

An Enabling Technology for New Planning and Scheduling Paradigms

John Jaap and Elizabeth Davis
Marshall Space Flight Center, Alabama

Abstract— The Flight Projects Directorate at NASA's Marshall Space Flight Center is developing a new planning and scheduling environment and a new scheduling algorithm to enable a paradigm shift in planning and scheduling concepts. Over the past 33 years Marshall has developed and evolved a paradigm for generating payload timelines for Skylab, Spacelab, various other Shuttle payloads, and the International Space Station. The current paradigm starts by collecting the requirements, called "task models," from the scientists and technologists for the tasks that are to be scheduled. Because of shortcomings in the current modeling schema, some requirements are entered as notes. Next, a cadre with knowledge of vehicle and hardware modifies these models to encompass and be compatible with the hardware model; again, notes are added when the modeling schema does not provide a better way to represent the requirements. Finally, the models are modified to be compatible with the scheduling engine. Then the models are submitted to the scheduling engine for automatic scheduling or, when requirements are expressed in notes, the timeline is built manually. A future paradigm would provide a scheduling engine that accepts separate science models and hardware models. The modeling schema would have the capability to represent all the requirements without resorting to notes. Furthermore, the scheduling engine would not require that the models be modified to account for the capabilities and limitations of the scheduling engine.

The enabling technology under development at Marshall has three major components. (1) A new modeling schema allows expressing all the requirements of the tasks without resorting to notes or awkward contrivances. The chosen modeling schema is both maximally expressive and easy to use. It uses diagrams to show hierarchies of task constraints and networks of temporal relationships. (2) A new scheduling algorithm automatically schedules the models without the intervention of a scheduling expert. The algorithm is tuned for the constraint hierarchies and the complex temporal relationships provided by the modeling schema. It has an extensive search algorithm that can exploit timing flexibilities and constraint and relationship options. (3) An innovative architecture allows multiple remote users to simultaneously model science and technology requirements and other users to model vehicle and hardware characteristics. The architecture allows the remote users to submit scheduling requests directly to the scheduling engine and immediately see the results. These three components are integrated so that science and technology experts with no knowledge of the vehicle or hardware subsystems and no knowledge of the internal workings of the scheduling engine have the ability to build and submit scheduling requests and see the results. The immediate feedback will hone the users' modeling skills and ultimately enable them to produce the desired timeline. This paper summarizes the three components of the enabling technology and describes how this technology would make a new paradigm possible.

1. Introduction

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 1-1, requires significant effort and flow time. The task experts often struggle to enter their requirements using a language that is limited – often resorting to notes to describe fully their requirements. The vehicle and hardware experts then convert and augment this knowledge to prepare further the models for scheduling. The scheduling team then feeds the models to the scheduling engine. Because the models are incomplete, the scheduling team often has to “steer” the scheduling algorithm or manually post-edit to produce an acceptable schedule.

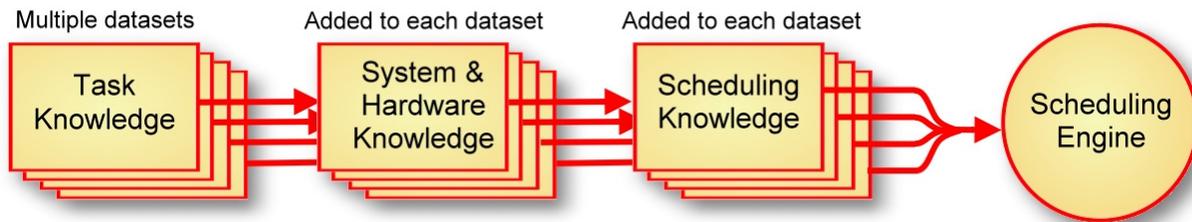


Figure 1-1 – State-of-the-Art Paradigm

This paradigm has begotten the “scheduling cadre,” a group of people which 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 schedule and a schedule editor (a mixed-initiative approach to scheduling).

The technology presented in this paper allows a streamlined paradigm as depicted in Figure 1-2. The vehicle experts would enter the system and hardware constraints independently of the task knowledge. The task experts would enter the task requirements. The modeling schema would allow them to specify all of the payload requirements without resorting to notes to the scheduling team; 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.

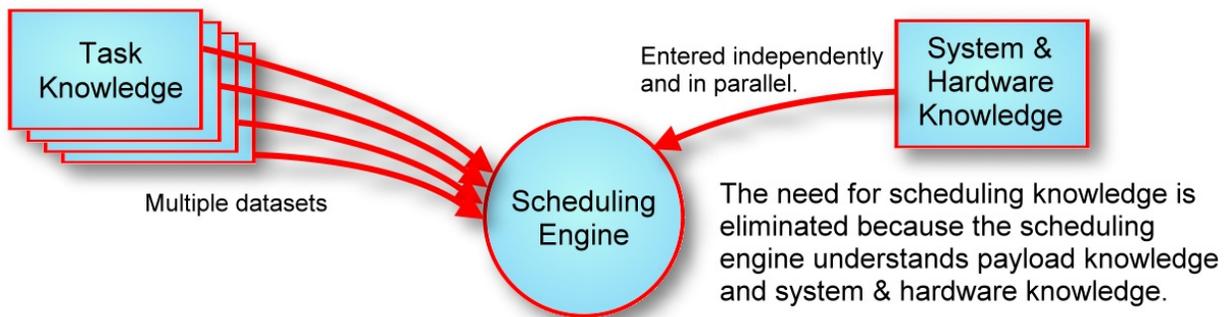


Figure 1-2 – Future Paradigm

Implementing the technology described in this paper can significantly reduce the manpower and flow time required to produce a schedule in a complex scheduling domain. An operations concept encompassing both the model schema presented in this paper and the corresponding scheduling engine has been proposed [3].

2. Maximally Expressive Modeling

Modeling even the simplest of tasks cannot be automated, 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 is a human enterprise – both an art and a science. The act of modeling is the act of expressing the requirements in the schema of the database. The schema must be “maximally expressive” to capture all the requirements without resorting to notes or awkward representations. Poor 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. The modeling schema must have an available representation for all the constraints and be friendly enough to allow the user to enter them all without excessive labor. 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 problem into salient components* — Operations are decomposed into activities that define resource requirements and sequences that define relationships between activities. Sequences can also contain other sequences, repeated activities and sequences, and optional activities and sequences.
- *Graphical paradigms* — Simple graphical paradigms such as outlines and networks are used to build and depict the models. Modeling itself is done using techniques such as drag-and-drop.
- *Modeling equipment modes* — Implicit resource requirements are defined by equipment mode models, thereby more closely representing the real world.
- *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.

The objective of the maximally expressive modeling schema is to capture easily *all* the requirements and constraints so that an automatic scheduler can produce a satisfactory schedule. The inspiration for the modeling schema is the real world that we interact with and observe each day. The schema is based on the scientist who goes to a lab to perform an experiment, the instructor who explains a complex

procedure to students, the housewife who prepares dinner for guests while helping with homework, and the various experiments that are performed on the International Space Station. 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].

Decomposition into Salient Components

This schema models scheduling requirements by defining "activities" and "sequences." Activities generally equate to the simplest or lowest-level tasks. A sequence of activities is usually required to represent a scheduling entity. Consider the following example: to collect rocks on the lunar surface, an astronaut must don a spacesuit, airlock out, drive the lunar rover to the site, collect the rocks, drive back, airlock in, and doff the spacesuit. Doing the activities out of sequence does not accomplish the objective. Activities define the resource requirements (with alternatives) and other quantitative constraints and requirements of the tasks to be performed. Activity requirements may be grouped into "all-of" groups or "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. Requirements include the specification of the minimum, maximum and preferred duration of the activity. Sequences define the temporal relationships between activities. In our rock collecting example, we discussed seven activities that would be done one after the other (in a sequence). Sequences may also define relationships with other sequences, as well as with events. For example: the rock collecting excursion is done after preparation (planning and scheduling, equipment checkout, etc) and before dinner (preparation, eating, cleanup). 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 but are outside the scope of this paper.

The discipline of Operations Research uses the terms "sequence" or "operations sequence." The discipline of Graph Theory would label these sequences, which have directed and weighted relationships, as "networks." The planning and scheduling cadre for the International Space Station uses the term "sequence." In this paper, the term "sequence" is used when discussing the user's perspective and the term "network" is used when discussing implementation.

Graphical Methods

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

Hierarchies of groups of requirements best describe the constraints of most non-trivial activities. The outline paradigm 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 2-1 compares two

representations of the “cooking a roast” example; notice the similarity between the two. The implementation provides a list of predefined constraints; clicking on one of the constraints adds it to the activity model at the current cursor location. “One of” and “all of” headings are also added by clicking on their respective icons. Constraints are arranged into groups and hierarchies via drag-and-drop. Values are added by double-clicking on a constraint item and entering data into a dialog box.

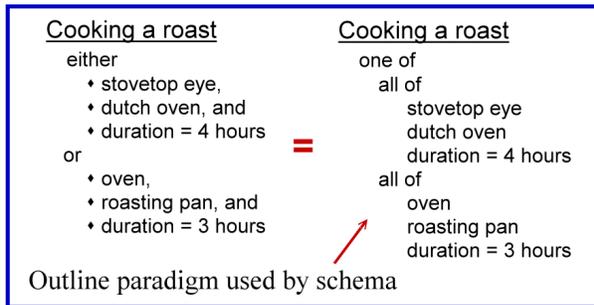


Figure 2-1 – Constraint Hierarchies

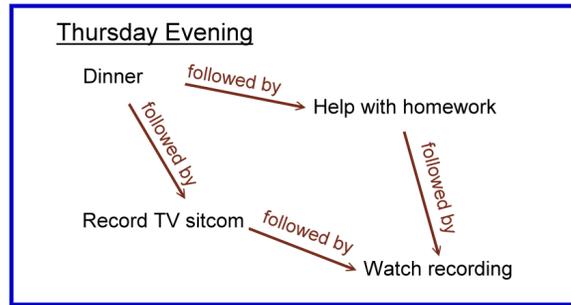


Figure 2-2 – Network Paradigm

Sequences use a “network” paradigm to define the relationships between tasks (activities and included sequences). The method consists of selecting tasks from a list and placing them on a drawing canvas. They are placed and connected using the mouse. When a connection is made, a dialog box appears to allow the user to specify the type of relationship and the time delays between tasks. Figure 2-2 shows a sequence for a Thursday evening in a hypothetical household. The “followed by” relationships in this example include slack time so that homework and the recording are not artificially forced to be the same duration.

Modeling Equipment Modes

Most tasks are accomplished by using equipment. Most equipment have various operating modes: e.g., a microwave oven in the lunar habitat has modes such as defrost, reheat, and cook. The power requirements of each mode are predefined. On the lunar base, the characteristics of each piece of equipment will be known to those building and integrating the equipment into the Lunar Habitat. The equipment and their modes can be modeled independently of the experiments that will use the equipment. Occasionally an experiment will need to use a piece of equipment in a new or novel manner; consequently, a new mode must be defined. Equipment mode models use an outline paradigm like that used by activity models.

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 sequence, resource usage, conditions required, etc.) are not visible to the

requesting sequence, but will be booked when scheduling. Public services are usually modeled in advance. An example from the International Space Station: One of the racks has an active system for damping vibrations. The model for the damping system has multiple activities. A user with an experiment in the rack does not need to model the damping system, but can request active damping of the rack by embedding the damping public service in the experiment model.

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 earth rise and vehicle landing).

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 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 are of 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: an excursion on the moon might need communication with earth for 60% of the excursion. This coverage 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.

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 don the space suit and the airlocking out, an item would be scheduled to show that the crewperson is not available. If airlocking out follows immediately after donning the spacesuit, then the standby item is not scheduled.

Fragmentable — When an activity may be fragmented into parts, an activity or sequence is scheduled to book the resources that are used during the interruption. For example, when the collected rocks are being cataloged, they would be laid out on the work bench. Sorting the collection could be fragmented into multiple short sessions, but between the sessions the work bench 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.

Cyclic — An item in a sequence may be repeated; minimum and maximum repetition counts may be specified. The frequency in hours, days or weeks may be specified. 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. Additionally, the temporal relationship of the repetitions can also be separated or overlapped with time constraints. When the minimum repetition count is zero, the item is considered optional.

Lock-In — If two activities in a sequence contain identical “one-of” selection groups, then the same constraints must be chosen when scheduling the sequence. For example: assume there is a choice of which notebook computer onto which the daily schedule can be downloaded and choice of which notebook to take on the rock collecting excursion. When scheduling the excursion, the same notebook must be chosen for both downloading and carrying.

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 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 embedded sequences — A sequence can specify that temporal relationships to it are to be applied to an item within itself. For example, a sequence to take a video might include video setup, video record, and video downlink activities; the video record would be specified as the item to which relationships are applied. Sequences embedding the video sequence and specifying the sequence is to run during an activity, would get video record during the specified activity; setup and downlink would occur before and after the activity to be recorded.

Modeling Flexibility and Nuances of the Tasks

Several of the features are especially useful for modeling flexibility. They are alternate choice of constraints (“one-of” groups) in the activity, variable durations of an activity, variable separations in a sequence, 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.

The temporal relationships defined by the schema are common sense relationships, not the classical (and sometimes esoteric) temporal relationships [2]. Sequential, separated, during, and overlap can be mapped directly to the classical relationships. The schema introduces percent coverage, fragmentable, and standby 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.

3. The Scheduling Engine (SATRN)

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 makes 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 process multiple requests simultaneously, and “mixed-initiative” which assist 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 3-1, recursion is used to process embedded networks. An activity’s requirements are checked by a depth-first search. 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 is 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).

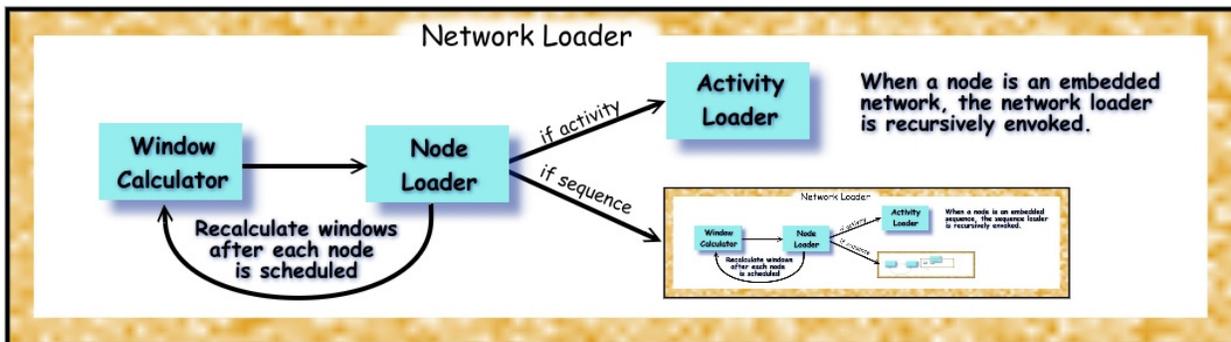


Figure 3-1 – Scheduling Algorithm Block Diagram

Computing the Windows

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

Asserting the Activities

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. The activity loaded 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.

4. Innovative Architecture

State-of-the-Art Implementation

A complete implementation of the technology would be a state-of-the-art web-based implementation as depicted in Figure 4-1. Further refinements could host the database and the scheduling engine on separate computers.

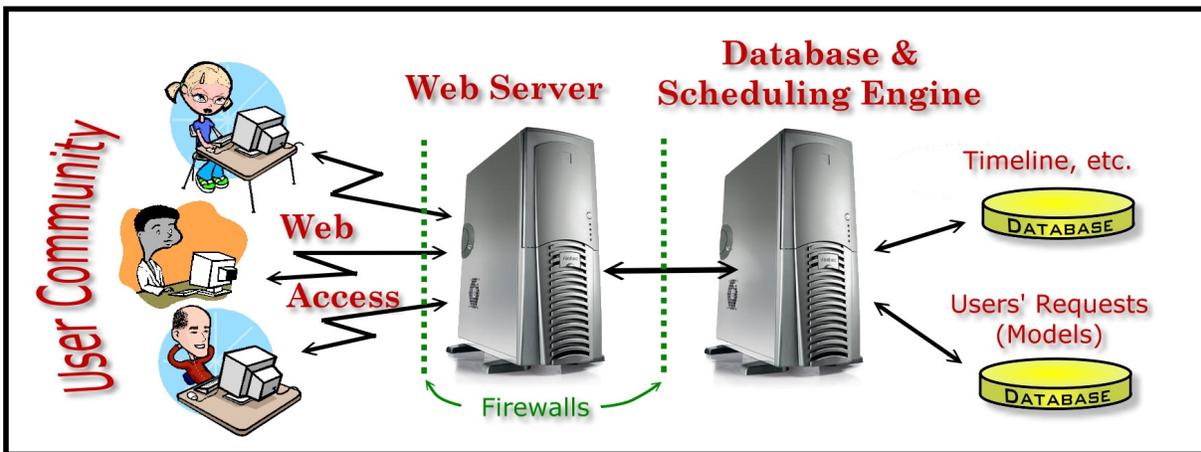


Figure 4-1 – Web-based Architecture

More compact implementations are possible. All of the components of the system could be hosted on one computer. A single-computer configuration might be useful for small projects or autonomous implementations. A pseudo-autonomous configuration might be used for scheduling activities in a remote operations site, such as a lunar base. All components would be installed locally at the remote site so that the remote site could operate autonomously, but the models would be built or uploaded from a base site. In the case of a lunar base, the complete system would be installed at the lunar base. The major part of task modeling and equipment modeling would be done by engineers and scientists on earth. The astronauts at the lunar base, based on their knowledge and experience, would make final adjustments to the models, and submit them to the scheduling engine to produce their daily schedules.

Just-in-time configuration control

The technology avoids the inconvenience associated with configuration control by delaying configuration control as late as possible. When a model is submitted for scheduling, a copy of the entire model is

made; if the model is successfully scheduled, the copy of the model is stored with the timeline and cannot be changed. However, the original model can still be changed. The user is not impacted by the configuration control. Configuration control of the equipment mode definitions is supported. Because resource requirements will usually be associated with the equipment model, the resource usage can be placed under full configuration control.

Contract-Oriented

When a user submits a request and receives a report showing when tasks in that request are scheduled, the user can be sure that those tasks will not be moved without his direct action. Multiple users can simultaneously build an integrated schedule by adding their tasks with assurances that the action of other users will not modify the schedule of their own tasks. The architecture is flexible; for example, an operations concept could use the scheduler only to build a preliminary schedule to be reworked extensively with a manual timeline editor.

5. Status (Spring of 2004)

The modeling schema is defined and implemented, the architecture is defined and demonstrated, and the scheduling engine is under development. The system is currently operating end-to-end. The equations to interpret the not-yet-implemented temporal relationships are being developed and implemented. The web server, the database and the scheduling engine are written in C++ and C# and are running on Windows 2000 Server platforms. The client application is written in J# and runs on Windows desktops.

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/iURC-1/iurc.html>

[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/SpaceOps2000/ROSEconcept.html>