Using the Project Management Tools in the SAS® System

Radhika Kulkarni and Gehan Corea,
Operations Research R & D,
SAS Institute Inc., Cary, North Carolina

Abstract

The project management procedures in SAS/OR® software have been growing in scope continuously over the years. The first part of this paper briefly lists some of the new options that are available in the CPM, GANTT, and NETDRAW procedures. The major focus of the paper, however, is to describe and illustrate some of the more advanced scheduling features of the CPM procedure.

The CPM procedure allows you to specify target dates, save baseline dates, update a project that is in progress, schedule subject to scarce resources, and perform several scheduling tasks. Each of these features of the procedure can be fine tuned according to your needs using the options available. To effectively use any project management system, it is essential to understand the heart of the system; in particular, you need to understand how all the scheduling options work together.

New Features

Several new options have been added in Release 6.07 (SAS Institute Inc., 1992). Some of the highlights are

- logic Gantt chart
- time scaled and zoned network diagram
- alternate resources in PROC CPM
- baseline schedules
- lag calendars
- activity delay

Software Scheduling Project

Consider a simple project in a software development department. The main resources that are required in a software project are the programmers. Suppose that there are three programmers: A, B, and C. The activities in the project, their durations in hours, precedence relationships, and the resource requirements are printed in Output 1. We shall use this project to illustrate several aspects of resource constrained scheduling used in conjunction with the BASELINE statement which allows you to examine different scheduling options.

Software Development
Project Data

OBS	ID	HOURS	ACT	S1	S2	A	B	C
1	Plans & Reqts	18	1	2	3	0.5	0.5	0.5
2	Product Design	20	2	4	5	1.0	0.5	.
3	Test Plan	20	3	6	7	0.5	.	0.5
4	Documentation	60	4	9	.	.	.	1.0
5	Code	80	5	8	.	1.0	.	.
6	Test Data	36	6	8	.	.	0.5	0.5
7	Test Routines	36	7	8	.	.	0.5	0.5
8	Test Product	40	8	9	.	0.5	1.0	.
9	Finish	0	9	.	.	.	.	.

Output 1 Project Data

The project is scheduled to start on April 1, 1992 and the programmers work from 8 a.m. to 5 p.m. on weekdays with an hour off for lunch and from 8 a.m. to 12 noon on Saturdays. There is a company holiday on April 17, 1992. The holiday and calendar data are printed in Output 2. The resource availability data set is printed in Output 3. Note that all three resources are identified as *replenishable* by the value `1` for each of the resources `A`, `B`, and `C` in the first observation.

Software Development

Shift Data

OBS	SHIFT1	SHIFT2
1	8:00	8:00
2	12:00	12:00
3	13:00	.
4	17:00	.

Calendar Data

OBS	_CAL_	_SUN_	_MON_	_TUE_	_WED_	_THU_	_FRI_	_SAT_
1	0	holiday	shift1	shift1	shift1	shift1	shift1	shift2

Holiday Data

OBS	HOLISTA	HOLIFIN
1	17APR92:08:00:00	20APR92:07:59:59

Output 2 Calendar and Holiday Data

Software Development
Resource Availability Data

OBS	OBSTYPE	DATE	A	B	C
1	restype	.	1	1	1
2	reslevel	01APR92	1	1	1

Output 3 Resource Availability Data

Initial Schedule with Replenishable Resource

First, the project is scheduled using the default scheduling rule, and the resulting resource constrained schedule is saved as a baseline schedule.

```
proc cpm data=software resin=softres        /* input data sets      */
         holidata=holiday caledata=softcal workdata=shifts
         interval=dthour                    /* unit of duration     */
         date='01apr92:08:00'dt             /* start date           */
         out=softout resout=resusg;         /* output data sets     */
     activity act;
     successor s1 s2;
     duration hours;
     id id;
     holiday holista/holifin=holifin;       /* holiday information  */
     resource a b c / obstype=obstype
                      period=date
                      delayanalysis;        /* delay diagnostics    */
     /* save S_START as baseline schedule and compare with E_START  */
     baseline / set=resource compare=early;
run;
```

The resource constrained schedule and the early start schedule are printed in Output 4 along with the delay information (variables R_DELAY and DELAY_R) and the variable S_VAR that indicates the difference between E_START and S_START. Note that the BASELINE variables (B_START and B_FINISH, not shown in the output) are set to S_START and S_FINISH, respectively. Note also that only the date part of the start and finish times has been printed. The variables R_DELAY and DELAY_R indicate the amount of delay (in hours) due to resource constraints and the name of the resource causing the delay, respectively.

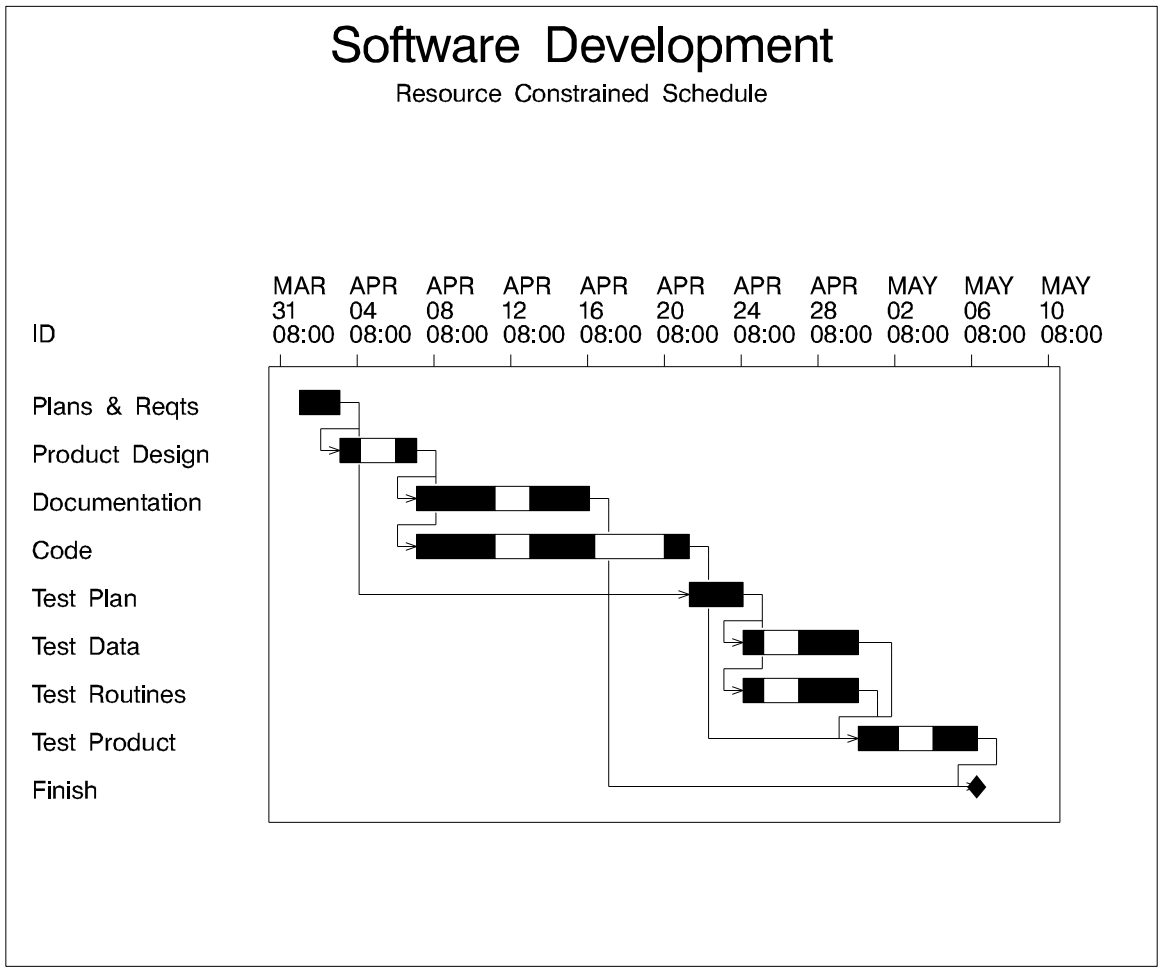
NOTE: R_DELAY represents the amount of delay caused by insufficient resources and does not include any delay caused by the delay in an earlier activity. Thus, for example, the activity `Test Data` has `R_DELAY = 0` even though `S_VAR = -100`. Note also that the project has been delayed by 56 hours (S_VAR = −56, for the activity `Finish`).

Software Development
Project Schedule

ID	S_START	S_FINISH	R_DELAY	DELAY_R	E_START	E_FINISH	S_VAR
Plans & Reqts	01APR92	03APR92	0		01APR92	03APR92	0
Product Design	03APR92	07APR92	0		03APR92	07APR92	0
Test Plan	21APR92	24APR92	100	A	03APR92	07APR92	-100
Documentation	07APR92	16APR92	0		07APR92	16APR92	0
Code	07APR92	21APR92	0		07APR92	21APR92	0
Test Data	24APR92	30APR92	0		07APR92	13APR92	-100
Test Routines	24APR92	30APR92	0		07APR92	13APR92	-100
Test Product	30APR92	06MAY92	0		21APR92	28APR92	-56
Finish	06MAY92	06MAY92	0		28APR92	28APR92	-56

Output 4 Project Schedule

Logic Gantt Chart and Time Scaled Network Diagram



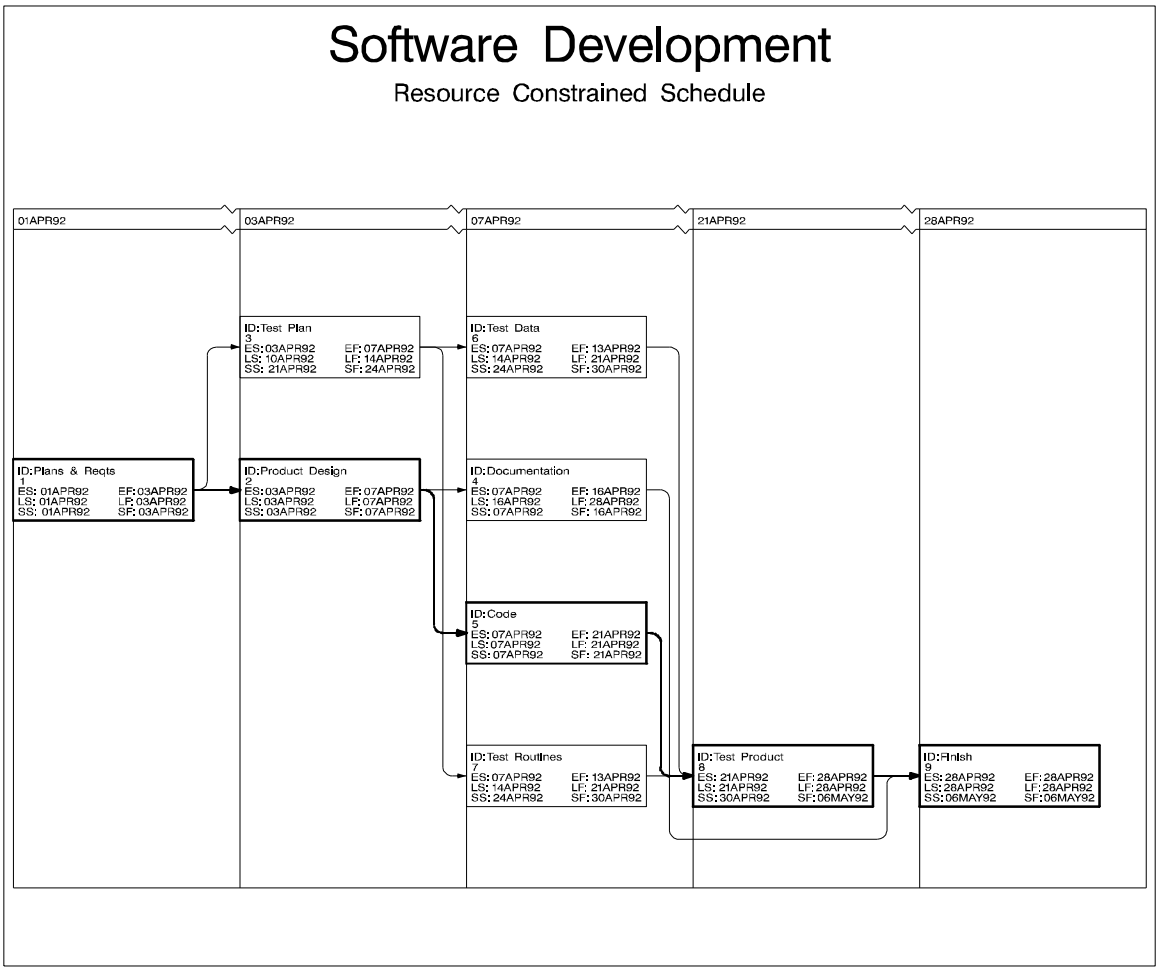
Output 5 Gantt Chart of Schedule

The following program uses the schedule data set, SOFTOUT, output by PROC CPM to produce the logic Gantt chart (Output 5) after the data are sorted by S_START and saved in the data set SCHED. The precedence information is conveyed to PROC GANTT via the ACTIVITY= and SUCCESSOR= options. The DURATION= option causes the zero-duration activities to be printed as milestones.

```
proc sort data=softout out=sched;
   by s_start;
   run;

title f=swiss 'Software Developent';
title2 f=swiss 'Resource Constrained Schedule';
pattern1 v=s c=black r=6; pattern2 v=e c=black; pattern3 v=s c=black;
proc gantt workdata=shifts holidata=holiday caledata=softcal
     data=sched(drop=e_: l_: b_:) /* omit early, late and baseline schedules */
     graphics;
   chart / activity=act successor=(s1 s2) /* logic information        */
```

```
          duration=hours font=swiss
          nolegend nojobnum markwknd compress
          holiday=(holista) holifin=(holifin);
   id id;
   run;
```



Output 6 Time Scaled Network Diagram

An alternate view of the project can be obtained using PROC NETDRAW. The following program invokes PROC NETDRAW with the TIMESCALE option to produce the network diagram shown in Output 6. Note that critical activities are indicated by thicker lines (using the LWCRIT= option in the ACTNET statement).

```
title f=swiss 'Software Development';
title2 f=swiss 'Resource Constrained Schedule';

pattern1 v=e c=black r=2;
proc netdraw data=sched graphics;
   actnet / act=act succ=(s1 s2)
            pcompress   /* compress: proportional transformation */
            separatearcs ybetween=8
            lwidth=1 lwcrit=8
```

```
            id=(id) font=swiss
            showbreak autoref timescale  /* timescale options    */
            cref=black;
   format e_: l_: s_: datetime7.;
   run;
```

Activity Splitting

Recall from the earlier discussion that the completion of the project has been delayed by 56 hours. A basic assumption in the default scheduling algorithm used by PROC CPM is that activities, once started, cannot be interrupted. Often, you may be able to get a shorter project duration by allowing activity preemption. The following program adds a variable MINDUR to the schedule data set SOFTOUT and uses the resulting data set as input to PROC CPM. The variable MINDUR specifies the minimum segment length into which activities can be split; thus, in this example, activities can be performed in four-hour segments, if necessary. PROC CPM is invoked with the MINSEGMTDUR= option, indicating that actvities can be split, if necessary.

```
data softwr1;
   set softout;
   if hours > 0 then mindur=4;
   run;

proc cpm data=softwr1 resin=softres
     holidata=holiday caledata=softcal workdata=shifts
     interval=dthour date='01apr92:08:00'dt
     out=softout1 resout=resusg;
     activity act;
     successor s1 s2;
     duration hours;
     id id;
     holiday holista/holifin=holifin;
     resource a b c / obstype=obstype
                      period=date
                      minsegmtdur=mindur  /* activity splitting option */
                      delayanalysis;
     baseline / compare=resource; /* compare new resource schedule
                                     with first resource schedule */
run;
```

Output 7 shows some of the relevant information from the output data set SOFTOUT1. Since activity splitting is allowed, the new variable SEGMT_NO indicates the index of split segments. The variables S_VAR and F_VAR indicate the difference between the current resource constrained schedule and *the saved baseline schedule*. Note that the project now finishes on 30APR92, reducing the project duration by 36 hours when compared with the earlier resource constrained schedule (which is the baseline schedule in this example). For activities that have been split, the F_VAR variable is different from the S_VAR variable, as is to be expected. You can also compare resource utilization for the two schedules using the resource usage output data sets (Hoopes 1992).

Software Development
Schedule with Activity Splitting

ID	B_START	B_FINISH	SEGMT_NO	S_START	S_FINISH	S_VAR	F_VAR
Plans & Reqts	01APR92	03APR92	.	01APR92:08:00:00	03APR92:09:59:59	0	0
Product Design	03APR92	07APR92	.	03APR92:10:00:00	07APR92:09:59:59	0	0
Test Plan	21APR92	24APR92	.	10APR92:15:00:00	16APR92:14:59:59	-52	-36
Test Plan	21APR92	24APR92	1	10APR92:15:00:00	13APR92:09:59:59	-52	-36
Test Plan	21APR92	24APR92	2	14APR92:10:00:00	15APR92:09:59:59	-52	-36
Test Plan	21APR92	24APR92	3	16APR92:10:00:00	16APR92:14:59:59	-52	-36
Documentation	07APR92	16APR92	.	07APR92:10:00:00	27APR92:14:59:59	0	56
Documentation	07APR92	16APR92	1	07APR92:10:00:00	10APR92:14:59:59	0	56
Documentation	07APR92	16APR92	2	13APR92:10:00:00	14APR92:09:59:59	0	56
Documentation	07APR92	16APR92	3	15APR92:10:00:00	16APR92:09:59:59	0	56
Documentation	07APR92	16APR92	4	24APR92:10:00:00	27APR92:14:59:59	0	56
Code	07APR92	21APR92	.	07APR92:10:00:00	24APR92:09:59:59	0	20
Code	07APR92	21APR92	1	07APR92:10:00:00	10APR92:14:59:59	0	20
Code	07APR92	21APR92	2	13APR92:10:00:00	14APR92:09:59:59	0	20
Code	07APR92	21APR92	3	15APR92:10:00:00	16APR92:09:59:59	0	20
Code	07APR92	21APR92	4	16APR92:15:00:00	24APR92:09:59:59	0	20
Test Data	24APR92	30APR92	.	16APR92:15:00:00	24APR92:09:59:59	-36	-36
Test Routines	24APR92	30APR92	.	16APR92:15:00:00	24APR92:09:59:59	-36	-36
Test Product	30APR92	06MAY92	.	24APR92:10:00:00	30APR92:14:59:59	-36	-36
Finish	06MAY92	06MAY92	.	30APR92:15:00:00	30APR92:15:00:00	-36	-36

Output 7 Schedule with Activity Splitting

Alternate Resources

Suppose now that it is possible for one programmer to do the work of another programmer, if necessary. However, the substitute programmer may not be as efficient as the original programmer. For example, programmer B may need to work full-time on an activity that could be accomplished by programmer A working half-time on the same activity. The data set SOFTRES2, printed in Output 8 , specifies the rate of substitution for each programmer, in addition to the resource availability data.

Software Development
Alternate Resource Information

OBS	OBSTYPE	DATE	RES	A	B	C
1	restype	.		1	1	1
2	reslevel	01APR92		1	1	1
3	altrate	.	a	.	2	2
4	altrate	.	b	2	.	2
5	altrate	.	c	2	2	.

Output 8 Alternate Resources

The following program uses the output data set SOFTOUT produced by the first invocation of PROC CPM to obtain an alternate schedule that allows substitution of resources as indicated by the data set SOFTRES2. To allow substitution of resources, PROC CPM is invoked with the RESID= option which indicates the variable in the data set SOFTRES2 that contains the names of the resources for which alternate resource information is provided in a given observation. The BASELINE statement is used to compare the new resource constrained schedule with the original schedule saved as a baseline in the data set SOFTOUT. The resulting schedule and a comparison with the earlier schedule is printed in Output 9.

```
proc cpm data=softout resin=softres2
     holidata=holiday caledata=softcal workdata=shifts
     interval=dthour date='01apr92:08:00'dt
     out=softout2 resout=resusg2;
     activity act;
     successor s1 s2;
     duration hours;
     id id;
     holiday holista/holifin=holifin;
     resource a b c / obstype=obstype
                      period=date
                      resid=res   /* triggers use of alternate resources */
                      delayanalysis;
     baseline / compare=resource; /* compare new resource schedule
                                     with first resource schedule */
run;
```

Software Development
Schedule with Alternate Resources

ID	B_START	B_FINISH	S_START	S_FINISH	S_VAR	F_VAR	A	B	C	UA	UB	UC
Plans & Reqts	01APR92	03APR92	01APR92	03APR92	0	0	0.5	0.5	0.5	0.5	0.5	0.5
Product Design	03APR92	07APR92	03APR92	07APR92	0	0	1.0	0.5	.	1.0	0.5	.
Test Plan	21APR92	24APR92	07APR92	09APR92	-80	-80	0.5	.	0.5	.	1.0	0.5
Documentation	07APR92	16APR92	15APR92	27APR92	56	56	.	.	1.0	.	.	1.0
Code	07APR92	21APR92	07APR92	21APR92	0	0	1.0	.	.	1.0	.	.
Test Data	24APR92	30APR92	09APR92	15APR92	-80	-80	.	0.5	0.5	.	0.5	0.5
Test Routines	24APR92	30APR92	09APR92	15APR92	-80	-80	.	0.5	0.5	.	0.5	0.5
Test Product	30APR92	06MAY92	21APR92	28APR92	-56	-56	0.5	1.0	.	0.5	1.0	.
Finish	06MAY92	06MAY92	28APR92	28APR92	-56	-56	.	.	.	.	.	.

Output 9 Schedule with Alternate Resources

Note that the project completion time has been reduced by allowing alternate resources. In fact, the project finishes on the same day as the unconstrained early start schedule (28APR92). The output data set contains new variables, UA, UB, and UC, which indicate the *actual* usage of the resources. For example, the activity `Test Plan` requires .5 of the resources A and C but can be performed using 1.0 unit of B and 0.5 units of C instead.

Manpower as a Consumable Resource

In projects that use manpower as a resource, in addition to the holidays and work shifts of the people involved, another factor that you may wish to control is the *changing* availability of the resources. For example, programmer A may be working on another project at the same time and may be able to devote only 50 percent of his time to the current project during certain weeks. There are several ways to model this scenario. One way is to use man-days as a consumable resource. We illustrate the concept with a simple example.

Suppose, for example, that an activity, `1`, requires programmer A throughout its duration. However, due to other commitments he is available only for 60 percent of the time during the week of April 13, 1992. In addition to the *replenishable* resource A, we can define a *consumable* resource, ADAYS, which accounts for the number of man-days expended by programmer A. Further, allow activity splitting, with minimum segment duration set to one day. In the resource data set, the availability of the consumable resource is adjusted according to the availability of programmer A. Thus, only three man-days (60 percent of a five-day work week) are available between the 13th and

the 20th, as indicated by the increase in ADAYS from 5 to 8 in observation 3, below. The activity data and resource data are shown in Output 10.

Manpower Scheduling
Activity Data

OBS	ACT	SUCC	DUR	A	ADAYS	MINDUR
1	1	.	10	1	1	1

Resource Data

OBS	OTYPE	PER	A	ADAYS
1	restype	.	1	2
2	reslevel	06APR92	1	5
3	reslevel	13APR92	.	8
4	reslevel	20APR92	.	10

Output 10 Activity and Resource Data

If activity splitting is not allowed, the activity will be scheduled to start on 13APR92 (the invocation of PROC CPM is not shown) and will finish on 24APR92; the schedule is in Output 11.

Manpower Scheduling
Activity Schedule

OBS	ACT	SUCC	DUR	A	ADAYS	S_START	S_FINISH	E_START	E_FINISH	L_START	L_FINISH
1	1	.	10	1	1	13APR92	24APR92	06APR92	17APR92	06APR92	17APR92

Output 11 Schedule with Replenishable and Consumable Resources

The following program invokes PROC CPM, with the MINSEGMTDUR= option, to produce the schedule shown in Output 12. Note that interval=weekday. The activity now finishes on 21APR92, and the availability information has been incorporated along with activity splitting to produce an earlier completion time.

```
proc cpm data=soft resin=res interval=weekday
     date='1apr92'd out=out2 resout=rout2;
     act act; dur dur; succ succ;
     res a adays/ per=per obstype=otype
                  minsegmtdur=mindur;  /* allow activity splitting */
run;
```

Manpower Scheduling
Activity Schedule: Splitting Allowed

OBS	ACT	SUCC	SEGMT_NO	DUR	A	ADAYS	S_START	S_FINISH	E_START	E_FINISH	L_START	L_FINISH
1	1	.	.	10	1	1	06APR92	21APR92	06APR92	17APR92	06APR92	17APR92
2	1	.	1	8	1	1	06APR92	15APR92	06APR92	17APR92	06APR92	17APR92
3	1	.	2	2	1	1	20APR92	21APR92	06APR92	17APR92	06APR92	17APR92

Output 12 Schedule Allowing Splitting

Annotated Gantt Chart

The most common application of PROC GANTT is to plot the project schedule produced by PROC CPM. However, it is not always necessary to invoke PROC CPM before drawing a Gantt chart, nor is it necessary to draw only project schedules with PROC GANTT. In this example, PROC GANTT is used to graphically represent alternate flight plans for flying from Raleigh-Durham to Honolulu. There are three alternate plans, two via Dallas and one via Chicago. The data set, FLTSCHED (printed in Output 13), contains the relevant information regarding the flight times (all converted to Eastern Standard Time) in a format that can be used by PROC GANTT.

Possible Flight Plans

OBS	FLTPLAN	DESCR	SEGMT_NO	S_START	S_FINISH	ROUTE
1	1	Flight Plan 1	1	7:00	9:08	RDU - ORD
2	1		2	10:15	17:54	ORD - HNL
3	2	Flight Plan 2	1	6:33	9:28	RDU - DFW
4	2		2	10:37	17:23	DFW - HNL
5	3	Flight Plan 3	1	8:02	11:07	RDU - DFW
6	3		2	12:20	18:56	DFW - HNL

Output 13 Data Set FLTSCHED

Each observation indicates the start and finish times (in variables S_START and S_FINISH) for a segment of journey (indexed by the variable SEGMT_NO) for a given flight; flights are indexed from 1 to 3. PROC GANTT uses the S_START and S_FINISH variables and the segment number information to plot a bar with breaks corresponding to each break in the journey (as for resource constrained schedules). The same input data set is used to create an ANNOTATE data set that is used by PROC GANTT to label the bars denoting the flight segments. The program creating the ANNOTATE data set and the invocation of PROC GANTT is shown below. The results appear in Output 14. The graph obtained enables you to compare flight times and layover times easily.

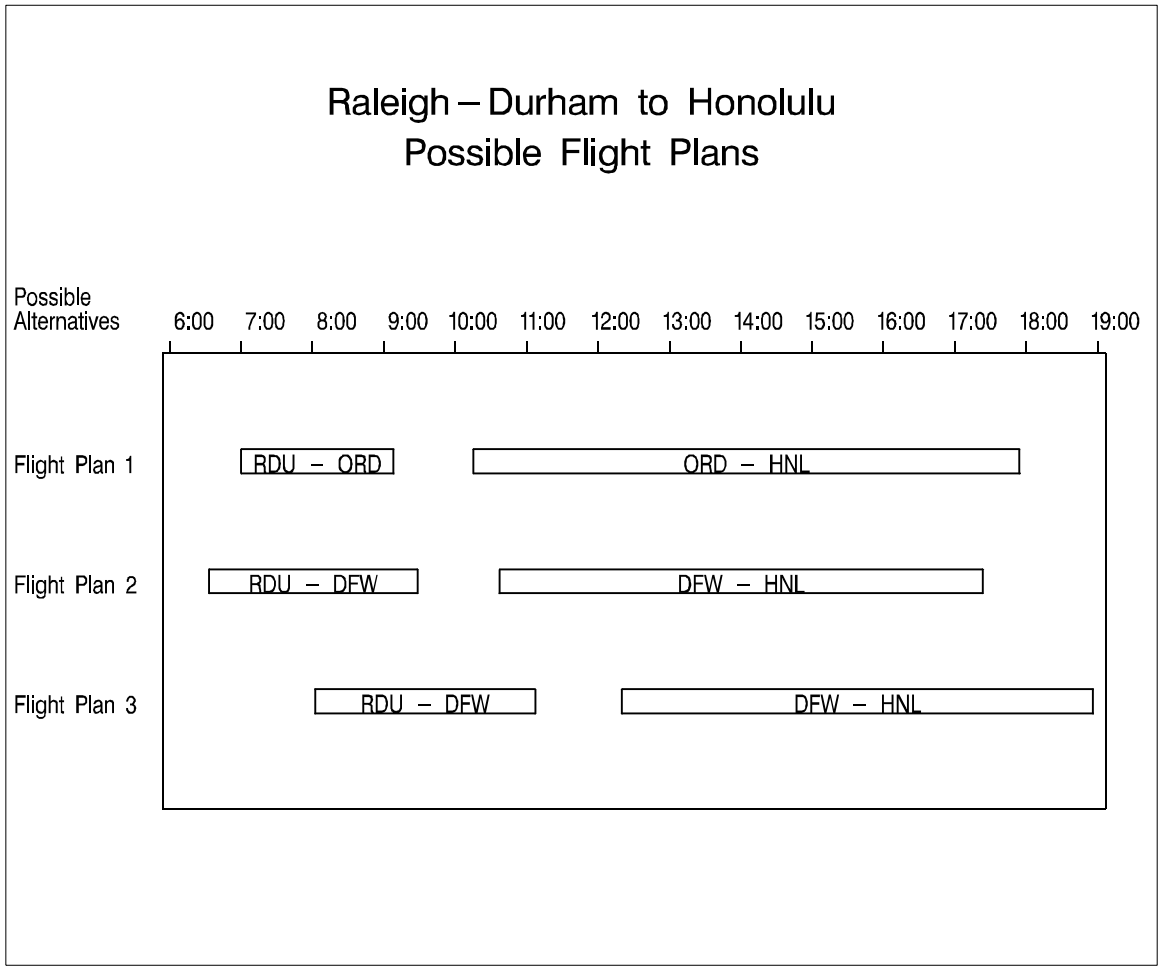
```
title1 h=2 ' ';
title2 h=1.5 'Raleigh-Durham to Honolulu';
title3 h=1.5 'Possible Flight Plans';

goptions hpos=120 vpos=40;
goptions ftext = swiss border;

/* Use the data set FLTSCHED to create an ANNOTATE data set */
/* containing labels for each flight segment                */
data anno;
   set fltsched;
   /* Set up required variable lengths, etc. */
   length function color style   $8;
   length xsys ysys hsys         $1;
   length when position          $1;
   xsys     = '2';  ysys     = '2';  hsys     = '4';
   when     = 'a';
   function = 'label   ';
   x = s_start + 0.5 * (s_finish - s_start); /* center text in bar */
   y = fltplan - .03;
   text = route;  size = 1;
   position = '5';
   run;
```

```
pattern1 v=e c=black r=8;

proc gantt graphics data=fltsched anno=anno;
   chart / font=swiss skip=4
           mininterval=hour
           compress
           nolegend nojobnum
           lwidth=3
           increment=1 scale=9
           mindate='06:00't;
id descr;
run;
```



Output 14 Flight Plans

Network Diagram of Training Schedule

This example illustrates a nonstandard use of PROC NETDRAW to draw a schedule of training courses. Suppose that the data set CRSSCHED, printed in Output 15 , contains the dates and locations for some SAS training courses. (See Kulkarni 1991 for a description of course scheduling using PROC CPM and alternate resources.)

Software Training Center
Schedule and Location for Courses

OBS	CRSNBR	SUCC	CRSDESC	LOCATION	PERIOD	START	FINISH
1	1	.	Color Graphics	Cary, N.C.	1	11FEB91	13FEB91
2	2	.	Color Graphics	Cary, N.C.	2	18FEB91	20FEB91
3	3	.	Color Graphics	Chicago, Il.	2	18FEB91	20FEB91
4	4	.	Color Graphics	New York City	2	18FEB91	20FEB91
5	5	.	Color Graphics	Washington, D.C.	2	18FEB91	20FEB91
6	6	.	Menu Driven Applications	Chicago, Il.	1	11FEB91	13FEB91
7	7	.	Menu Driven Applications	New York City	1	11FEB91	13FEB91
8	8	.	Project Management	Cary, N.C.	3	25FEB91	27FEB91
9	9	.	Project Management	Cary, N.C.	4	07MAR91	09MAR91
10	10	.	Project Management	Washington, D.C.	3	25FEB91	27FEB91

Output 15 Data Set CRSSCHED

The following program defines a format that is associated with each course period and then invokes PROC NETDRAW with the ZONE= and ALIGN= options. The resulting diagram is printed in Output 16. Since there are no successors in the SUCC= variable, the network does not have any arcs. Each node is positioned in the column corresponding to the appropriate time and the row corresponding to the appropriate location.

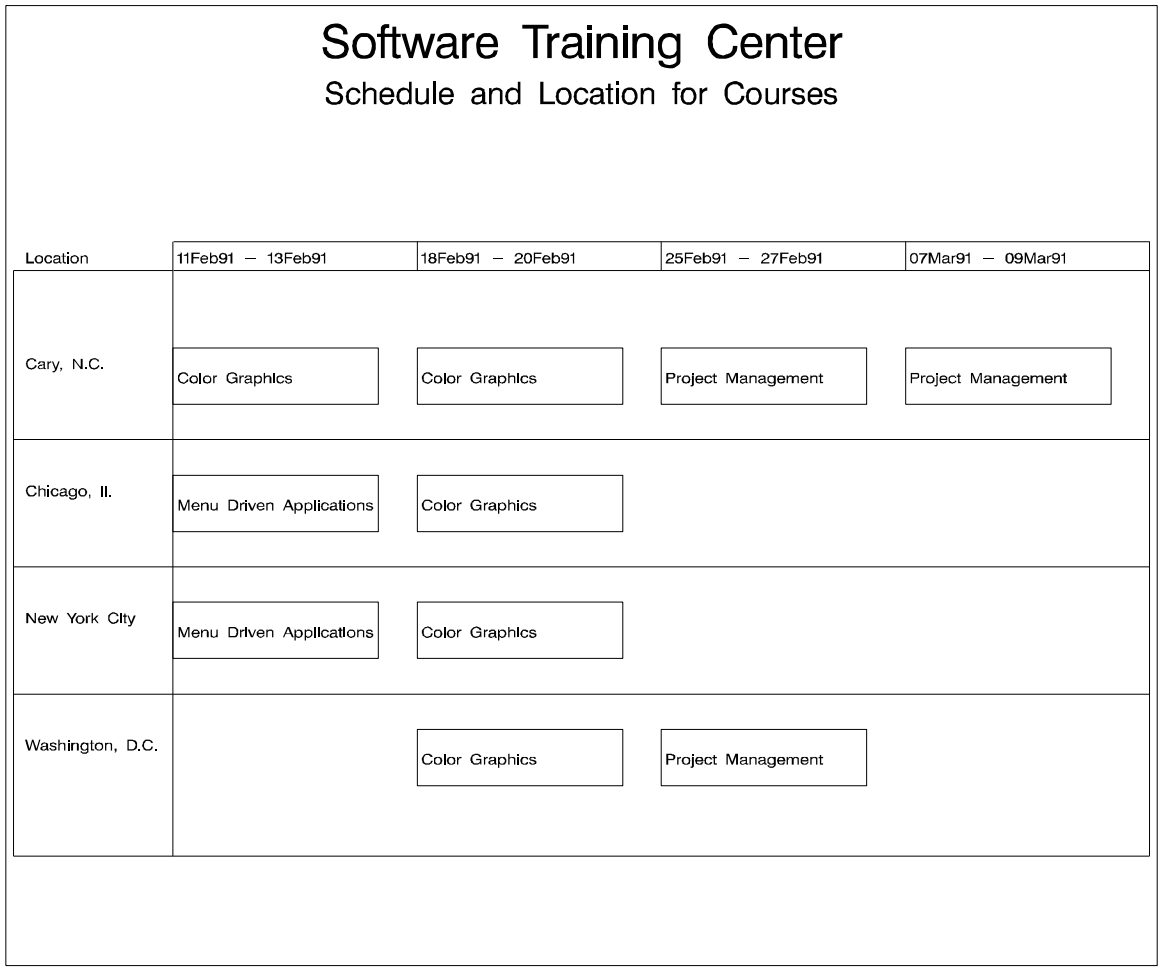
```
proc format;
   value crsdates 1 = '11Feb91 - 13Feb91'
                  2 = '18Feb91 - 20Feb91'
                  3 = '25Feb91 - 27Feb91'
                  4 = '07Mar91 - 09Mar91';
run;

goptions ftext = swiss border;
pattern1 v=e c=black;
goptions hpos=80 vpos=32;
proc netdraw graphics data=crssched;
   actnet / act=crsnbr  succ=succ
            nodefid nolabel id=(crsdesc) boxht=3 pcompress
            zone=location align=period useformat;
   format period crsdates.;
run;
```

Conclusion

In this paper, we have illustrated some of the capabilities of SAS/OR software for some standard as well as some nonstandard project management tasks. The software development example is considered in greater detail by Hoopes who includes discussion of how you can manage multiple software projects drawing from the same pool of programmers. The last two examples illustrate

that the software is not limited to project management tasks; the procedures can be used for a wide variety of applications.



Output 16 Training Schedule

References

1. Hoopes, B. (1992), *Project Management: Selected Examples Using the SAS System*, Operations Research Department, Applications Division, SAS Institute Inc.

2. Kulkarni, R. (1991), Scheduling with the CPM Procedure, *Proceedings of the Sixteenth Annual SAS Users Group International Conference.*

3. SAS Institute Inc. (1989), *SAS/OR User's Guide, Version 6, First Edition*, Cary, NC: SAS Institute Inc.

4. SAS Institute Inc. (1992), *Update to Version 6 SAS/OR User's Guide, Project Management Procedures*, Operations Research Department, Applications Division, SAS Institute Inc.