N93-11956

# COMPASS
# An Ada Based Scheduler

**Mary Beth McMahon**
Mc Donnell Douglas Space Systems Company
16055 Space Center Boulevard
Houston, TX 77062

**Chris Culbert**
Software Technology Branch
NASA - Johnson Space Center

*Abstract: COMPASS is a generic scheduling system developed by Mc Donnell Douglas and funded by the Software Technology Branch of NASA-Johnson Space Center. The motivation behind COMPASS is to illustrate scheduling technology and provide a basis from which custom scheduling systems can be built. COMPASS was written in Ada to promote readability and to conform to DOD standards. COMPASS has some unique characteristics that distinguishes it from commercial products. This paper discusses these characteristics and uses them to illustrate some differences between scheduling tools.*

## Introduction

COMPASS is an automated scheduling system which is highly interactive. It allows the scheduler to control the placement of activities on the schedule while maintaining the integrity of the schedule by adhering to resource and temporal constraints. It is generic in the sense that it was not developed for any specific problem domain. Rather, any scheduling problem that can be described by activities, resources, and their constraints can be modelled using COMPASS.

The motivation behind COMPASS was to provide scheduling technology to the various NASA groups working on scheduling problems. It was written in Ada to promote readability and reusability. It is public domain software. Anyone working on a government project may obtain a copy of the executable and the source code to read or modify as needed. It is very portable and currently runs on a variety of machines. To support a larger number of users, two versions of COMPASS were produced: a graphical version and an ASCII version. The graphical version is based on X-Windows and runs stand-alone on a SUN3/SUN4, SPARC station, IBM RS6000 and DEC VAX. In the client/server mode, it can be run over internet using an Apollo or a MacIntosh to display the image while executing on any of the stand-alone machines. For those with minimal computing resources, an ASCII version is available which prompts the user for commands. This version has the same scheduling power as the graphical version.

This paper will describe some scheduling concepts and will explore traits of COMPASS that distinguish it from commercially available products. This knowledge is useful in analyzing scheduling problems and determining which tools and algorithms best fit a particular scheduling need. The traits discussed in this paper also serve as a measure by which to compare many of the commercial scheduling tools currently available.

## Data Representations

There are four primary data objects that COMPASS uses to describe a scheduling world: Activities, Resources, Conditions and Time. Activities are the items which must be scheduled. Resources are the objects which must be present in order for the activity to be performed. These include such things as tools, electricity, and people. Conditions describe the state of the scheduling world. Activities may require that certain conditions exist before they are scheduled. Conditions may include things such as in-orbit, daytime, Phase III, etc. Time is used in all scheduling packages; however, the way it is represented affects the largest and smallest time units that may be used by the problem.

### The Activity Data Structure

COMPASS provides thirteen fields to describe an activity. All but two are optional. This allows the user to create a data structure that closely fits the scheduling problem. The two mandatory fields are the activity name and the duration of the activity. The following is a list of all of the fields of the activity data structure:

Name - *string - this is used as the unique id.*
Duration - *time type.*
Priority - *integer.*
Earliest Start - *time type.*
Latest Finish - *time type.*
Predecessors - *list of activity names that must precede this activity.*
Successors - *list of activity names that must follow this activity.*
Temporal Constraints - *timing constraints between the start or finish of one activity and the start or finish of another. There are four types of relations that may be represented: start/start, start/finish, finish/start, and finish/finish.*
Nonconcurrent Activities - *list of activity names that may not occur the same time as this activity.*
Preferred Intervals - *list of times which it is preferred that the activity occur. These times must fall between the earliest start and latest finish.*
Excluded Intervals - *list of times between the earliest start and latest finish in which the activity may not occur.*
Resource Requirements - *names of resources and their quantities required by this activity.*
Condition Requirements - *names of conditions and the times which they must exist in order for this activity to be scheduled.*

Most scheduling tools support the following fields: name, duration, priority, earliest start, latest finish, resource requirements, predecessors, successors and to some degree temporal constraints. The other fields are unique to COMPASS and have been included as optional fields in response to actual scheduling problems. Due to the modularity in the design of the activity data structure, it is trivial to add new fields to the activity. The scheduling engine is easily modified to conform to the requirements of the new field.

### The Resource Data Structure

A resource is typically represented by the resource name and a quantity for a given time period. COMPASS uses this resource structure is used in three places- in the initial and net resource availability description and in the activity data structure to describe which resources are needed and in what quantities.

There are several characteristics of resources that must be considered when describing a resource need. Resource descriptions may be broken down into two types: Piecewise Constant and Piecewise Linear. Piecewise constant resource descriptions allow the use to request a quantity over an interval of time. More complex piecewise constant descriptions allow several quantities over several intervals of time. Piecewise linear descriptions specify a rate of consumption or production. An example might be the consumption of water - an activity might use a quart of water per minute for 1.5 hours. The closest approximation to this using piecewise constant is to describe it using "stair steps". However, this is more difficult to read and understand and is also not as accurate. A piecewise linear description would be a straight line whose second endpoint is 90 quarts lower and 1.5 hours later from its first endpoint.

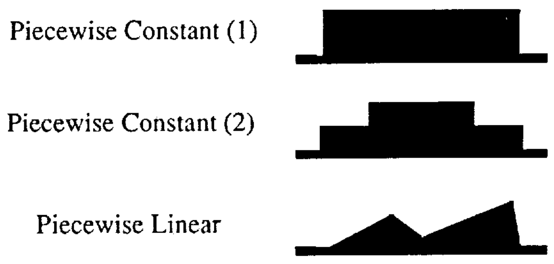

Figure 1. Resource Descriptions

Another characteristic of resources to consider is that they may be assignable, consumable or producible. An assignable resource is one that is used for the duration of the activity and then returned. A consumable resource is one that is consumed during the resource; therefore, when the activity finishes, that resource is no longer available. A producible resource is one that is produced by an activity.

It does not exist before the activity, but after the activity is under way the resource is available to be used by other activities. An example of an activity that both produces and consumes resources is the electrolysis of water. Water is consumed (at a rate) by the activity while hydrogen and oxygen are produced.

Most PC products support only the assignable resources and the first instance of piecewise constant resource descriptions. Larger software products usually only support assignable resources and both instances of piecewise constant resource descriptions. COMPASS supports all of the above types of resources and resource descriptions.

## The Condition Data Structure

A concept that is not commonly supported by most scheduling tools is the ability to schedule activities only when certain conditions exist. COMPASS provides this capability using a data structure called Conditions. Conditions are used to describe the state of the world. Conditions are propositions whose values change between true, false and undefined over time. A condition is represented by a name and the intervals of time when the condition is true, false or undefined. If an activity requires that a certain condition be true, then the scheduling algorithm only schedules that activity during the span of time when the said condition is true.

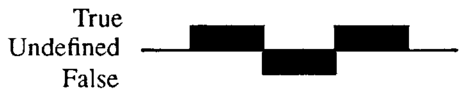

Figure 2. Condition Representation

## The Time Data Structure

Most scheduling packages use calendars to describe the times when resources are available and dates to specify when activities should begin and end. Typically, the smallest unit of time is one minute and the largest unit of time one year. Dates can range from the late 1900's to the early 2000's.

COMPASS supports date representations and relative time representations. Relative time allows the user to specify time requirements in relative measures. For example, the earliest start for an activity might be two weeks. This means that the earliest start time for the activity is two weeks after "time zero", where time zero is an anchor which does not correspond to a particular date. This allows the user to schedule activities two weeks before launch or one week after launch, without knowing exactly when launch is. Relative dates may be positive (eg. Launch plus one week) or negative (eg. Launch minus 3 days).

## Characteristics of Scheduling Tools

There are three characteristics of a scheduling process which affect the way the schedule is built and the quality of the resulting schedule. These characteristics have minimal support, if any, in most commercial products; however, they are fully supported by COMPASS.

The first characteristic to look for is whether or not the scheduling tool is interactive. Can the scheduler affect the placement of activities when the schedule is being built? With most commercial products, all of the data is entered and then the schedule is created automatically placing each activity on the schedule as early as possible. COMPASS allows the user to put activities on the schedule individually or in groups. They can be placed as early as possible or as late as possible or at any feasible time in between. Activities are easily unscheduled and rescheduled to respond to various scenarios.

Another characteristic COMPASS supports is that of incremental scheduling. This allows the user to build a schedule in one activity at a time. It is not necessary to have all of the activities defined before starting to build a schedule. Once an activity is placed on the schedule, adding new activities does not interfere with those already on the schedule. That is, inserting an activity into the schedule does not change the starting times of any of the activities already on the schedule. This allows a user to make late additions to the schedule without affecting already scheduled activities. It also allows the user to place high priority items at preferred times with the guarantee that lower priority items will not move them from their slots.

A third characteristic is COMPASS supports non chronological scheduling. Non chronological scheduling allows the user to select any time in the future to place an activity. In chronological scheduling, a user starts at a designated time, schedules any activities that can be scheduled at that time, then moves forward in time until an event occurs which allows other activities to be scheduled. This may be compared to planning a weeks worth of activities. The non chronological method would provide a calendar on which you could see the entire week ahead of you and write down which activities you wish to do in any order on the days you wish to accomplish them. The chronological approach would require that you start writing down on Monday, determine which activities to do, write those down then move on to Tuesday, look at what is left to do, write those down then move to Wednesday, etc. until all of the activities have been scheduled. You could not say in advance that "I will do this Friday at 9:00", rather you would have to wait until you have scheduled everything from Monday to Thursday night, then Friday if it is still left to be done, then you can scheduled it. In this example, I moved forward one day at a time. In actuality, time is moved forward by the smallest increment or by events.

## The Scheduling Algorithm

Another distinguishing characteristic of COMPASS is the scheduling engine. Most schedulers are based on the Critical Path Method (CPM) algorithm. CPM is a chronological based scheduling algorithm. CPM schedules each activity at its earliest starting time. The earliest starting time of each activity is determined by the latest completion time of all of its predecessors. CPM does not take into account any of the resource constraints. The result is a scheduling which finishes in the shortest amount of time, but oversubscribes resources. This type of scheduling algorithm is good for estimations and loading studies. A loading study is used to determine the number of resources needed to complete a schedule by a given date. But with most scheduling problems, the number of resources are limited. Therefore, oversubscribing the resources produces an infeasible schedule.

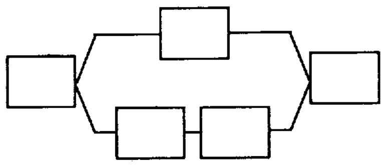

Figure 3. Critical Path Method

To alleviate the problem of oversubscription, most CPM tools also provide resource leveling. This technique shifts activities forward in time until the resources are available.

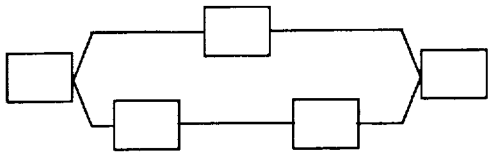

Figure 4. CPM with Resource Leveling

COMPASS does not use CPM. It uses a scheduling algorithm which takes into account both the temporal constraints and resource constraints. Given an activity, it determines all of the feasible times that the activity may be scheduled. Then the user provides some direction on the placement of the activity. This algorithm allows the user to select the order in which activities are placed on the schedule, as well as influence the placement of each as it is being scheduled. The feasible intervals of time in which an

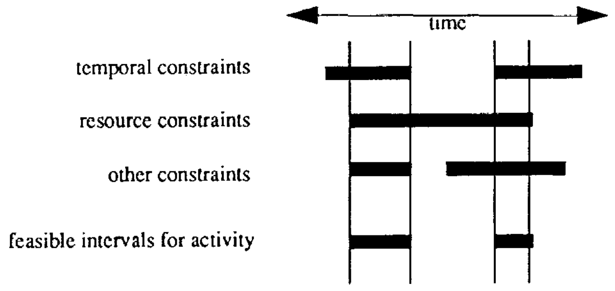

Figure 5. Computing Feasible Intervals

activity may be scheduled is determined by calculating all of the intervals of time that are feasible for each constraint and then taking the intersection. This gives the flexibility of choosing which constraints to enforce. It also provides the user with information on all feasible times that the activity may be scheduled and allows the user to select the placement.

Another distinguishing feature of the scheduling algorithm used by COMPASS is that it allows the user to constrain both the temporal and resource constraints simultaneously. Most commercial products will create a schedule abiding by the timing constraints and oversubscribing the resources where necessary. Then if the user wishes, resource leveling can be applied. There are two ways resources can be leveled. First the activities can be moved forward in time to the point where enough resources are available to satisfy the activity. This causes the successors of this activity also to be moved forward in time also. Second, the quantities required by the activity can be reduced and the duration increased proportionally. This kind of resource leveling produces two potentially undesirable results. First, it changes the quantities and durations of activities. In some instances, this would not produce feasible results. Second, it may push activities forward in time past their deadlines. If there are hard limits on both time and resources CPM and resource leveling will not produce feasible schedules, because on an oversubscribed problem it will violate either the timing or resource constraints.

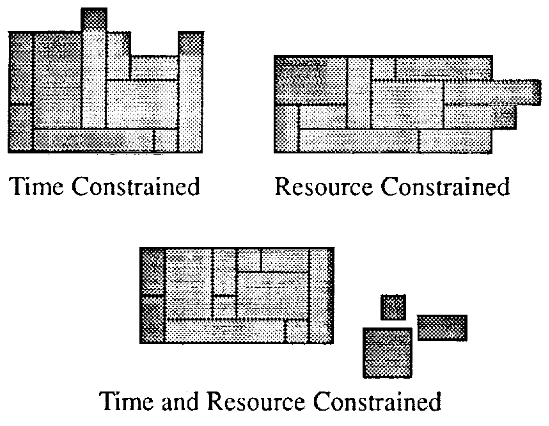

Figure 6. Three Ways to Handle Oversubscription

COMPASS will not violate the timing or resource constraints. Instead, it will inform the user that the activity cannot be scheduled. One reason for taking this approach is that most space related scheduling problems have hard limits on both the time and resources. For example, when scheduling space station or shuttle activities, the duration of the mission is fixed in advance and the number of resources is limited.

## Conclusions

COMPASS is unique in that it provides the user more control over the placement of activities than most commercial products do. This is due to its unique scheduling algorithm which determines all of the feasible intervals of an activity then allows the user to affect the placement of it

COMPASS is also unique in that it will constrain both the resources and temporal relations simultaneously. This addresses those scheduling problems which have a fixed time to complete and fixed resources.

## Acknowledgments

This development of COMPASS and related scheduling research is supported by NASA under NAS9-17885.

## References

1. Fox, B. R., *Mixed Initiative Scheduling*, AAAI - Spring Symposium on AI in Scheduling, Stanford, CA, 1989.

2. Fox, B. R., *Non-Chronological Scheduling*, AAAI - Spring Symposium on AI in Scheduling, Stanford, CA, 1989.