Maximally Expressive Modeling

John Jaap, Elizabeth Davis, Lea Richardson

Flight Projects Directorate
Marshall Space Flight Center
National Aeronautic and Space Administration
John.Jaap@nasa.gov

Abstract. Planning and scheduling systems organize "tasks" into a timeline or schedule. Tasks are logically grouped into containers called models. Models are a collection of related tasks, along with their dependencies and requirements, that when met will produce the desired result. One challenging domain for a planning and scheduling system is the operation of on-board experiments for the International Space Station. Another domain is planning and scheduling the tasks to be done by humans on the surface of the Moon, on Mars, or in a trans-Mars spacecraft. In these domains, the equipment used will be among the most complex hardware ever developed; the information sought will be at the cutting edge of scientific endeavor; and the procedures will be intricate and exacting. Scheduling will be made more difficult by a scarcity of resources. The models to be fed into the scheduler must describe both the complexity of the tasks and procedures (to ensure a valid schedule) and the flexibilities of the procedures and the equipment (to effectively utilize available resources). Ideally, remote platforms should be autonomous — the crew should schedule their own tasks. The great distance from earth, the long-duration of the mission, the need to cut cost, and the desire for autonomy call for an operations concept that is based on an automatic scheduler and a modeling schema that captures all the requirements.

1 Introduction

Marshall Space Flight Center has a long history of planning and scheduling manned space missions, including Skylab, Spacelab, Shuttle, and the International Space Station. The Ground Systems Department is building on its experience in developing scheduling engines to develop a new scheduling system that is highlighted by a "maximally expressive" modeling schema.

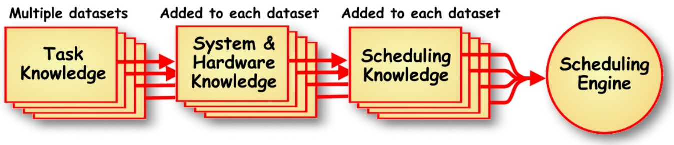


Figure 1 — State-of-the-Art Paradigm

The current state-of-the-art in modeling methodologies and scheduling engines results in a linear paradigm with knowledge contributed by task experts (scientists, etc.), vehicle experts, and scheduling engine experts. This paradigm, depicted in Figure 1, requires significant effort and flow-time. The task experts often struggle to enter their requirements using a language that is limited – frequently resorting to notes to fully describe their requirements. The vehicle and hardware experts then convert and augment this knowledge to prepare the models for scheduling. The scheduling team then feeds the models to the scheduling engine. Since the models are incomplete, they often have to "steer" the scheduler to produce an acceptable schedule.

This paradigm has begotten the "scheduling cadre," a group of people that digests all the requirements, builds the best models allowed by the current schema, makes notes containing the remainder of the requirements, and then generates the timeline using a combination of an automatic scheduler and a schedule editor (a mixed-initiative approach to scheduling).

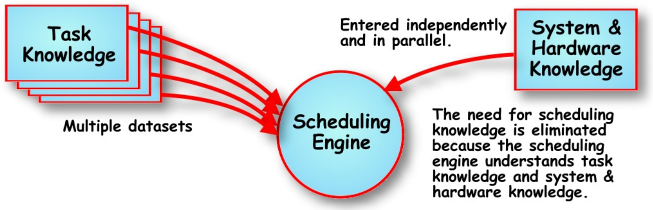

Task Knowledge
Multiple datasets
Entered independently and in parallel.
System & Hardware Knowledge
Scheduling Engine
The need for scheduling knowledge is eliminated because the scheduling engine understands task knowledge and system & hardware knowledge.

Figure 2 — Future Paradigm

The modeling schema presented in this paper (and the corresponding scheduling engine — see Appendix A) allows a streamlined paradigm as depicted in Figure 2. The vehicle experts would enter the system and hardware constraints independently of the task knowledge. The task experts would enter the tasks' requirements. The modeling schema would allow them to specify all of the task requirements without resorting to notes – these models would be ready for the scheduling engine. Having models that express all the constraints allows the scheduling engine to operate automatically without human intervention.

The modeling schema described in this paper can significantly reduce the manpower and flow-time required to produce a schedule in a complex scheduling domain.

2 The Schema

Inadequate modeling is the downfall of automatic scheduling. If all the requirements are not included in the model, then the scheduler has little chance of producing a satisfactory schedule. But modeling even the simplest of tasks cannot be automated because no sensor can be attached to a piece of equipment that can discern how to use that piece of equipment, and no camera can quantify how to operate a piece of equipment. Modeling (expressing the requirements in the schema) is a human enterprise – both an art and a science. The modeling schema must be "maximally expressive" so that *all* the requirements can be captured without resorting to notes or awkward representations, and the schema must be friendly enough to allow the user to enter the requirements without excessive labor or training. The schema which meets the "maximally expressive" criteria is a synergy of technological advances and domain-specific innovations. Some of the key features are given below:

- *Decomposition of the operations into salient components* — Operations are decomposed into activities that define resource requirements and sequences that define relationships between activities.[1] Sequences can also contain other sequences, repeated activities and sequences, and optional activities and sequences.
- *Graphical interfaces* — Simple graphical interfaces such as outlines and networks are used to build and depict the models. Modeling itself is done using techniques such as drag-and-drop.
- *Intuitive and rich expression of the relationships between components* — The schema employs common-sense representations of temporal relationships using everyday concepts like sequential, during, and overlap. Innovative enhancements to represent the continuance of resource usage between tasks, the interruption of tasks, minimal percent coverage, and temporal relationships to outside tasks have been added to the modeling schema.
- *Public services* — The schema also introduces the concept of public services – models that are scheduled at the request of another model.
- *Modeling equipment modes* — Implicit resource requirements are defined by equipment mode models, thereby more closely representing the real world.
- *Flexibility and Nuances* — Variable timing, alternate resources and sequences, optional items, and other nuances are modeled.

The modeling schema is an evolutionary improvement of the modeling schema currently used by the Marshall Space Flight Center for International Space Station payloads [1].

2.1 Decomposition into Salient Components

This schema models scheduling requirements by defining "activities" and "sequences of activities." Activities generally equate to the simplest or lowest-level tasks. A sequence of activities is usually required to represent scheduling entities. Consider the following example: to do the laundry a housewife must wash, dry, and put away the clothes. Doing the activities out of order or standalone does not accomplish the objective.

Activities define the resource requirements (with alternatives) and other quantitative constraints of the tasks to be performed. Activity requirements may be grouped into "all-of" groups and "one-of" groups. Groups may be hierarchical. For example, the housewife can use the oven or the stovetop to cook a roast; however, the duration would be different, and a different pan would be used. Activity requirements include the specification of the minimum, maximum, and preferred duration of the activity. Resource modeling is described in the paragraph titled "Resource and Condition Modeling." at subsection 2.8.

Sequences define the temporal relationships between activities. In our laundry example, we discussed three activities that would be done one after the other; i.e., in a sequence. Sequences may also define relationships with other sequences, as well as with events. For example, laundry is done after taking the children to school, and dinner is served between the evening news and primetime TV. Temporal relationships include during, sequential, separated, overlap, standby, fragmentable, percent coverage, and (for repeated items) cyclic. Resource lock-in and one-to-one relationships are also included. Other temporal constraints are modeled for the International Space Station to match the scheduling engine in use; these relationships are outside the scope of this paper.

A sequence may have alternate ways of accomplishing its objective; these alternatives are called scenarios. Each scenario of a sequence is independent of the others and may have all the features of a sequence that does not have multiple scenarios.

[1] The "Activity" and "Sequence" nomenclature chosen for this paper matches that used by the International Space Station.

Sequences may have temporal constraints that apply to the sequence as a whole. These are described in the paragraph titled “Other Sequence Constraints” at sub-section 2.7.

2.2 Graphical Interfaces

The schema is implemented using graphical interfaces to interact with the user – both for presenting and editing the data. An outline interface is used for activities and a network interface is used for sequences.

Hierarchies of groups of requirements best describe the constraints of most non-trivial activities. The outline interface is well-suited to modeling hierarchies of groups because it can be manipulated by a drag-and-drop interface and nested to any depth without ambiguity. Figure 8 (in Section 3) illustrates the use of the outline hierarchy. Constraints are selected from a list and added to the hierarchy canvas. A dialog box is used to specify the mode or values.

Sequences use a “network” interface to define the relationships between tasks (activities and embedded sequences). The user selects from a list of tasks, placing the item onto a canvas. The user then positions the items as needed, creates relationships by connecting them together, and differentiates between relationships to pre-existing tasks and to-be-scheduled tasks. A dialog box is used to specify the type of the relationship and its details. Table 1 shows the parameters for each relationship and Figure 3 shows the features of an item that can be set by the user. Figure 4 shows the visual cues attached to a sequence embedded in the displayed sequence. The example in Figure 5 (section 3) has several embedded sequences.

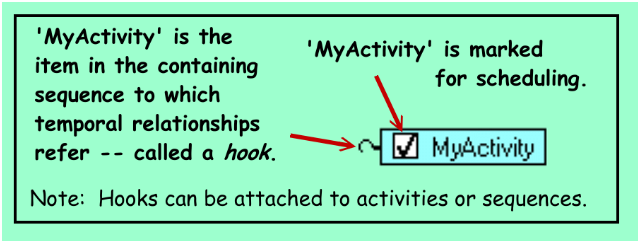

'MyActivity' is the item in the containing sequence to which temporal relationships refer -- called a *hook*.
'MyActivity' is marked for scheduling.
MyActivity
Note: Hooks can be attached to activities or sequences.

Figure 3 — Features of an Item in a Sequence

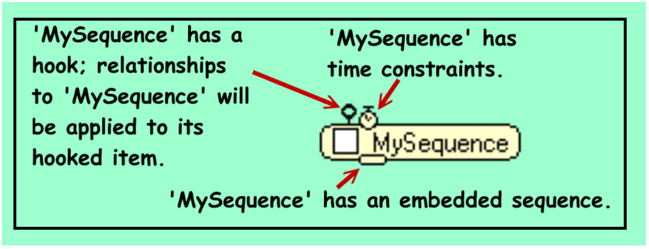

'MySequence' has a hook; relationships to 'MySequence' will be applied to its hooked item.
'MySequence' has time constraints.
MySequence
'MySequence' has an embedded sequence.

Figure 4 — Indicators on an Embedded Sequence

2.3 Intuitive and Rich Expression of the Relationships

The sequence model may include one or more of the relationships listed below. As stated earlier, sequences may contain activities, other sequences, public services, and external events (such as launch and docking).

Sequential — Items follow each other. Minimum and maximum separations may be specified.

Separated — Items may not overlap, but the order of execution is not defined. Minimum and maximum separations may be specified.

Avoid — An item to be scheduled may not overlap any instance of the avoided item. This constraint applies to items not scheduled by this sequence and is also enforced on not-yet-scheduled instances of the avoided item. Minimum before and after separations may be specified.

During — Items occur simultaneously; when items have different durations, one contains the other. Which item is during the other may be specified. Minimum and maximum separations of both the start and end times may be specified.

Overlap — Items overlap; which item starts first may be defined. Minimum and maximum durations of the overlap may be specified.

Percent Coverage — One item must be scheduled during another item so that it covers a certain percentage of the duration of the other item. For example, about 60% of the time, a parent needs to provide assistance to a certain young child playing on the computer. This time may be broken into reasonably short segments. The minimum coverage, the maximum number of segments, the minimum duration of a segment, and the maximum separation between segments may be specified.

Fragmentable — When an activity may be broken into parts, an activity or sequence is scheduled to book the resources that are used during the interruption. For example, when a stamp collection is being organized, it could be laid out on the kitchen table. Sorting the collection could be broken into multiple short sessions, but between the sessions the table is in use and cannot be used for anything else. The maximum number of fragments, the minimum duration of a fragment, and the maximum duration of an interruption may be specified.

Standby — During a delay between sequential or separated items, a standby item is scheduled to book (consume) the resources that are used during the delay. For example, if there is delay between washing the clothes and drying the clothes, an item would be scheduled to show that the washer is in use. If drying follows immediately after washing, then the standby item is not scheduled.

Repeated — An item in a sequence may be repeated; minimum and maximum repetition counts may be specified. When the minimum repetition count is 0, the item is considered optional. The temporal relationship of the repetitions can be sequential, overlapped, or cyclic (see below).

Cyclic — A frequency can be specified for repeated items. The frequency may be specified in hours, days, or weeks. For the daily and weekly options, the time of day (with variation) may be specified. For the weekly option, days of the week may be specified

Lock-In — If two activities in a sequence contain identical "one-of" selection groups, then the same constraints may be chosen when scheduling the sequence. For example, assume there is a choice of which car to drive to the grocery and which car to drive from the grocery. If lock-in is requested when scheduling the grocery shopping sequence, the same car must be chosen for both the trip to and the trip from the market.

One-to-One — When an item is to be done multiple times, and each repetition of this item is related to a pre-existing item in the timeline, but only one instance of the

Table 1 — Temporal Relationships of the Schema

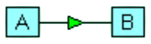
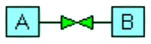
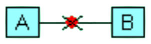
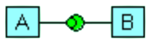
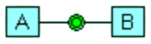
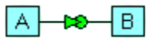
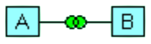
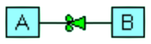
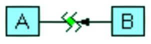
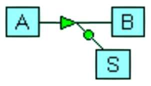
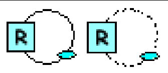
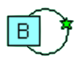
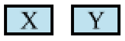
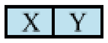
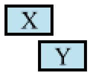
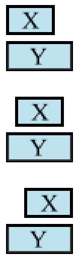
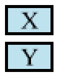
Relationship	Graphics	Order			Parameters
		A,B	B,A	either	
Sequential		✓	✓		Minimum & maximum separation
Separated				✓	Minimum & maximum separation
Avoid		Implied			Buffer before and after
During		✓	✓		Minimum & maximum start time separation Minimum & maximum end time separation
				✓	
Overlap		✓	✓		Minimum & maximum overlap
				✓	
Percent Coverage		✓	✓		Maximum number of occurrences Minimum duration of an occurrence Maximum separation between occurrences Minimum percent coverage
Fragmentable		✓	✓		Maximum number of fragments Minimum duration of a fragment Maximum duration of an interruption
Standby		Shown with a sequential relationship.			None
Repeated		When the minimum number of repetitions is 0, then the item is optional and the loop is dashed. All repeated relations include a temporal relationship such as separated, overlap, or cyclic.			Minimum & maximum number of repetitions
Cyclic					Hourly: Hours between repetitions Time variance
					Daily: Days between repetitions Desired center time with variance
					Weekly: Weeks between repetitions Day(s) of the week Desired center time with variance

Table 2 — Schema Relationships Compared to Classical Relationships

Interval	Endpoint Relations	Pictorial	Schema Equivalent
X < Y	$X_E < Y_S$		sequential: $wn \neq 0$
X meets Y	$X_E = Y_S$		sequential or overlap: $wn = wx = 0$
X overlaps Y	$X_S < Y_S$ and $X_E > Y_S$ and $X_E < Y_E$		overlap: $wn \neq 0$ and $w < min(X_D, Y_D)$
X during Y	$X_S > Y_S$ & $X_E \leq Y_E$ or $X_S \geq Y_S$ & $X_E < Y_E$		during: $wn_S + wn_E > 0$
X = Y	$X_S = Y_S$ and $X_E = Y_E$		during: $wx_S = wx_E = 0$
S and E indicate the start and end. wn and wx indicate the minimum and maximum separations (waits). D indicates the duration.			

item is to be scheduled for each instance of the pre-existing item, then a one-to-one relationship is required. For example, many pictures of the crew having breakfast are to be taken, but only one picture is to be taken per meal.

Relationships to Internal Items of Sequences — A sequence can specify that temporal relationships to itself are to be applied to an item within itself. For example, a sequence named Video might include the following three activities: video-setup, video-record, and video-downlink; the video-record activity could be specified as the internal item to which the temporal relationships are applied. When a task in another sequence specifies a during relationship with the Video sequence, only the video-record activity would be required to run during the that task; video-setup and video-downlink would occur before and after the task to be taped. The item in the sequence to which the relationships are to be applied is called the "hooked" item and on the drawing canvas is marked with a hook; see Figures 3 and 7.

The parameters that may be specified for each temporal relationship are shown in Table 1.

The temporal relationships defined by the schema are common-sense relationships, not the classical (and sometimes esoteric) temporal relationships. Sequential, separated, during, and overlap can be mapped directly to the classical relationships as shown in Table 2 and to Allen's thirteen relationships[2] as shown in Table 3. The schema introduces percent coverage, fragmentable, and standby relationships that are not in the classical set of temporal relationships. The schema also includes a cyclic relationship that is not in the classical set but can be found in virtually every calendar program.

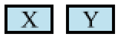
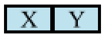
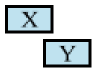
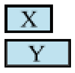
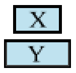
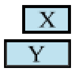
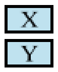
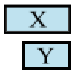
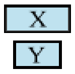
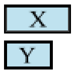
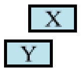
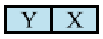
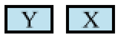
2.4 Public Services

A public service is a task (usually a sequence) that can be scheduled in conjunction with a user's sequence. When a user includes a public service in a sequence, the process of scheduling the sequence will also cause the public service to be scheduled. Note that the details of the public service (such as tasks of the public service, resource usage, conditions required, etc.) are not visible to the requesting sequence, but will be booked when scheduling the request. Public services are usually modeled in advance. For example, a housewife might ask her husband to bring home a loaf of bread for dinner. She does not need to define where to get the bread or how to get there. She only needs to request the bread.

2.5 Modeling Equipment Modes

Most tasks are accomplished using equipment of some sort. Most equipment have various operating modes; e.g., a microwave oven has modes such as defrost, reheat, and cook. The microwave oven's power requirement for each mode is predefined. On a space platform, the characteristics of each piece of equipment are well known to those building and integrating the equipment into the platform systems. The equipment and their modes may be modeled independently of the tasks that will use the equipment. Equipment mode models may use multiple resources and may use other equipment in a specified mode. Equipment mode models can describe a hierarchy of constraints and use an outline interface like that used by activity models. Equipment models allow low-level resources, such as power, to be hidden from the task modeler. The task modeler can only request these low-level resources by selecting equipment that use them.

Table 3 — Schema Relationships Compared to Allen's Relationships

Allen's Relation	Symbol	Endpoint Relations	Pictorial	Schema Equivalent
X before Y	<, b	$X_E < Y_S$		sequential (X before Y): $wn \neq 0$
X meets Y	m	$X_E = Y_S$		sequential or overlap (X before Y): $wn = wx = 0$
X overlaps Y	o	$X_S < Y_S$ & $X_E > Y_S$ & $X_E < Y_E$		overlap (X before Y): $wn \neq 0$, $w < min(X_D, Y_D)$
X starts Y	s	$X_S = Y_S$ & $X_E < Y_E$		during (X during Y): $wn_S = 0$, $wn_E \neq 0$ or overlap (X before Y): $wn \neq 0$, $w = X_D, <Y_D$ [1]
X during Y	d	$X_S > Y_S$ & $X_E < Y_E$		during (X during Y): $wn_S \neq 0$, $wn_E \neq 0$
X finishes Y	f	$X_S > Y_S$ & $X_E = Y_E$		during (X during Y): $wn_S \neq 0$, $wn_E = 0$ or overlap (Y before X): $wn \neq 0$, $w = X_D, <Y_D$ [2]
X equal Y	=	$X_S = Y_S$ & $X_E = Y_E$		during: $wx_S = wx_E = 0$ or overlap: $wn \neq 0$, $w = X_D, = Y_D$ [3]
X finished by Y	fi	$X_S < Y_S$ & $X_E = Y_E$		during (Y during X): $wn_S \neq 0$, $wn_E = 0$ or overlap (X before Y): $wn \neq 0$, $w = Y_D, <X_D$ [4]
X contains Y	di	$X_S < Y_S$ & $X_E > Y_E$		during (Y during X): $wn_S \neq 0$, $wn_E \neq 0$
X started by Y	si	$X_S = Y_S$ & $X_E > Y_E$		during (Y during X): $wn_S = 0$, $wn_E \neq 0$ or overlap (Y before X): $wn \neq 0$, $w = Y_D, <X_D$ [5]
X overlapped by Y	oi	$X_S > Y_S$ & $X_S < Y_E$ & $X_E > Y_E$		overlap (Y before X): $wn \neq 0$, $w < min(X_D, Y_D)$
X met by Y	mi	$X_S = Y_E$		sequential or overlap/ (Y before X): $wn = wx = 0$
X after Y	>, a	$X_S > Y_E$		sequential (Y before X): $wn \neq 0$

S and E indicate the start and end.
wn and wx indicate the minimum and maximum separations (waits)
D indicates the duration.

[1] Overlap becomes *starts* when the overlap = the duration of the shorter earlier item.
[2] Overlap becomes *finishes* when the overlap = the duration of the shorter later item.
[3] Overlap becomes *equal* if the scheduled overlap and both durations are equal.
[4] Overlap becomes *finished by* when overlap = duration of the shorter later item.
[5] Overlap becomes *started by* when overlap = duration of shorter earlier interval.

2.6 Modeling Flexibility and Nuances

Several of the features that have been defined are especially useful for modeling flexibility. They are alternate choice of constraints ("one-of" groups) in the activity, variable durations of an activity, variable timing in relationships, sequence scenarios, and optional items in a sequence. The subtle nuances of tasks can be modeled with features like lock-in, standby, fragmentation, and percent-coverage relationships.

2.7 Other Sequence Constraints

In addition to the temporal relationships between the items within a sequence, a sequence can have other constraints. A given sequence can appear in the timeline multiple times; for example, crew meal. Each appearance is called a performance. Multiple performances can occur when a top-level sequences is submitted multiple times or when a given sequence is embedded in other sequences that have multiple performances or when a given sequence is included multiple times within a single sequence.

Separation between Performances — A minimum and maximum separation between performances may be specified. If this separation is not specified, then performances may overlap each other.

Performance Duration — The modeling schema allows a lot of flexibility and variability in a sequence (paragraph 2.6). Often it is necessary to constrain the overall duration of a sequence. Consider the sequence with three sequential tasks, where each task has a duration of one hour and the delay between each task can vary from zero to two hours. Then the implied minimum duration of the sequence is three hours and maximum is seven hours. The schema allows the overall duration to be limited; for example, it could be forced to be exactly five hours. Then the sum of the delays would be exactly two hours.

Performance Windows — Performance windows bound the earliest start and latest end of a sequence. Performance windows are specified relative to an external event (such as launch) or to a specified epoch (such as 15 April, 2008). In addition, a maximum number of performances to schedule within a window may be specified. Windows may overlap and are attempted in the order listed.

Scenario Strategy — The preferred order of the scenarios may be specified.

2.8 Resource and Condition Modeling

All resource usage is decomposed into usage of nondepletables, depletables, and conditions.

Nondepletables — A nondepletable is a resource whose availability is restored when the consuming activity terminates. Examples are a crewperson, workbench, video camera, power drawn from a solar array, etc.

Depletable — A depletable resource is a resource whose availability is not restored when the consuming activity terminates. These are often called consumable resources. Examples are stored water, stored argon, electrical energy drawn from a battery, etc. Depletables can be replenished by a discrete event; for example, the water on the International Space Station is replenished when the Shuttle visits.

Conditions — A condition is a "resource" which has state. Examples are sunlight (yes or no), vibration (above or below a given threshold), communication link, etc. An activity or equipment model may specify that certain conditions may be required, avoided, or induced.

In addition, resources may be interdependent. For example, energy is power over time. Modeling compound resources, such as hot water, are usually modeled as equipment. For example, to model *hot* water, which uses both water and electrical power, a hot water spigot (a piece of equipment) might be modeled. It would use water (a depletable), power (a nondepletable), and possibly energy (a depletable). The usage of electrical energy is computed based on electrical power and activity duration.

3 An Example

Example Overview — The hypothetical Atmospheric Contamination Experiment (ACE) is an International Space Station payload that is designed to monitor both ionic and particulate contamination of the air inside the International Space Station. The hardware will be brought up on a Shuttle visit and returned to earth about three months later. The hardware consists of a base unit and six remote sensors. The base unit is attached via Velcro at a well-exposed location inside the main module and connected to both the power output receptacle and a data input receptacle. The base unit records data from the sensors in flash memory and periodically dumps the data to the ground.

The six remote sensors are attached at various locations within the module. The remote sensors are battery-operated and communicate with the base unit via infrared signals. The base unit has cradles for recharging the remote sensors; it contains changeable filters and a small fan to force air through the filters as each is exposed. There is a hydrogen sulfide (H_2S) generator for a special test. The models required to represent this experiment are

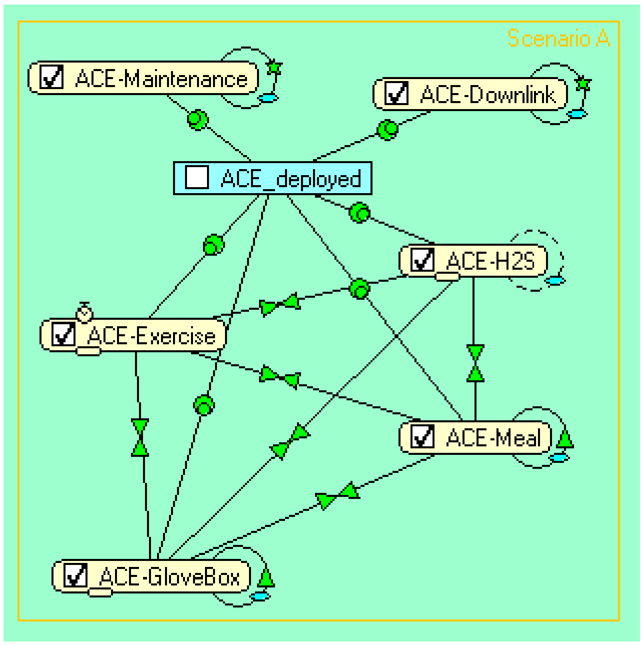


Figure 5 — The Main Model for ACE

discussed below beginning with a global model and then the details.

3.1 The Sequence Model

The model shown in Figure 5 depicts the relationships of the various sequences of the ACE experiment to one another. In the sequence shown, the ACE-Exercise, ACE-H2S, ACE-Meal and ACE-GloveBox tasks are separated (not allowed to overlap). The ACE-H2S sequence is optional, and the ACE-Meal, ACE-GloveBox, ACE-Maintenance and ACE-Downlink sequences are repeated multiple times, either with a cyclic relationship or a sequential relationship. It also shows that these tasks are

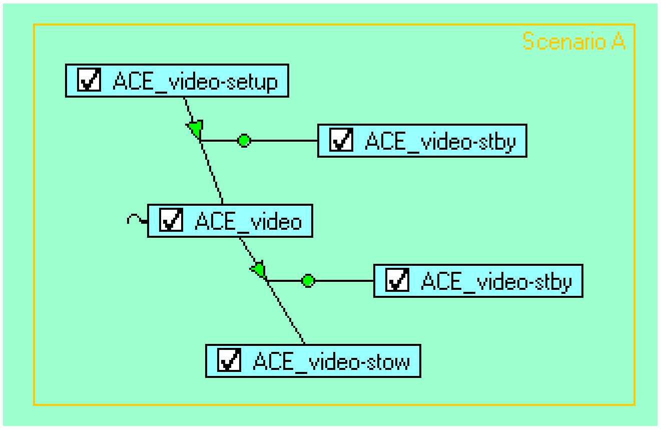


Figure 7 — ACE-Video-Conf Model

to be done while the hardware is deployed (the ACE_deployed activity is not checked for scheduling, but is scheduled by another sequence.) The embedded sequence, ACE-H2S, is shown in Figure 6. This sequence contains two scenarios or alternate ways of accomplishing the objective: one with a video conference and one without. It also has an avoid relationship with Meal. The video conference sequence is shown in Figure 7. ACE-Video-Conf has a hook on the ACE_video activity, indicating that relationships to this sequence are treated as relationships to the ACE_video activity. A graphics interface is used to build and edit a sequence. The items are selected from a list by clicking. They are positioned on the canvas by dragging and dropping, and they are connected by dragging between items (using another mouse key or a mouse modifier key). Details (see Table 1) are entered by double-clicking to invoke a dialog box.

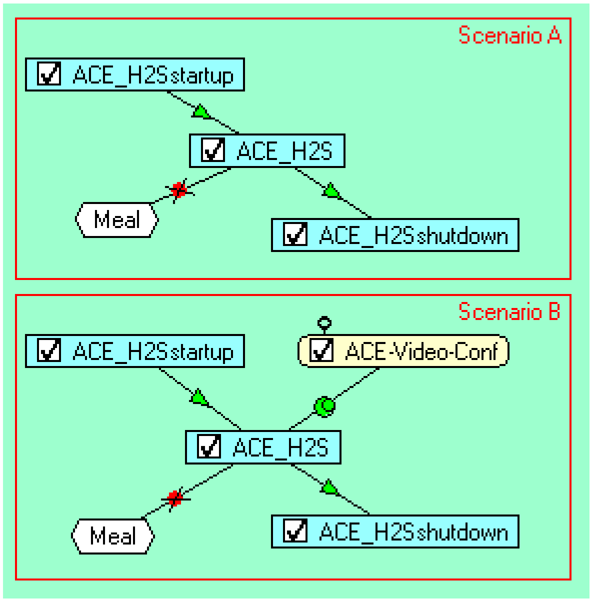


Figure 6 — ACE-H2S Model

3.2 The Activity Model

The model for ACE_video-setup is shown in Figure 8. This activity shows a hierarchy of constraints; if the activity is done by Crew: SC2, the duration is 20 minutes; if done by Crew: SC3, the duration is 15 minutes. It also uses the USL Camcorder in checkout mode and requires Orbital Day. A graphics interface is used to build and edit an activity model. The requirements and the

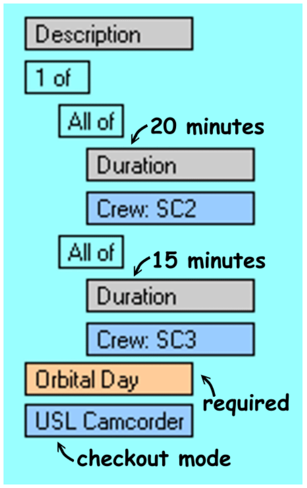


Figure 8 — ACE-video-setup

"one-of" and "all of" headers are selected from a list by clicking. The hierarchy is built by positioning the items on the canvas. Requirement quantities are entered by double clicking to invoke a dialog box.

3.3 The Mode Model

The model (not shown) for USL Camcorder equipment has five modes: "setup," which uses no resources, "checkout," which uses no resources, "record," which uses no resources, "battery charge," which uses power, and "download." which uses power and the video channel (all modes implicitly use the camcorder itself and all have different procedures). Mode models may contain hierarchies of requirements similar to activity models.

4 Summary

The modeling schema presented in this paper can enable new planning and scheduling paradigms. The objective of the maximally expressive modeling schema is to allow a person who has detailed knowledge of the task objectives and minimal knowledge of the hardware systems and scheduling technology to build usable models. Technicians with detailed knowledge of the system and experiment hardware would build the equipment mode models. They alone would work with the low level details. Scientists, technicians, or astronauts with detailed knowledge of the tasks would use these equipment mode models to build and submit scheduling requests.

The modeling schema is simultaneously robust (can represent complex requirements) and easy to use (relies on everyday terminology, constructs, and interfaces). Simple everyday relationships, such as during, overlap, etc., are employed; graphics paradigms are used to enter and display the information; and the nuances of the tasks can be directly represented. The schema is truly "maximally expressive."

The modeling schema presented in this paper is a key component of a system that could enable new scheduling paradigms. An operations concept based on the modeling schema and scheduling engine has been proposed [3].

Appendix A The Scheduling Engine

It is not enough to have good models with all the requirements expressed quantitatively; the scheduling engine must be able to process this data and produce the desired/expected schedule. The use of nested networks to capture temporal relationships, the complexity, and the variety of the relationships make all existing scheduling engines obsolete. A new scheduling engine designed specifically for the new modeling schema is being developed. The name "Scheduling Algorithm for Temporal Relation Networks" (SATRN) has been given to this new scheduling engine.

Overview

The scheduling engine is an incremental engine. It processes one request at a time, possibly adding multiple tasks to the timeline, and then waits for the next request. The timeline is incrementally built. Other classes of scheduling engines are "batch," which processes multiple requests simultaneously, and "mixed-initiative," which assists the user to build or edit a timeline.

The scheduling request submitted to SATRN is a network of nodes and temporal relationships between the nodes. A node can be an embedded network or an activity (activities specify resource usage). The algorithm in SATRN converts the temporal relations in a request to time bounds on the nodes. As each node is scheduled, the bounds on not-scheduled nodes are shrunk. As shown in Figure A, recursion is used to process embedded networks. Variable activity durations are utilized to stay within the time bounds. When a node cannot be "asserted" (scheduled but not committed to the timeline), already asserted nodes are adjusted to free up resources or smart backtracking and reordering are used to unshrink the bounds on the hard-to-schedule node. Backtracking is also used to explore alternate requirements ("1-of" groups, scenarios, optional nodes, and relationships to pre-existing items with multiple instances).

The window calculator interprets the temporal relationships in a sequence. It accounts for relationships to pre-existing items after choosing a specific instance. It

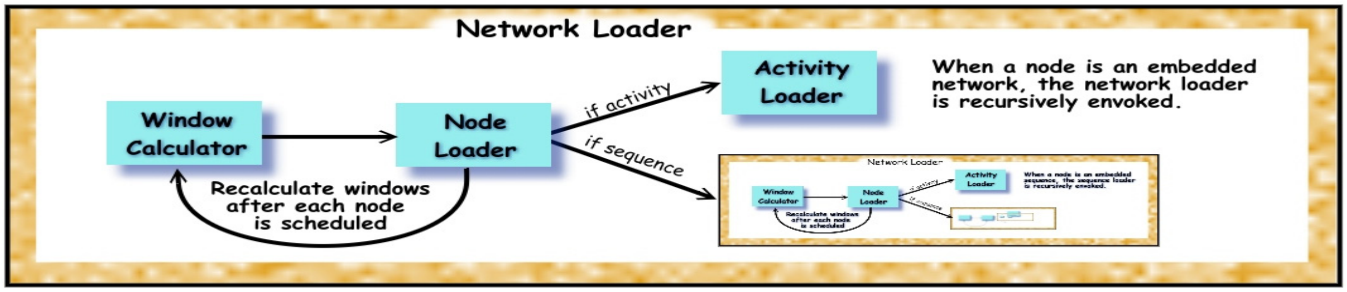

Network Loader
Window Calculator
Node Loader
if activity
Activity Loader
if sequence
When a node is an embedded network, the network loader is recursively envoked.
Recalculate windows after each node is scheduled

Figure A — Scheduling Algorithm Block Diagram, Computing the Windows

also accounts for relationships to already-asserted items. For each node to be scheduled, it computes a candidate window which ensures that, if the item is asserted in the window, windows for other items can be calculated. A candidate window is defined by an earliest start, latest start, earliest end, and latest end times.

Finding where on the schedule an activity can be scheduled within its candidate window is accomplished by standard solutions with the enhancements necessary to accommodate the modeling innovations. To check constraints, the activity loader uses a depth-first search (a standard Constraint Satisfaction Problem solution) but has been enhanced to accommodate modeling innovations such as variable duration and duration constraints that are dependent on which resource is chosen from a "1-of" group.

References

[1] J. Jaap, P. Meyer and E. Davis, "Using Common Graphics Paradigms to Represent Complex Scheduling Requirements," Workshop Notes - International Workshop on Planning and Scheduling for Space Exploration and Science, (28-30 October, 1997). http://nexus.nasa.gov/publications

[2] J. F. Allen, "Maintaining Knowledge about Temporal Intervals," Vol 26, No 11. (Communications of the ACM, November 1983,)

[3] J. Jaap and K. Muery, "Putting ROSE to Work: A Proposed Application of a Request-Oriented Scheduling Engine for Space Station Operations," (SpaceOps 2000, 19-23 June, 2000) http://nexus.nasa.gov/publications

Biographies

John Jaap has been employed with NASA's Marshall Space Flight Center since 1980. He has worked virtually all tasks related to planning and scheduling, serving as lead developer, designer and user of scheduling tools. He has designed and developed web applications since 1995. He has received NASA's Silver Snoopy, the NASA Exceptional Achievement Medal, the Space Act Award, and is a Space Flight Awareness Honoree. He received a Bachelor of Arts degree in Mathematics from Mississippi State University in 1966.

Elizabeth Davis began her employment with NASA at the Marshall Space Flight Center in 1980. She has developed mission planning software since 1977. In 1978, she began concentrating on space activity scheduling software. She has been involved in web application development since 1997. She is the recipient of numerous awards, including the Silver Snoopy and the Space Act Award. She graduated summa cum laude with a Bachelor of Science degree in Mathematics from Middle Tennessee State University.

Lea Richardson has been employed with the Marshall Space Flight Center since 1990. She has worked numerous console positions during space-flight operations, collected and modeled science requirements for Spacelab missions, and developed space-flight software. She has been developing mission planning and web application software since 1996. She graduated cum laude with a Bachelor of Science degree in Mathematics and a minor in Computer Science from the University of Alabama in Huntsville