem, then, involves scheduling mission launches to provide cost-effective utilization of the spacelab inventory consistent with constraints such as launch window limitations and competition for Space Shuttle resources by non-Spacelab projects.

At the Marshall Space Flight Center in Huntsville, several programs are available to generate proposed flight schedules for the Spacelab missions. These programs are designed to answer "what if" type questions involving the impact on launch schedules caused by the deletion, addition, or changing of acquisition times for the various pieces of flight and ground hardware. These programs are organized under an executive operating system called GROPE (Ground Resource Operations Planning Executive). GROPE was designed to facilitate the interfacing of the various scheduling programs under its control and with the numerous data files containing information on flight schedules, flow relationships, timelines, and hardware requirements.

By using the scheduling programs contained in GROPE, the analysts at Marshall are able to respond quickly to requests from NASA headquarters concerning the impact on flight schedules caused by changes in flight and ground hardware schedules.

Each time a new schedule was generated by the GROPE system in response to changes in flight hardware, a new simulation model was also required as a cross check. While GROPE is designed for easy modification, the modification or generation of simulation models was more difficult and time consuming. Often the results of the studies had to be sent to NASA headquarters without validation by the simulation models.

This paper describes the Special Purpose Simulation Language (SPSL) which was designed to eliminate many of the problems encountered in generating models using GPSS and other general purpose simulation languages. We begin by giving a brief overview of the Space Shuttle and Spacelab programs. This is followed by a description of timeline analysis using the GROPE system and simulation problem areas encountered in using GPSS for validating Spacelab timelines. Finally, a description of the SPSL language is presented.

SPACE SHUTTLE AND SPACELAB (1)

The Space Shuttle is composed of a retrievable Orbiter, an expendable External Tank, containing propellants used by the Orbiter main engines and two refurbishable Solid Rocket Boosters. The Solid Rocket Boosters and the Orbiter's main engine operate simultaneously during the first portion of flight and the Solid Rocket Boosters are jettisoned. The Orbiter main engines continue to operate until the desired suborbital conditions are achieved and the External Tank is jettisoned. The Orbital Maneuvering system is then used to place the Orbiter in the desired earth orbit.

The Orbiter is a manned vehicle designed to carry a maximum total cargo of 65,000 lbs. at lift off. The cargo bay located midship, is outfitted with large clam-shell type doors that open to provide access to the entire upper half of the cargo. A typical cargo manifest consists of an Interim Upper Stage, and associated spacecraft, single or multiple automated spacecraft (free flyers), or a Spacelab and experiment equipment alone or in combination. Specialized payloads and payload support equipment may also be carried in the cargo bay area.

The Shuttle concept substantially reduces cost of space operations by repetitive use of most of its

components. It also saves money for the experimenter because instrumentation and equipment can be returned to earth for reuse or repair rather than be abandoned in orbit.

The Spacelab is a space-borne laboratory being built by the European Space Agency. Spacelab will be transported to earth orbit by the NASA Space Shuttle for missions lasting up to 30 days. The Spacelab is a major Shuttle element being developed to transport people and material more economically and routinely between the ground and earth orbit. Spacelab will have facilities and equipment similar to laboratories on earth but adapted for zero gravity. In any of several configurations to meet specific experiment requirements it will provide a shirt-sleeve environment, like that in a passenger airliner, for both male and female experimenters. Spacelab will remain attached to the Orbiter, where the experimenters will eat and sleep, throughout the mission. On return to earth, Spacelab will be removed from the Orbiter and outfitted for its next assignment.

Spacelab is conceived as having two main elements: the pressurized laboratory module and an external instrument platform or pallet for large man-directed instruments such as telescopes or antennas requiring direct exposure to space or a broad field view. Pallet-mounted instruments normally will be supplied and controlled from within the pressurized module but could be attended from the Orbiter cabin in a "pallet-only" mode. The pallet can accommodate a variety of experiments where manned attendance is unnecessary but certain services such as temperature control and electrical power are needed. Both the pressurized portion of Spacelab and the pallet are modular in concept so as to be able to adjust to varying demands.

To meet a variety of experimental needs within the pressurized module, each Spacelab will include such features as: standard laboratory instruments; equipment racks; work benches; windows; data recording and processing equipment; and extendable booms for remote positioning of experiments requiring special placement. Continuous voice and data transmission capability is contemplated to allow experimenters to maintain contact with colleagues on the ground for consultation and rapid reorientations.

Flight equipment recovered from previous missions are saved, deserviced and maintained/refurbished as required. The refurbished solid rocket motors, the Orbiter, and a new External Tank are assembled on the Mobile Launch Platform in the Vertical Assembly Building and transported to the launch pad with the crawler transporter.

TIMELINE ANALYSIS USING GROPE (2, 3)

Long range policy and decision making for programs such as Spacelab must be supported by a detailed and realistic analysis capability. This ability to evaluate program concepts and the impact of changes to operating procedures and constraints can be abstracted to a very large scheduling problem: to develop a set of mission launch dates consistent with launch window constraints and effective utilization of resource inventories. Moreover, since the specific analysis problems rarely take on the same form, the general analysis capability must have an inherent flexibility and adaptability.

Simulation was utilized in early attempts to address this problem. A GPSS (4) model of the Shuttle/Spacelab operations was developed and used with some success. Unfortunately, it did not provide the necessary adaptability to cope with frequent changes in the problem statement. Consequently, other analysis methods were sought out and tested and the simulation capability was maintained only as a cross-check on the results of the other study methods. In fact, as the analysis system grew and made increasing use of heuristic methods, the need for a simulation as a means of validating conclusions became correspondingly greater.

The analysis system currently in use at NASA's Marshall Space Flight Center consists of a number of application programs which operate on a data base under master control of an executive program. Collectively known by the acronym GROPE (Ground Resource Operations Planning Executive), the system was developed over several years by NASA and NASA contractors. It originally consisted of incompatible segments addressed at individual pieces of the total problem, but now is a fully integrated package. The main feature of the system that is of interest here is the organization of information within the data base.

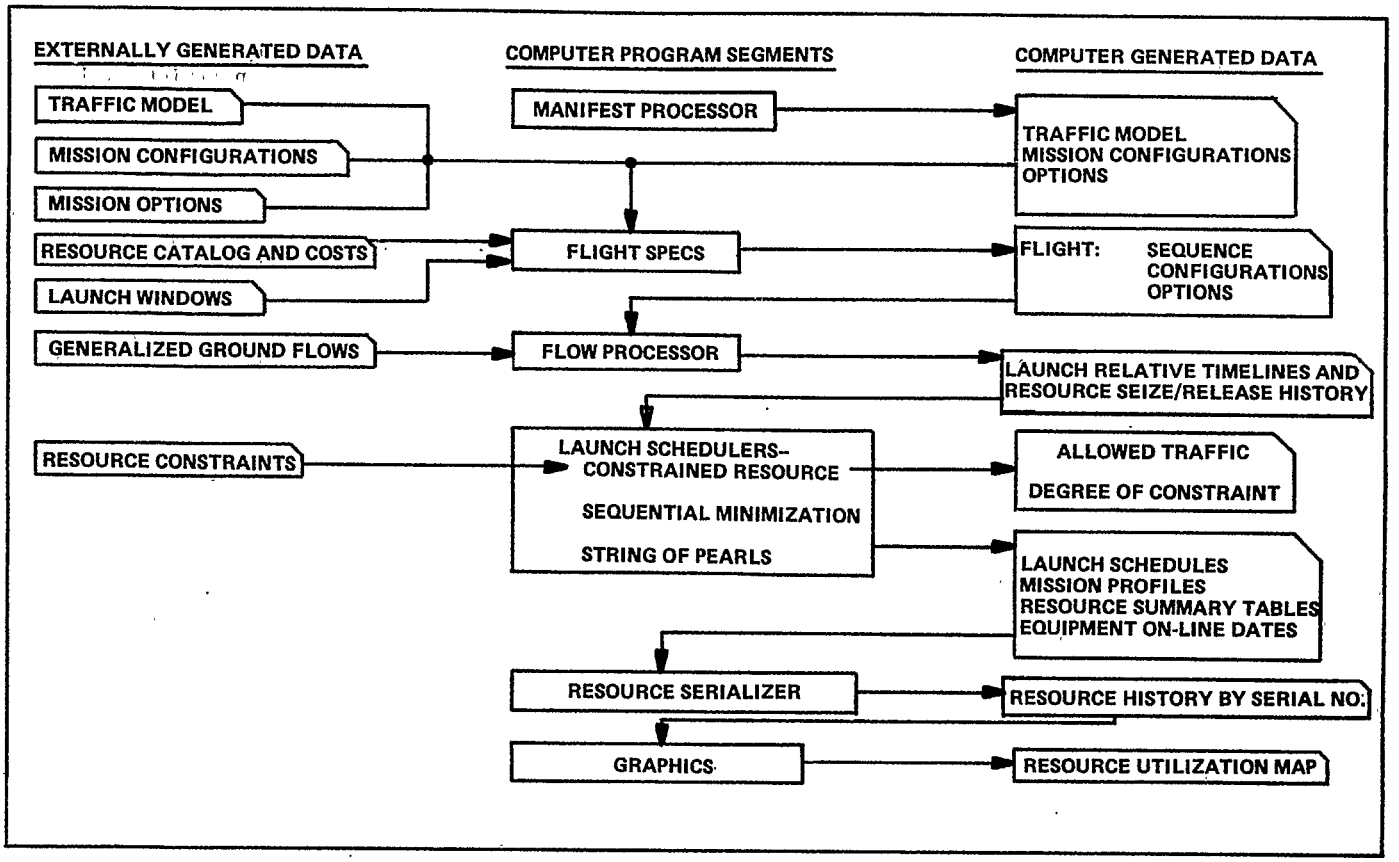
The information in the GROPE data base models the important aspects of the ground operations for the Shuttle/Spacelab system. The data can be grouped into four subdivisions to provide a coherent picture of the system:

- Ground Operations Model
- Mission Model
- Resource Model
- Traffic Model.

The Ground Operations Model consists of a complete description of the ground operations for Spacelab and significant activities of the Shuttle and some non-Spacelab missions. These activities are organized into several flows with individual tasks treated as timeline activities. The task data is made up of descriptive information, resource seize-release flags, limited predecessor-successor data, activity duration information and options relating the task to particular mission sets.

The mission sets are defined in the Mission Model. Missions are categorized by payload configuration and other factors which influence the ground support operations. Over 90 distinct mission sets are included. The primary information in the Mission Model is the inventory requirements, by mission, and the option data which indicates a mission's requirement for particular tasks in the Ground Operations Model.

Figure 1. NASA Spacelab Inventory Analysis System



The Resource Model defines the set of resources for which Shuttle missions contend. The resources include flight hardware, ground support hardware, and support facilities; and include both constrained and unconstrained items. Significant information in the Resource Model consists of a resource reference table, a summary of projected resource levels (quantities) by year and a table of resource acquisition dates and quantities.

The Traffic Model defines the Shuttle workload through lists of mission launch dates.

The models contain much more information as they exist in the GROPE data base, but only items pertinent to the simulation problem were mentioned. The information ascribed to the models consists both of source data and results from the several analysis programs, since the simulation goal is to verify rather than define resource utilization and traffic capability. The overall GROPE analysis process is illustrated in Figure 1.

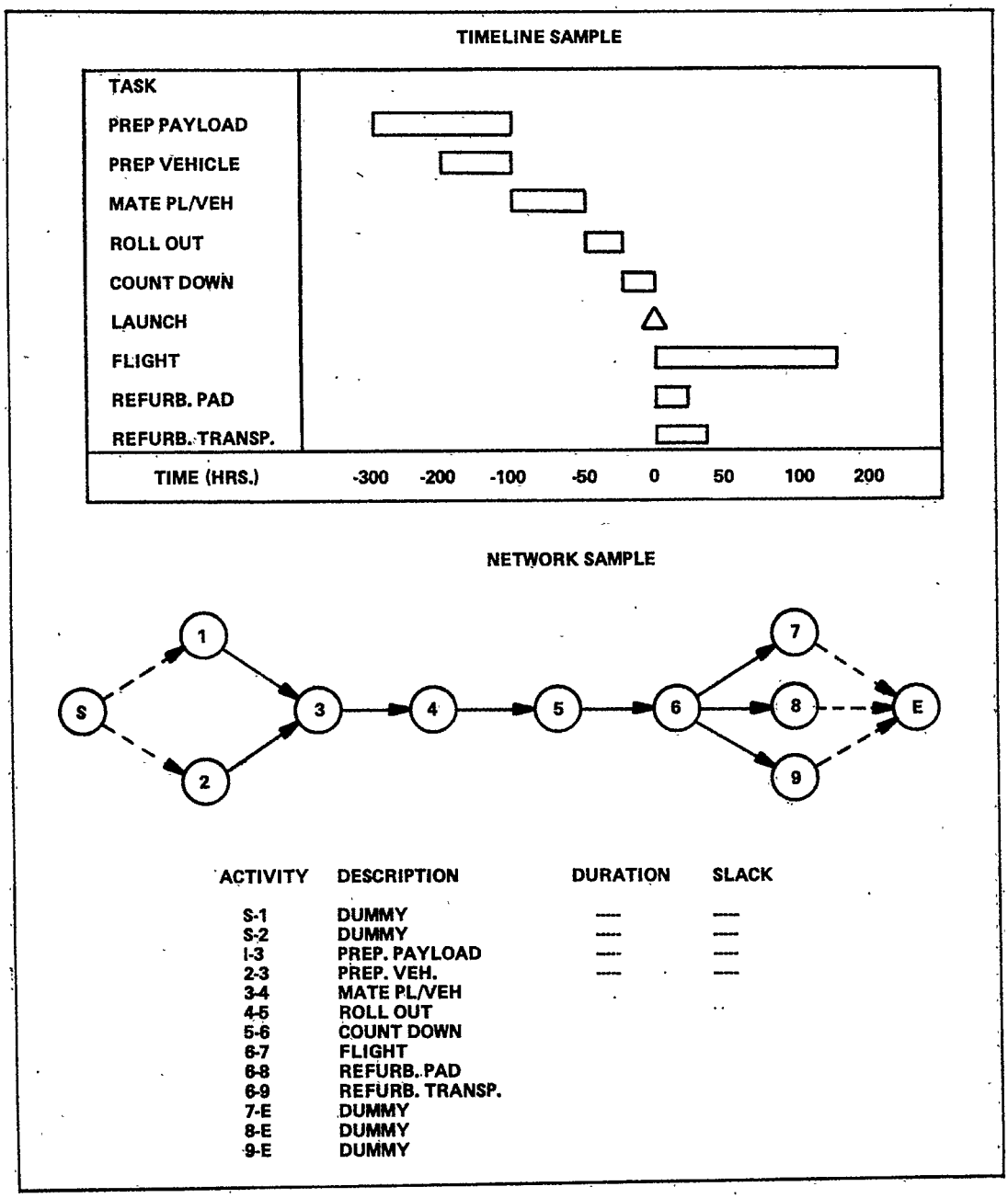
Much of the report generation capability within the GROPE system is aimed at presenting study results in a concise and succinct form. The timeline is a natural medium for conveying this type of planning information and it is clearly a popular one. It displays on a time scale the relative scheduling of activities, their durations, and the occurrence of key milestones of a project. Together with cost and resource data, timelines have been used extensively to provide management with the "big picture" of a project. The timeline concept has carried over into many aspects of the simulation.

Figure 2 illustrates some of the differences between timeline and network representations. The timeline bar is reflected by the network arrow, but the network nodes do not appear in the timeline explicitly except for key milestones. The primary information conveyed by the node--the activity interconnections--is only suggested by the timeline. Conversely, the temporal relationships among the tasks is only suggested by the network.

The information in the data base is sufficient to completely describe either a network or timeline. There are many factors which led to the use of the timeline concept. The timeline task consists of more than a simple timed activity. Resources required by a task may be seized at its beginning and released at its end. PERT and CPM methods name activities by their start and stop nodes and do not permit distinct activities to begin and end on the same nodes. Also, PERT and CPM require a single start and a single stop node for the network as a whole. Finally, PERT and CPM are aimed at non-repetitive projects--everything is done only once.

The ground operations for Spacelab are always described in terms of the tasks and not where the tasks originate or end. There are many occurrences of parallel tasks which, in a network, would start

Figure 2. Graphic Comparison of Timeline and Network Concepts



ACTIVITY	DESCRIPTION	DURATION	SLACK
S-1	DUMMY	—	—
S-2	DUMMY	—	—
1-3	PREP. PAYLOAD	—	—
2-3	PREP. VEH.	—	—
3-4	MATE PL/VEH		
4-5	ROLL OUT		
5-6	COUNT DOWN		
6-7	FLIGHT		
6-8	REFURB. PAD		
6-9	REFURB. TRANSP.		
7-E	DUMMY		
8-E	DUMMY		
9-E	DUMMY		

and stop on the same nodes. Typically, there are many "danglers," or tasks which have either no predecessors or no successors. The tasks in the Ground Operations Model are repetitive--operation of the system is dependent on the competition for resources by many missions repeated over and over in the course of several years. One last reason for the timeline concept over a network is that many tasks in the model are mutually exclusive so that a single, self-consistent network cannot be constructed.

SIMULATION PROBLEM AREAS

GPSS simulation models have been used with varying degrees of success in analyzing various aspects of the Ground Operations process. The GPSS models were used to determine if the launch schedules generated by the GROPE system were feasible.

In order to produce GPSS models in the least amount of time, a well defined procedure for conversion of a Spacelab timeline into a GPSS

model was needed. This led to the adoption of standardized coding techniques aimed at simplifying the conversion process. Even though these coding techniques eliminated much of the time required to build a model, the time required to debug and run the model was increased. The increase in production time of these models was due to the increase of core and run time caused by the coding techniques adopted to reduce the time required to code a model.

The problems encountered in converting Spacelab timelines to GPSS models are best illustrated by example. Each task in the Ground Operations Model represents the following sequence of steps: required resources are seized; time is spent performing the task; and resources no longer required are released. Since the quantity of resources to be seized or released is a function of mission type, a table look-up function must be built which incorporates the Mission Model resource requirements table. Activity duration, in general, is not constant for each task, but must be scaled to include effects of learning and the number of shifts per day that the task may be performed. Thus, a function must be constructed to recompute the activity duration each time the task is to be performed. In general, these two functions are best performed by support subroutines, via the GPSS HELP block, which can be initialized by input data extracted from the Ground Operations Model. Unfortunately, they add greatly to the complexity and run time of the GPSS model. The simulation analyst, therefore, will avoid the functions where possible, but finds that from case to case the necessity for using the functions may change.

To simplify the structure of the GPSS model, the tasks are encoded using the GPSS macro feature. This enables the task related code to be separated from the code controlling the flow of transactions. The net effect of structuring the model in this manner was an increase in size and run time but a marked improvement in the clarity and maintainability of the model. This aspect of the modeling process is mentioned because it is a feature we felt should be retained in any new simulation system.

The network, or flow, logic presents a number of more serious problems. As mentioned above, the Ground Operations Model does not provide a single network, but rather functions as an inventory of tasks from which flows are constructed for individual missions. In the GROPE system, the flow processor program actually builds a separate flow for each mission, but this would be prohibitive in any simulation. The approach taken to deal with this problem evolved as follows: First, a network was developed which linked all tasks and then logic was added to limit transaction flow to appropriate paths. The nature of the logic is such that only trivial cases could actually be simulated without overloading the model with code for this single function. The problem points are those places where parallel activities merge into a single flow. In GPSS it is necessary to know exactly how many transactions will arrive at each assembly, or merge, point. Given the number of different mission types, it is difficult to determine this number.

To get around the assembly problem, the flow was set up so that transactions would be routed to all tasks and logic was added to the task macros to test the Mission Model option set and shunt transactions around the task code if it represents a mission which does not require the task. Thus, assembly points always merge a fixed number of transactions.

Two more problems appear at this point. The GPSS ASSEMBLE block transmits the first transaction to arrive in each set to be assembled. Since a transaction moving through a sub-flow which it does not require will be shunted immediately to an ASSEMBLE block, it will be the transaction to continue on once the assembly conditions are satisfied. In other words, the transactions which actually go through the task processing could be destroyed in favor of a copy which has not performed any actual processing. Consequently, any information stored in a transaction parameter during a sub-flow is in danger of being lost. Also, GPSS requires resources to be released by the transaction which actually seized them, and not a copy. These problems were dealt with, again, by the addition of logic and data tables.

One final problem in the network which has not yet been completely solved involves the location of slack. In the GROPE system all tasks with slack are positioned to occur as close to launch as possible. Since launch is usually in the center of the flow, prelaunch tasks occur as late as possible and postlaunch tasks occur as early as possible. Within the simulation network, however, all tasks will occur as early as possible. This can result in resources being seized much earlier than necessary. Extreme cases are adjusted manually, but in most situations the problem is simply not treated.

A final difficulty with GPSS is the implementation of changes in resource levels at predetermined times. It is necessary in GPSS to build a separate flow to affect these changes. A simpler method would be desirable.

Most of the problems discussed can be dealt with successfully, but only by the addition of overhead logic to the simulation. While the task and network flow can be easily encoded, the overhead logic requires close attention by the analyst. It is precisely this overhead which detracts from the flexibility of the simulation. The experience gained and problems encountered in our attempts to use GPSS were instrumental in developing the concepts and requirements for a more direct form of simulation based on timelines and the GROPE data base organization. This led to the development of the Special Purpose Simulation Language (SPSL) described in the next section.

THE SPECIAL PURPOSE SIMULATION LANGUAGE (SPSL)

Given the scope and difficulties of the modeling problem and the continuing need for the simulation capability, an effort was undertaken to find and implement a more viable approach. Alternatives fell into three groups: use of a more suitable simulation system, modification of an existing system, and development of a totally new system.

This last option was not seriously considered due to the size of such an undertaking.

The options for simulation languages centered around those already available to us, namely GPSS, Q-GERT (5), GASP-IV (6), and SIMSCRIPT I.5 (7). The problems with GPSS have already been mentioned and similar problems were encountered with Q-GERT. Of the event-oriented languages, GASP was favored over the older SIMSCRIPT I.5. In either of these cases, the question is whether an event oriented language can efficiently and flexibly simulate a system already modeled in a timeline fashion.

In a timeline or network diagram, milestones, or events, occur at the end points of a task--the nodes of a network. Since GASP considers an event to be any point in time at which a change in the state of the system could occur, there appears to be a natural correspondence between a timeline model and a GASP model. Unfortunately, each node is a separate event and would require a separate event subroutine. This would not improve the original problem of simulation flexibility.

What is necessary is a generalized event routine, or some minimal set of event routines. A task is not an event since it involves the time to perform the function, but a task could be described by two events, task start and task stop, displaced in time by the task duration. The task start event would be concerned with resource acquisition and the stop event would process resource releases and scheduling of successor tasks. More problems arise, however, and these two events are quickly broken into several more specialized events.

It was at this point in our analysis that the prospect of proliferating event routines combined with the experience being gained in the development of a new, GASP based simulation language, SLAM (8), produced the idea of modifying, or adapting, GASP or Q-GERT to provide a simulation capability closely tailored to our problem. As design requirements, such a system would have to provide or resolve the following key items:

- Reduce core and run time requirements, compared to GPSS
- Provide a structured framework to separate task description from the network flow
- Internally control network routing and task option selection
- Provide selective transfer of attribute information at flow merge points
- Provide for indexed or serialized resources
- Provide internal handling of resource capacity changes
- Incorporate automatic scaling of activity durations due to learning, and
- Make efficient use of information from the data base.

The resulting system, which we called the Special Purpose Simulation Language (SPSL), combines features of GPSS and GASP. It is written in FORTRAN and uses a slightly modified version of GASP-IV as the system host. Simulations are produced by developing an input data deck which SPSL interprets and processes.

The SPSL data statements are organized to provide a one-to-one correspondence between the information requirements of SPSL and the existing GROPE source data base. In addition, the modeling task is simplified by having the overall problem description decomposed into separate but related elements. The task flow sequencing, as defined by predecessor and successor relationships between tasks, are prescribed in the Network Description section. The detailed ground operations associated with each task are defined separately in the Macro Definition section. The available resources such as orbiters, pallets, etc., for which the individual tasks compete, are defined in the Resource Definition section. Information transfers between tasks are provided for by attributes associated with transactions routed through the task, and by SAVEX variables defined in the SAVEX Definition section of the SPSL program. Finally, an Option Set Definition section is provided to allow the modeler to define the members of each option set.

NETWORK DESCRIPTION

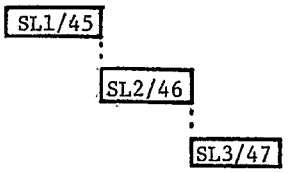
The network section describes the linkage and operational conditions required for each task. The information defining the task flow sequence included in the network section consists of:

- The number of transactions from different tasks that are required to start the task
- Which tasks are to be executed after the current task is completed
- Which option set member value is required for the task to be executed once started
- The attribute whose value must be the same in each transaction which forms the set of transactions to be merged.

There are two types of flows which are typically encountered in timeline analysis. There are serial or sequential flows in which the tasks are executed in sequential order, and parallel flows in which the tasks operate independently until separate flows must merge. The network section of SPSL is designed to accommodate both types of timeline flows.

To illustrate the modeling of a sequential flow, consider a series of three tasks consisting of Prep Pallet for Shipment, Level IV Integration, Ship Pallets to KSC, and denoted as Task SL1, SL2,

and SL3 respectively. The duration and sequencing of the tasks are depicted graphically as follows:



The length and location of each task describes the duration and starting time of each task. The dashed lines connecting each task denote a predecessor-successor relationship between the tasks. The number following the slash (/) is the task or sequence number. The corresponding SPSL network description is given as follows:

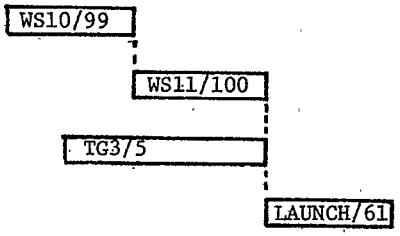
```
NETWORK
  .
  .
  .
SL1/45, 0, 1,46*      Task Number 45 has no predecessors
                      and one successor which is Task 46.

SL2/46, 1, 1, 47*     Task Number 46 has one predecessor
                      and one successor which is Task 47.

SL3/47, 1*            Task Number 47 has one predecessor
                      and no successors.
```

The * is used to denote the end of the executable statement and all data following * is comments.

Parallel flows which merge are as easily written as the serial flow. The following example illustrates the merging of two serial tasks (WS10 and WS11) from western Shuttle flow with a task from the Tug flow. The graphic representation for the flow sequence is depicted below:



Once task WS11 and TG3 are finished, the two tasks are merged and the launch task is started. This example illustrates how these two independent flows (one for the orbiter and one for the tug) are combined for launch. The corresponding network description would be coded as:

```
NETWORK

WS10/99, 0, 1, 100*      Task Number 99 has no predecessors
                         and one successor which is Task 100.

WS11/100, 1, 1, 61*      Task Number 100 has one predecessor
                         and one successor which is Task 61.

TG3/3, 0, 1, 61*         Task 3 has no predecessor and one
                         succesor which is Task 61.

LAUNCH/61, 2*            Task 61 has 2 predecessors and no
                         successors.
```

Suppose we require the two transactions causing the launch task to be started be from the same mission. The launch task would be changed to:

```
LAUNCH, 61, 2, 0, N*
```

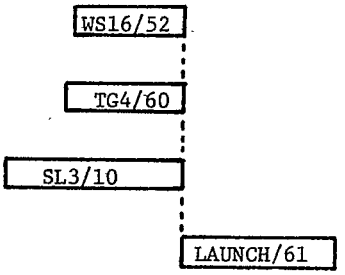
where N is the attribute of the transaction which contains the mission number.

As mentioned previously, a transaction moves through all branches of a flow but may not execute the block cards within each task. The following shows a change in the TG3 task requiring an option match:

```
TG3, 5, 0, 1, 61, N*
```

N is the option value which must match a member of the option set specified by the transaction entering the task.

Another unique feature of SPSL is the method by which the analysts can determine which transaction data is to be saved after two or more flows are merged. Consider the case where three flows are merged; data from each flow is required in the transaction which continues after the merge. Consider the merging of the following flows:



Suppose the transaction coming from SL3 is to be saved after the merge. Also suppose the information contained in attribute 3 of WS16 and attribute 5 of TG4 are needed in the merged flow. The statement to accomplish this operation would be as follows:

```
LAUNCH/61, 10, 3, 52, 16, 5, 60, 17*
```

This card specifies: First, the transaction coming from Task number 10 (SL3) is to be saved after the merge, second, the value contained in attribute 3 of the transaction coming from Task number 52 (WS16) is to be copied to attribute 16 of the saved transaction, and third, the value contained in attribute 5 of the transaction coming from Task number 60 (TG4) is to be copied to attribute 17 of the saved transaction.

If copy information is not required on the saved transaction, the transaction saved is the last one that satisfies the merge conditions.

Table 1. Excerpt from Ground Operations Catalog

SEQ	TASK	OPT	TASK DESCRIPTION	SHIFT	LEARN	CYCLE	DUR-P	DUR-M	DUR-C	PRECEDENT TASKS	GOTO	RES. ACTION/RESET	QTY/CAT NO.
20	WS0	0	TRANSPORT ORBITER TO WTR	3	0	0	24	0	0		WS1	+0	1

MACRO DEFINITION

The Macro Definition section is a group of GPSS-like block cards which define the operations performed within each task. The organization of the task is similar to the macro definition sequences found within GPSS with one major exception. In GPSS, macro's are called in line with the regular GPSS block cards. In SPSL, each macro call is initiated according to the linkage defined within the Network Definition section.

The block cards and the macro organization were designed to exploit the best features found within GPSS. The block cards contained in SPSL are a modified subset of the block cards found in GPSS. One design criteria of SPSL which led to the use of a GPSS-like format for the Macro Definition section was to have a language which would simplify the modeling process yet avoid extensive retraining of the analysts involved.

The block types found in SPSL provide for the basic functions required to model the ground operations process. These cards provide for the following operations:

- Seizing and releasing of resources
- Branching within the task based on comparisons of values contained in the resource, SAVEX, or attribute variables
- Replacing and/or incrementing the value of system variables by values of other system variables
- Delaying the transaction by a quantity obtained from the value contained in an attribute, SAVEX, user-defined function, or one of ten statistical functions available in GASP-IV.

One factor which simplifies the modeling of the ground operations tasks is that the structure of the majority of the tasks is identical. The main difference between these tasks is in terms of the data used. The format of this structure is:

- Store the task number
- Store the nominal time for execution
- Seize or release resource
- Delay the transaction
- Release resource and/or exit the task.

All the data required to write these block cards is available from the Ground Resources Catalog and the Ground Operations Catalog.

Table 1 is an excerpt from the Ground Operations Catalog of Western Shuttle activities:

The data in this table describes the operations associated with Task WSO/20. The task is Task Number 20, has a nominal time of 24 hours, and requires one orbiter for initiation of the task. The actual duration or delay time for the task is the nominal time scaled to include the effects of learning and the number of shifts per day that the task may be performed. The SPSL statements required to model the task are shown below:

```
MACRO
  .
  .
  .
WSO/20                    *    TASK CARD
*TRANSPORT ORBITER TO WTR
ASSIGN,ATRIB(2),20        *    STORE TASK NUMBER IN
                               ATTRIBUTE 2
ASSIGN,ATRIB(3),24.       *    STORE NOMINAL TIME
                               IN ATTRIBUTE 3
SEIZE,ORBITER,1           *    ACQUIRE ONE ORBITER
DELAY,USER(1)             *    CALCULATE DELAY
                               BASED ON LEARNING
                               NOMINAL TIME AND
                               SHIFT DIFFERENTIAL
```

The card types coded in SPSL represent the minimum number which will provide the modeling flexibility required within the constraints on core size and run time. There are several operations that are not easily performed using SPSL block cards. Provisions for these operations for one reason or another were intentionally not incorporated in the SPSL block card structure. However, two block card types are provided in the Macro section which allow the user to interface with GASP-IV and user-generated FORTRAN routines. The block cards are the User Function and Activity blocks. By using these two block types, problems previously encountered with the calculation of delay times based on learning and the changing of resource capacity is minimized. Another important aspect of these blocks is the ability to start any task within the network by user-written FORTRAN routines. This ability allows the user to introduce transactions into the model at the proper time, based on data accessible through FORTRAN read statements.

RESOURCE DEFINITION

The Resource Definition section contains those cards which define resource variables and assigns an initial value to each. Resource variables perform the same function as facility and storage variables in GPSS.

The SPSL structure contains several specific features which facilitate the use of resources within the Ground Operations model. Resource features incorporated include:

- Resources can be serialized or indexed
- A transaction that releases a resource does not have to be the one that seized it
- Dynamic resource capacity change
- Resource quantities are real valued

Resource names can be indexed or nonindexed. If indexed, the value of the index can be determined at run time from the value of a SAVEX or attribute.

For example, if the SAVEX variable PAYLOADNUM has been assigned a value of 3, then the statement:

```
SEIZE, ORBITER (PAYLOADNUM),1.*
```

would result in ORBITER (3) being seized. Likewise, if attribute three of the transaction entering the SEIZE block is equal to five, then

```
SEIZE, ORBITER (ATRIB=3),1*
```

would result in ORBITER (5) being seized.

In many cases, the transaction needs to seize the first available resource of a serialized group instead of a specific one. SPSL contains a method by which the processor determines the first available resource, seizes it, and stores the value of the index of the resource in an attribute of the transaction.

For example, assume we want only the first available orbiter. We can accomplish this with the following SPSL statement:

```
SEIZE, ORBITER (-3),1.*
```

SPSL will seize the first available ORBITER and store the index value of the ORBITER in attribute three. The corresponding release of the ORBITER would be:

```
RELEASE, ORBITER (ATRIB=3),1.*
```

In GPSS, when two flows merge only one transaction may continue. If transactions from these tasks are holding previously seized resources which are still required in the flow, one or both of the transactions must release the resource. Once assembly is performed, the saved transaction must seize the resources released before the merge. In SPSL any transaction can release any resource. This eliminates the need for the user to ensure that the transaction which caused the seize be preserved and routed to the appropriate release block.

Resource capacities can be changed at run time by the use of the FORTRAN routines and/or block cards in the Macro section. Another important feature of the resource variable is that the resource capacity is real valued and the quantity of resource may be determined at run time as well as compile time.

SAVEX DEFINITION

In the SAVEX Definition section, the user defines names and the initial values for SAVEX variables. The SAVEX variable is a global variable similar in function to the GPSS SAVEX variable.

OPTION SET DEFINITION

All branching between tasks in SPSL is deterministic. A transaction can enter a task; however, unless an option code assigned to that task matches a member of an option set specified by the transaction, all operations within the task are skipped. The option set definition section defines the option set members associated with each option set.

SUMMARY

The main conclusion reached in our development of SPSL is that by employing a language such as GASP-IV as a host, it may be practical to develop simulation languages for specialized applications. Future plans for SPSL include a new processor which will automatically generate SPSL programs from data contained within the GROPE system. This will allow planners to more efficiently and cost-effectively plan for future space programs requiring timeline analysis.

BIBLIOGRAPHY

1. Space Shuttle, U.S. GPO 740-049/144, NASA Marshall Space Flight Center, Alabama, 1977.
2. KSC Ground Operations Plan, NASA Report SP-PAY-56-76, Kennedy Space Center, Florida, June 2, 1976.
3. Spacelab Payload Accommodations Handbook, ESA Ref. No. SLP/2104, European Space Agency, January 1977.
4. General Purpose Systems Simulator (GPSS 1100) Programmer Reference, UP-7883, Rev. 1-A, Sperry Rand Corporation, 1974.
5. The Q-GERT Users Manual, Pritsker & Associates, Lafayette, Indiana, 1974.
6. The GASP-IV Simulation Language, Pritsker, A. A. B., John Wiley and Sons, New York, 1974.
7. SIMSCRIPT I.5 Programmer Reference, UP-7885, Sperry Rand, 1974.
8. Introduction to Simulation and SLAM, Pritsker, A. A. B., and Pegden, C. D., Halstead Press and Systems Publishing Corporation, 1978.

ACKNOWLEDGEMENT

Portions of the work was sponsored by NASA, George C. Marshall Space Flight Center, Alabama, under contract NAS8-31640.