# A NEW TECHNOLOGY FOR MANUFACTURING SCHEDULING DERIVED FROM SPACE SYSTEM OPERATIONS



**R.S. Hornstein**
**Office of Space Communications**
**NASA Headquarters**
**Washington, DC, USA**

**J.K. Willoughby**
**President**
**Avyx, Inc.**
**Englewood, CO, USA**

## ABSTRACT

A new technology for producing finite capacity schedules has been developed in response to complex requirements for operating space systems such as the Space Shuttle, the Space Station, and the Deep Space Network for telecommunications. This technology has proven its effectiveness in manufacturing environments where popular scheduling techniques associated with Materials Resources Planning (MRP II) and with factory simulation are not adequate for shop-floor work planning and control.

The technology has three components. The first is a set of data structures that accommodate an extremely general description of a factory's resources, its manufacturing activities, and the constraints imposed by the environment. The second component is a language and set of software utilities that enable a rapid synthesis of functional capabilities. The third component is an algorithmic architecture called the Five Ruleset Model which accommodates the unique needs of each factory.

Using the new technology, systems can model activities that generate, consume, and/or obligate resources. This allows work-in-process (WIP) to be generated and used; it permits constraints to be imposed on intermediate as well as finished goods inventories. It is also possible to match as closely as possible both the current factory state and future conditions such as promise dates. Schedule revisions can be accommodated without impacting the entire production schedule.

Applications have been successful in both discrete and process manufacturing environments. The availability of a high-quality finite capacity production planning capability enhances the data management capabilities of MRP II systems. These schedules can be integrated with shop-floor data collection systems and accounting systems. Using the new technology, semi-custom systems can be developed at costs that are comparable to products that do not have equivalent functional capabilities and/or extensibility.

## BACKGROUND

The operations of a space system such as the Space Shuttle, the Space Station, or a telecommunications satellite network have surprising similarities to running a manufacturing facility. Both space systems and manufacturing plants use scarce and expensive resources to satisfy objectives as efficiently as possible. Both must revise their expected activities when equipment malfunctions. Both must respond to opportunities that present themselves unexpectedly. In the realm of space systems operations, the resources may be satellites, antennae, or astronauts; whereas in a factory the resources are production lines, machines, and skilled laborers. A target of opportunity such as a solar flare, the creation of a distant black hole, or an Atlantic Ocean hurricane are to space operations what special orders or unforecasted sales demands are to manufacturing. In both domains, the goal is to get as much from the limited resources as possible, and to do so in a manner that responds to a changing environment.

Recently, the needs for "Finite Capacity Planning" and "Finite Capacity Scheduling" have been recognized by the

manufacturing community [1]. Although scheduling has long been a part of manufacturing support software such as an MRP II system, the logic used in those systems does not adequately model the limitations that exist hour-by-hour on the shop floor. These limitations represent the finite capacity that must be modeled accurately and updated frequently in order to plan and replan the production activities.

The finite capacity of space systems has been the driver for developing a new technology for scheduling and rescheduling operational activities. This technology has been applied successfully to several complex manufacturing environments. Very few requirements from manufacturing environments have stretched the generality and completeness of the technology that has emerged from the space operations domain. Therefore, manufacturers can benefit by inheriting the capabilities embodied in the new technology with minimum costs for customization.

There are several components to the scheduling technology that has emerged from space operations. The single greatest design driver for all of these components has been the need to adapt easily from one application to another. In the 1960s and 1970s, each new space system required a start-from-scratch design of the software systems needed to support operations. It was generally conceded that the approaches used for the Apollo (Moon Landing) Program would not work well for operating the new Space Station Freedom, or that planning the communications with satellites using orbiting relay satellites would require different software than that used to plan the communications with ground-based antennae. The costs for each new application were very high. As a result, NASA sought ways to abstract the planing and scheduling problem, i.e., to find a generic way to describe and solve these problems that could be applied to any new space program.

The analogy to manufacturing is again apparent. Analysts have regarded the differences among production environments to be sufficient to justify custom development of finite capacity shop floor planning and scheduling systems. For example, the details of an aircraft brake manufacturing plant were not seen as similar to the processes for producing soups and canned vegetables. The search for generalizations and descriptive abstractions was not seen as a feasible task.

## DESCRIPTIVE ELEMENTS

In Table 1, elements of a descriptive vocabulary are shown which are domain independent. The Table is not a complete presentation of all possible descriptive elements, but gives some examples from both manufacturing and the space operations world. The authors have been involved in the development and the application of planning and scheduling techniques to several space and manufacturing systems. Although the use of one single system for all of these environments is not (yet) possible, the degree of reusability of concepts, data structures, system architectures, algorithmic components, and software modules is remarkably high, and still increasing rapidly. Knowledge gained from one application suggests an approach that can be generalized; the result is that each successive application benefits from a rapid accumulation of reusable software features and modules.

| GENERIC DESCRIPTIVE ELEMENT | EXAMPLE FROM SPACE OPERATIONS | EXAMPLE FROM MANUFACTURING |
|---|---|---|
| Resource (Pooled) | Propellant<br>Bandwidth<br>Electrical Power | Labor Skills<br>Chemicals<br>Work-in-process (WIP) Base Brand |
| Resource (Individual) | Crew Person<br>Antenna<br>Tape Recorder | Milling Machine<br>Furnace Crane |
| Activity | Playback Recorder<br>Crews Sleep<br>Send Command Sequence | Make Subassembly X<br>Ship Product X<br>Perform Preventive Maintenance |
| Condition | Daylight Only<br>Alternate Orbits<br>Every 3 Earth Days | First Shift Only<br>Not on Weekends |
| Temporal Relations | Record Before Playback<br>Exercise Before Eating | Routings<br>PM at Least Every 3 Days |

**Table 1: Analogies Between Space Operations and Manufacturing**

The accumulation of generic insights has produced a set of descriptive data structures that are inherently hierarchic and asymmetric. For example, the generalized description of an activity to be scheduled, whether it be in space operations or manufacturing, can be captured in the data structure shown in Figure 1. Experience has shown that different application domains will require more or less information at any level in this tree-like structure, but will not require new levels in the data structure. Some applications will have activities that result in broad bushy activity trees; others will use narrow or sparce structures. Note that the number of branches and levels in some parts of the data structure are not the same as those in other parts. Hence, the observation that the data structures are asymmetric. This characteristic makes scheduling data difficult to manipulate in traditional tables or matrices which are the fundamental data structures of modern relational data base systems. We have found repeatedly that although relational data base systems are very appropriate for storing, retrieving, and reporting the inputs and outputs from a scheduling process, they are inappropriate for supporting the computational process of generating or revising a schedule.

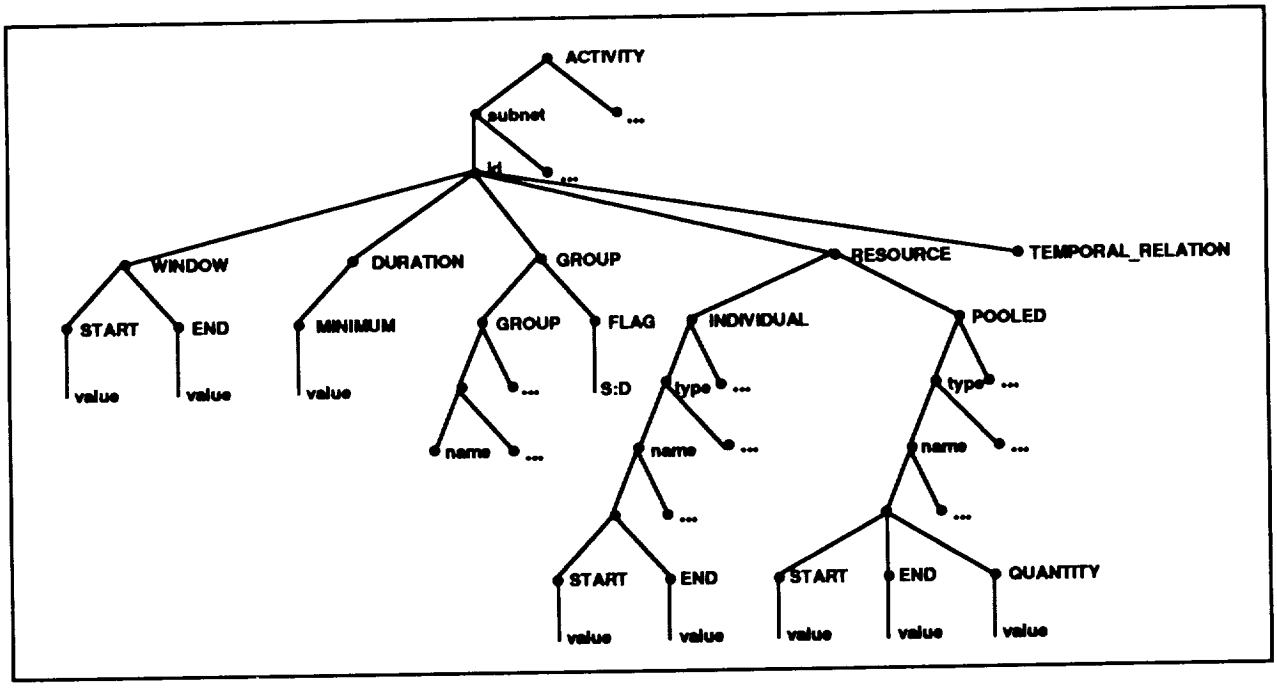


**Figure 1: Hierarchic Asymmetric Structure of Activities to be Scheduled**

Information about resource limitations (finite capacity descriptions) and timing OR sequencing constraints are also easily represented by asymmetric hierarchic data structures. Some of these structures are shown generically in Figures 2a and 2b. As a result, the descriptive mechanisms that have proven to be the most transportable, i.e., the easiest to apply in a very broad range of application domains are these hierarchic tree-like structures. These structures fit well with the concepts of object-oriented software development.

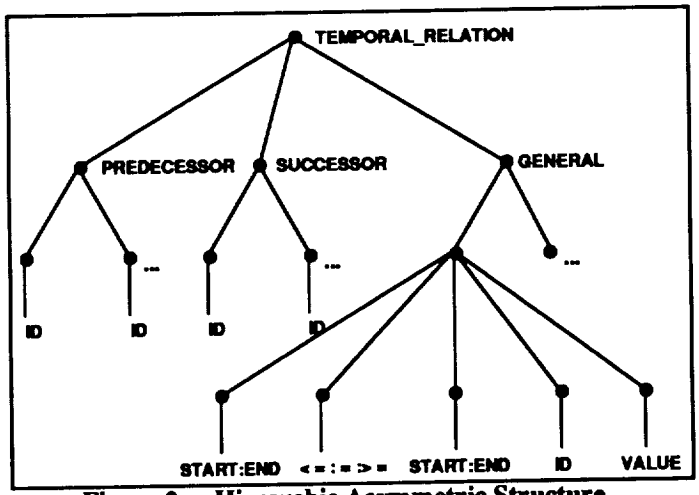


**Figure 2a: Hierarchic Asymmetric Structure of Temporal Constraint Data**

The generic descriptive framework provided by these data structures provides several advantages specifically for manufacturing environments. Among them is the generality of the resource modeling. Most manufacturing support software makes distinctions among different types of resources that the generic data structures do not. There need not be modeling differences between inventories, machines, power, raw materials, supplies, work-in-process (WIP), or labor using the generic resource data

model. Since any resource can be obligated, generated, consumed, or have its attributes transformed, the modeling flexibility is enormous. For example, one step in a manufacturing process can create a new resource which a subsequent step can consume. This allows modeling of flexible routings without complex sequence relationships. In applications of generic resource data models, the authors have found significant efficiencies that could not have been possible if fixed routings had been imposed by the limitations of a scheduling system.

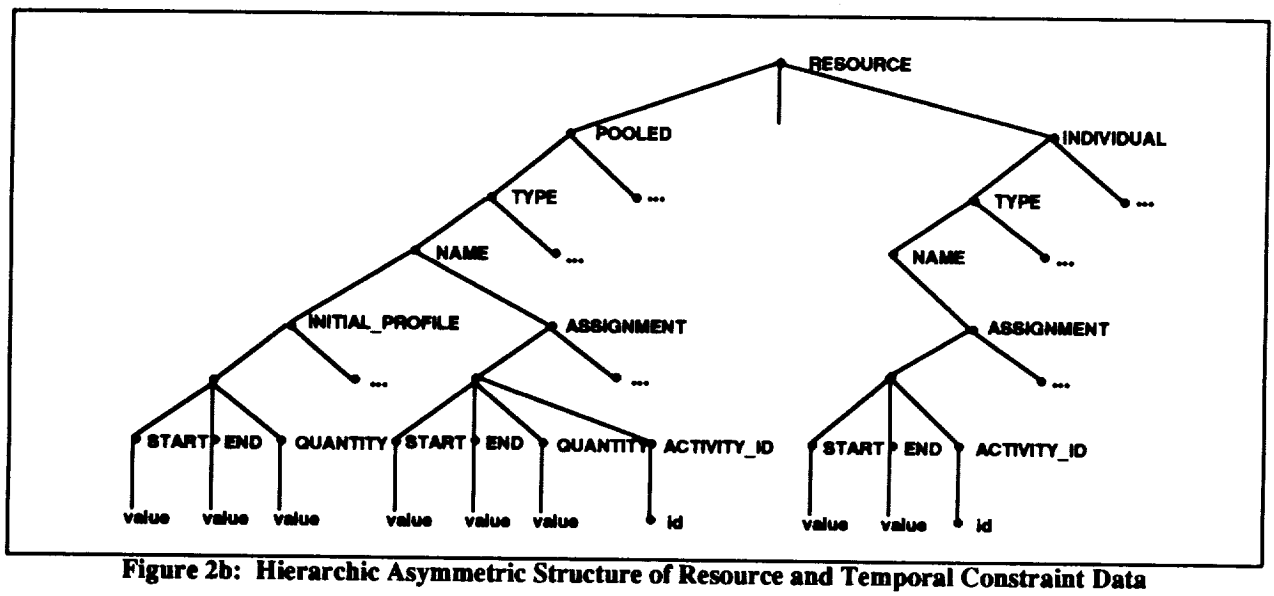


**Figure 2b: Hierarchic Asymmetric Structure of Resource and Temporal Constraint Data**

Another example of the resource data model flexibility is the use of generic descriptors with any resource. These descriptors can cause a part of any order to be tracked from one "location" to another or from one state of "completion" to another. The location and/or completion descriptors are simply attached to the resource. Any number of such descriptors can be used on any resource.

## A LANGUAGE AND UTILITY LIBRARY

The solution to planning and scheduling problems in either space operations or manufacturing requires the manipulation of these tree-like data structures. Goal-directed research in the 1970s and 1980s along with field testing and revision led to a simple realization. If the output of a scheduling process, i.e., a schedule, was a hierarchic asymmetric data structure that looked a lot like the input data structures, then the scheduling process should be describable as a systematic manipulation of input data structures into output data structures. The concept is illustrated in Figure 3.

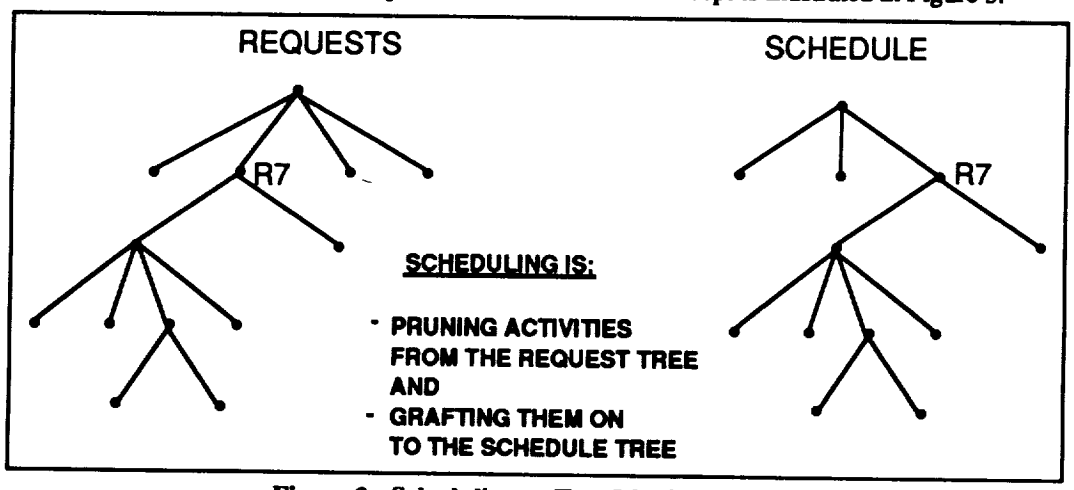


**Figure 3: Scheduling as Tree Manipulation**

Our premise has become the following:

> Scheduling can be described as the systematic manipulation of tree-like data structures in such a way that objectives are met and constraints are satisfied.

A software programming language that was idealized for manipulating these tree-like structures was devised in order to test this premise. Over twenty scheduling applications have been developed using this language [2] with the result that the average size and development time for applications have been reduced by a factor of approximately twenty when compared to custom-built systems from the 1970s and 1980s. This language has now evolved to be a set of data-structure manipulators written in C++ [3]. Scheduling systems are currently under development using these tree-manipulation capabilities in C++ for both space operations and manufacturing applications.

## ALGORITHMIC ARCHITECTURE

To complement the data manipulation capabilities, a set of scheduling utilities has also been developed. These utilities are software modules that find frequent use in all scheduling applications. These reusable modules perform constraint checking, interval and set algebra, and data management operations that are independent of any application domain. The names of these modules are shown in Table 2. The module names suggest their functionality.

| | |
|---|---|
| **Scheduling Routines** | **Activity Management Routines** |
| FindEarliestTime | InvertRelations |
| FindLatestTime | OrderByRelations |
| ScheduleActivity | **Constraint Enforcement Routines** |
| UnscheduleActivity | EnforceRelations |
| **Resource Management Routines** | EnforceConditions |
| AddResource Assignment | EnforceResources |
| AssignResources | **Profile Management Routines** |
| DetermineConsumableAvailability | AddProfile |
| DetermineReusableAvailability | CollapseProfile |
| FreeResource | ComplementProfile |
| UpdateResourceAvailability | ComputeProfileEnd |
| UpdateResourceStates | ComputeSegmentArea |
| **Interval Algebra Routines** | ComputeSlope |
| CollapseIntervals | GetSegmentTime |
| ComplementInterval | GetSegmentQuantity |
| ComputeIntervalDuration | IntersectProfileSegments |
| IntersectIntervals | IntersectProfiles |
| UnionIntervals | SetProfile |
| **Miscellaneous Routines** | SubtractProfile |
| GetForestVersion | UnionProfiles |

**Table 2: Contents of a Scheduling Utility Library**

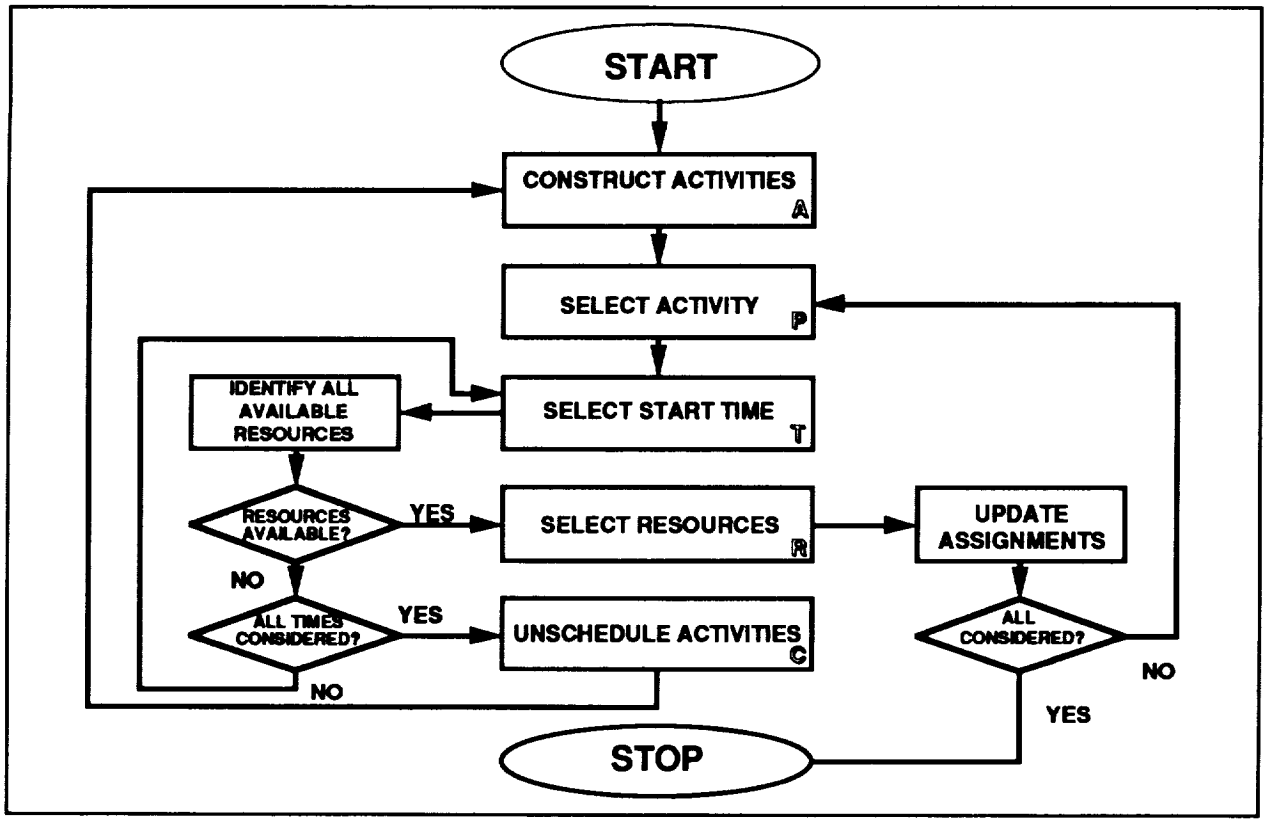


**Figure 4: The Five Ruleset Model Architecture**

In addition to the descriptive data structures and the language for application building already described, a third component of the technology is an algorithmic architecture called the Five Ruleset Model. This Model is a framework for describing the decision-making processes used in a broad range of scheduling algorithms. As shown in Figure 4, the Five Ruleset Model decomposes the solution logic of an algorithm into five nearly-decoupled sets of decisions called rulesets. The concept is that once each of these rulesets is specified, a unique algorithm is completely specified. All of the non-decision-making software can be pre-built and available as reusable code. The majority of most scheduling applications has proven to be in constraint checking and book-keeping of the obligations of the resources. These portions of the scheduling algorithm can be pre-built using the data structures, the language, and the utility library approach previously described. The remaining tasks in implementing a system for a new environment are the specification and implementation of the unique decision-making rulesets for the Five Ruleset Model. Once determined, these rulesets can be inserted into the architecture as shown in Figure 5.

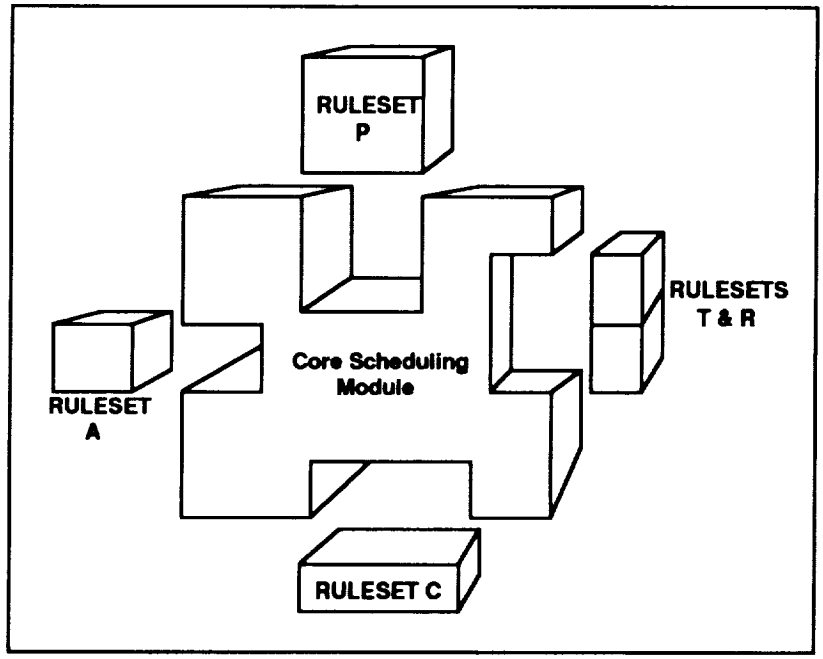


**Figure 5: Reuse of the Five Ruleset Model for Different Applications**