Dual Use Space Technology Conference

NASA Scheduling Technologies

A compilation of information concerning
planning and scheduling tools designed by NASA

Jerry R. Adair
Intelligent Systems Branch
NASA/JSC

February 1994

Table of Contents

Abstract

This paper is a consolidated report on ten major planning and scheduling systems that have been developed by the National Aeronautics and Space Administration (NASA). A description of each system, its components, and how it could be potentially used in private industry is provided in this paper. The planning and scheduling technology represented by the systems ranges from activity based scheduling employing artificial intelligence (AI) techniques to constraint based, iterative repair scheduling. The space related application domains in which the systems have been deployed vary from Space Shuttle monitoring during launch countdown to long term Hubble Space Telescope (HST) scheduling. This paper also describes any correlation that may exist between the work done on different planning and scheduling systems. Finally, this paper documents the lessons learned from the work and research performed in planning and scheduling technology and describes the areas where future work will be conducted.

Introduction

NASA has performed an extensive amount of work in the area of planning and scheduling technology both in terms of research and real-world applications. This paper will present this work and discuss the goals, strategies, methods of implementation and results of each project on a case by case basis. Each project will be described with introductory remarks, planning and/or scheduling technology details of implementation, and closing remarks. In addition, the relation of a project to the private sector and how it could be used therein (i.e., its dual-use potential) will be outlined.

NASA Planning and Scheduling Projects

SPIKE

We begin with an activity-based scheduler called SPIKE that was designed to perform scheduling tasks for the Hubble Space Telescope (HST). It was developed at the Space Telescope Science Institute (STScI) in Baltimore, Maryland. SPIKE exploits AI techniques for constraint representation and for scheduling search. The aim of SPIKE is to allocate observations to timescales of days to a week, observing all scheduling constraints, and maximizing preferences that help ensure that observations are made at optimal times. SPIKE has been in use operationally for HST since shortly after the observatory was launched in April 1990. Although developed specifically for HST scheduling, SPIKE was carefully designed to provide a general framework for similar (activity-based) scheduling problems.

HST scheduling is a large scale problem: some 10,000 to 30,000 observations per year must be scheduled, each subject to a large number of operational and scientific constraints. The scheduling of the HST has been divided into two processes: the first is long-term scheduling, which allocates observations to week-long segments over a scheduling period of one year or more in duration.

This task is the responsibility of the SPIKE system. Individual weeks are then scheduled in more detail by the Science Planning and Scheduling System (SPSS), which orders observations within the week and generates a detailed command sequence for the HST control center at NASA Goddard Space Flight Center.

Scientists who wish to use the HST write observing programs to achieve their goals and these are sent to STScI in machine-readable form over national and international computer networks. They are then translated by an expert system called Transformation into a form suitable for scheduling within SPIKE. SPIKE treats the construction of a schedule as a constrained optimization problem and uses a heuristic repair-based scheduling search technique. The SPIKE algorithm has desirable "anytime" characteristics: at any point in the processing after the initial guess has been constructed, a feasible schedule can be produced simply by removing any remaining activities with constraint violations. The repair heuristics used by SPIKE are based on a very successful neural network architecture developed for SPIKE and later refined into a simple symbolic form which has made the original neural network obsolete. The adopted initial guess heuristic selects the most-constrained activities to assign first, where the number of minimum conflicts times is used as the measure of degree of constraint. If there remain gaps when all conflicting activities have been deleted (removed), then a simplistic best-first pass through the unscheduled activities is used to fill them.

SPIKE provides support for rescheduling operations in several ways. Two worth mentioning in particular are the task locking and conflict-cause analysis features. Tasks or sets of tasks can be locked into place on the schedule, and will thereafter not be considered during search or repair. These tasks represent the fixed points on the schedule. Conflict-cause analysis permits the user to force a task onto the schedule, and then display both what constraints have been violated and by which other tasks. The conflicting tasks can be unassigned if desired either individually or as a group and returned to the unscheduled tasks pool. The primary lesson learned from SPIKE development (which applies to similar systems) is to build in the expectation of change from the outset.

The implementation of SPIKE started in early 1987 and was initially based upon Texas Instruments Explorers as the hardware and software environment. The SPIKE graphical user interface was implemented in KEE Common Windows, but the remainder of the system, used only Common Lisp and the Flavors object system. At HST launch, STScI had a complement of 8 TI Explorers and microExplorers user for SPIKE operation, development and testing. In late 1991 SPIKE was moved from Explorers to Sun SparcStation IIs as the primary operations and development workstation. All of the Flavors code was automatically converted to the Common Lisp Object System. The Lisp used on the SparcStation is Allegro Common Lisp from Franz Inc. Allegro CL supports a version of CommonWindows based on X-windows, and so the user interface continues to operate on UNIX platforms as it did on the Explorers.

The dual use potential for SPIKE lies in large-scale scheduling problems that need to employ heuristics in the search algorithm. SPIKE can be modified to accommodate a new problem domain that might be found in private industry. For more information on SPIKE, contact the Space Telescope Science Institute, 3700 San Martin Drive, Baltimore, Maryland, 21218. The contact is Mark D. Johnston.

Additional HST Scheduling Research

Next, we take a look at research work performed for HST scheduling by the Robotics Institute at Carnegie Mellon University which is related to the SPIKE system described in our previous project description. This work focuses on the short term scheduling of the HST and can best be described through an examination of its attributes: decomposability and scalability.

Existing and planned space-based observatories vary in structure and nature, from very complex and general purpose, like the HST, to small and targeted to a specific scientific program, like the Submillimeter Wave Astronomy Satellite (SWAS). However the fact that they share several classes of operating constraints (i.e., periodic loss of target visibility, limited onboard resources, etc.) does suggest the possibility of a common approach. The complexity of this problem stems from two sources. First, both types of observatories have the difficulty of the classical scheduling problems: optimization of objectives relating to overall system performance (e.g., maximizing return of science data), while satisfying all constraints imposed by the observation programs (e.g., precedence and temporal separation among observations) and by the limitations on the availability of capacity (e.g., observations requiring different targets cannot be executed simultaneously). Secondly, a safe mission operation requires the detailed description of all the transitions and intermediate states that support the achievement of observing goals and are consistent with an accurate description of the dynamics of the observatory; it is this aspect that constitutes a classical planning problem. Yet another characteristic of the problem is its large scale. To effectively deal with problems of this size, it is essential to employ problem and model decomposition techniques, which is where the research in this project was focused.

The problem in the HST short term scheduling domain is the efficient generation of schedules that account for the major telescope's operational constraints and domain optimization objectives. The basic assumption is to treat resource allocation, or scheduling, and auxiliary task expansion, or planning, as complementary aspects of a more general process of constructing behaviors of a dynamic system. The natural approach to solving the problem is then an iterative posting of constraints extracted either from the external goals or from the description of the system dynamics; consistency is tested through constraint propagation. In addition, the use of abstraction, model decomposability and incremental scaling, and exploiting opportunism to generate good solutions are techniques researched in this effort. These three are described in more detail in the following paragraph.

Briefly, where models are expressed in terms of the interacting state variables of different components of the physical system and its operating environment, an abstract model is one which summarizes system dynamics in terms of more aggregate structural components or selectively simplifies the represented system dynamics through omission of one or more component state variables. In HST, problems are naturally approached by decomposing them into smaller sub-problems separately and then assembling the sub-solutions. We can judge how the problem solving framework supports modularity and scalability by two criteria: 1) the degree by which heuristics dealing with each sub-problem need to be modified when adding sub-problem assembly heuristics to the problem solver, 2) the degree of increase of the computational effort needed to solve the problem versus the one needed to solve the component sub-problems. Finally, for overall sequence development, a simple dispatch-based strategy was used: simulating forward in time at the abstract level, the candidate observation estimated to incur the minimum amount of wait time (due to HST reconfiguration and target visibility constraints) was repeatedly selected and added to the current sequence. This heuristic strategy, termed "nearest neighbor with look-ahead" (NNLA), attends directly to the global objective of maximizing the time spent collecting science data. However, one critical tradeoff that must be made in space-based observatory scheduling is between maximizing the time spent collecting science data and satisfying absolute temporal constraints associated with specific user requests.

A second sequencing strategy (to NNLA) of comparable computational complexity that directly attends to the objective of minimizing rejection of absolutely constrained goals is "most temporally constrained first" (MCF). Under this scheme, the sequence is built by repeatedly selecting and inserting the candidate goal that currently has the tightest execution bounds.

Both NNLA and MCF manage combinatorics by making specific problem decomposition assumptions and localizing search according to these decomposition perspectives. The difference between these two comes in that NNLA assumes an event based decomposition while MCF assumes that the problem is decomposable by degree of temporal contrainedness.

The research in this effort is for work done in the HST scheduling project, but like the SPIKE effort, could be applied to similar systems (i.e., large scale) in the private sector. More information on this research work can be found through The Robotics Institute at Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, Pennsylvania, 15213. The contacts are Nicola Muscettola and Stephen F. Smith.

ERE

The next project is the work performed at NASA Ames Research Center in Moffett Field, California. It is related to previous work mentioned in that it is an integrated system and involves the planning, scheduling, and control for automatic telescopes. The project is called the Entropy Reduction Engine (ERE) project and is focusing on the construction of integrated planning and

scheduling systems. This project serves as the focus for extending classical AI techniques involving automatic planners (i.e., systems that synthesize a plan to solve a problem) in a variety of ways. In the context of closed-loop plan execution, the ERE project is a focus for research as it relates to planning and scheduling. The eventual goal of the ERE project is a set of software tools for designing and deploying integrated planning and scheduling systems that are able to effectively control their environments. ERE has two important sub-goals: first the integration of planning and scheduling and second the study of plan execution as a problem of discrete event control.

The ERE project defines an integrated planning and scheduling system as a system that would able to consider alternative sets of actions, unlike the stand-alone scheduler, which is unable to deviate from its given action set. Moreover, the ERE project views plan execution as a problem in discrete event control; specifically, it formalizes a plan as a simple type of feedback controller, and this yields a new view on plan execution.

ERE itself is an architecture for producing systems that look ahead into the future, and by so doing, choose actions to perform. The essential idea is as follows: if a system has a limited amount of time to plan, and having planned, is allowed to plan no further, then it makes sense for the system to make the best use of the available time by incrementally improving its current plan until time runs out. The algorithm ERE employs is called traverse and robustify and follows this idea. ERE can provide a solid base for the development of integrated telescope planning, scheduling, and control systems that help to make this simplified management structure a reality.

ERE could be applied to private industry where a need for scheduling of automatic telescopes in an integrated planning and scheduling environment is required. Additional information on ERE can be obtained at NASA Ames Research Center, Mail Stop: 269-2, Moffett Field, California, 94035. The contact is Keith Swanson.

OMP

Next is the application and its underlying research which were performed by the Jet Propulsion Laboratory in Pasadena, California. It deals with a scheduling approach called Iterative Refinement and the application is called the Operations Mission Planner (OMP).

Iterative refinement is a heuristics-based approach to the scheduling of deep space missions which are modeled on the approach used by expert human schedulers in producing schedules for planetary encounters. Whenever the Voyager spacecraft encounters a planet, the science experiments to be conducted must be preplanned and ready to execute. This is a difficult scheduling problem due to the number and complexity of the experiments and the extremely limited resources of such a spacecraft. In general, not only are the schedules oversubscribed, they are also dynamic. As the scientists learn more about their objectives, the experiment requests themselves are updated.

Thus, the mission schedule is a dynamic entity. To solve the problem, OMP is centered on minimally disruptive replanning and the use of heuristics to limit the scheduler's search space.

In OMP, a resource tracks how a variable describing a state of the system changes through time and the steps which presently reserve this resource. There are five fundamental types of resources to OMP: capacity, consumable, renewable, continuous-state, and discrete-state. A brief description of each follows: a capacity resource is basically a pooled resource but can have non-integer value and may have a time varying initial capacity. A consumable resource is one for which there is a limited supply, and once it is used, it is no longer available (e.g., spacecraft fuel). A renewable resource is a generalization of a consumable resource, where the resource can be replenished (e.g., storage tape; it is used up during recording, and "replenished" during playback). A state resource represents a resource whose state (i.e., configuration, position, etc.) must be a certain value in order to support some activity. A continuous-state resource is a resource in which the state of the resource can best be described by a continuous variable (e.g., the direction that an antenna is pointing). A discrete-state resource however is represented by discrete values (e.g., on/off, low-gain/medium-gain/high-gain).

Associated with each of these types of resources is its definition of conflict. A conflict for a capacity resource occurs if the system reserves more than the limit of the pool at any moment in time. The resource is in conflict at the temporal interval for which a oversubscription occurs.

For additional definitions, OMP defines a step as a temporal interval which contains resource reservations. To OMP, an activity is a set of steps and a set of constraints that link the steps together. The temporal constraints are the "glue" that bind the steps into a logical unit. The user views an activity as the "primitive" action that must be scheduled to satisfy a user scheduling request.

OMP's iterative planning consists of a series of techniques. Each technique is responsible for a different aspect of the overall planning process. The first of these techniques roughs out the plan and identifies areas of high resource-conflict. The later techniques use the knowledge of the resource conflicts to refine the plan and solve many of the schedule problems. The final techniques try to solve the last of the conflicts and "optimize" the plan.

The OMP load phase is responsible for drafting an initial schedule. During the resource centered phase, OMP becomes resource oriented. The scheduler focuses upon a resource region which contains conflicts and uses quick and simple techniques to fix these regions before processing another resource. By focusing on just one resource region at a time the scheduler may fix one portion of the schedule but create additional conflicts in other regions. The scheduler discovers the bottlenecks by tracking these interactions between the separate regions. Once a bottleneck has been identified, it is classified and OMP attempts to resolve that bottleneck using techniques specialized for the type of bottleneck.

Once the conflict regions of the schedule have been resolved, OMP takes another look at the high priority activities which have been deleted from the schedule and tries to fit them in. At this point, OMP will perform its deepest search in an effort to schedule just one more activity. This phase is called the Optimization, although it doesn't produce a truly optimal schedule as would be defined in an operations research sense. Rather, it refers to fitting in additional activities after a conflict-free schedule has been produced. By specializing the planning techniques, each technique can be made more efficient.

The basic concept of self-reflective search in OMP is focusing the search by using knowledge gained from monitoring the search process. The OMP architecture, operating as outlined in the previous discussion, provides the mechanisms for supporting self-reflective search: the chronologies gather the raw information, the assessment heuristics analyze the information and feed the results to the control heuristics which focus the dispatch heuristics.

During the scheduling process, OMP keeps a chronology of the effort expended to resolve resource conflicts. During the resource centered phases, OMP focuses on a temporal interval within a given resource that is in conflict. Simple heuristics are used to reduce the level of conflict in the focus region. The chronologies keep track of the effect of these actions within the region and on other regions which are changed as a result of the scheduling actions. OMP first attempts to find a set of resource assignments which reduces the total amount of conflict in the entire schedule. If the system cannot lower the total conflict then it will increase the effort level for the focus region. This process will eventually cause OMP to cycle through the same regions.

The iterative planning approach to scheduling employed by OMP arose from attempts to heuristically control the search space of mission scheduling. The source of the heuristics were the human schedulers of Voyager, Viking, and SpaceLab who provided information on the stages of those scheduling processes.

Most of these heuristics in OMP assume that the scheduler knows which resources are the bottlenecks and which tasks are causing the most difficulty for the scheduler. The iterative planning approach assumes that the information gained by earlier techniques can be used by the later techniques to constrain the search space. Iterative planning also assumes that the schedule will not be changed dramatically by the later techniques.

OMP could be applied to the private sector in cases where an iterative approach utilizing heuristics for scheduling search is required to schedule in most any domain. The platform for OMP is a Mac II fx or Quadra. Additional information can be obtained by contacting Eric Biefeld at the Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, California, 91109-8099.

APS

Next is the Autonomous Power System (APS) project developed at the NASA Lewis Research Center. APS is designed to demonstrate the applications of integrated intelligent diagnosis, control and scheduling techniques to space power distribution systems. The APS project consists of three elements: the Autonomous Power Expert System (APEX) for Fault Diagnosis, Isolation, and Recovery (FDIR); the Autonomous Intelligent Power Scheduler (AIPS) to efficiently assign activities, start times and resources; and power hardware (Brassboard) to emulate a space-based power system. The AIPS portion of APS served as a learning tool and an initial scheduling testbed for the integration of FDIR and automated scheduling systems. Many lessons were learned from the AIPS scheduler and were integrated into a new scheduler called SCRAP (Scheduler for Continuous Resource Allocation and Planning). The APS project domain is the intelligent hardware and software of an electrical power system.

Similar to the scheduling of a telescope, the resources onboard a complex spacecraft will be vastly oversubscribed, having many times more resource requests than available resources. This makes it a paramount objective to efficiently utilize the available resources in order to complete as many activities as possible. The goal of the APS project is automated scheduling for space systems with proof-of-concept demonstrations on a power system testbed. In this scenario, the scheduler must not only know how to generate the schedule, but must also know how to implement the schedule, and how to recover from system or load induced deviations in the schedule.

The APS Brassboard is a power system testbed that contains a set of power supplies, switchgear, and loads that simulate a space-based power system. In order to more closely emulate this system, each load is given a set of attributes resembling those of the space-based system. Each activity (i.e., load) has a time varying profile of power demand, earliest start time and latest completion time constraints, a priority, and a temporal placement preference. In general, two modes of schedule generation are needed for any integrated scheduling system. The ability to generate an initial schedule and the ability to modify (i.e., reschedule) an already executing schedule in the case of an anomaly. The AIPS scheduler has two modes of schedule generation used for scheduling and rescheduling. The scheduling engine itself is an incremental scheduler that uses a set of activity selection and placement heuristics.

Finally, the SCRAP scheduling tool employs the theory of delineating two general categories in scheduling: predictive and reactive systems. Specifically, predictive scheduling allows the efficient allocation of available resources to activities by generating schedules based on predicted knowledge of the activity and resource states. On the contrary, reaction provides easier implementation in dynamic domains, but sacrifices resource usage efficiency caused by the lack of knowledge used to generate schedules. SCRAP also employs the theory that it may not be necessary to construct the initial schedule with a great level of detail. Therefore SCRAP schedules far term activities with less effort or detail than near term activities. It accomplishes this by using multiple levels of

abstraction when scheduling activities. Further into the future the schedule is constructed abstractly, while nearer to the execution time more precision is used.

The APS project is an ongoing effort to demonstrate the use of knowledge-based diagnosis and scheduling software in advanced space-based electrical power systems. The SCRAP paradigm will allow for more efficient use of the available resources in such a system.

The APS project has dual use potential in electrical power system domains that can be modeled and which would allow for less detail in long term scheduling and more detail in the short term. Additional information can be obtained through Mark J. Ringer, Sverdrup Technology Inc., NASA Lewis Research Center Group, Cleveland, Ohio, 44135.

MAESTRO

Next is a scheduling and resource management system named MAESTRO developed by the Martin Marietta Astronautics Group in Denver Colorado. As its problem domain, MAESTRO was interfaced with a Space Station Module Power Management and Distribution (SSMPMAD) breadboard at the Marshall Space Flight Center (MSFC). The resulting combined system serves to illustrate the integration of planning, scheduling, and control in a realistic, complex domain.

Briefly, the functionality of MAESTRO is as follows: 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 schedulers 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). The Fault Recovery And Management Expert System (FRAMES) performs fault isolation, diagnosis and recovery for the power scheduler during real-time contingencies.

During normal operations, 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 power system anomaly occurs, MAESTRO will get a set of information from FRAMES through the TM. MAESTRO follows a three-step process to handle these messages and revise the schedule. It a) 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, b) tries to find ways to create and schedule continuations for interrupted activities, and c) tries to schedule any activities that can take advantage of the resources released by the interruption of others.

Insofar as planning is concerned, activity continuation is the single automated planning function within MAESTRO. However, in terms of scheduling, MAESTRO can represent temporal constraints between activities, sometimes necessitating the consideration of more than one continuation model at once.

MAESTRO is a tool that could be applied to a need for a system executive that could monitor and control individual subsystems of a commercial rocket launch. Additional information on MAESTRO can be obtained from Daniel L. Britt and Amy L. Geoffroy at the Martin Marietta Astronautics Group, P. O. Box 179, Mail Stop: XL4370, Denver, Colorado, 80201.

GERRY

Next is a well-known scheduling tool called GERRY that was also developed at NASA Ames Research Center. GERRY is a general scheduling system applied to the Space Shuttle ground processing problem at Kennedy Space Center (KSC). It employs the approach called constraint-based iterative repair that provides the ability to satisfy domain constraints, address optimization concerns, minimize perturbation to the original schedule, and produce modified schedules quickly; all of which rescheduling systems should be capable of in general. GERRY utilizes a novel approach to rescheduling that addresses the concerns and gives the user the ability to individually modify each criteria's relative importance.

In terms of problems specific to GERRY's domain, there is an additional complicating factor introduced by the Space Shuttle problem and that is preemption. In preemptive scheduling, each task is associated with a calendar of legal work periods that determine when the task must be performed. Preemption effectively splits a task into a set of subtasks. It requires additional computational overhead since for each task the preemption times must be computed and appropriate constraint manipulation for each time assignment must be performed.

In the Space Shuttle processing domain, rescheduling is necessitated by changes that occur in the environment. In response, GERRY adopts the approach of repairing the constraints that are violated in the schedule. Constraint-based iterative repair itself begins with a complete schedule of unacceptable quality and iteratively modifies it until its quality is found satisfactory. In GERRY repairs are associated with constraints. Repairing any violation typically involves moving a set of tasks to different times: at least one task participating in the constraint violation is moved, along with any other tasks whose temporal constraints would be violated by the move. Simply put, all temporal constraints are preserved after the repair. Similar to many scheduling evaluation tools, at the end of each iteration, the system re-evaluates a cost

function to determine whether the new schedule resulting from the repairs is better than the current solution. The system sometimes accepts a new solution that is worse than the current solution in order to escape local minima and cycles. In summary, the algorithm is interruptable, restartable, and outputs a solution when terminated.

To examine the effects of this scheduling strategy, the STS-43 Space Shuttle processing flow was used as an experiment. The results were very positive and the experiments suggest that the constraint framework and the knowledge encoded in this framework is an effective search tool that allows one to adjust the importance of schedule perturbation and other objective criteria.

GERRY could be adapted to an environment in the private sector which employed assembly line processing for construction of a product. For more information concerning GERRY, contact Eugene Davis, Brian Daun, or Michael Deale at the NASA Ames Research Center, Mail Stop: 269-2, Moffett Field, California, 94035.

TMSA

Next is a concept prototype known as Time Management Situation Assessment (TMSA) developed by the Advanced Computing Technologies Group at Boeing at the Kennedy Space Center in Florida. TMSA was designed to support NASA Test Directors (NTDs) in schedule execution monitoring during the later stages of a Shuttle countdown. The system detects qualitative and quantitative constraint violations in near real-time.

The problem domain involves the NTDs primary concern with the orderly and timely execution of the countdown process. The cognitive model they reason with is relatively high-level and includes a nominal (i.e., planned) model of the countdown and a set of qualitative and quantitative constraints that define such a countdown by specifying temporal duration and ordinal relationships between countdown events. The NTDs monitor the current countdown and assess its compliance with their nominal countdown model. The countdown schedule may be revised by reordering events and/or adjusting the durations of intervals between events. The characteristics of the domain are as follows:

A) The situation is highly structured with a well formulated, proven set of constraints on the schedule, and

B) Although this is an advisory system used by experts, the criticality of the situation places a premium on timeliness and correctness beyond that of many applications.

The verification and validation issues in TMSA's software environment in conjunction with the above mentioned characteristics led to an approach of the problem algorithmically and avoided using heuristics.

Furthermore, while the countdown is formulated in terms of both events and intervals, the constraints between intervals are such that intervals were represented as start and end pairs of events. From the NTD's perspective

countdown time is discrete. Pseudo events are used and are defined as events that are not members of the universe of countdown events employed by the NTDs. Uncertainty arises in the countdown schedule situation in that many of the qualitative constraints between countdown events are ambiguous. In addition, ambiguity also occurs in some quantitative duration constraints on the length of intervals.

To solve this problem, two algorithms were developed for TMSA. These form the reasoning kernel of the system and are designed to monitor and interpret the legality of the temporal duration and sequential unfolding of a countdown. The first algorithm, called the ConstraintChecker, is used to maintain a qualitative representation of the current status of a countdown and to check the consistency of that status with the qualitative constraints that define the legality of a countdown. The second algorithm, known as the ScheduleMaintainer, is used to maintain both a qualitative and quantitative representation of a countdown. This representation includes both the current status of the countdown and the quantitative constraints that define the legality of a countdown. The representation is also used to generate relational assertion vectors as input to the consistency checking algorithm. Together, the ConstraintChecker and the ScheduleMaintainer form the core of TMSA and exchange relations between the ordering of events.

The TMSA prototype is implemented in Smalltalk and runs on a 25 mhz 486, under MS DOS. TMSA could be applied to a private industry domain with similar constraints and events found in that of a Space Shuttle countdown, like a launch of a commercial rocket perhaps. It could also be utilized in an environment where testing of a device of some sort underwent a well-defined procedure that employed a set of rules for its own execution. An example of this might be the process testing of an airplane in a wind tunnel. For further information concerning TMSA, contact Michael B. Richardson of the Advanced Computing Technologies Group, Boeing Aerospace Operations, FA-71, Kennedy Space Center, Florida, 32899.

COMPASS / ISAID

Finally, there is an interactive and highly flexible scheduling system called COMPASS developed by the McDonnell Douglas Space Systems Corporation in Houston, Texas. COMPASS is a constraint-based scheduler and with its interactive trait, it provides an environment where a mixed initiative is possible; that is, it lets the computer do what it does best (i.e., check constraints and calculate feasible intervals) and lets the human do what he/she does best (i.e., provide heuristic and subjective inputs into the schedule). This results in a cooperative production of a schedule which reflects both the hard constraints and subjective preferences. Written in ADA with the user interface in C and X-Windows, COMPASS is a well-designed, highly flexible scheduler that could be modified to fit the needs of a wide variety of scheduling problems. It operates on a number of platforms, including Sun3/4, Sun Sparc, Rational, RS6000, and VAX/VMS. COMPASS provides both forward and backward scheduling modes and displays a schedule in the Gantt chart format. In addition, a histogram of

resource usage can be displayed. An example of COMPASS's modification ease follows.

COMPASS was used as the scheduling engine in a highly successfully project known as the Interactive Scheduling AID (ISAID). The target environment for this tool was the Systems Engineering Simulator (SES) located at NASA Johnson Space Center (JSC). The SES is a facility which houses the software and hardware for a variety of simulation systems such as the Manned Maneuvering Unit and the Remote Manipulator System. Due to the highly flexible and well thought out design of COMPASS, it was modified to meet the requirements of the SES. This entailed work in three discrete areas:

A) A new user interface. The SES scheduling experts had grown accustomed to performing scheduling duties with paper "worksheets" which represented a schedule in development in a calendar-type format. The default interface for COMPASS represented activities in a Gantt chart format (H bars) which was unacceptable to the SES users. Therefore, a completely new interface which appears like a calendar (very much like the previous method) was developed.

B) Implementation of domain-specific SES constraints. These constraints (or "rules") were incorporated into the scheduling engine of COMPASS to emulate the procedure the SES scheduling experts used when developing a schedule.

C) Optimization. This feature had not been implemented into COMPASS prior to the ISAID project. The addition of this feature gave COMPASS the ability to allow the user to interleave schedule creation, revision, and optimization. The optimization procedure itself was based on the simulated annealing of metal. It begins with high temperatures and allows many changes to the schedule to occur, evaluates the result according to an objective function, compares that to the best schedule generated so far, and decrements the temperature as time passes. At low temperatures, very little change is allowed to occur to the schedule.

With these three enhancements to COMPASS, the ISAID tool found much success and is still used on a weekly basis to generate SES schedules.

COMPASS could be applied to many scheduling domains in the private sector due to its highly flexible design and ease of modification. For more information about COMPASS, contact Barry Fox of the Planning and Scheduling Technology Group at the McDonnell Douglas Space Systems Company, 16055 Space Center Boulevard, Houston, Texas, 77062.

Conclusion

This paper has presented a number of scheduling applications and research areas, all of which are capable of being applied to the private sector. They were designed with the capability of being modified to meet the requirements of an application domain other than the one for which they were developed.

Furthermore, the projects described in this paper use various planning and scheduling strategies to meet particular goals and therefore one of them will fulfill a number of scheduling needs in the private community. A contact person or persons was (were) listed at the end of each project description for your convenience.

References

All technical information contained in this paper was referenced through:

NASA Ames Research Center, Artificial Intelligence Research Branch. *Technical Report FIA-92-17: Working Notes form the 1992 AAAI Spring Symposium on Practical Approaches to Scheduling and Planning.* NASA Ames Research Center, 1992.

CONFIG - INTEGRATED ENGINEERING OF SYSTEMS AND OPERATIONS

ABSTRACT. CONFIG 3 is a prototype software tool that supports integrated conceptual design evaluation from early in the product life cycle, by supporting isolated or integrated modeling, simulation, and analysis of the function, structure, behavior, failures and operation of system designs. Integration and reuse of models is supported in an object-oriented environment providing capabilities for graph analysis and discrete event simulation. Integration is supported among diverse modeling approaches (component view, configuration or flow path view, and procedure view) and diverse simulation and analysis approaches. Support is provided for integrated engineering in diverse design domains, including mechanical and electro-mechanical systems, distributed computer systems, and chemical processing systems.

INTRODUCTION. The core of engineering design and evaluation focuses on analysis of physical design. Thus, today's computer-aided engineering software packages often do not provide enough support for conceptual design early in the life cycle or for engineering for operation, fault management, or supportability (reliability and maintainability). Benefits of engineering for operations and supportability include more robust systems that meet customer needs better and that are easier to operate, maintain and repair. Benefits of concurrent engineering include reduced costs and shortened time for system development. Integrated modeling and analysis of system function, structure, behavior, failures and operation is needed, early in the life cycle.

Conventional system modeling approaches are not well suited for evaluating conceptual designs early in the system life cycle. These modeling approaches require more knowledge of geometric or performance parameters than is usually available early in design. More abstracted models can support early conceptual design definition and evaluation, and also remain useful for some later analyses. Component-connection models provide one such useful abstraction, and discrete events are another. Discrete event simulation technology combines both abstractions, for evaluation of conceptual designs of equipment configurations in operations research (3). CONFIG uses these abstractions, with some enhancements, to define and evaluate conceptual designs for several types of systems.

The initial CONFIG project goal was to support simulation studies for design of automated diagnostic software for new life-support systems (9). The problem was to design an "expert system" on-line troubleshooter before there was an expert. The design engineer could use a model of the system to support what-if analyses of failure propagation, interaction, observability and testability. This activity is similar to Failure Modes and Effects Analysis (5), but uses comparative simulations of failure effects to develop diagnostic software. Conventional simulation software was not up to this challenge, but discrete event simulation software has been. CONFIG supports the use of qualitative models for applying discrete event simulation to continuous systems.

A major design goal for CONFIG is to support conceptual design for operations and safety engineering. Major tasks in conceptual design are design definition, evaluation (by simulation and analysis) and documentation. Operations engineering focuses on the design of systems and procedures for operating, controlling and managing the system in normal or faulty conditions. Safety engineering focuses on prevention of

hazardous effects and conditions in the physical system or its operation. In these types of engineering, complex interactions and interfaces among system components and operations must be a focus.

Another design goal of CONFIG is to bridge the gaps between physical design engineering and other types of engineering. Component-connection representations are well suited for modeling and defining physical system designs (as structures of interacting components) and operations designs (as structures of interacting actions), as well as interactions between system components and operational actions. Discrete event models have been used for this type of modeling for queueing and scheduling problems, but can be extended to support conceptual modeling in operations and safety engineering. This type of modeling is also compatible with systems engineering function diagrams (1).

CONFIG 3. The project approach has been to incrementally integrate advanced modeling and analysis technology with more conventional technology. The prototype integrates qualitative modeling, discrete event simulation and directed graph analysis technologies for use in analyzing normal and faulty behaviors of dynamic systems and their operations. The prototype has been designed for modularity, portability and extensibility. A generic directed graph element design has been used to standardize model element designs and to promote extensibility. This directed graph framework supports integration of levels of modeling abstraction and integration of alternative types of model elements.

Enhanced Discrete Event Simulation Capabilities. In traditional discrete event modeling and simulation, state changes in a system's entities, "events", occur discretely rather than continuously, at nonuniform intervals of time. Throughout simulation, new events are added to an event list that contains records of events and the times they are scheduled to occur. Simulation processing jumps from one event to the next, rather than occurring at a regular time interval. Computation that results in creation of new events is localized in components, which are connected in a network. Any simulation run produces a particular history of the states of the system. Statistical simulation experiments, using random variables in repeated simulation runs, are used to compare design alternatives.

To enhance this discrete event simulation approach to accommodate continuous systems, a number of new concepts and methods were developed. These include component models with operating modes, types of links connecting components ("relations" and "variable clusters"), state transition structures ("processes"), methods for representing qualitative and quantitative functions ("process language"), and a new simulation control approach.

Digraph Analysis Capabilities. The CONFIG Digraph Analyzer (DGA) makes graph analysis techniques available for evaluating conceptual designs of systems and their operations. The DGA is based on reachability search, and is implemented generically for application to the many types of graph data structures in CONFIG. DGA can support analyses of completeness, consistency and modularity. Analysis of failure sources and impacts can be done by tracing the paths from a given failure.

System Modeling. Devices are the basic components of a CONFIG system model, which are connected together in topological model structures with Relations. Device behavior is defined in operating and failure Modes, which contain mode dependent and mode transition Processes. Modes are connected together in a mode transition digraph which delineates the transition dependencies among the individual modes.

Device Processes define change events in device variables, and are conditionally invoked and executed with appropriate delays during a simulation. Processes define time-related behavioral effects of changes to device input variables, both direction of change and the new discrete value that will be reached, possibly after a delay. Faults and failures can be modeled in two distinctly different ways. Failure modes can be used to model device faults. Mode-transition processes can be used to model latent device failures that cause unintended mode changes. Relations connect devices via their variables, so that state changes can propagate along these relations during simulations. Related variables are organized into variable clusters, to separate types of relations by domain (e.g., electrical vs. fluid connections). Relations can also connect Devices with device-controlling Activities in operations models.

Flow Path Modeling. Flow is a property of many systems, whether the substance flowing is a liquid or information. There are two difficulties in modeling flows with local device processes. First, flow is a global property of the topology of the modeled system and the substances flowing within it. Second, while dynamic changes in system structure and flow can occur during operations, process descriptions involving flow must often rely on assumptions of static system topology. These factors would limit the reusability of device descriptions to a limited set of system structures.

A flow-path management module (FPMM) has been implemented to address these problems. The FPMM is separate from the module implementing local device behavior, but the two modules are interfaced via flow-related state variables in the devices. When FPMM is notified during simulation of a local change in device state, it recomputes the global effects on flows produced by the local state change. The FPMM then updates the state of flow in all affected devices. This design permits the user to write reusable local device process descriptions that do not depend on any assumptions concerning the system topology.

FPMM uses a simplified representation of the system, as a collection of aggregate objects, or "circuits." Further abstraction is achieved by identifying serial and parallel clusters in the circuit (6). In many cases, configuration determination alone is sufficient to verify flow/effort path designs, to establish flow paths for a continuous simulation, for reconfiguration planning, and for troubleshooting analysis (see Ref. [2] on cluster-based design of procedures for diagnosis, test, repair and work-around in a system).

Operations modeling. Activities are the basic components of a CONFIG operations model, and are connected together in action structures with Relations. They represent procedures or protocols that interact with the system, to control and use it to achieve goals or functions. Each activity model can include specifications for what it is intended to achieve or maintain. Activity behavior is controlled in a sequence of phases, ending in an evaluation of results. Activity behavior is defined by processes that model direct effects of actions, or that control device operation and mode

transitions to achieve activity goals. Relations define sequencing and control between activities and connect Devices with device-controlling Activities.

Operations models are designed to support operation analysis with procedure models. These models are designed to support analysis of plans and procedures for nominal or off-nominal operation. The procedure modeling elements are designed for reuse by intelligent replanning software, and for compatibility with functional modeling in systems engineering.

Model Development & Integration Capabilities and Approach. CONFIG provides intelligent automation to support nonprogrammer and nonspecialist use and understanding. CONFIG embeds object-oriented model libraries in an easy-to-use toolkit with interactive graphics and automatic programming.

CONFIG provides extensive support for three separable yet tightly integrated phases of user operation during a modeling session: Library Design, Model Building, and Simulation and Analysis. This includes a graphical user interface for automated support of modeling during each of the phases including the development of object-oriented library element classes or templates, the construction of models from these library items, model inspection and verification, and running simulations and analyses.

The integration between the phases enables an incremental approach to the modeling process. Lessons learned from analyses can be repeatedly and rapidly incorporated by the user into an initially simple model. Support for these phases as separate user activities fosters the achievement of concurrent engineering goals. Different users can define library elements, build models, and analyze models at different times depending on area of expertise and availability of resources. Support for the model building phases spans all types of modeling that can be performed in CONFIG including component structure, behavior and flow, and activity goals and structure.

Hosting. CONFIG is implemented in software that is portable to most Unix work stations. The Common LISP Object System (CLOS) is a highly standardized language, with compilers for most of the commonly available work stations. The user interface was implemented using the Common LISP Interface Manager (CLIM), another standardized tool built on CLOS.

CONCLUSIONS. CONFIG is designed to model many types of systems in which discrete and continuous processes occur. The CONFIG 2 prototype was used to model and analyze: 1) a simple two-phase thermal control system based on a Space Station prototype thermal bus, 2) a reconfigurable computer network with alternate communications protocols, and 3) Space Shuttle Remote Manipulator System latching and deployment subsystems (7). The core ideas of CONFIG have been patented (8). CONFIG 3 has added capabilities for graph analysis and for modeling operations and procedures.

The CONFIG prototype demonstrates advanced integrated modeling, simulation and analysis to support integrated and coordinated engineering. CONFIG supports qualitative and symbolic modeling, for early conceptual design. System models are component structure models with operating modes, with embedded time-related

behavior models. CONFIG supports failure modeling and modeling of state or configuration changes that result in dynamic changes in dependencies among components. Operations and procedure models are activity structure models that interact with system models. The models support simulation and analysis both of monitoring and diagnosis systems and of operation itself. CONFIG is designed to support evaluation of system operability, diagnosability and fault tolerance, and analysis of the development of system effects of problems over time, including faults, failures, and procedural or environmental difficulties.

REFERENCES

1. Alford, M. Strengthening the System Engineering Process, Engineering Management Journal, Vol. 4, No. 1, March, 1992 , pp 7-14.

2. Farley, A. M. Cluster-based Representation of Hydraulic Systems, Proc. 4th Conference on AI Applications, March, 1988, pp. 358-364.

3. Fishman, G. S. Principles of Discrete Event Simulation. New York, NY: Wiley, 1978.

4. Forbus, K. Qualitative Physics: Past, Present, and Future. In Exploring Artificial Intelligence (H. Shrobe and AAAI, eds.). San Mateo, CA: Morgan Kaufmann, 1988.

5. Fullwood, R. R. and Hall, R. E. Probabilistic Risk Assessment in the Nuclear Power Industry: Fundamentals and Applications. Pergamon Press, 1988.

6. Liu, Z. and Farley, A. M. "Structural Aggregation in Common-Sense Reasoning". Proc. 9th National Conference on Artificial Intelligence (AAAI-91), 1991, pp. 868-873.

7. Malin, J. T., B. D. Basham and R. A. Harris, "Use of Qualitative Models in Discrete Event Simulation for Analysis of Malfunctions in Continuous Processing Systems." Artificial Intelligence in Process Engineering (M. Mavrovouniotis, ed.), Academic Press, pp. 37-79, 1990.

8. Malin et al., U. S. Patent 4,965,743, "Discrete Event Simulation Tool for Analysis of Qualitative Models of Continuous Processing Systems" October, 1990.

9. Malin, J. T., and Lance, N. "Processes in construction of failure management expert systems from device design information". IEEE Trans. on Systems, Man, and Cybernetics, 1987, SMC-17, 956-967.

10. Malin, J. T. and Leifker, D. B. "Functional Modeling with Goal-Oriented Activities for Analysis of Effects of Failures on Functions and Operations". Informatics and Telematics, 1991, 8(4), pp 353-364.

ACKNOWLEDGEMENTS. The authors wish to thank Bryan Basham, Leslie Ambrose, Ralph Krog, Debra Schreckenghost, Brian Cox, Daniel Leifker and Sherry Land for their contributions to CONFIG design and implementation.