AUTOMATED PLATFORM MANAGEMENT SYSTEM SCHEDULING

Larry G. Hull
Code 522
Goddard Space Flight Center
Greenbelt, Maryland 20771

ABSTRACT

The Platform Management System was established to coordinate the operation of platform systems and instruments. The management functions are split between ground and space components. Since platforms are to be out of contact with the ground more than the manned base, the on-board functions are required to be more autonomous than those of the manned base. Under this concept, automated replanning and rescheduling, including on-board real-time schedule maintenance and schedule repair, are required to effectively and efficiently meet Space Station Freedom mission goals.

In a FY88 study, we developed several promising alternatives for automated platform planning and scheduling. We recommended both a specific alternative and a phased approach to automated platform resource scheduling. Our recommended alternative was based upon use of exactly the same scheduling engine in both ground and space components of the platform management system. Our phased approach recommendation was based upon evolutionary development of the platform.

In the past year, we developed platform scheduler requirements and implemented a rapid prototype of a baseline platform scheduler. Presently we are rehosting this platform scheduler rapid prototype and integrating the scheduler prototype into two Goddard Space Flight Center testbeds, as the ground scheduler in the Scheduling Concepts, Architectures, and Networks Testbed and as the on-board scheduler in the Platform Management System Testbed. Using these testbeds, we will investigate rescheduling issues, evaluate operational performance and enhance the platform scheduler prototype to demonstrate our evolutionary approach to automated platform scheduling.

The work described in this paper was performed prior to Space Station Freedom rephasing, transfer of platform responsibility to Code E, and other recently discussed changes. We neither speculate on these changes nor attempt to predict the impact of the final decisions. As a consequence some of our work and results may be outdated when this paper is published.

INTRODUCTION

The Platform Management System (PMS) has been established to coordinate the operation of platform systems and instruments. The management functions are split between ground and space. The ground segment is designated the Platform Management Ground Application (PMGA). The space segment is the Platform Management Application (PMA). The PMS Definition Document (Reference 1) prescribes that each application includes seven functions. Two of these functions are associated with the job of maintaining a platform resource schedule. The Platform Management System must only alter this resource schedule in response to change requests and changes in resource availabilities.

Schedule generation is not a function allocated to the Platform Management System but rather it is performed by a Platform Support Center scheduler which furnishes a short term plan. The PMS manages the short term plan and performs rescheduling (the PMS conflict recognition and resolution function). Rescheduling is of particular interest because it is initiated from three sources: instrument, end user, and the platform itself. As shown in Figure 1, there are three schedulers of different capabilities involved.

- o An on-board scheduler is part of the PMA. Initially, the on-board scheduler will only reschedule to the extent necessary to ensure platform and instrument safety until the next contact.

- o A ground scheduler is part of the PMGA. This scheduler is more capable than the on-board scheduler and will integrate downlinked changes and uplink a revised short term plan.

- o A ground scheduler, shown in Figure 1 as the planning function, is in the Platform Support Center. This scheduler is the most capable of the three and generates and maintains the initial schedule, and furnishes the short term plan to the PMS.

Platforms will be out of contact with the ground more than the manned base. As a consequence, platform operations management functions, both ground and space, need to be more autonomous than those of the manned base to effectively and efficiently meet mission goals. Automated replanning and rescheduling, including on-board real-time schedule maintenance and schedule repair, are required to support autonomous operation of platform systems and instruments.

PLATFORM SCHEDULING

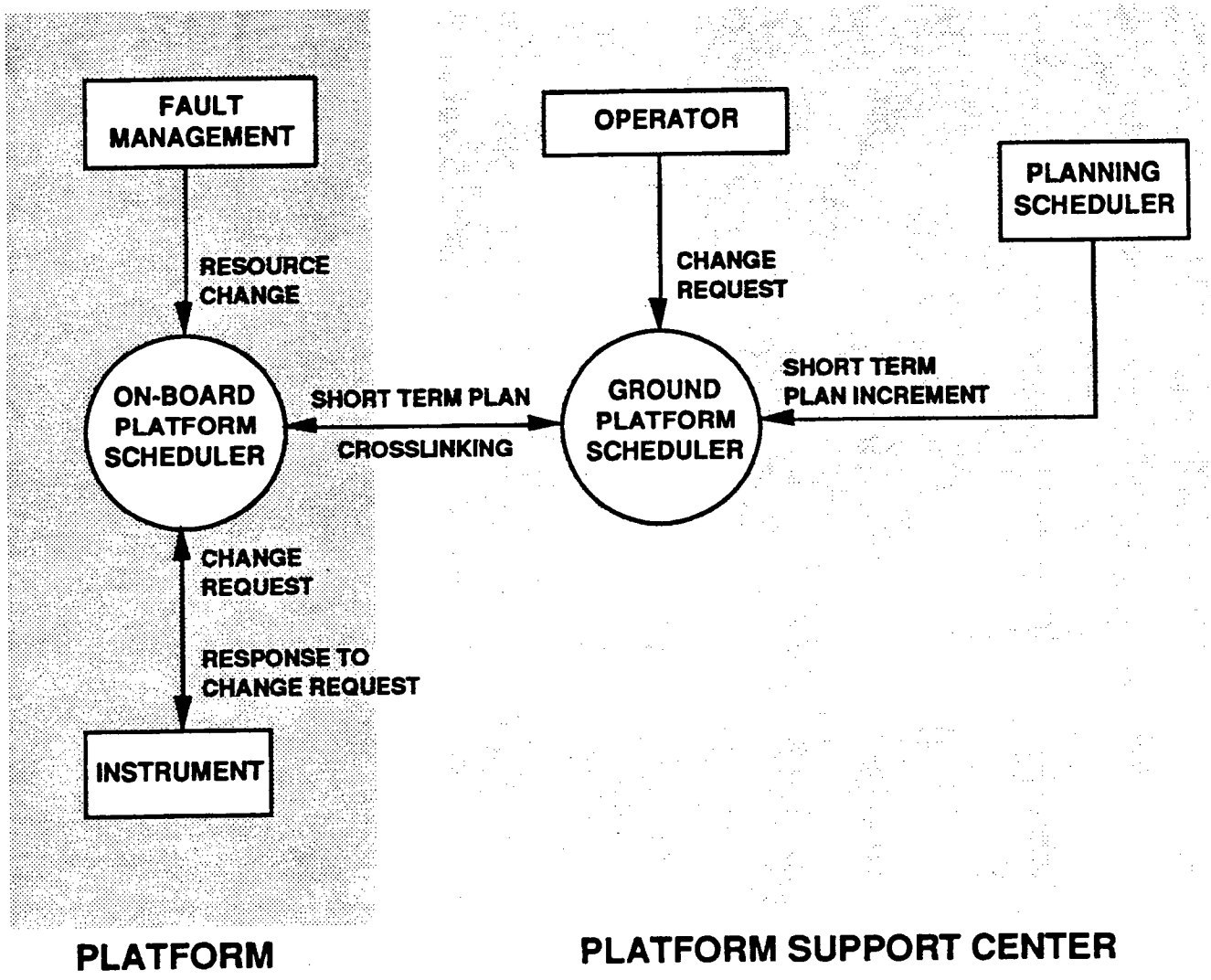

Figure 1

OBJECTIVES

Our FY88 study objectives were to analyze platform resource management, to generate functional requirements for platform scheduling and on-board plan management, and to develop promising alternatives for automation. We recommended both a specific alternative and a phased approach to automated platform resource scheduling. Our recommended alternative was based upon use of exactly the same scheduling engine in both ground and space components of the platform management system. Our phased approach recommendation was based on evolutionary development of the platform. The results of this study were published (References 2 and 3) and distributed in early 1989.

Our FY89 work focused upon implementation of our recommendation for platform resource scheduling in a manner that follows the phased approach and permits the scheduler to evolve over the life of the platform. We generated requirements specifications and designed a prototype platform management system scheduler. We also built a rapid prototype of this scheduler to explore some of the questions raised during the requirements and design work.

ORGANIZATION OF THE PAPER

The first half of this paper provides our rationale for the use of exactly the same scheduling engine for both components of the platform management system and our recommendation for evolutionary development. We begin with a definition of platform scheduling. Next, we introduce the twin problems of schedule maintenance and scheduler coordination. Having established the necessary foundation, we provide our rationale and recommendation.

The second half of this paper discusses our prototype platform management system scheduler. We describe the requirements for this platform scheduler, for on-board processing, and for ground processing. Next, we provide the requirements for crosslinking, a concept that we feel is essential to scheduler coordination. Following a brief description of our rapid prototype, we discuss our conclusions and one particularly subtle open issue under the heading of hooks and scars.

OBJECTIVES

FY88

- o Analyze platform resource management
- o Generate functional requirements
  - platform scheduling
  - on-board plan management
- o Develop automation alternatives
- o Recommend specific alternative/approach

FY89

- o Implement platform scheduler prototype
  - Generate requirements
  - Provide hooks and scars
- o Follow recommended phase approach

PLATFORM SCHEDULING

We define the platform schedule and both ground and on-board segments of this schedule as a set of envelopes arranged on a timeline. An "envelope", or "operations envelope", is a request for a set of resources to be allocated to instrument or platform for some period of time. Operations envelopes do not include commands to conduct the activity. A "resource" is either a measurable quantity or an environment in which to perform an activity that is provided by the platform to an instrument, e.g., an environmental right.

A schedule or short term plan is said to bear "conflicts" when either resources are oversubscribed or an environment is provided to one instrument that is not compatible with the desired environment of another instrument. In the case of the short term plan, conflicts may arise from three sources: instrument, end user, and platform. An example of an end user induced conflict is a request for more of a resource than is currently available, perhaps generated in response to a target of opportunity or other real-time event. A platform induced conflict results from unanticipated reduction in a resource.

Schedule Maintenance

We now define the maintenance problem for a platform scheduler: given a schedule, identify a segment of the schedule that contains conflicts and resolve those conflicts without affecting envelopes outside of the identified segment.

This task differs from that of a "planning" scheduler which generates the initial schedule. For comparison, we provide our definition of the schedule generation problem: given a set of requests, investigate different possible schedules in a search for a schedule that maximizes some figure of merit, e.g., number of requests scheduled.

Scheduler Coordination

We must also consider how the ground scheduler and the on-board scheduler will cooperate. The question of a scheme for cooperation arises because the on-board scheduler and the ground scheduler both have access to a copy of the short term plan and both receive requests to change it. This dual access poses the risk that both schedulers will alter their copies of the on-board plan at the same time. One new plan might not be compatible with the other.

PLATFORM SCHEDULING DEFINITIONS

OPERATIONS ENVELOPE

Request for a set of resources to be allocated to an instrument for some period of time

SCHEDULE / SHORT TERM PLAN

Set of envelopes arranged on a timeline

INITIAL SCHEDULE GENERATION

Given a set of requests, search for a schedule that maximizes some figure of merit

CONFLICT

A resource is oversubscribed or a an environment provided for one instrument is not compatible with the environment desired by another instrument

SCHEDULE MAINTENANCE

Given a schedule, identify a segment that contains conflicts and resolve without affecting envelopes outside the identified segment

SCHEDULER COORDINATION

Given two copies of a schedule, keep the copies compatible in the face of asynchronous and independent requests to change the schedule

We considered three possible ways to carve up the scheduling labor:

o Concurrent Scheduling

The ground scheduler alters its copy of the short term plan when it receives a request. This is driven by a perceived need to be able to immediately tell a user who makes a change request whether or not the request can be scheduled. In the case of changes to the on-board portion of the plan, the ground scheduler incorporates the changes into its version of the plan.

o Local Scheduling

The on-board scheduler schedules all of the requests that affect the on-board portion of the short term plan, and the ground scheduler handles all requests that affect the rest of the short term plan. This scheme prevents the system from being able to immediately tell users the status of their requests to change the on-board portion of the short term plan.

o Pseudo-scheduling

The ground scheduler assists the on-board scheduler in making changes to the short term plan. When the ground scheduler receives a request that falls within the on-board span of the short term plan, it looks at its copy and determines how it would adjust the plan to accommodate the request. The ground scheduler does this without changing its copy of the short term plan. It creates a working copy. When the ground scheduler determines that it could satisfy the request, it saves the sequence of actions used along with the original request. If the ground scheduler is again asked to modify the short term plan, it repeats the procedure, but uses the working copy.

At the next contact, the on-board scheduler downlinks the master short term plan, and receives requests and sequences of actions from the ground scheduler. The ground scheduler then discards the working copy, and begins anew with the current on-board short term plan. When the on-board scheduler receives the request, it first tries the same sequence of actions taken by the ground scheduler. If it can do this without having a conflict occur, the request is scheduled in the way that the ground scheduler determined. If it cannot, then the on-board scheduler decides how to schedule the request on its own.

DIVISION OF SCHEDULING LABOR

CONCURRENT SCHEDULING

Ground Scheduler

- Alters its copy of the short term plan
- Provides user with immediate feedback

On-board Scheduler

- Provides on-board changes to ground
- Receives updated, altered plan from ground

LOCAL SCHEDULING

Ground Scheduler

- Alters only short term plan not yet uplinked
- Uplinks requests to change on-board plan

On-board Scheduler

- Alters only on-board portion of short term plan
- Downlinks requests to change remaining plan

PSEUDO-SCHEDULING

Ground Scheduler

- Assists on-board scheduler

On-board Scheduler

- Mimics ground scheduler's actions

PHASED APPROACH

We developed a conceptual model for implementation of the platform scheduler and for automation of platform scheduling. Our model is based upon an assumption that the platform itself will evolve over time. Our conceptual model provides for three stages of development over the life of the platforms. We do not presume to establish dates for each stage in the lifetime of the platform but simply name the stages of development: baseline, midterm, and final. These stages of development are shown in Figure 2 and discussed below.

o Baseline

Initially, we see both on-board and ground platform schedulers as simple schedule managers. Either local or concurrent scheduling may be followed. Given the need to be able to immediately tell a user who makes a change request whether or not the request may be scheduled, we assume that concurrent scheduling will be followed. The ground scheduler maintains the master copy of the short term plan and uplinks replacement for the on-board plan after first incorporating any on-board changes (simple safing actions) since the last contact.

o Midterm

At this stage, we see the on-board scheduler as a more sophisticated schedule manager with limited automated scheduling capability (enhanced safing) while ground scheduling is automated, but not yet autonomous. Pseudo-scheduling is followed with the ground scheduler uplinking both change requests and the sequences of actions that will schedule these requests provided the segment of the on-board plan affected has not changed since the last contact.

o Final

In the final stage, we see platform scheduling as both automated and autonomous. The platform scheduler takes the entire short term plan into account in resolving conflicts rather than dealing with limited segments. The platform scheduling requirement for scheduler coordination is satisfied by providing exactly the same scheduling engine in space and ground applications.

PLATFORM SCHEDULER DEVELOPMENT

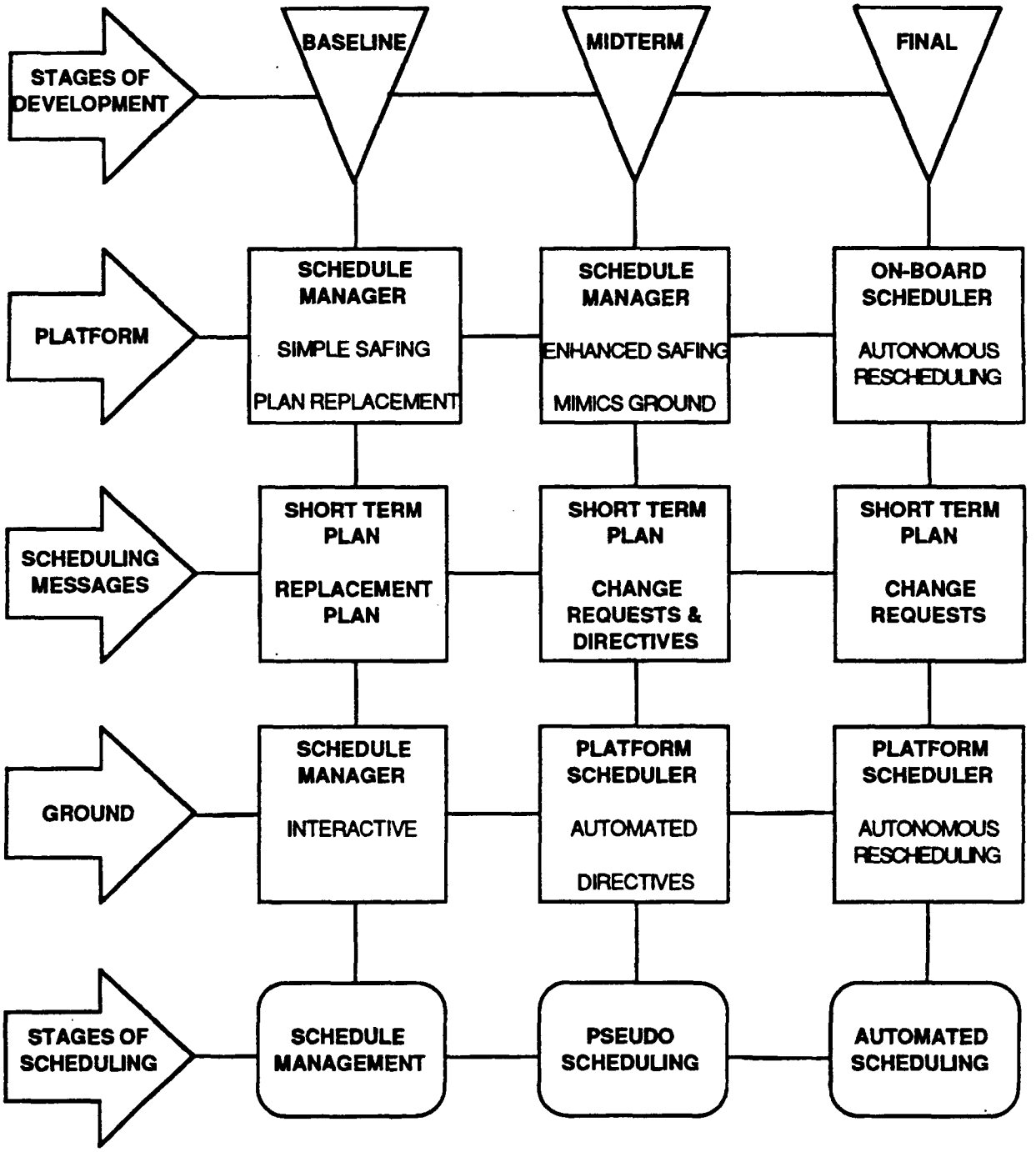

Figure 2

PROTOTYPE

As we have discussed, two platform schedulers are altering two schedules, with one schedule a subset of the other. The system must act in such a way that the ground and space components both agree on the on-board schedule immediately after each contact.

Prototype Operations Concept

Principal investigators submit requests for resources to the planning scheduler in the Platform Support Center. The planning scheduler generates the initial schedule and maintains the schedule through the start of the short term plan furnished to the platform schedulers. The planning scheduler forwards all requests that have a start time within the span of the short term plan.

Each request has a priority that the platform schedulers use to adjudicate conflicts. If two requests have the same priority and are in contention for the same resources, then we use the order in which the requests are received by the scheduler to determine a unique effective priority. A high priority, late arriving request can cause an existing but low priority request to be removed from the schedule.

A "smart" instrument may submit a change request to the on-board scheduler. This scheduler processes the request if it has a start time that falls within the current span of the on-board short term plan. If the request has a later start time, the on-board scheduler defers it to the ground scheduler at the next contact.

When fault management detects a change in platform resource capacities, it provides the on-board scheduler with the new resource availabilities. At the next contact with the ground, the on-board component "crosslinks" the schedules so that the space and ground applications have identical copies of the on-board short term plan and identical knowledge of the resource availabilities.

PROTOTYPE OPERATIONS CONCEPT

PLANNING SCHEDULER

- o Generates and maintains initial schedule
- o Furnishes short term plan to platform schedulers
- o Passes user change requests within span of plan

GROUND SCHEDULER

- o Processes all change requests within span of plan
- o Uses request priority to adjudicate conflicts
- o Crosslinks schedules and resource requests

ON-BOARD SCHEDULER

- o Processes only change requests within span of on-board plan
- o Uses request priority to adjudicate conflicts
- o Knows present platform resource availabilities
- o Crosslinks schedules and resource availabilities

BASELINE REQUIREMENTS

The operations concept discussed above allows many different sets of requirements, especially in connection with crosslinking. We used prototyping to identify one set of requirements that will allow this high-level operations concept. The requirements provided here are not the only requirements that will enable this operations concept.

Requirements on the Scheduling Engine

Our scheduling engine is a simple priority scheduler that allocates resources to requests depending upon resource availability and the priority of the request. For baseline capability, the scheduler needs to process only very simple kinds of requests. Each request has a specific start-time and duration, and includes a specification of all resources needed to accomplish some activity and the required environment conditions.

We assume that the baseline scheduler should allow the expression of some scheduling constraints in connection with the placement of a request on the timeline relative to other requests. However, these constraints have not yet been defined and our rapid prototype does not presently allow such scheduling directions.

With these simple requests, the scheduling engine satisfies three baseline requirements:

- o Do not schedule a request if that will oversubscribe resources.
- o Do not schedule a lower priority request if a higher priority request can be scheduled.
- o Maintain the schedule so that as many requests as possible are scheduled at all times.

BASELINE REQUIREMENTS

SCHEDULING REQUESTS

- Priority
- Start time
- Duration
- Resources
- Constraints
- Environmental Conditions

SCHEDULING ENGINE

- Do not schedule a request if that will oversubscribe resources
- Do not schedule a lower priority request if a higher priority request can be scheduled
- Maintain the schedule so that as many requests as possible are scheduled at all times

On-board Processing Requirements

The baseline on-board scheduler only adds or defers change requests from instruments. The on-board scheduler processes all requests that fall within the span of the on-board short term plan as well as those that fall outside the time span of the on-board plan by less than the period between regularly scheduled contacts. Those change requests with start times within one contact period of the end of the on-board short term plan would otherwise have to be downlinked, processed, and uplinked during the crosslink process, which is not necessarily going to be feasible.

The on-board scheduler must alter the priorities of requests dynamically if, as in our prototype, a simple priority scheduler is to be used. It is the simplest way to prevent the ground scheduler from removing requests scheduled on-board. It ensures that the ground and space components have the same version of the on-board schedule immediately after each contact.

No request is submitted to our prototype, whether acting as the on-board scheduler or the ground scheduler, with a priority greater than 4. We increase the priority of any request scheduled on-board so that it is in a range from 5-9. Further, a request that is active (start time less than current time) is given the highest priority of 10. This scheme, while not the only possible alternative, does guarantee two necessary characteristics of the schedule maintained by our priority-based scheduler prototype:

- Since active envelopes are given the highest priority, the scheduler will remove active envelopes from the schedule in response to a degradation in resources only as a last resort.

- When the schedules (on-board and ground versions of the short term plan) are merged on the ground, all requests scheduled on-board will be scheduled as well by the ground scheduler.

BASELINE REQUIREMENTS

ON-BOARD PROCESSING

- Add or defer change requests from instruments
- Process requests within the time span of the on-board plan
- Process requests within one contact period beyond the time span of the present on-board plan
- Defer all requests beyond the present span plus the time between ground contacts (nominally one orbit)
- Alter the priorities of the scheduled requests
- Remove active envelopes from the schedule only as a last resort

Ground Processing Requirements

The baseline ground scheduler both adds and deletes requests. The ground scheduler processes all requests that fall within the span of the short term plan. It also merges the on-board versions of the short term plan into the ground short term plan during crosslink.

The requests that the ground scheduler processes (ending with a status of either scheduled or not) and that fall within the span of the on-board short term plan are uplinked at the next contact period. The on-board short term plan time span is extended by the time between contacts at the start of each contact, just prior to crosslink.

Crosslinking Requirements

The crosslink process is the sequence of steps that the on-board and the ground scheduling systems must accomplish to ensure that the on-board short term plan and the corresponding portion of the ground short term plan are exactly the same immediately after each contact.

Our scheduler prototype implements the crosslink process in three steps:

- o The crosslink is made at a regularly scheduled contact time (perhaps once each orbit) and both schedulers increase the time span of the on-board short term plan by one contact period.

- o The ground scheduler uplinks all requests with a start-time that falls within the (updated) span of the on-board short term plan. The on-board scheduler adds them to the schedule one-by-one and screens for conflicts after each addition. At the completion of this process, the platform has an executable on-board short term plan.

- o As the final step, the on-board scheduler sends the resource availabilities, the on-board short term plan, and all deferred and unscheduled requests to the ground. The ground scheduler merges the present on-board plan with the rest of the short term plan, screens the new schedule against the current resource availabilities, and processes all deferred requests.

BASELINE REQUIREMENTS

GROUND PROCESSING

- Add or delete change requests
- Process requests within the time span of the short term plan
- Uplink requests within the time span of the on-board schedule at the next contact

CROSSLINKING

- Crosslink at a regularly scheduled contact time
- Increase time span of the on-board plan by the interval between contacts prior to crosslink
- Uplink all requests with a start time that falls within this time span
- Downlink on-board plan, deferred requests, unscheduled requests and resource availabilities

RAPID PROTOTYPE DESCRIPTION

We built a rapid prototype of the platform scheduler to explore some of the questions raised during the requirements and design work. This rapid prototype is designed to be both the on-board scheduler and the ground scheduler. As the on-board scheduler, the prototype acts in exactly the same way as the ground scheduler except that it dynamically adjusts the priorities of requests it can schedule and in execution. We use this dynamic adjustment of priorities to prevent requests that are scheduled on-board from being unscheduled on the ground and to guarantee, that in instances of resource degradation, the on-board scheduler will not remove active requests except as a last resort.

Our rapid prototype implements both request management and conflict recognition and resolution functions. Request management first determines whether to process (add, delete, replace) a request or to defer a request. A request is deferred if it falls outside the span of the current short term plan. After all requests have been processed, the conflict recognition and resolution function is called to ensure a conflict free plan. If a conflict is found, this function resolves it by unscheduling all requests at that time and then attempting to add them back to the schedule in priority order.

Unscheduling differs from deleting a request. The rapid prototype will unschedule lower priority requests to accommodate a higher priority request. However, our prototype does not remove the unscheduled requests from the schedule. It only changes the status of these requests. We retain unscheduled requests since subsequent changes may allow these requests to be rescheduled, e.g., higher priority requests may be unscheduled or deleted.

The rapid prototype is menu-driven as shown in Figure 3. Our implementation allows the user to crosslink at any time. When crosslink is selected, the rapid prototype sequences through the crosslink steps waiting only for the user to grant permission to proceed. This manual capability enables us to easily demonstrate crosslinking. A fully automated capability will be needed to support emergency crosslink.

RAPID PROTOTYPE MENU SYSTEM

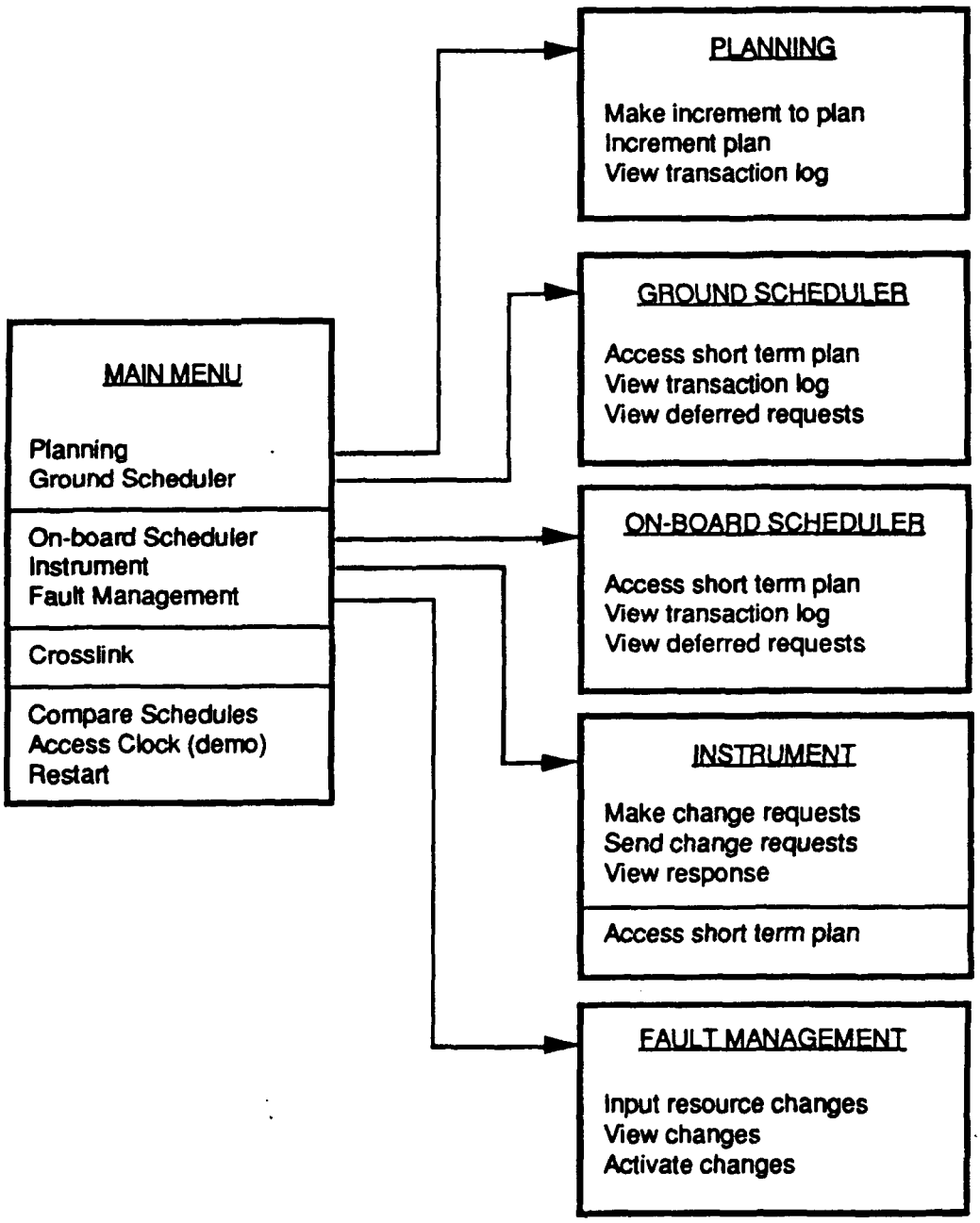

Figure 3

HOOKS AND SCARS

The prototype platform scheduler work and our rapid prototype were guided by the requirements generated in our FY88 study. These requirements, and the accompanying methodology for evolution, did not use hooks as the mechanism for evolving the capabilities of the scheduler. We relied on module replacement.

Module replacement is a reasonable strategy for evolution, but requires sufficiently powerful data structures at the beginning of the life cycle. These data structures should, even in the baseline, provide all of the information to the platform scheduler that it will need in order to automatically, autonomously reschedule.

The envisioned data structures will express all possible ways that the platform scheduler can satisfy the need for resources in support of an activity. By the final stage in development, if any activity must be removed to make room for a higher priority request, the scheduler will look at the request for this lower priority activity to see how it can be rescheduled.

As emphasized, unscheduling differs from deleting a request in our rapid prototype. The prototype will unschedule requests in order to accommodate a higher priority request. Unscheduling only changes the status of these requests. Request management tries to add these requests back to the schedule when either resource availabilities change or a higher priority request is deleted from the schedule.

One complication in making changes to the schedule may not be immediately obvious. A request for one resource can imply a request for another resource. Power may be an implied resource. The actual ceiling for power varies not only because of the need for power to run the platform and operate instruments, but also for requested resources that only imply the use of power, such as a tape recorder. Further work is needed on this issue.

HOOKS AND SCARS

MODULE REPLACEMENT

- Provides a reasonable strategy for evolution
- Requires sufficiently powerful data structures at baseline

ENVISIONED DATA STRUCTURES

- Express all possible ways to satisfy a request
- Reduce number of unscheduled requests
- Improve utilization of platform resources

UNSCHEDULING REQUESTS

- Differs from deleting requests
  - Status flag is changed
  - Request may still be accessed
- May reschedule previously unscheduled requests
  - Higher priority request is itself deleted or unscheduled
  - Resource availabilities change

IMPLIED RESOURCES

- Implied in a request for another resource

ACKNOWLEDGEMENTS

The work described in this paper involves a large number of people. This work could not have been performed without the support of Gregg Swietek, Strategic Plans and Programs Division, Office of Space Station. Specific thanks are due to Don Rosenthal, Ames Research Center, whose presentation of lessons learned at the Boulder workshop (Reference 4) helped to guide this work. Finally, credit for specific concepts and for implementation of the rapid prototype of the schedule belongs to James Retter, System Sciences Division, Computer Sciences Corporation.

REFERENCES

1. Goddard Space Flight Center, "Platform Management System (PMS) Definition Document", Revision 3, September 1987.

2. Computer Sciences Corporation, "Methodology for Platform Management System (PMS) Scheduling Automation", CSC/TM-88/6144, March 1989.

3. Computer Sciences Corporation, "Platform Management System (PMS) Scheduling Requirements Study", CSC/TM-88/6036, March 1989.

4. Rosenthal, Donald, "Lessons Learned from HST Scheduling", Workshop on Operations Planning and Scheduling Systems for the Space Station Era, University of Colorado at Boulder, Boulder, Colorado, August 1987.

5. NASA, "Space Station Program Definition and Requirements Document", Section 4, Part 3: Space Operations Requirements, Revision B, SSP 30000, Space Station Program Office, Reston, Virginia, October 1988.