

524-63
N93-18683
37250

REAL-TIME CONTINGENCY HANDLING IN MAESTRO

Daniel L. Britt and Amy L. Geoffroy
Martin Marietta Astronautics Group
P.O. Box 179, ms XL4370
Denver, CO 80201

Abstract

A scheduling and resource management system named MAESTRO has been interfaced with a Space Station Module Power Management and Distribution (SSMPMAD) breadboard at Marshall Space Flight Center (MSFC). The combined system serves to illustrate the integration of planning, scheduling and control in a realistic, complex domain. This paper briefly describes the functional elements of the combined system, including normal and contingency operational scenarios, then focusses on the method used by the scheduler to handle real-time contingencies.

I. Introduction

For the past six years a team at Martin Marietta has been developing an integrated approach to scheduling, resulting in the implementation of a robust prototype scheduling system called MAESTRO [Geoffroy, Gohring & Britt, 1991]. During the same time frame another group at Martin Marietta has been building a hardware/software testbed to study various concepts in the automation of electrical power management, the Space Station Module Power Management and Distribution (SSMPMAD) system. In 1988 an initial version of the SSMPMAD system integrated with MAESTRO was delivered to Marshall Space Flight Center. Since then both the SSMPMAD system and the scheduler have gone through several revisions, and a major delivery of new software occurred in June of 1991. This paper describes that combined system, highlighting those aspects of it that illustrate concepts in integrated planning, scheduling and control. We focus on the replanning and rescheduling processes used in MAESTRO to respond to real-time contingencies, unexpected changes in the state of the power system that cause a schedule currently being executed to become invalid.

Section II defines some terms used in the rest of the paper. Section III describes the functional architecture of the system. In section IV are presented two operational scenarios, one for normal operations and one which describes a possible contingency. Section V provides a description of the processes carried out by the scheduler to effect real-time replanning and rescheduling, including timing issues. In section VI we conclude with indications of possible future directions for this research.

II. Definitions

For the purposes of this discussion we will make use of the following restricted definitions. *Planning* is defined to be the process of specifying goals to be achieved onboard a spacecraft, and further, of specifying the activities which will achieve those goals. This involves determining these activities' structure as well as constraints on the execution of them. *Activities*, in turn, are defined to be sequences of subtasks which accomplish the desired goal. *Scheduling* is defined to be the process of selecting some subset of these activities and specifying exact start/end times and resource assignments for their component subtasks. A *valid schedule* is a specification of start/end times and resource assignments for a set of activities such that the activities may be executed as scheduled. A *contingency* arises when a previously valid schedule becomes invalid as a result of a change in the assumptions upon which that schedule was based. The term *real-time* is used here to mean "during execution of the activities on a schedule". This does not have the connotation from control theory that a real-time event must be responded to within microseconds, but rather is used to differentiate between actions that are occurring at the moment as opposed to those that will occur at some point in the future. A *load* is the use of electrical power by a piece of equipment.

The authors would like to acknowledge John Gohring of Martin Marietta Western Internal Systems and Joel Riedesel of Martin Marietta Astronautics Group for their significant contributions to this report and to the work described herein.

III. Functional Architecture

Figure 1 depicts the functional elements of the combined SSMPMAD/MAESTRO system and relationships among these elements. Briefly, the Activity Editor is used to create definitions for activities which accomplish goals desired by the user. MAESTRO is used to select and schedule a subset of these activities, and to save the resultant schedule(s) out to files. The Transaction Manager (TM) serves as a communications port, facilitating specific types of communications between MAESTRO and the rest of the system during breadboard operation. The Front End Load Enable Scheduler (FELES) creates schedules of power system events (such as closing switches) from saved schedule files. The Communications and Algorithmic Controller (CAC) distributes

schedules among Load Centers (LCs), into which are incorporated Lowest Level Processors (LLPs). These LLPs actually control hardware switches on the power system breadboard, as well as monitoring the states of various sensors distributed throughout the system. The Fault Recovery And Management Expert System (FRAMES) performs fault isolation, diagnosis and recovery for the power system, and communicates with the scheduler during real-time contingencies. The Load Priority List Management System (LPLMS) maintains a list of active loads in a prioritized order such that if there is a need to quickly reduce power consumption in a portion of the breadboard, loads can be shed (turned off) in an order that minimizes the impact of this load shedding.

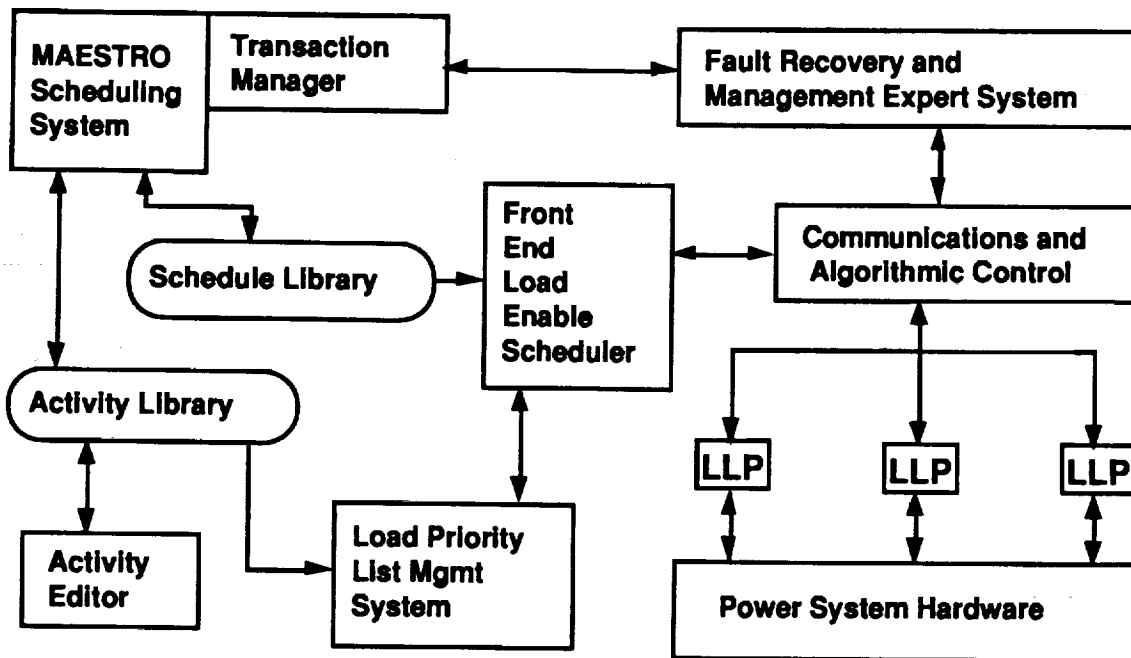


Figure 1. Functional architecture of the MAESTRO/SSMPMAD combined system.

A portion of the actual power circuits on the breadboard is depicted in figure 2. Note that several 1-kilowatt Remote Power Controllers (RPCs) can be attached to a single 3-kilowatt RPC. Thus it is possible to overload an intermediate RPC without

overloading any of the lower-level RPCs connected to it. For this reason it is necessary to represent the entire power path for each power-using resource to the scheduling system, rather than just representing total power consumed by each activity.

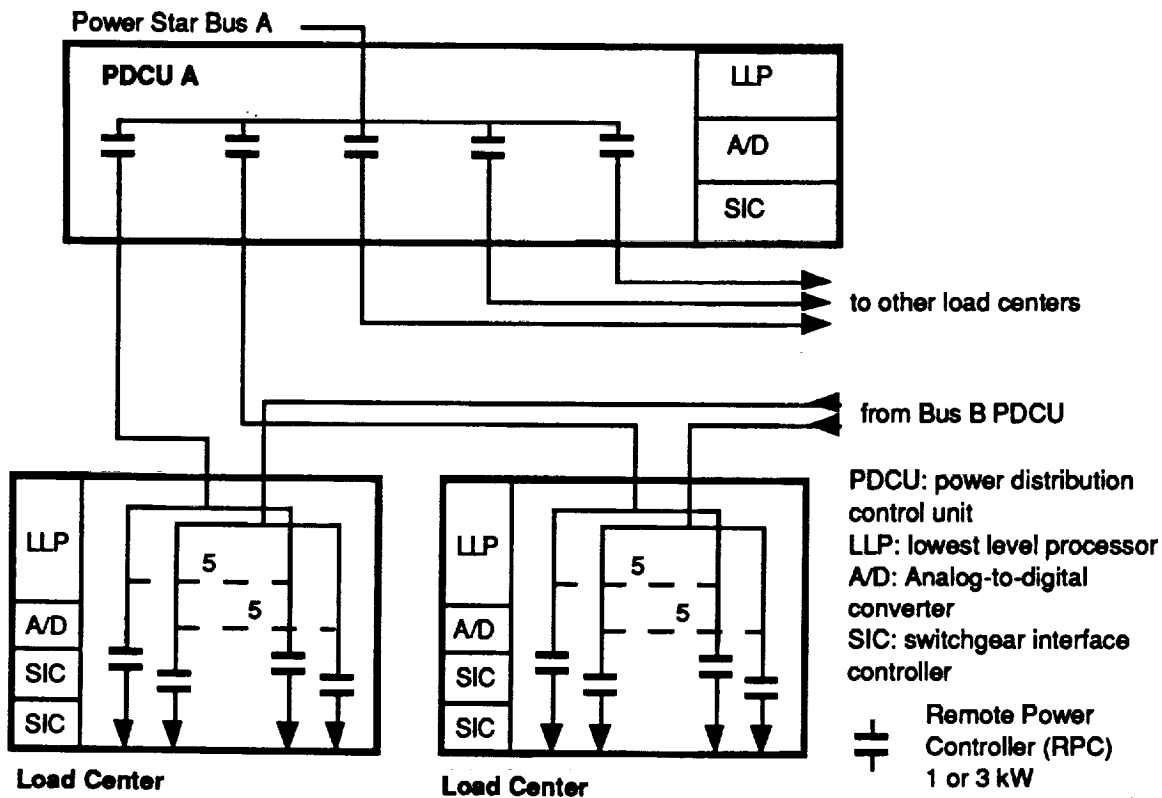


Figure 2. Representative schematic of a portion of the SSMPMAD breadboard.

IV. Operational Scenarios

Normally, a user will interact with the activity editor to create a set of activities to be scheduled, saving these activities' definitions in an activity library. In that or another session, the user will run the scheduler to create one or more initial schedules of these activities. These schedules will be saved into a schedule library. When a user wishes to operate the power system breadboard, s/he uses the SSMPMAD interface to select a saved schedule, initialize the system and execute that schedule. The FELES first obtains a saved schedule and translates a portion of it (roughly one-half hour of activity) into a series of power system events, specifying at what times and power levels each RPC is to be turned on. The LPLMS takes this schedule of power system events and creates a list of loads to shed in an emergency power reduction. The event schedule and priority list are transmitted to the CAC, which distributes them among the LLPs as appropriate. The CAC also maintains a system clock, coordinating timing for the various elements.

Execution of the distributed schedule proceeds with the LLPs directing the RPCs to close and open switches at the times specified by their respective event schedules. The RPCs monitor voltage, current, temperature and other parameters of their operations.

Prior to the expiration of the timeline increment being executed, the FELES will acquire another increment from the saved schedule, translate it into power system events, and transfer it to the CAC, which distributes it to the LLPs. At a specified time, the LLPs stop executing the old increment event list and begin executing the new one.

When an anomalous condition (such as over-current or under-voltage at a switch) is detected by one of the RPCs, it automatically takes a safing action, if possible. The LLP controlling it reports this event to FRAMES, which gathers together all available information about the fault, isolates it, and compiles a list of system configuration status changes resulting from the fault. These changes can include a load being switched to a

redundant power source (redundancy switching), an RPC going out of service, the deliberate shutdown of a load to reduce power consumption (load shedding), or a reduction in power available at an RPC and the expected duration of that reduction. This list of changes is then communicated to the scheduler, which revises the activity schedule to reflect the changes and makes the new schedule available to the FELES. It creates a new event list, which is distributed to the LLPs along with a time tag indicating when to begin executing the new schedule.

V. The Real-Time Rescheduling Process

When a power system anomaly occurs, MAESTRO will get a set of information from FRAMES through the TM. This information will include the current time in addition to redundancy switch, load shed, power availability change, and RPC out-of-service messages. These messages will include the time the event occurred, and if applicable the duration of the change. MAESTRO follows a three-step process to handle these messages and revise the schedule. It 1) modifies the schedule to reflect changes made to it by the power system and to remove resource and temporal constraint violations for activities not yet begun, 2) tries to find ways to create and schedule continuations for interrupted activities, and 3) tries to schedule any activities that can take advantage of the resources released by the interruption of others. The first step results in a valid but possibly not very efficient schedule. It is carried out as quickly as possible to ensure that a workable schedule can be in place soon, reducing the likelihood that adherence by the power system to the old (invalid) schedule will result in a cascade of faults registered by that system. The second and third steps will only be attempted if there is sufficient time to get something useful done. Management of its own computation time is a difficult issue for a real-time rescheduler. It must project a time when it will have a valid schedule available, including the time it takes to transmit that schedule to the entities responsible for carrying it out, then not make changes to the schedule (other than those already made by the power system) that would need to be acted upon before they are

received by the power system. For example, if at 10:00 a contingency occurs, and the scheduler determines that an interrupted activity can be continued at 10:05, but this information cannot be transmitted to the power system until 10:08, then the schedule is invalid the moment the system begins to execute it. In this example the scheduler could specify that the activity be continued at 10:08, but not before.

The actual structure used to control the three-step process mentioned above is a prioritized list of command queues. As information comes in from FRAMES, it is routed to one of several command queues, for action as soon as MAESTRO has nothing more important to take care of. Resource availability changes appear in one queue, while redundancy switches are in another and load sheds in a third, for example. MAESTRO will be in a wait state until something appears on one of its command queues, at which time it will process a command from the highest priority queue that has an item, then check all the queues again for new items, returning to the wait state when no items remain.

MAESTRO will add items to its command queues as a result of its own processing. Handling a resource availability change, for example, will cause MAESTRO to add a command to check for resource constraint violations. If a violation is found and an activity interrupted, MAESTRO will add a command to try to plan and schedule a continuation of that activity.

Activity continuation is the single automated planning function within MAESTRO. When initially creating activities, the user specifies ways and conditions under which each subtask may be continued if it is temporarily interrupted. Three continuations are currently represented for each subtask. These are effectively operators that can be selectively applied to achieve the goal of a completed activity performance. First, the unexecuted portion of an interrupted subtask may be skipped, with a parameter stating how much time the subtask must execute prior to the interruption. A data collection subtask could be terminated early and data analysis begun, for

example. Second, a subtask may be continued after a sufficiently brief interruption. Finally, the interrupted subtask may be started over again, making use of states set by previous subtasks but not using the progress gained in the interrupted subtask.

The scheduler will create a new activity model appropriate for a particular type of continuation using information from the interrupted activity and possible continuations specified by the user for that activity. Each of the above continuations has different implications for the rescheduling of the subtasks following the interrupted one, so MAESTRO must try various options in order to find a viable placement for the new activity. MAESTRO can represent temporal constraints between activities, sometimes necessitating the consideration of more than one continuation model at once. This complexity combines with the time limitations on rescheduling to prohibit MAESTRO from finding the "best" way to continue an activity - it simply accepts the first viable continuation found. Attempts are heuristically ordered such that higher-value continuations are tried earlier, however. Note that in many cases no continuation will be possible, in which case the work done to represent the current state of the system is all that can be accomplished for a particular activity. Note also that safing actions are not scheduled but rather are carried out immediately and automatically by the subsystems involved.

As each continuation attempt is made, the system consults the system clock, abandoning further attempts at the point where they would cause changes made to the schedule to be unimplementable. When all continuation attempts have been tried (and there may be none tried), if there is still time, the scheduler will attempt to add new performances of activities to the schedule. System time is checked after each schedule addition, and this process ends when time runs out or no more activities can be added to the schedule. At that point the schedule is made available to the FELES, and schedule execution proceeds as previously described.

VI. Future Directions and Related Work
Work is continuing on MAESTRO, as it is on SSMPMAD. The scheduler needs to be enhanced to manage the timing and consistency issues that arise when a user wishes to alter a schedule that is currently executing. We also intend to enhance the representational as well as computational power of the system. The current methods for finding a way to continue an interrupted activity are cumbersome and depend too much on initial user input into the representation of the subtasks. A more appropriate method would be to have an intelligent system monitoring each experiment or other major activity, with the capability to plan continuations based on an accurate assessment of the state of the activity.

We have begun a task similar to the MAESTRO/SSMPMAD integration for Kennedy Space Center under the Advanced Launch Processing (ALP) contract. In that effort we will build a system executive capable of coordinating the actions of multiple Knowledge-Based Autonomous Test Engineer (KATE) systems [Parrish & Brown, 1991]. These systems are used to monitor and control individual launch vehicle subsystems during testing and launch, but are independent of one another. The system executive will interface with the KATE systems as well as with higher-level launch flow management functions, enhancing integrated vehicle systems tests and reducing launch costs.

VII. References

Geoffroy, A.L. Gohring, J.R. & Britt, D.L. (1991) Sharing intelligence: Decision-making interactions between users and software in MAESTRO. *Telematics and Informatics*. 8 (3/4).

Parrish, C.L. & Brown, B.L. (1991) Knowledge-Based Autonomous Test Engineer (KATE). *Technology 2001 Conference*. NASA. December, 1991, San Jose, CA.