Chapter 1
Introduction to Project Management

Chapter Contents

OVERVIEW 17

DATA FLOW 17
- The CPM Procedure 18
- The GANTT Procedure 19
- The NETDRAW Procedure 20
- The PM Procedure 22
- Communication between Procedures 23

DECISION SUPPORT SYSTEMS 24

DECISION ANALYSIS 24
- The DTREE Procedure 25

EXAMPLES 26
- Example 1.1. Project Definition 26
- Example 1.2. Work Breakdown Structure 29
- Example 1.3. Project Scheduling and Reporting 30
- Example 1.4. Summary Report 32
- Example 1.5. Resource-Constrained Scheduling 34
- Example 1.6. Multiple Projects 38
- Example 1.7. Sequential Scheduling of Projects 47
- Example 1.8. Project Cost Control 49
- Example 1.9. Subcontracting Decisions 57

PROJECT MANAGEMENT SYSTEMS 60

THE PROJMAN APPLICATION 61

WEB-BASED SCHEDULING SYSTEMS 61

MICROSOFT PROJECT CONVERSION MACROS 62

REFERENCES 62

Chapter 1
Introduction to Project Management

Overview

This chapter briefly describes how you can use SAS/OR software for managing your projects. This chapter is not meant to define all the concepts of project management; several textbooks on project management explain the basic steps involved in defining, planning, and managing projects (for example, Moder, Phillips, and Davis 1983). Briefly, a *project* is defined as any task comprising a set of smaller tasks that need to be performed, either sequentially or in parallel. Projects can be small and last only a few minutes (for instance, running a set of small computer programs), or they can be mammoth and run for several years (for example, the construction of the Channel Tunnel).

SAS/OR software has four procedures that can be used for planning, controlling, and monitoring projects: the CPM and PM procedures for scheduling project activities subject to precedence, time, and resource constraints; the GANTT procedure for displaying the computed schedule; and the NETDRAW procedure for displaying the activity network. These procedures integrate with the SAS System so that you can easily develop a customized project management system. The Projman application, a user-friendly graphical user interface included as part of SAS/OR software, is one such system.

This chapter gives a brief introduction to the CPM, GANTT, NETDRAW, and PM procedures and shows how you can use the SAS System for project management.

In addition to these four procedures and the Projman application, which are the major tools for the *traditional* functions associated with project management, SAS/OR software also contains a procedure for decision analysis, the DTREE procedure. *Decision analysis* is a tool that attempts to provide an analytic basis for management decisions under uncertainty. It can be used effectively as an integral part of project management methods. A brief introduction to PROC DTREE is provided in the "Decision Analysis" section on page 24.

Data Flow

This section provides an overview of how project information is stored in the SAS System in data sets and how these data sets are used by the CPM, GANTT, NETDRAW, and PM procedures. Maintaining the project information in SAS data sets enables you to merge project information easily from several sources, summarize information, subset project data, and perform a wide variety of other operations using any of the many procedures in SAS software. Each of the SAS/OR procedures also defines a SAS macro variable that contains a character string indicating whether

or not the procedure terminated successfully. This information is useful when the procedure is one of the steps in a larger program.

The CPM Procedure

PROC CPM does the project scheduling and forms the core of the project management functionality in SAS/OR software. It uses activity precedence, time, and resource constraints, and holiday and calendar information to determine a feasible schedule for the project. The precedence constraints between the activities are described using a network representation, either in Activity-On-Arc (AOA) or Activity-On-Node (AON) notation, and input to PROC CPM in an Activity data set. The two different representations are described in Chapter 2, "The CPM Procedure." The Activity data set can also specify time constraints on the activities and resource requirement information. The Activity data set is required. Resource availability information can be specified using another data set, referred to here as the Resource data set. Holiday, workday, and other calendar information is contained in the Holiday, Workday, and Calendar data sets; each of these data sets is described in detail in Chapter 2, "The CPM Procedure." The schedule calculated by PROC CPM using all the input information and any special scheduling options is saved in an output data set, referred to as the Schedule data set. For projects that use resources, individual resource schedules for each activity can be saved in a Resource Schedule output data set. Resource usage information can also be saved in another output data set, referred to as the Usage data set. Figure 1.1 illustrates all the input and output data sets that are possible with PROC CPM. In the same figure, _ORCPM_ is the SAS macro variable defined by PROC CPM.

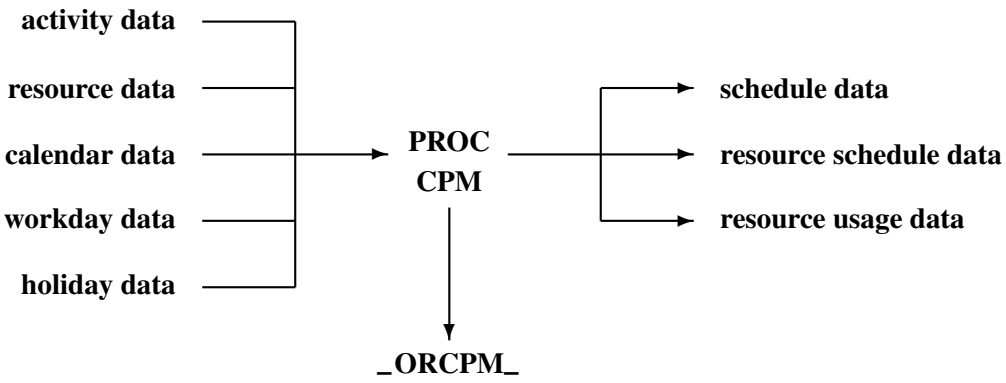


Figure 1.1. Input and Output Data Flow in PROC CPM

The three output data sets produced by PROC CPM contain all the information about the schedule and the resource usage; these data sets can be used as input to either PROC GANTT or PROC NETDRAW or to any of the several reporting, charting, or plotting procedures in the SAS System.

The Schedule data set can also contain additional project information such as project ID, department and phase information, accounting categories, and so on, in the form of ID variables passed to it from the Activity input data set with the ID statement. These variables can be used to produce customized reports by reordering, subsetting, summarizing, or condensing the information in the Schedule data set in various ways.

The GANTT Procedure

PROC GANTT draws, in line-printer, high-resolution graphics, or full-screen mode, a bar chart of the schedules computed by PROC CPM. Such a bar chart is referred to as a Gantt chart in project management terminology. In addition to the Schedule data set, PROC GANTT can also use the Calendar, Workday, and Holiday data sets (that were used by PROC CPM when scheduling the activities in the project) to mark holidays and weekends and other nonwork periods appropriately on the Gantt chart. You can indicate target dates, deadlines, and other important dates on a Gantt chart by adding CHART variables to the Schedule data set. Furthermore, the GANTT procedure can indicate milestones on the chart by using a DURATION variable in the Schedule data set.

Precedence information can be used by PROC GANTT in either Activity-on-Node or Activity-on-Arc format to produce a Logic bar chart that shows the precedence relationships between the activities. The precedence information, required for drawing the network logic, can be conveyed to PROC GANTT using the Activity data set or a Logic data set, as described in Chapter 4, "The GANTT Procedure."

The Gantt procedure also supports an automatic text annotation facility, using the Label data set, which is designed specifically for labeling Gantt charts independently of the SAS/GRAPH Annotate facility. The specifications in this data set enable you to print label strings with a minimum of effort and data entry while providing the capability for more complex chart labeling situations.

The Gantt procedure is Web-enabled. The HTML= option enables you to specify a variable in the Schedule data set that defines a URL for each activity. If you route the Gantt chart to an HTML file using the Output Delivery System, then you can click on a schedule bar and browse text or other descriptive information about the associated activity. You also use this information to create custom HTML files with drill-down graphs. PROC GANTT also produces an Imagemap data set that contains the outline coordinates for the schedule bars used in the Gantt chart that can be used to generate HTML MAP tags.

As with PROC CPM, PROC GANTT also defines a macro variable named _ORGANTT that has a character string indicating if the procedure terminated successfully. Figure 1.2 illustrates the flow of data in and out of PROC GANTT.

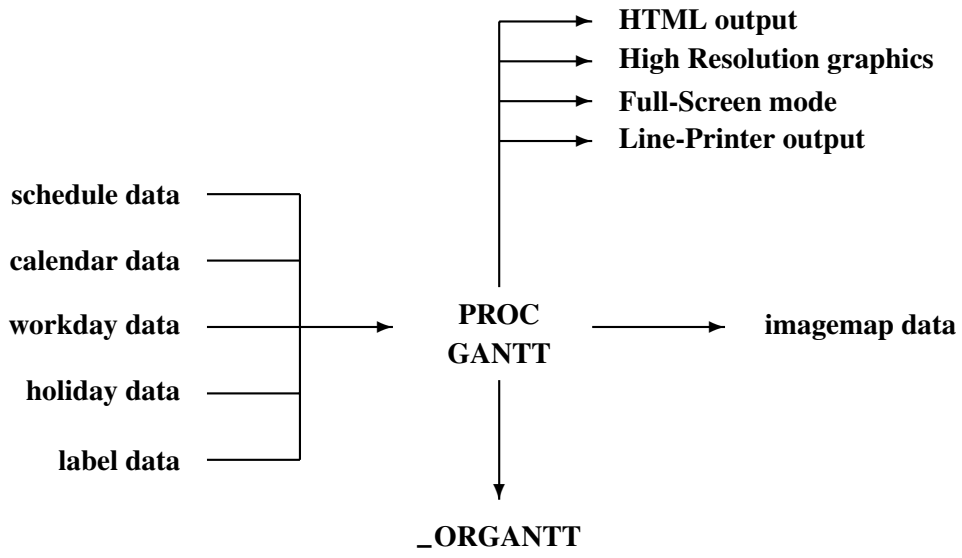


Figure 1.2. Input and Output Data Flow in PROC GANTT

The NETDRAW Procedure

PROC NETDRAW draws project networks. The procedure automatically places the nodes in the network and draws the arcs connecting them, using the (activity, successor) relationship as specified by the Network data set described in Chapter 5, "The NETDRAW Procedure." The Network data set, used as input to PROC NETDRAW, can be an Activity data set, a Schedule data set, or a Layout data set, as described in Chapter 5. If a Schedule data set, output by PROC CPM, is used as the Network data set, the network diagram also contains all the schedule times calculated by PROC CPM. The procedure can draw the diagram in line-printer mode as well as in high-resolution graphics mode. Further, you can invoke the procedure in full-screen mode, which enables you to scroll around the network to view different parts of it; in this mode, you can also modify the layout of the network by moving the nodes of the network.

By default, PROC NETDRAW uses the topological ordering of the activity network to determine the X coordinates of the nodes. In a time-based network diagram, the nodes can be ordered according to any SAS date, time, or datetime variable in the Network data set. In fact, PROC NETDRAW enables you to align the nodes according to any numeric variable in this data set, not just the start and finish times.

You can produce a zoned network diagram by identifying a ZONE variable in the input data set, which divides the network into horizontal bands or zones. This is useful in grouping the activities of the project according to some appropriate classification. The NETDRAW procedure also draws tree diagrams. This feature can be used to draw work breakdown structures or other organizational diagrams (see Example 1.2).

PROC NETDRAW produces an output data set (Layout data set), which contains the positions of the nodes and the arcs connecting them. This output data set can also be used as an input data set to PROC NETDRAW; this feature is useful when the same project network is drawn several times during the course of a project. You

may want to see the updated information drawn on the network every week; you can save computer resources by using the same node placement and arc routing, without having the procedure recompute it every time. PROC NETDRAW defines the macro variable _ORNETDR, which contains a character string indicating if the procedure terminated successfully.

The NETDRAW procedure is also Web-enabled (like PROC GANTT), and it supports the HTML= and IMAGEMAP= options.

Figure 1.3 illustrates the flow of data in and out of PROC NETDRAW.

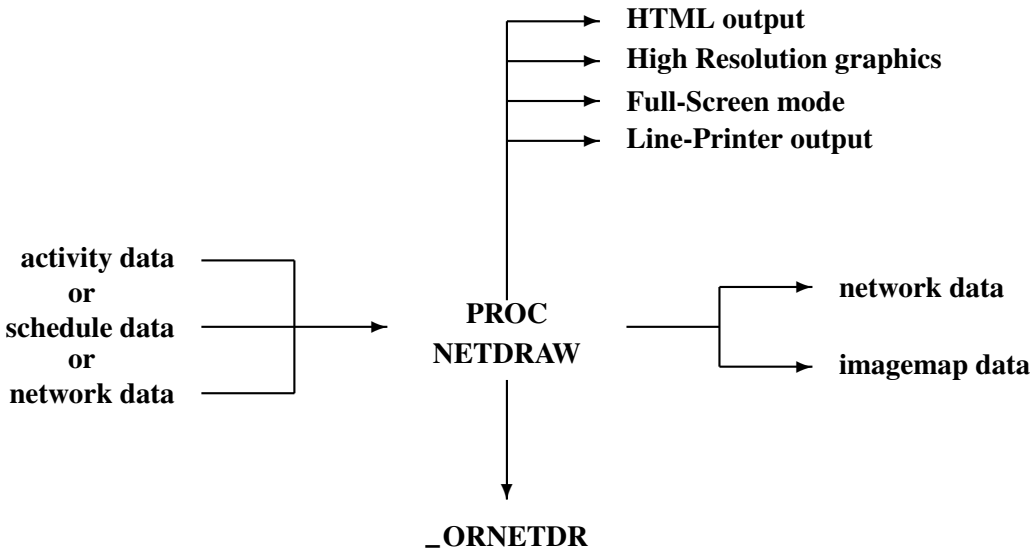


Figure 1.3. Input and Output Data Flow in PROC NETDRAW

The PM Procedure

PROC PM is an interactive procedure that can be used for planning, controlling, and monitoring a project. The syntax and the scheduling features of PROC PM are virtually the same as those of PROC CPM; there are a few differences, which are described in Chapter 6, "The PM Procedure." As far as the flow of data is concerned (see Figure 1.4), the PM procedure supports an additional data set that can be used to save and restore preferences that control the project view. The scheduling engine used by the PM procedure is the same as the one used by PROC CPM; the same macro variable, _ORCPM_, is used to indicate if the schedule was computed successfully.

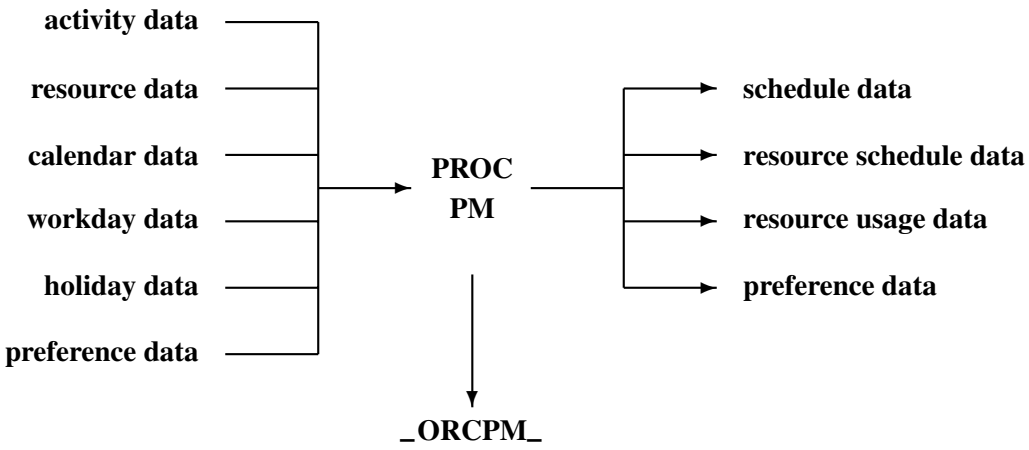


Figure 1.4. Input and Output Data Flow in PROC PM

Communication between Procedures

Figure 1.1, Figure 1.2, Figure 1.3, and Figure 1.4 illustrate the data flow going in and out of each of the four procedures: CPM, GANTT, NETDRAW, and PM, respectively. The data sets described in the previous sections store project information and can be used to communicate project data between the procedures in the SAS System. Figure 1.5 shows a typical sequence of steps in a project management system built around these procedures.

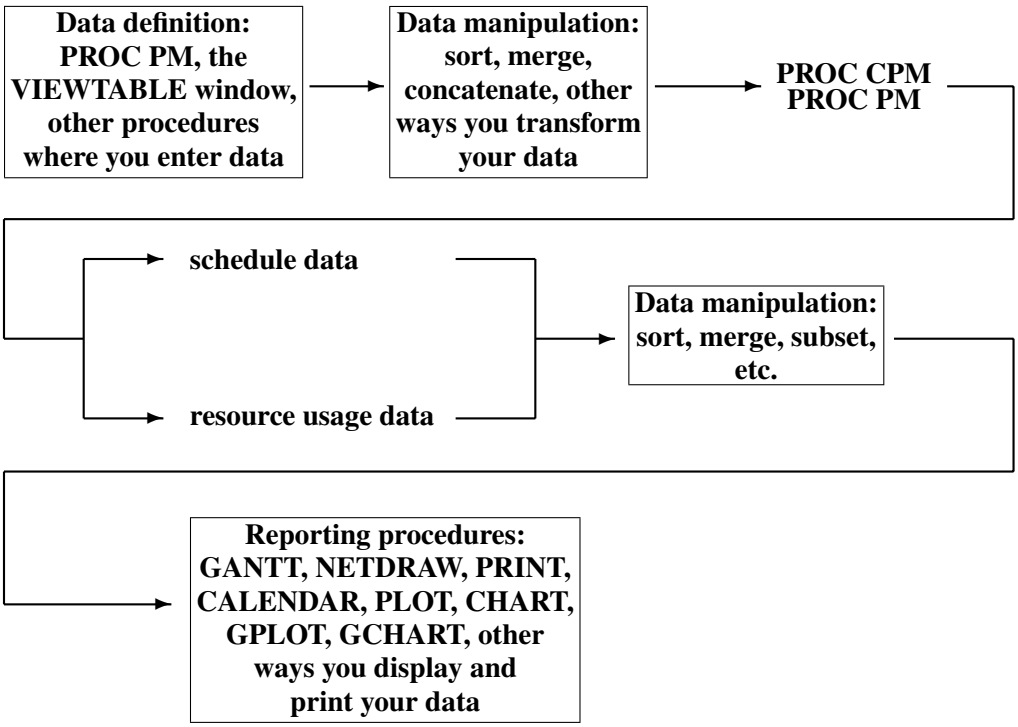


Figure 1.5. Using the SAS System for Project Management

Of course, this is only one possible scenario of the use of these procedures. In addition, you may want to use PROC NETDRAW to check the logic of the network diagram before scheduling the project using PROC CPM. Further, the data flow shown in Figure 1.5 may represent only the first iteration in a continuous scheme for monitoring the progress of a project. As the project progresses, you may update the data sets, including actual start and finish times for some of the activities, invoke PROC CPM again, produce updated Gantt charts and network diagrams, and thus continue monitoring the project.

For example, a project management system designed for scheduling and tracking a major outage at a power plant may include the steps illustrated in Figure 1.6. In the sequences of steps illustrated in both these figures, you can use PROC PM to update most of the activity information using the procedure's graphical user interface.

Thus, SAS/OR software provides four different procedures designed for performing many of the traditional project management tasks; these procedures can be combined in a variety of ways to build a customized comprehensive project management sys-

tem. The "Examples" section beginning on page 26 illustrates several applications of the procedures in typical project management situations.

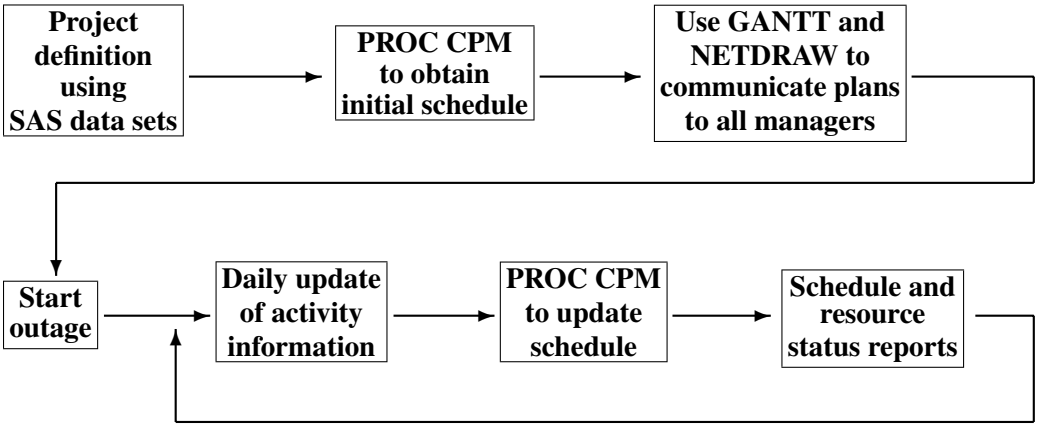


Figure 1.6. Scheduling a Power Plant Outage

Decision Support Systems

In addition to the CPM, GANTT, NETDRAW, and PM procedures, which are the major tools for the traditional functions associated with project management, SAS/OR software has several procedures that can be used to create a many-faceted Decision Support System. Traditional CPM/PERT techniques form only one part of effective project management and may be considered as a specialized application of Decision Support Systems (Williams and Boyd 1990). SAS/OR software contains several mathematical programming procedures that can be used to design effective systems for solving inventory control, transportation, network flow, transshipment, product-mix, cutting stock problems, and so on. These procedures are discussed in detail in the *SAS/OR User's Guide: Mathematical Programming*.

Decision analysis is another important tool that is receiving recognition as a component of project management. The next section briefly describes PROC DTREE and the role it can play in making important decisions in project management.

Decision Analysis

There are several stages in the course of a project when critical decisions are to be made regarding the future path that is to be followed. In fact, the most crucial decision might be to decide at the beginning whether to embark on the project or not. Other important decisions that could benefit from using decision analysis tools may be subcontract awarding, subproject termination in a research and development (R&D) environment, what-if analysis, and so on. Decision analysis techniques can be used effectively in such situations to help make decisions under uncertainty.

The DTREE Procedure

PROC DTREE interprets a decision problem represented in SAS data sets, finds the optimal decisions, and plots on a line printer or a graphics device the decision tree showing the optimal decisions. A decision tree contains two types of nodes: decision nodes and chance nodes. A decision node represents a stage in the problem where a decision is to be made that could lead you along different paths through the tree. A chance node represents a stage in the problem where some uncertain factors result in one of several possible outcomes, once again leading you to different branches of the tree, with associated probabilities.

The structure of a decision model is given in the STAGEIN= data set. This data set, described in detail in Chapter 3, "The DTREE Procedure," specifies the name, type, and attributes of all outcomes for each stage in your model. This is the only data set that is required to produce a diagrammatic representation of your decision problem. To evaluate and analyze your decision model, you need to specify the PROBIN= and PAYOFFS= data sets. The PROBIN= data set specifies the conditional probabilities for every event in your model. The PAYOFFS= data set specifies the value of each possible scenario (sequence of outcomes). The objective is to use the information summarized in these data sets to determine the optimal decision based on some measure of performance. One common objective is to maximize the expected value of the return. Figure 1.7 illustrates the data flow for PROC DTREE.

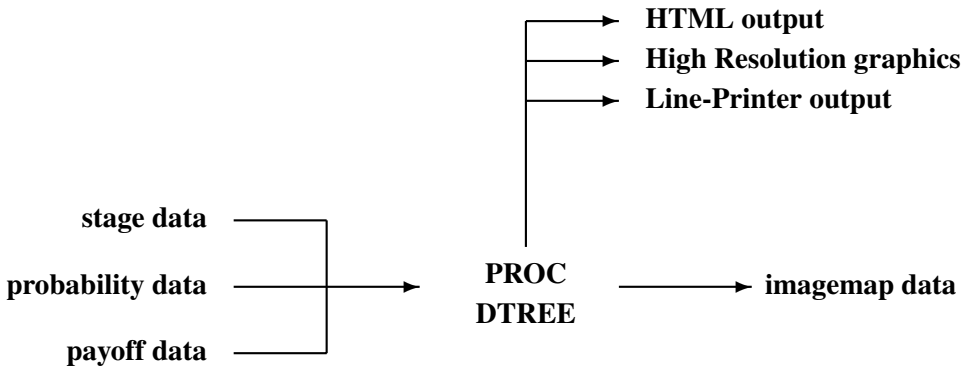


Figure 1.7. Input and Output Data Flow in PROC DTREE

You can use PROC DTREE to display, evaluate, and summarize your decision problem. The procedure can be used to plot the decision tree in line-printer or graphics mode. The optimal decisions are highlighted on the output. Further, a summary table can be displayed listing all paths through the decision tree along with the cumulative reward and the evaluating values of all alternatives for that path. The summary table indicates the optimal evaluating value for each path with an asterisk. The procedure can also perform sensitivity analysis and what-if analysis. A simple decision problem is described in Example 1.9.

Examples

In this section, a few simple examples illustrate some of the basic data flow concepts described in this chapter. More detailed examples of each procedure are provided in the corresponding chapters and can also be found in *SAS/OR Software: Project Management Examples*.

Example 1.1. Project Definition

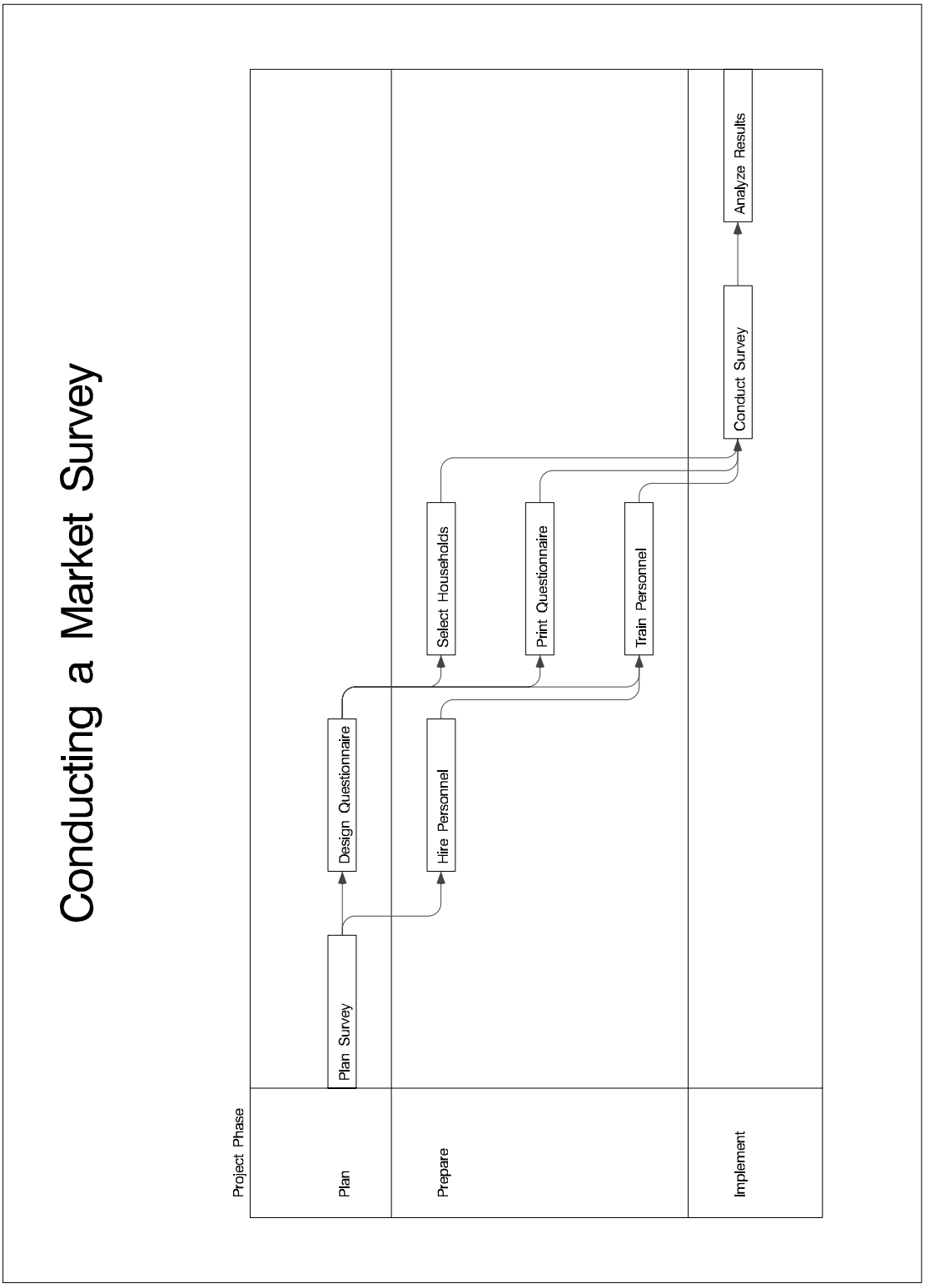
Suppose you want to prepare and conduct a market survey (Moder, Phillips, and Davis 1983) to determine the desirability of launching a new product. As a first step, you need to identify the steps involved. Make a list of the tasks that need to be performed and obtain a reasonable estimate of the length of time needed to perform each task. Further, you need to specify the order in which these tasks can be done. The following DATA step creates a SAS data set, survey, representing the project. This Activity data set contains a representation of the Survey project in Activity-On-Node format; a brief discussion of the two types of representations is given in Chapter 2, "The CPM Procedure." The data set contains a variable activity listing the basic activities (tasks) involved, a variable duration specifying the length of time in days needed to perform the tasks, and, for each task, the variables succ1–succ3, which indicate the immediate successors. An ID variable is also included to provide a more informative description of each task. Thus, the activity 'Plan Survey' takes four days. Once the planning is done, the tasks 'Hire Personnel' and 'Design Questionnaire' can begin. The Activity data set also contains a variable named phase associating each activity with a particular phase of the project.

```
data survey;
   format id $20. activity $8. succ1-succ3 $8. phase $9. ;
   input id & activity & duration succ1 & succ2 & succ3 & phase $ ;
   label phase = 'Project Phase'
         id    = 'Description';
   datalines;
Plan Survey           plan sur   4  hire per  design q  .         Plan
Hire Personnel        hire per   5  trn per   .         .         Prepare
Design Questionnaire  design q   3  trn per   select h  print q   Plan
Train Personnel       trn per    3  cond sur  .         .         Prepare
Select Households     select h   3  cond sur  .         .         Prepare
Print Questionnaire   print q    4  cond sur  .         .         Prepare
Conduct Survey        cond sur  10  analyze   .         .         Implement
Analyze Results       analyze    6  .         .         .         Implement
;
```

The data set survey can be input to PROC CPM, which calculates how long the project will take given the current estimates of the durations. As a first step, you may want to graph the project network using PROC NETDRAW. In the initial stages of defining the tasks in a project, it is useful to see how the tasks relate to each other and perhaps modify some of the relationships. The following program invokes PROC NETDRAW; the ZONE= option is used to create a zoned network diagram with the activities grouped according to the phase of the project to which they correspond. The network diagram is shown in Output 1.1.1.

```
title ' ';
title2 h=3 c=black f=swiss 'Conducting a Market Survey';


goptions hpos=100 vpos=65 border;

proc netdraw data=survey graphics;
   actnet/act=activity font=swiss
          succ=(succ1-succ3)
          separatearcs
          xbetween=3
          id=(id)
          nodefid
          nolabel
          zone=phase
          zonepat
          frame;
   run;
```

Output 1.1.1. Network Diagram of SURVEY Project

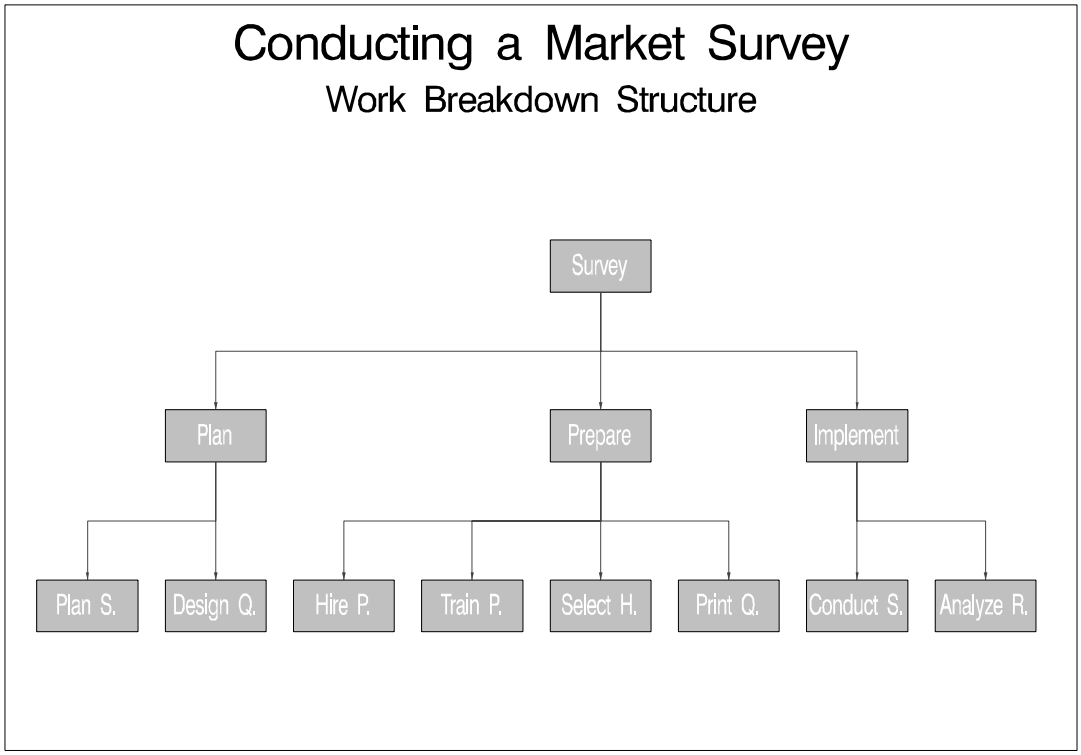


Example 1.2. Work Breakdown Structure

A tree diagram is a useful method of visualizing the work breakdown structure (WBS) of a project. For the survey project, the activities are divided into three phases. In this example, the NETDRAW procedure is used to represent the work breakdown structure of the project. The following program saves the data in a Network data set that is input to PROC NETDRAW. The TREE option is used to draw the WBS structure in the form of a tree (Output 1.2.1).

```
data wbs;
   format parent $10. child $10. ;
   input parent & child & style;
   datalines;
Survey      Plan          1
Survey      Prepare       1
Survey      Implement     1
Plan        Plan S.       2
Plan        Design Q.     2
Prepare     Hire P.       3
Prepare     Train P.      3
Prepare     Select H.     3
Prepare     Print Q.      3
Implement   Conduct S.    4
Implement   Analyze R.    4
Plan S.     .             2
Design Q.   .             2
Hire P.     .             3
Train P.    .             3
Select H.   .             3
Print Q.    .             3
Conduct S.  .             4
Analyze R.  .             4
;

goptions vsize=5.75 in hsize=4.0 in border;

title  a=90 h=1  f=swiss 'Conducting a Market Survey';
title2 a=90 h=.8 f=swiss 'Work Breakdown Structure';


proc netdraw data=wbs graphics;
   actnet/act=parent succ=child tree
          rotate rotatetext coutline=black
          ctext=white font=swiss rectilinear
          htext=2 compress
          xbetween=15 ybetween=3 pattern=style
          centerid;
   run;
```

Output 1.2.1. Work Breakdown Structure of SURVEY Project



Example 1.3. Project Scheduling and Reporting

Having defined the project and ensured that all the relationships have been modeled correctly, you can schedule the activities in the project by invoking PROC CPM. Suppose the activities can occur only on weekdays, and there is a holiday on July 4, 2003. Holiday information is passed to PROC CPM using the Holiday data set holidata. The following statements schedule the project to start on July 1, 2003. The early and late start schedules and additional project information are saved in the output data set survschd. The output data set produced by PROC CPM can then be used to generate a variety of reports. In this example, the data set is first sorted by the variable E_START and then displayed using the PRINT procedure (see Output 1.3.1).

```
data holidata;
   format hol date7.;
   hol = '4jul03'd;
   run;
```

```
proc cpm data=survey date='1jul03'd out=survschd
         interval=weekday holidata=holidata;
   activity   activity;
   successor  succ1-succ3;
   duration   duration;
   id         id phase;
   holiday    hol;
   run;

proc sort;
   by e_start;
   run;

proc print;
   run;
```

Output 1.3.1. Project Schedule: Listing

Conducting a Market Survey
Early and Late Start Schedule

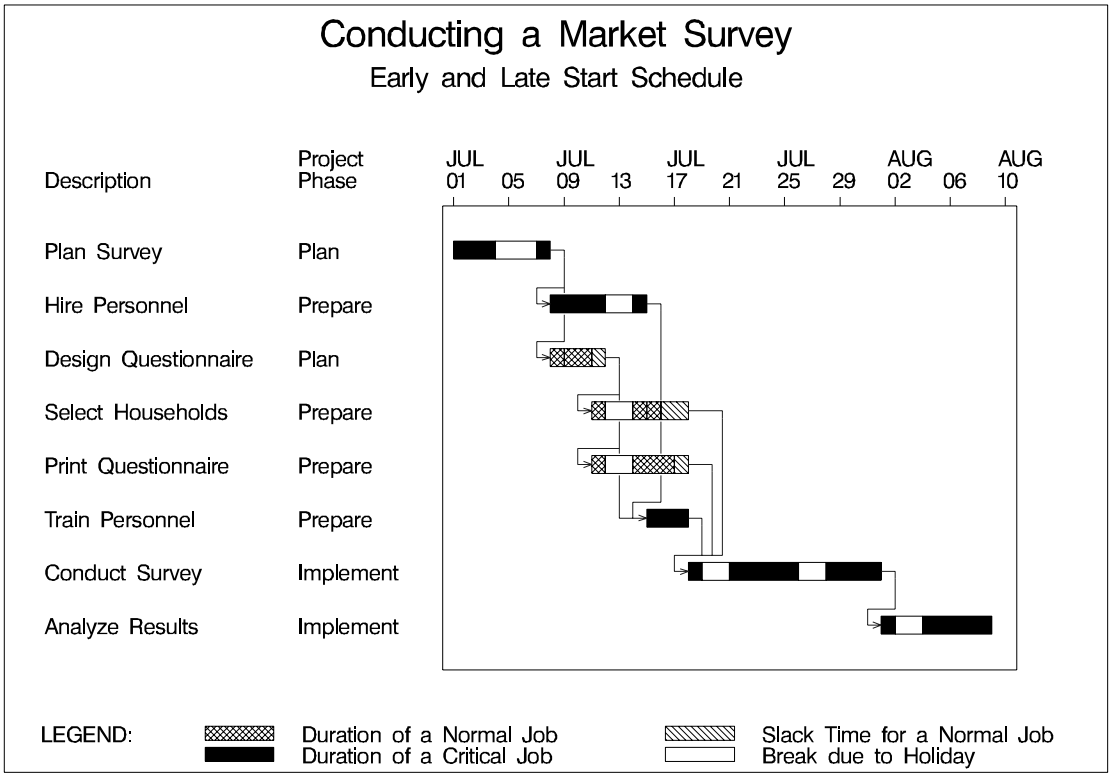
Obs	activity	succ1	succ2	succ3	duration	id
1	plan sur	hire per	design q		4	Plan Survey
2	hire per	trn per			5	Hire Personnel
3	design q	trn per	select h	print q	3	Design Questionnaire
4	select h	cond sur			3	Select Households
5	print q	cond sur			4	Print Questionnaire
6	trn per	cond sur			3	Train Personnel
7	cond sur	analyze			10	Conduct Survey
8	analyze				6	Analyze Results

Obs	phase	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT
1	Plan	01JUL03	07JUL03	01JUL03	07JUL03	0	0
2	Prepare	08JUL03	14JUL03	08JUL03	14JUL03	0	0
3	Plan	08JUL03	10JUL03	09JUL03	11JUL03	1	0
4	Prepare	11JUL03	15JUL03	15JUL03	17JUL03	2	2
5	Prepare	11JUL03	16JUL03	14JUL03	17JUL03	1	1
6	Prepare	15JUL03	17JUL03	15JUL03	17JUL03	0	0
7	Implement	18JUL03	31JUL03	18JUL03	31JUL03	0	0
8	Implement	01AUG03	08AUG03	01AUG03	08AUG03	0	0

The schedule produced by PROC CPM is then graphed by invoking PROC GANTT, as shown in the following code. The CALENDAR procedure or NETDRAW procedure can also be used to display the schedule. The Gantt chart produced is shown in Output 1.3.2. Note that the precedence relationships are displayed on the Gantt chart.

```
goptions hpos=80 vpos=43;

title c=black f=swiss 'Conducting a Market Survey';
title2 c=black f=swiss h=1.5 'Early and Late Start Schedule';
proc gantt graphics data=survschd holidata=holidata;
   chart / holiday=(hol) interval=weekday
           font=swiss skip=2 height=1.2 nojobnum
           compress noextrange
           activity=activity succ=(succ1-succ3)
           cprec=blue caxis=black ;
   id   id phase;
   run;
```

Output 1.3.2. Gantt Chart of SURVEY Project



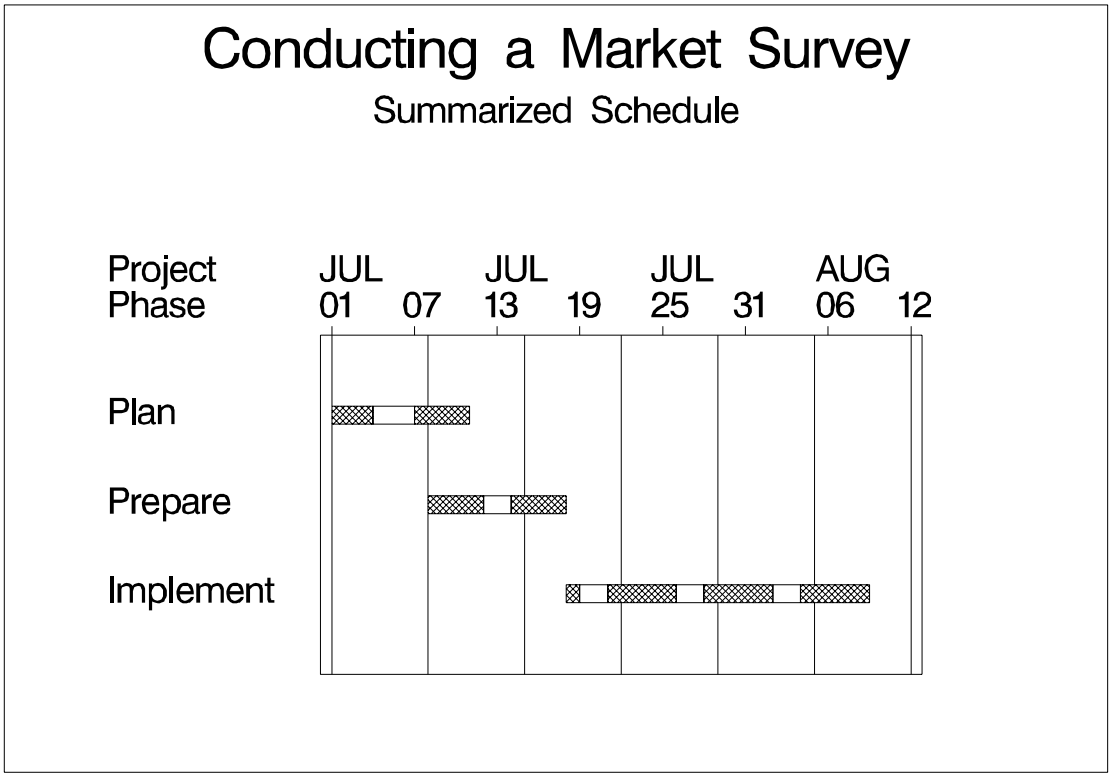
Example 1.4. Summary Report

As mentioned in the "Data Flow" section beginning on page 17, the output data set can be manipulated in several different ways. You can subset the project data to report progress on selected activities, or you can produce reports sorted by a particular field or grouped according to a natural division of the project activities. For large projects, you may want to get a summarized view of the schedule, with the start and finish times of only the major phases of the project.

For the survey project, suppose that you want a condensed report, containing only information about the start and finish times of the three different phases of the project. The following program summarizes the information in the data set survschd and produces a Gantt chart of the summarized schedule (shown in Output 1.4.1).

```
proc sort data=survschd;
   by phase;
   run;

proc summary data=survschd;
   by phase;
   output out=sumsched min(e_start)= max(e_finish)= ;
   var e_start e_finish;
   run;

proc sort data=sumsched;
   by e_start;
   format e_start e_finish date7.;
   run;


goptions hpos=80 vpos=43;
title c=black f=swiss  h=3 'Conducting a Market Survey';
title2 c=black f=swiss h=2 'Summarized Schedule';

proc gantt data=sumsched graphics
     holidata=holidata;
   id phase;
   chart / nojobnum
           nolegend font=swiss
           interval=weekday
           height=2 skip=4
           ref='01jul03'd to '15aug03'd by week
           caxis=black
           holiday=(hol);
   run;
```

Output 1.4.1. Summary Gantt Chart of SURVEY Project



Example 1.5. Resource-Constrained Scheduling

The previous two examples illustrated some of the reports that can be generated using the Schedule output data set produced by PROC CPM. This section illustrates the use of PROC CPM to perform resource-constrained scheduling and to obtain a resource Usage output data set for generating reports of resource utilization during the course of a project. A primary concern in data processing centers is the number of processors needed to perform various tasks. Given a series of programming tasks, a common question faced by a data center operator is how to allocate computer resources to the various tasks.

Consider a simple job that involves sorting six data sets A, B, C, D, E, and F, merging the first three into one master data set, merging the last three into another comparison data set, and then comparing the two merged data sets. The precedence constraints between the activities (captured by the variables task and succ), the time required by the activities (the variable dur), and the resource required (the variable processor) are shown in the following code:

```
data program;
   format task $8. succ $8. ;
   input task & succ & dur processor;
   datalines;
Sort A    Merge 1    5   1
Sort B    Merge 1    4   1
Sort C    Merge 1    3   1
Sort D    Merge 2    6   1
Sort E    Merge 2    4   1
Sort F    Merge 2    6   1
Merge 1   Compare    5   1
Merge 2   Compare    4   1
Compare   .          5   1
;
```

If the programming project is scheduled (in absolute units) without any resource constraints, it will take 15 time units for completion and will require a maximum availability of six processors. Suppose now that only two processors are available. The resin data set limits the availability of the resource to 2, and PROC CPM is invoked with two input data sets (Activity data set program and Resource data set resin) to produce a resource-constrained schedule.

PROC CPM produces two output data sets. The Schedule data set (progschd) contains the resource-constrained schedule (S_START and S_FINISH variables) in addition to the early and late start unconstrained schedules. The Usage data set (progrout) shows the number of processors required at every unit of time, if the early start schedule or the late start schedule or the resource-constrained schedule were followed, in the variables eprocessor, lprocessor, and rprocessor, respectively; the variable aprocessor shows the number of processors remaining after resource allocation. The two output data sets are displayed in Output 1.5.1.

```
data resin;
   input per processor;
   datalines;
0  2
;

proc cpm data=program resin=resin
         out=progschd resout=progrout;
   activity  task;
   duration  dur;
   successor succ;
   resource  processor/per=per;
   run;

title 'Scheduling Programming Tasks';
title2 'Data Set PROGSCHD';
proc print data=progschd;
   run;

title2 'Data Set PROGROUT';
```

```
proc print data=progrout;
   run;
```

The Schedule and Usage data sets, displayed in Output 1.5.1, can be used to generate any type of report concerning the schedules or processor usage. In the following program, the unconstrained and constrained schedules are first plotted using PROC GANTT (see Output 1.5.2).

Output 1.5.1. Data Sets PROGSCHD and PROGROUT

Scheduling Programming Tasks

Data Set PROGSCHD

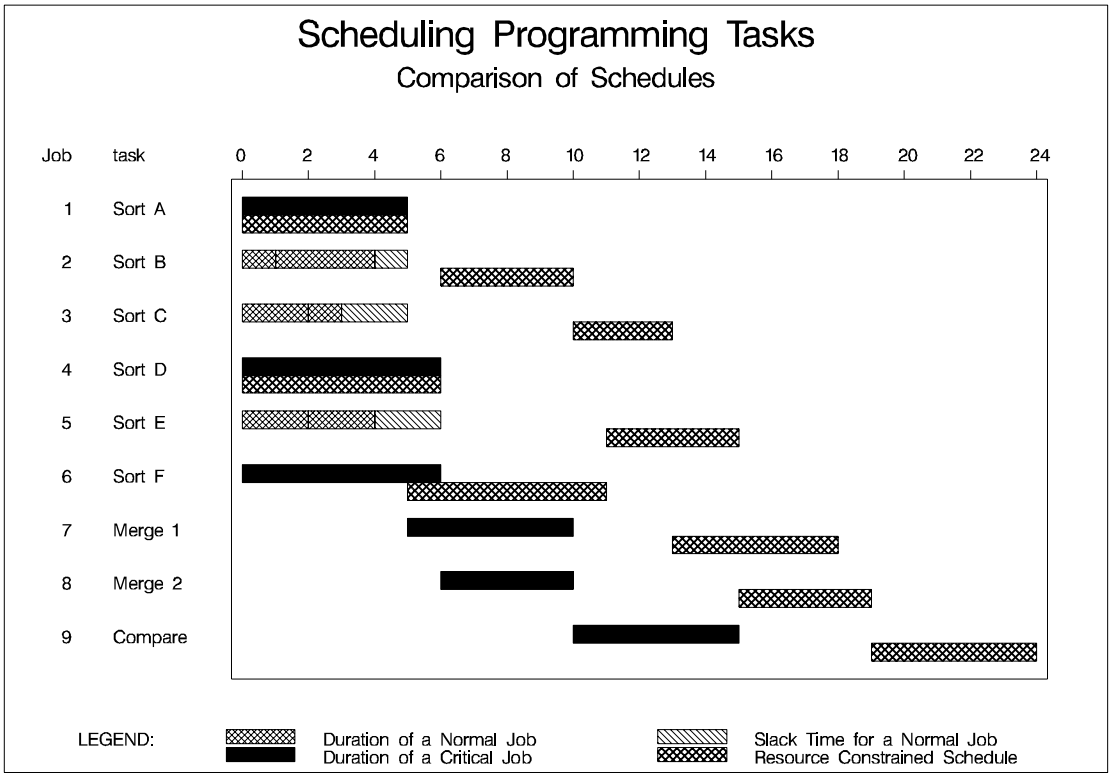
Obs	task	subc	dur	proc	S_START	S_FINISH	E_START	E_FINISH	L_START	L_FINISH
1	Sort A	Merge 1	5	1	0	5	0	5	0	5
2	Sort B	Merge 1	4	1	6	10	0	4	1	5
3	Sort C	Merge 1	3	1	10	13	0	3	2	5
4	Sort D	Merge 2	6	1	0	6	0	6	0	6
5	Sort E	Merge 2	4	1	11	15	0	4	2	6
6	Sort F	Merge 2	6	1	5	11	0	6	0	6
7	Merge 1	Compare	5	1	13	18	5	10	5	10
8	Merge 2	Compare	4	1	15	19	6	10	6	10
9	Compare		5	1	19	24	10	15	10	15

Data Set PROGROUT

Obs	_TIME_	Eprocessor	Lprocessor	Rprocessor	Aprocessor
1	0	6	3	2	0
2	1	6	4	2	0
3	2	6	6	2	0
4	3	5	6	2	0
5	4	3	6	2	0
6	5	3	4	2	0
7	6	2	2	2	0
8	7	2	2	2	0
9	8	2	2	2	0
10	9	2	2	2	0
11	10	1	1	2	0
12	11	1	1	2	0
13	12	1	1	2	0
14	13	1	1	2	0
15	14	1	1	2	0
16	15	0	0	2	0
17	16	0	0	2	0
18	17	0	0	2	0
19	18	0	0	1	1
20	19	0	0	1	1
21	20	0	0	1	1
22	21	0	0	1	1
23	22	0	0	1	1
24	23	0	0	1	1
25	24	0	0	0	2

```
goptions hpos=80 vpos=43;
title f=swiss 'Scheduling Programming Tasks';
title2 f=swiss h=1.5 'Comparison of Schedules';
proc gantt data=progschd graphics;
   chart / font=swiss increment=2 caxis=black;
   id task;
   run;
```

Output 1.5.2. Gantt Chart Comparing Schedules

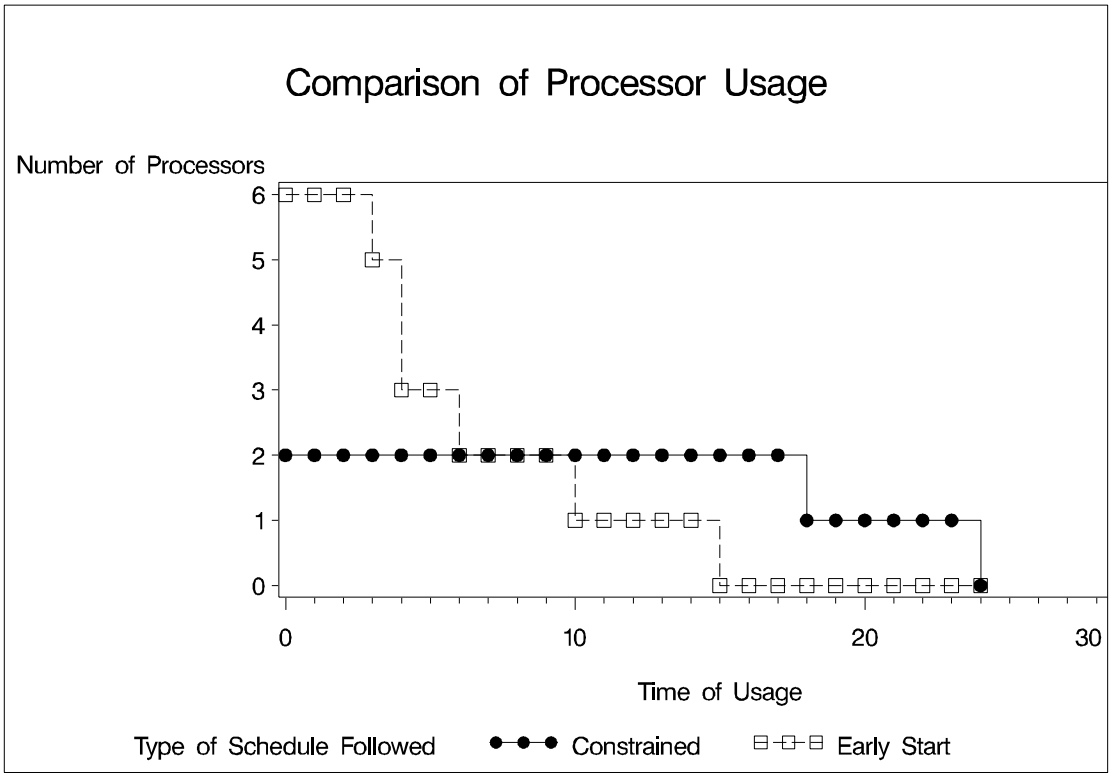


Next, the GPLOT procedure is invoked using the Usage data set to compare the unconstrained and the constrained usage of the resource (see Output 1.5.3).

```
/* Create a data set for use with PROC GPLOT */
data plotout;
   set progrout;
   label _time_='Time of Usage';
   label processor='Number of Processors';
   label resource='Type of Schedule Followed';
   resource='Constrained';
   processor=rprocessor; output;
   resource='Early Start';
   processor=eprocessor; output;
   run;
```

```
axis1 minor=none width=3;
axis2 length=80 pct;
symbol1 i=steplj;
symbol2 i=steplj l=3;

goptions ftext=swiss;
title2 h=1.5 'Comparison of Processor Usage';
proc gplot data=plotout;
   plot processor * _time_ = resource/ vaxis=axis1
                                       haxis=axis2
                                       caxis=black;
   run;
```

Output 1.5.3. Plot Comparing Resource Usage



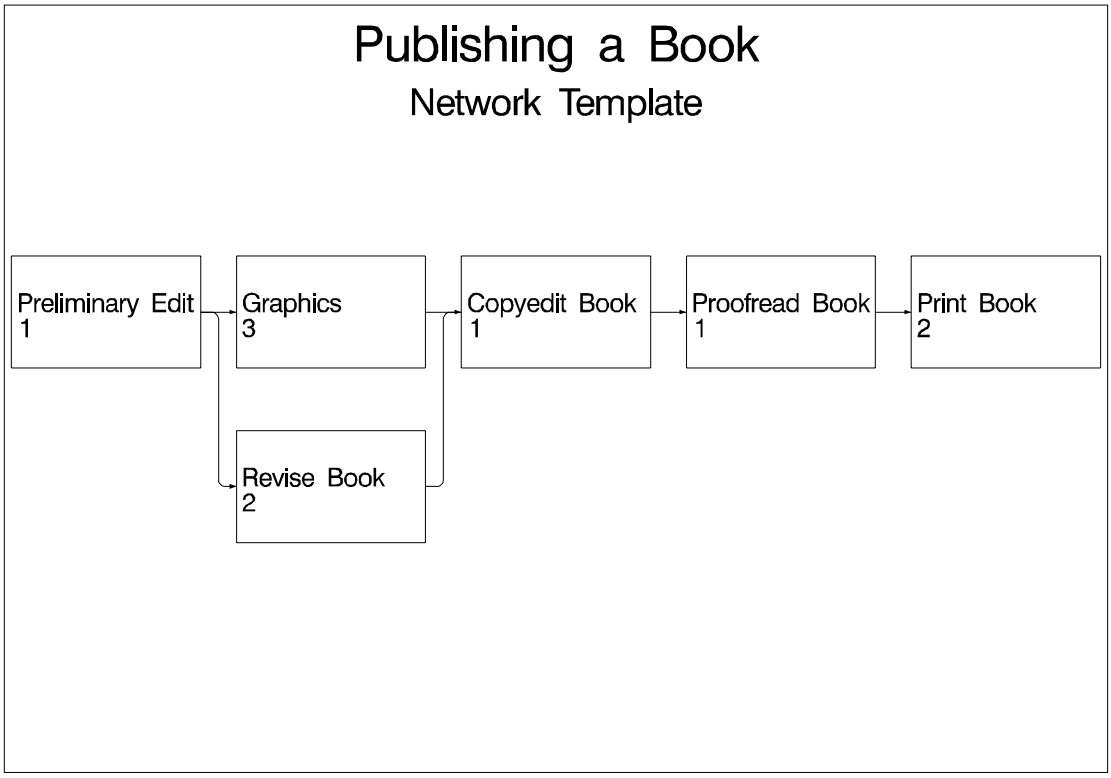
Example 1.6. Multiple Projects

Often a project is divided into several subprojects, each of which is then broken into activities with precedence constraints. For reporting or accounting purposes, it may be essential to group activities or to aggregate the information pertaining to activities in a given group. Sometimes, totally different projects use a common pool of resources and you may want to schedule all the projects using the common pool; you may want to vary the priority with which the resources are allotted to the activities on the basis of the projects to which they belong. Often, you have several projects that are essentially the same, with only a few minor differences; these projects may also share a common pool of resources. In such cases, you may want to have a project *template* listing all the activities and their precedence relationships; for each specific

project you can copy the template, make any modifications that are necessary for the given scenario, and determine the project schedule accordingly.

This example illustrates some of these possibilities for a multiproject scenario. The project is first scheduled using PROC CPM, and then the PM procedure is used with the same input data set to illustrate the project displayed in the PM Window.

Output 1.6.1. Network Diagram for Project Book



Consider a publishing company that accepts manuscripts from different authors for publication. The publication of each book can be treated as a project. Thus, at a given point in time, several projects, almost identical in nature, may be in progress. Some of the resources that may be needed are a technical editor, a copyeditor, and a graphic artist. All the books that are currently being worked on share a common pool of these resources. This example uses a simplified version of such a scenario to illustrate some of the ways in which you can handle multiple projects competing for the same pool of resources.

The network in Output 1.6.1 represents some of the tasks required to publish one book and the precedence constraints among these tasks; the durations in the diagram are in weeks. Suppose that the *generic* project data are in the data set book, which is displayed in Output 1.6.2. This data set is used as a *template* for creating the Activity data set for any book publishing project.

Suppose that the company is working on two books simultaneously. The editor and artist must now allocate their time between the two books. The following program uses the *template* data set book to create Activity data sets book1 and book2 corresponding to the publication of each book. Any modifications to the generic project

data can be made in the DATA step or by using PROC PM. In this example, the duration for the first activity, 'Preliminary Edit,' is changed to two weeks for the second book. The two Activity data sets book1 and book2 are also displayed in Output 1.6.2.

```
data book1;
   length act $6. succ $6.;
   set book;
   subproj = "Book 1";
   act = "B1"||task;
   if succ ^= " " then succ = "B1"||succ;
   run;

title 'Publishing Book 1';
proc print data=book1;
   var subproj task act succ id dur editor artist;
   run;

data book2;
   length act $6. succ $6.;
   set book;
   subproj = "Book 2";
   act  = "B2"||task;
   if act = "B2PEDT" then dur = 2;
   if succ ^= " "    then succ = "B2"||succ;
   run;

title 'Publishing Book 2';
proc print data=book2;
   var subproj task act succ id dur editor artist;
   run;
```

Output 1.6.2. Template and Activity Data Sets for Book Publishing Example

Publishing a Book
Template Data Set

Obs	id	task	dur	succ	editor	artist
1	Preliminary Edit	PEDT	1	REV	1	.
2	Preliminary Edit	PEDT	1	GRPH	1	.
3	Revise Book	REV	2	CEDT	1	.
4	Graphics	GRPH	3	CEDT	.	1
5	Copyedit Book	CEDT	1	PRF	1	.
6	Proofread Book	PRF	1	PRNT	1	.
7	Print Book	PRNT	2		.	.

Publishing Book 1

Obs	subproj	task	act	succ	id	dur	editor	artist
1	Book 1	PEDT	B1PEDT	B1REV	Preliminary Edit	1	1	.
2	Book 1	PEDT	B1PEDT	B1GRPH	Preliminary Edit	1	1	.
3	Book 1	REV	B1REV	B1CEDT	Revise Book	2	1	.
4	Book 1	GRPH	B1GRPH	B1CEDT	Graphics	3	.	1
5	Book 1	CEDT	B1CEDT	B1PRF	Copyedit Book	1	1	.
6	Book 1	PRF	B1PRF	B1PRNT	Proofread Book	1	1	.
7	Book 1	PRNT	B1PRNT		Print Book	2	.	.

Publishing Book 2

Obs	subproj	task	act	succ	id	dur	editor	artist
1	Book 2	PEDT	B2PEDT	B2REV	Preliminary Edit	2	1	.
2	Book 2	PEDT	B2PEDT	B2GRPH	Preliminary Edit	2	1	.
3	Book 2	REV	B2REV	B2CEDT	Revise Book	2	1	.
4	Book 2	GRPH	B2GRPH	B2CEDT	Graphics	3	.	1
5	Book 2	CEDT	B2CEDT	B2PRF	Copyedit Book	1	1	.
6	Book 2	PRF	B2PRF	B2PRNT	Proofread Book	1	1	.
7	Book 2	PRNT	B2PRNT		Print Book	2	.	.

As a next step, the data sets for the two subprojects are combined to form an Activity data set for the entire project. A variable priority is assigned the value '1' for activities pertaining to the first book and the value '2' for those pertaining to the second one. In other words, Book 1 has priority over Book 2. The Resource data set specifies the availability for each of the resources to be 1. The input data sets, books and resource, are displayed in Output 1.6.3.

```
data books;
   set book1 book2;
   if subproj = "Book 1" then priority = 1;
   else priority = 2;
   run;

title 'Publishing Books 1 and 2';
proc print data=books;
   var subproj priority task act succ id dur editor artist;
   run;
```

```
data resource;
   input avdate & date7. editor artist;
   format avdate date7.;
   datalines;
1jan03   1   1
;

title 'Resources Available';
proc print data=resource;
   run;
```

Output 1.6.3. Input Data Sets for Book Publishing Example

Publishing Books 1 and 2

Obs	subproj	priority	task	act	succ	id	dur	editor	artist
1	Book 1	1	PEDT	B1PEDT	B1REV	Preliminary Edit	1	1	.
2	Book 1	1	PEDT	B1PEDT	B1GRPH	Preliminary Edit	1	1	.
3	Book 1	1	REV	B1REV	B1CEDT	Revise Book	2	1	.
4	Book 1	1	GRPH	B1GRPH	B1CEDT	Graphics	3	.	1
5	Book 1	1	CEDT	B1CEDT	B1PRF	Copyedit Book	1	1	.
6	Book 1	1	PRF	B1PRF	B1PRNT	Proofread Book	1	1	.
7	Book 1	1	PRNT	B1PRNT		Print Book	2	.	.
8	Book 2	2	PEDT	B2PEDT	B2REV	Preliminary Edit	2	1	.
9	Book 2	2	PEDT	B2PEDT	B2GRPH	Preliminary Edit	2	1	.
10	Book 2	2	REV	B2REV	B2CEDT	Revise Book	2	1	.
11	Book 2	2	GRPH	B2GRPH	B2CEDT	Graphics	3	.	1
12	Book 2	2	CEDT	B2CEDT	B2PRF	Copyedit Book	1	1	.
13	Book 2	2	PRF	B2PRF	B2PRNT	Proofread Book	1	1	.
14	Book 2	2	PRNT	B2PRNT		Print Book	2	.	.

Resources Available

Obs	avdate	editor	artist
1	01JAN03	1	1

PROC CPM is then invoked to schedule the project to start on January 1, 2003. The PROJECT statement is used to indicate the subproject to which each activity belongs. The data set bookschd (displayed in Output 1.6.4) contains the schedule for the entire project. The ADDACT option on the PROC CPM statement adds observations for each of the subprojects, 'Book 1' and 'Book 2,' as well as one observation for the entire project. These observations are added at the end of the list of the observations corresponding to the observations in the input data set. The Usage data set booksout is also displayed in Output 1.6.4.

```
proc cpm data=books resin=resource
         out=bookschd resout=booksout
         date='1jan03'd interval=week
         addact;
   act      act;
   dur      dur;
   succ     succ;
   resource editor artist / per=avdate avp rcp
                            rule=actprty actprty=priority
                            delayanalysis;
   id       id task;
   project  subproj;
   run;
```

Compare the E_START and S_START schedules (in the data set bookschd) and note that on January 1, the activity 'B1PEDT' for Book1 is scheduled to start while the preliminary editing of book 2 (activity B2PEDT) has been postponed, due to subproject 'Book 1' having priority over subproject 'Book 2.' On January 22, there is no activity belonging to subproject 'Book 1' that demands an editor; thus, the activity 'B2PEDT' is scheduled to start on that day. As a result, the editor is working on an activity in the second project for two weeks starting from January 22, 2003; when 'B1CEDT' is ready to start, the editor is not available, causing a delay in this activity. Thus, even though the first book has priority over the second book, the scheduling algorithm does not keep a resource waiting for activities in the first project. However, if you enable activity splitting, you can reclaim the resource for the first book by allowing activities in the second project to be split, if necessary. For details regarding the scheduling algorithm allowing splitting of activities, see Chapter 2, "The CPM Procedure."

Note: The entire project finishes on April 1, 2003; resource constraints have delayed project completion by four weeks. The variable R_DELAY in the Schedule data set bookschd indicates the amount of delay in weeks caused by resource constraints. The value of R_DELAY does not include any delay in the activity that is caused by a resource delay in one of its predecessors. See Example 2.15 in Chapter 2, "The CPM Procedure," for more details about the R_DELAY variable.

Output 1.6.4. Data Sets BOOKSCHD and BOOKSOUT

Schedule for Project BOOKS

Obs	subproj	PROJ_DUR	PROJ_LEV	act	succ	dur	id	task	editor	art	_START_
1	Book 1	.	2	B1PEDT	B1REV	1	Preliminary Edit	PEDT	1	.	01JAN03
2	Book 1	.	2	B1PEDT	B1GRPH	1	Preliminary Edit	PEDT	1	.	01JAN03
3	Book 1	.	2	B1REV	B1CEDT	2	Revise Book	REV	1	.	08JAN03
4	Book 1	.	2	B1GRPH	B1CEDT	3	Graphics	GRPH	.	1	08JAN03
5	Book 1	.	2	B1CEDT	B1PRF	1	Copyedit Book	CEDT	1	.	05FEB03
6	Book 1	.	2	B1PRF	B1PRNT	1	Proofread Book	PRF	1	.	12FEB03
7	Book 1	.	2	B1PRNT		2	Print Book	PRNT	.	.	19FEB03
8	Book 2	.	2	B2PEDT	B2REV	2	Preliminary Edit	PEDT	1	.	22JAN03
9	Book 2	.	2	B2PEDT	B2GRPH	2	Preliminary Edit	PEDT	1	.	22JAN03
10	Book 2	.	2	B2REV	B2CEDT	2	Revise Book	REV	1	.	19FEB03
11	Book 2	.	2	B2GRPH	B2CEDT	3	Graphics	GRPH	.	1	05FEB03
12	Book 2	.	2	B2CEDT	B2PRF	1	Copyedit Book	CEDT	1	.	05MAR03
13	Book 2	.	2	B2PRF	B2PRNT	1	Proofread Book	PRF	1	.	12MAR03
14	Book 2	.	2	B2PRNT		2	Print Book	PRNT	.	.	19MAR03
15		9	1	Book 1		.			.	.	01JAN03
16		10	1	Book 2		.			.	.	22JAN03
17		13	0			.			.	.	01JAN03

Obs	_START_	_EFINISH_	EARLYSTART	EARLYFINISH	LATESTART	LATEFINISH	RESDELAY	SUPPLY
1	07JAN03	01JAN03	07JAN03	08JAN03	14JAN03	0		
2	07JAN03	01JAN03	07JAN03	08JAN03	14JAN03	0		
3	21JAN03	08JAN03	21JAN03	22JAN03	04FEB03	0		
4	28JAN03	08JAN03	28JAN03	15JAN03	04FEB03	0		
5	11FEB03	29JAN03	04FEB03	05FEB03	11FEB03	1	editor	
6	18FEB03	05FEB03	11FEB03	12FEB03	18FEB03	0		
7	04MAR03	12FEB03	25FEB03	19FEB03	04MAR03	0		
8	04FEB03	01JAN03	14JAN03	01JAN03	14JAN03	3	editor	
9	04FEB03	01JAN03	14JAN03	01JAN03	14JAN03	3	editor	
10	04MAR03	15JAN03	28JAN03	22JAN03	04FEB03	2	editor	
11	25FEB03	15JAN03	04FEB03	15JAN03	04FEB03	0		
12	11MAR03	05FEB03	11FEB03	05FEB03	11FEB03	0		
13	18MAR03	12FEB03	18FEB03	12FEB03	18FEB03	0		
14	01APR03	19FEB03	04MAR03	19FEB03	04MAR03	0		
15	04MAR03	01JAN03	25FEB03	08JAN03	04MAR03	0		
16	01APR03	01JAN03	04MAR03	01JAN03	04MAR03	3		
17	01APR03	01JAN03	04MAR03	01JAN03	04MAR03	0		

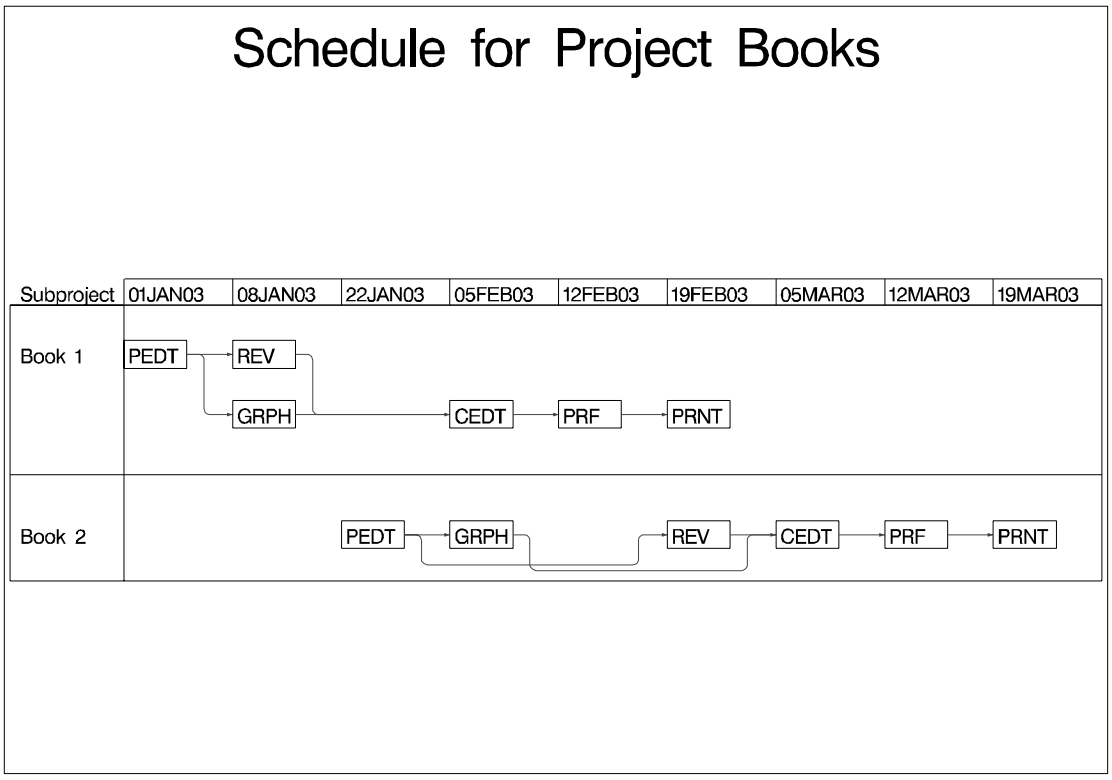
Resource Usage for Project BOOKS

Obs	_TIME_	Reditor	Aeditor	Rartist	Aartist
1	01JAN03	1	0	0	1
2	08JAN03	1	0	1	0
3	15JAN03	1	0	1	0
4	22JAN03	1	0	1	0
5	29JAN03	1	0	0	1
6	05FEB03	1	0	1	0
7	12FEB03	1	0	1	0
8	19FEB03	1	0	1	0
9	26FEB03	1	0	0	1
10	05MAR03	1	0	0	1
11	12MAR03	1	0	0	1
12	19MAR03	0	1	0	1
13	26MAR03	0	1	0	1
14	02APR03	0	1	0	1

The output data sets bookschd and booksout can be used to produce graphical reports of the schedule and the resource usage. In particular, the Schedule data set can be used to produce a zoned, time-scaled network diagram as shown in Output 1.6.5. The program used to produce the network diagram is shown in the following code. In this example, only the leaf tasks (those without any subtasks) are used to draw the network diagram. Further, the activities are aligned according to the resource-constrained start times and grouped according to the subproject.

```
goptions hpos=98 vpos=60;
pattern1 v=e c=green;
pattern2 v=e c=red;
title c=black f=swiss h=4 'Schedule for Project Books';

proc netdraw data=bookschd(where=(proj_dur=.) graphics;
   actnet / act=task succ=succ font=swiss
            id=(task) nodefid nolabel
            xbetween=8 htext=3 pcompress
            zone=subproj zonepat zonespace
            align=s_start separatearcs;
   label subproj = 'Subproject';
   run;
```

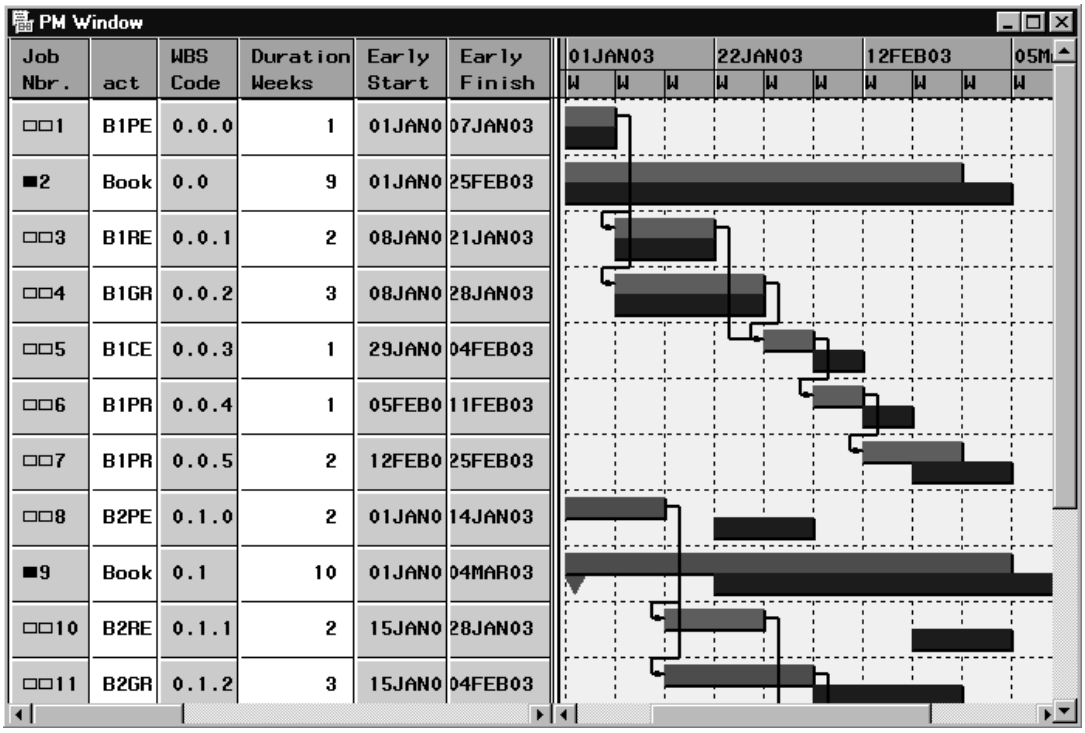
Output 1.6.5. Resource Constrained Schedule for Project Books



The same project can also be scheduled using the PM procedure, as shown in the following statements. The resulting PM Window is shown in Output 1.6.6. The advantage with using PROC PM is that you can use the PM Window to edit the activity information, such as the durations, resource requirements, and so forth.

```
proc pm data=books resin=resource
        out=pmsched resout=pmrout
        date='1jan03'd interval=week;
   act       act;
   dur       dur;
   succ      succ;
   resource editor artist / per=avdate
                            avp rcp
                            rule=actprty
                            actprty=priority
                            delayanalysis;
   id        id task;
   project   subproj;
   run;
```

Output 1.6.6. PM Window on Book Project



Example 1.7. Sequential Scheduling of Projects

Suppose the schedule displayed in Output 1.6.4 is not acceptable; you want the first book to be finished as soon as possible and do not want resources to be claimed by the second book at the cost of the first book. One way to accomplish this is to enable activities related to the second book to be split whenever the first book demands a resource currently in use by the second book. If you do not want activities to be split, you can still accomplish your goal by sequential scheduling. The structure of the input and output data sets enables you to schedule the two subprojects sequentially.

This example illustrates the sequential scheduling of subprojects 'Book 1' and 'Book 2.' The following program first schedules the subproject 'Book 1' using the resources available. The resulting schedule is displayed in Output 1.7.1. The Usage data set bk1out is also displayed in Output 1.7.1.

```
/* Schedule the higher priority project first */
proc cpm data=book1 resin=resource
         out=bk1schd resout=bk1out
         date='1jan03'd interval=week;
   act      act;
   dur      dur;
   succ     succ;
   resource editor artist / per=avdate avp rcp;
   id       id;
   run;
```

Output 1.7.1. Sequential Scheduling of Subprojects: Book 1

Schedule for sub-project BOOK1

Obs	act	succ	dur	id	editor	artist	S_START
1	B1PEDT	B1REV	1	Preliminary Edit	1	.	01JAN03
2	B1PEDT	B1GRPH	1	Preliminary Edit	1	.	01JAN03
3	B1REV	B1CEDT	2	Revise Book	1	.	08JAN03
4	B1GRPH	B1CEDT	3	Graphics	.	1	08JAN03
5	B1CEDT	B1PRF	1	Copyedit Book	1	.	29JAN03
6	B1PRF	B1PRNT	1	Proofread Book	1	.	05FEB03
7	B1PRNT		2	Print Book	.	.	12FEB03

Obs	S_FINISH	E_START	E_FINISH	L_START	L_FINISH
1	07JAN03	01JAN03	07JAN03	01JAN03	07JAN03
2	07JAN03	01JAN03	07JAN03	01JAN03	07JAN03
3	21JAN03	08JAN03	21JAN03	15JAN03	28JAN03
4	28JAN03	08JAN03	28JAN03	08JAN03	28JAN03
5	04FEB03	29JAN03	04FEB03	29JAN03	04FEB03
6	11FEB03	05FEB03	11FEB03	05FEB03	11FEB03
7	25FEB03	12FEB03	25FEB03	12FEB03	25FEB03

Resource Usage for sub-project BOOK1

Obs	_TIME_	Reditor	Aeditor	Rartist	Aartist
1	01JAN03	1	0	0	1
2	08JAN03	1	0	1	0
3	15JAN03	1	0	1	0
4	22JAN03	0	1	1	0
5	29JAN03	1	0	0	1
6	05FEB03	1	0	0	1
7	12FEB03	0	1	0	1
8	19FEB03	0	1	0	1
9	26FEB03	0	1	0	1

The Usage data set produced by PROC CPM has two variables, Aeditor and Aartist, showing the availability of the editor and the artist on each day of the project, *after* scheduling subproject 'Book 1.' This data set is used to create the data set remres, listing the remaining resources available, which is then used as the Resource input data set for scheduling the subproject 'Book 2.' The following program shows the DATA step and the invocation of PROC CPM.

The schedule for publishing 'Book 2' is displayed in Output 1.7.2. The Usage data set bk2out is also displayed in Output 1.7.2. Note that this method of scheduling has ensured that 'Book 1' is not delayed; however, the entire project has been delayed by two more weeks, resulting in a total delay of six weeks.

```
/* Construct the Resource availability data set */
/* with proper resource names                   */
data remres;
   set bk1out;
   avdate=_time_;
   editor=aeditor;
   artist=aartist;
   keep avdate editor artist;
   format avdate date7.;
   run;
```

```
proc cpm data=book2 resin=remres
       out=bk2schd resout=bk2out
       date='1jan03'd interval=week;
   act      act;
   dur      dur;
   succ     succ;
   resource editor artist / per=avdate avp rcp;
   id       id;
   run;
```

Output 1.7.2. Sequential Scheduling of Subprojects: Book 2

Schedule for sub-project BOOK2

Obs	act	succ	dur	id	editor	artist	S_START
1	B2PEDT	B2REV	2	Preliminary Edit	1	.	12FEB03
2	B2PEDT	B2GRPH	2	Preliminary Edit	1	.	12FEB03
3	B2REV	B2CEDT	2	Revise Book	1	.	26FEB03
4	B2GRPH	B2CEDT	3	Graphics	.	1	26FEB03
5	B2CEDT	B2PRF	1	Copyedit Book	1	.	19MAR03
6	B2PRF	B2PRNT	1	Proofread Book	1	.	26MAR03
7	B2PRNT		2	Print Book	.	.	02APR03

Obs	S_FINISH	E_START	E_FINISH	L_START	L_FINISH
1	25FEB03	01JAN03	14JAN03	01JAN03	14JAN03
2	25FEB03	01JAN03	14JAN03	01JAN03	14JAN03
3	11MAR03	15JAN03	28JAN03	22JAN03	04FEB03
4	18MAR03	15JAN03	04FEB03	15JAN03	04FEB03
5	25MAR03	05FEB03	11FEB03	05FEB03	11FEB03
6	01APR03	12FEB03	18FEB03	12FEB03	18FEB03
7	15APR03	19FEB03	04MAR03	19FEB03	04MAR03

Resource Usage for sub-project BOOK2

Obs	_TIME_	Reditor	Aeditor	Rartist	Aartist
1	12FEB03	1	0	0	1
2	19FEB03	1	0	0	1
3	26FEB03	1	0	1	0
4	05MAR03	1	0	1	0
5	12MAR03	0	1	1	0
6	19MAR03	1	0	0	1
7	26MAR03	1	0	0	1
8	02APR03	0	1	0	1
9	09APR03	0	1	0	1
10	16APR03	0	1	0	1

Example 1.8. Project Cost Control

Cost control and accounting are important aspects of project management. Cost data for a project may be associated with activities or groups of activities, or with resources, such as personnel or equipment. For example, consider a project that consists of several subprojects, each of which is contracted to a different company. From the contracting company's point of view, each subproject can be treated as one cost item; all the company needs to know is how much each subproject is going to cost. On the other hand, another project may contain several activities, each of which requires two types of labor, skilled and unskilled. The cost for each activity in the project may have to be computed on the basis of how much skilled or unskilled labor that activity uses. In this case, activity and project costs are determined from the resources

used. Further, for any project, there may be several ways in which costs need to be summarized and accounted for. In addition to determining the cost of each individual activity, you may want to determine periodic budgets for different departments that are involved with the project or compare the actual costs that were incurred with the budgeted costs.

It is easy to set up cost accounting systems using the output data sets produced by PROC CPM, whether costs are associated with activities or with resources. In fact, you can even treat cost as a consumable resource if you can estimate the cost per day for each of the activities (see Chapter 2, "The CPM Procedure," for details on resource allocation and types of resources). This example illustrates such a method for monitoring costs and shows how you can compute some of the standard cost performance measures used in project management.

The following three measures can be used to determine if a project is running on schedule and within budget (see Moder, Phillips, and Davis 1983, for a detailed discussion on project cost control):

- *Actual cost of work performed* (**ACWP**) is the actual cost expended to perform the work accomplished in a given period of time.
- *Budgeted cost of work performed* (**BCWP**) is the budgeted cost of the work *completed* in a given period of time.
- *Budgeted cost of work scheduled* (**BCWS**) is the budgeted cost of the work *scheduled* to be accomplished in a given period of time (if a baseline schedule were followed).

Consider the survey example described earlier in this chapter. Suppose that it is possible to estimate the cost per day for each activity in the project. The following data set survcost contains the project data (activity, succ1–succ3, id, duration) and a variable named cost containing the cost per day in dollars. In order to compute the BCWS for the project, you need to establish a baseline schedule. Suppose the early start schedule computed by PROC CPM is chosen as the baseline schedule. The Resource data set costavl establishes cost as a consumable resource, so that the CPM procedure can be used to accumulate costs (using the CUMUSAGE option).

The following program invokes PROC CPM with the RESOURCE statement and saves the Usage data set in survrout. The variable ecost in this Usage data set contains the cumulative expense incurred for the baseline schedule; this is the same as the budgeted cost of work scheduled (or BCWS) saved in the data set basecost.

```
data survcost;
   format id $20. activity $8. succ1-succ3 $8. ;
   input id & activity & duration succ1 & succ2 & succ3 & cost;
   datalines;
Plan Survey           plan sur   4 hire per  design q  .         300
Hire Personnel        hire per   5 trn per   .         .         350
Design Questionnaire  design q   3 trn per   select h  print q   100
Train Personnel       trn per    3 cond sur  .         .         500
Select Households     select h   3 cond sur  .         .         300
Print Questionnaire   print q    4 cond sur  .         .         250
Conduct Survey        cond sur  10 analyze   .         .         200
Analyze Results       analyze    6 .         .         .         500
;


data holidata;
   format hol date7.;
   hol = '4jul03'd;
   run;

data costavl;
   input per & date7. otype $ cost;
   format per date7.;
   datalines;
.       restype   2
1jul03  reslevel  12000
;

proc cpm date='1jul03'd interval=weekday
         data=survcost resin=costavl   holidata=holidata
         out=sched     resout=survrout;
   activity   activity;
   successor  succ1-succ3;
   duration   duration;
   holiday    hol;
   id         id;
   resource   cost / period  = per
                     obstype = otype cumusage;
   run;

data basecost (keep = _time_ bcws);
   set survrout;
   bcws = ecost;
   run;
```

Suppose that the project started as planned on July 1, 2003, but some of the activities took longer than planned and some of the cost estimates were found to be incorrect. The following data set, actual, contains updated information: the variables as and af contain the actual start and finish times of the activities that have been completed or are in progress. The variable actcost contains the revised cost per day for each activity. The following program combines this information with the existing project data and saves the result in the data set update, displayed in Output 1.8.1. The Resource data set costavl2 (also displayed in Output 1.8.1) defines cost and actcost as consumable resources.

```
data actual;
   format id $20. ;
   input id & as & date9. af & date9. actcost;
   format as af date7.;
   datalines;
Plan Survey            1JUL03   8JUL03   275
Hire Personnel         9JUL03  15JUL03   350
Design Questionnaire  10JUL03  14JUL03   150
Train Personnel       16JUL03  17JUL03   800
Select Households     15JUL03  17JUL03   450
Print Questionnaire   15JUL03  18JUL03   250
Conduct Survey        21JUL03   .        200
;

data update;
   merge survcost actual;
   run;

title 'Activity Data Set UPDATE';
proc print;
   run;

data costavl2;
   input per & date7. otype $ cost actcost;
   format per date7.;
   datalines;
.       restype   2      2
1jul03  reslevel  12000  12000
;

title 'Resource Data Set COSTAVL2';
proc print;
   run;
```

Output 1.8.1. Project Cost Control: Progress Update

Activity Data Set UPDATE

Obs	id	activity	duration	succ1
1	Plan Survey	plan sur	4	hire per
2	Hire Personnel	hire per	5	trn per
3	Design Questionnaire	design q	3	trn per
4	Train Personnel	trn per	3	cond sur
5	Select Households	select h	3	cond sur
6	Print Questionnaire	print q	4	cond sur
7	Conduct Survey	cond sur	10	analyze
8	Analyze Results	analyze	6	

Obs	succ2	succ3	cost	as	af	actcost
1	design q		300	01JUL03	08JUL03	275
2			350	09JUL03	15JUL03	350
3	select h	print q	100	10JUL03	14JUL03	150
4			500	16JUL03	17JUL03	800
5			300	15JUL03	17JUL03	450
6			250	15JUL03	18JUL03	250
7			200	21JUL03	.	200
8			500	.	.	.

Resource Data Set COSTAVL2

Obs	per	otype	cost	actcost
1	.	restype	2	2
2	01JUL03	reslevel	12000	12000

Next, PROC CPM is used to revise the schedule by using the ACTUAL statement to specify the actual start and finish times and the RESOURCE statement to specify both the budgeted and the actual costs. The resulting schedule is saved in the data set updsched (displayed in Output 1.8.2) and the budgeted and the actual cumulative costs of the project (until the current date) are saved in the data set updtrout. These cumulative costs represent the budgeted cost of work performed (BCWP) and the actual cost of work performed (ACWP), respectively, and are saved in the data set updtcost. The two data sets basecost and updtcost are then merged to create a data set that contains the three measures: bcws, bcwp, and acwp. The resulting data set is displayed in Output 1.8.3.

```
proc cpm date='1jul03'd interval=weekday
         data=update    resin=costavl2
         out=updsched   resout=updtrout
         holidata=holidata;
   activity   activity;
   successor  succ1-succ3;
   duration   duration;
   holiday    hol;
   id         id;
   resource   cost actcost / per     = per
                             obstype = otype
                             maxdate = '21jul03'd cumusage;
   actual / a_start=as a_finish=af;
   run;
```

```
title 'Updated Schedule: Data Set UPDSCHED';
proc print data=updsched;
   run;

data updtcost (keep = _time_ bcwp acwp);
   set updtrout;
   bcwp = ecost;
   acwp = eactcost;
   run;

/* Create a combined data set to contain the BCWS, BCWP, ACWP */
/* per day and the cumulative values for these costs.         */
data costs;
   merge basecost updtcost;
   run;

title 'BCWS, BCWP, and ACWP';
proc print data=costs;
   run;
```

Output 1.8.2. Project Cost Control: Updated Schedule

```
                    Updated Schedule: Data Set UPDSCHED

      a                               d
      c                               u
      t                               r      S
      i        s        s        s     a      T      A
      v        u        u        u     t      A      _
O     i        c        c        c     i      T      D
b     t        c        c        c     o      U      U i
s     y        1        2        3     n      S      R d

 1 plan sur hire per design q          4 Completed   5 Plan Survey
 2 hire per trn per                    5 Completed   5 Hire Personnel
 3 design q trn per  select h print q  3 Completed   3 Design Questionnaire
 4 trn per  cond sur                   3 Completed   2 Train Personnel
 5 select h cond sur                   3 Completed   3 Select Households
 6 print q  cond sur                   4 Completed   4 Print Questionnaire
 7 cond sur analyze                   10 In Progress . Conduct Survey
 8 analyze                             6 Pending     . Analyze Results

                        A               S               E               L
       a        A      _       S       _       E       _       L       _
       c        _      F       _       F       _       F       _       F
       t        S      I       S       I       S       I       S       I
   c   c        T      N       T       N       T       N       T       N
O  o   o        A      I       A       I       A       I       A       I
b  s   s        R      S       R       S       R       S       R       S
s  t   t        T      H       T       H       T       H       T       H

 1 300 275 01JUL03 08JUL03 01JUL03 08JUL03 01JUL03 08JUL03 01JUL03 08JUL03
 2 350 350 09JUL03 15JUL03 09JUL03 15JUL03 09JUL03 15JUL03 09JUL03 15JUL03
 3 100 150 10JUL03 14JUL03 10JUL03 14JUL03 10JUL03 14JUL03 10JUL03 14JUL03
 4 500 800 16JUL03 17JUL03 16JUL03 17JUL03 16JUL03 17JUL03 16JUL03 17JUL03
 5 300 450 15JUL03 17JUL03 15JUL03 17JUL03 15JUL03 17JUL03 15JUL03 17JUL03
 6 250 250 15JUL03 18JUL03 15JUL03 18JUL03 15JUL03 18JUL03 15JUL03 18JUL03
 7 200 200 21JUL03       . 21JUL03 01AUG03 21JUL03 01AUG03 21JUL03 01AUG03
 8 500   .       .       . 04AUG03 11AUG03 04AUG03 11AUG03 04AUG03 11AUG03
```

Output 1.8.3. Project Cost Control: BCWS, BCWP, ACWP

BCWS, BCWP, and ACWP

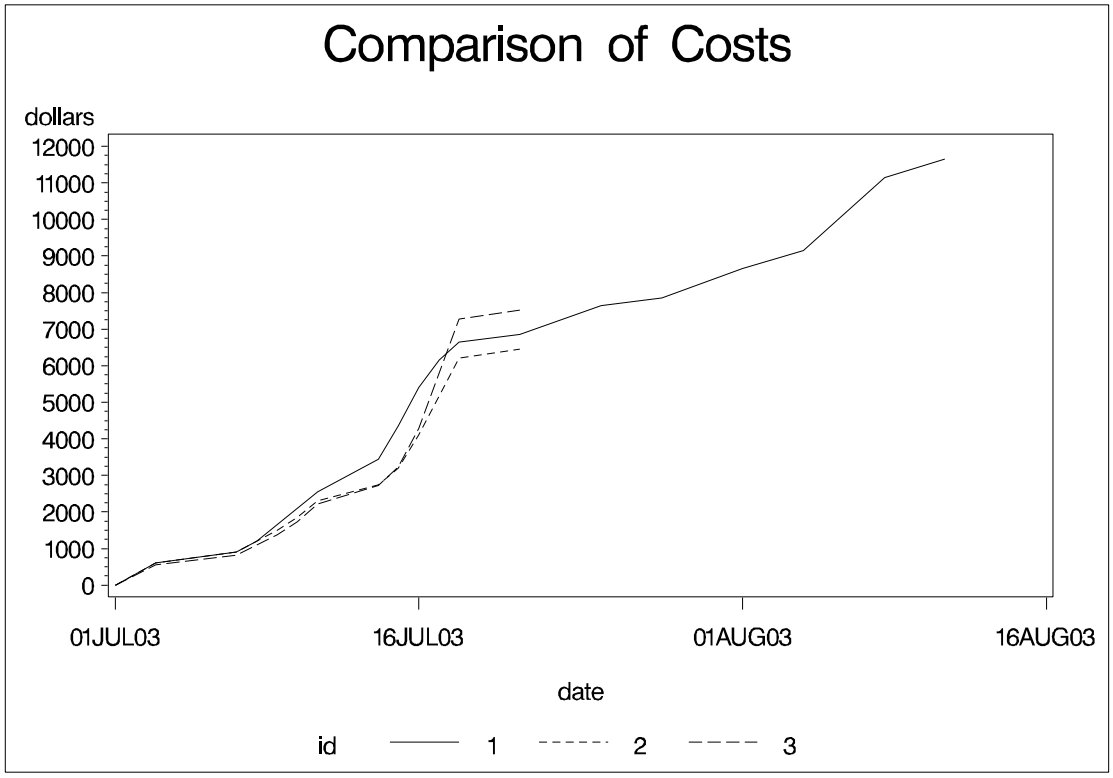
Obs	_TIME_	bcws	bcwp	acwp
1	01JUL03	0	0	0
2	02JUL03	300	300	275
3	03JUL03	600	600	550
4	07JUL03	900	900	825
5	08JUL03	1200	1200	1100
6	09JUL03	1650	1500	1375
7	10JUL03	2100	1850	1725
8	11JUL03	2550	2300	2225
9	14JUL03	3450	2750	2725
10	15JUL03	4350	3200	3225
11	16JUL03	5400	4100	4275
12	17JUL03	6150	5150	5775
13	18JUL03	6650	6200	7275
14	21JUL03	6850	6450	7525
15	22JUL03	7050	.	.
16	23JUL03	7250	.	.
17	24JUL03	7450	.	.
18	25JUL03	7650	.	.
19	28JUL03	7850	.	.
20	29JUL03	8050	.	.
21	30JUL03	8250	.	.
22	31JUL03	8450	.	.
23	01AUG03	8650	.	.
24	04AUG03	9150	.	.
25	05AUG03	9650	.	.
26	06AUG03	10150	.	.
27	07AUG03	10650	.	.
28	08AUG03	11150	.	.
29	11AUG03	11650	.	.

The data set costs, containing the required cost information, is then used as input to PROC GPLOT to produce a plot of the three cumulative cost measures. The plot is shown in Output 1.8.4.

Note: BCWS, BCWP, and ACWP are three of the cost measures used as part of *Earned Value Analysis*, which is an important component of the *Cost/Schedule Control Systems Criteria* (referred to as C/SCSC) that was established in 1967 by the Department of Defense (DOD) to standardize the reporting of cost and schedule performance on major contracts. Refer to Fleming (1988) for a detailed discussion of C/SCSC. Similar methods, such as the ones described in this example, can be used to calculate all the relevant measures for analyzing cost and schedule performance.

```
/* Plot the cumulative costs */
data costplot (keep=date dollars id);
   set costs;
   format date date7.;
   date = _time_;
   if bcws ^= . then do;
      dollars = BCWS; id = 1; output;
      end;
   if bcwp ^= . then do;
      dollars = BCWP; id = 2; output;
      end;
   if acwp ^= . then do;
      dollars = ACWP; id = 3; output;
      end;
   run;

legend1 frame
        value=(f=swiss c=black j=l f=swiss 'BCWS' 'BCWP' 'ACWP')
        label=(f=swiss c=black);

axis1 width=2
      order=('1jul03'd to '1aug03'd by week)
      length=60 pct
      value=(f=swiss c=black)
      label=(f=swiss c=black);

axis2 width=2
      order=(0 to 12000 by 2000)
      length = 55 pct
      value=(f=swiss c=black)
      label=(f=swiss c=black);

symbol1 i=join v=none c=green  w=4 l=1;
symbol2 i=join v=none c=blue   w=4 l=2;
symbol3 i=join v=none c=red    w=4 l=3;
title f=swiss c=black 'Comparison of Costs';

proc gplot data=costplot;
   plot dollars * date = id / legend=legend1
                              haxis=axis1
                              vaxis=axis2;
   run;
```

Output 1.8.4. Plot of BCWS, BCWP, and ACWP



Example 1.9. Subcontracting Decisions

Making decisions about subcontracting forms an important part of several medium-to-large scale projects. For example, in the pharmaceutical industry, the analysis of clinical trials may be a part of the drug development project that could either be accomplished by the company's statistical group or be subcontracted to a statistical consulting firm. The decision may hinge upon how busy the local statistical group is with other projects that may delay the results of the analysis for the drug in question. Further, there may be more than one firm that is a likely candidate for performing the analysis. As a prerequisite for deciding whether to assign the *analysis* subproject to an external firm, you need to obtain a *bid* in the form of estimates of the cost and project duration from the competing firms as well as a corresponding estimate from the in-house team.

The cost corresponding to each possible subcontracting firm may be a combination of the actual costs (consulting fees and so on) and the tardiness of the project (tardiness being measured as the time difference between when the results are expected to be available and the target date for the availability of the results). The information required could be provided in terms of Gantt charts and cost analysis charts. Using this information, the project manager for the drug development project can use the principles of decision analysis to determine whether to do the analysis in-house or assign it to an outside consulting firm and to pick the firm to which the subcontract is to be assigned. Some of these ideas are illustrated in the following example.

Output 1.9.1. Input Data Sets for Decision Problem

Subcontracting Decision
The Stage Data Set

Obs	_STNAME_	_STTYPE_	_OUTCOM_	_REWARD_	_SUCCES_
1	Assignment	D	In_House	.	Complete
2			Consult1	-20,000	Act_Finish
3			Consult2	-17,500	Act_Finish
4	Complete	C	On_Time	.	Cost
5			Delay	-10,000	Cost
6	Act_Finish	C	Early	.	
7			Late	.	
8			Delay2	-1,000	
9	Cost	C	High	.	
10			Low	.	

Subcontracting Decision
The Probability Data Set

Obs	_GIVEN_	_EVENT_	_PROB_
1		High	0.50
2		Low	0.50
3		On_Time	0.60
4		Delay	0.40
5	Consult1	Early	0.60
6	Consult1	Late	0.35
7	Consult1	Delay2	0.05
8	Consult2	Early	0.50
9	Consult2	Late	0.40
10	Consult2	Delay2	0.10

Subcontracting Decision
The Payoffs Data Set

Obs	_STATE1	_STATE2	_VALUE_
1	On_Time	High	-12,000
2	On_Time	Low	-9,500
3	Delay	High	-15,000
4	Delay	Low	-11,500
5	Early		3,500
6	Late		1,500
7	Delay2		0

The stages of the decision problem are identified by the STAGEIN= data set, stage, displayed in Output 1.9.1. As a first step, the drug company needs to decide whether to perform the analysis in-house or to assign it to one of two consulting firms. If the in-house team is chosen, the resulting stage is a chance node, called 'Complete,' with two possible outcomes: 'On-Time' or 'Delay'; if there is a delay, the resulting cost to the drug company is $\$10,000$. For each of these two outcomes, there is a second chance event corresponding to the cost of the analysis. For each of the two consulting firms, the outcome can be one of three possibilities: 'Early,' 'Late,' or 'Delay2'; if there is a delay, the drug company imposes a delay penalty of $\$9,000$ on the firm, resulting in a net reward of $-\$1,000$ (penalty of $\$9,000$ minus the cost of $\$10,000$).

The PROBIN= data set, prob, identifies the various probabilities associated with the different possible outcomes at each of the chance events. The prob data set is also displayed in Output 1.9.1.

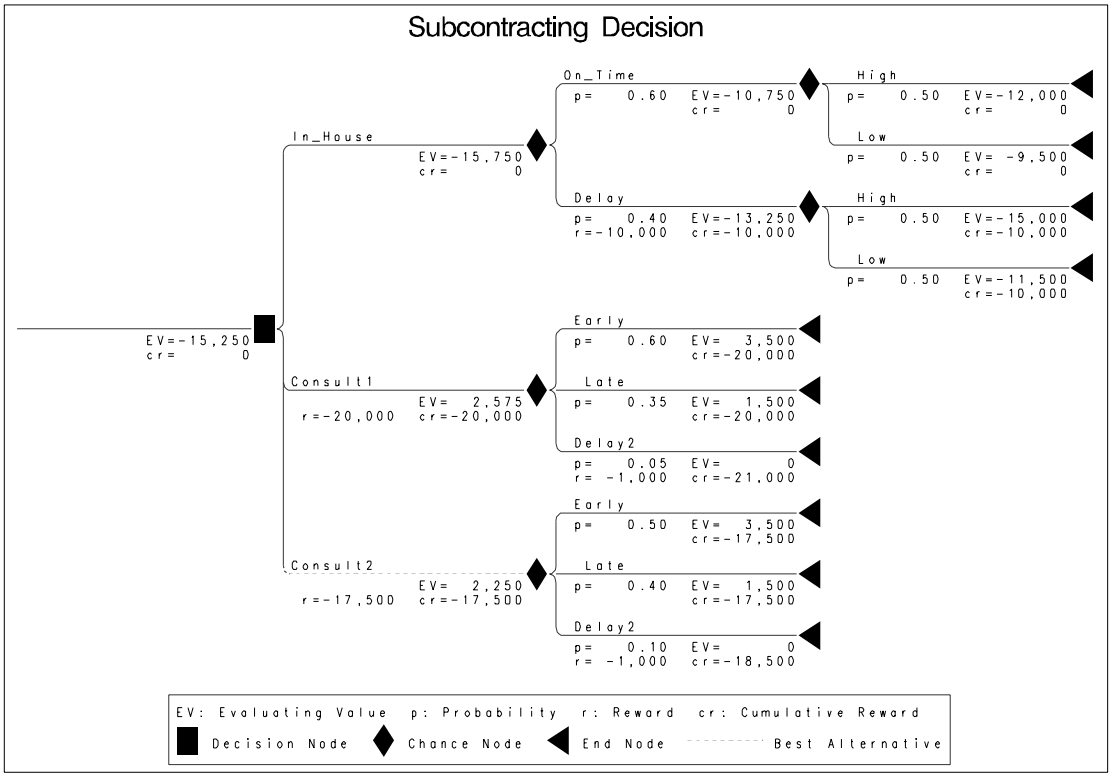
The rewards (or payoffs) associated with each of the end stages are listed in the PAYOFFS= data set, payoff (also listed in Output 1.9.1). For example, for the in-house team, the high (low) cost associated with completing the analysis on time is $\$12,000$ ($\$9,500$), and so on.

The following program invokes PROC DTREE to solve the decision problem. The complete decision tree, displayed in Output 1.9.2, represents the various stages and outcomes of the problem and identifies the optimal decision. In this example, the drug company should award the consulting contract to the second consulting firm as indicated by the dashed line for the corresponding branch of the tree.

See Chapter 3, "The DTREE Procedure," for details about the DTREE procedure.

```
title f=swiss "Subcontracting Decision";

goptions ftext=simplexu;
symbol1 f=marker v=P;
symbol2 f=marker v=U;
symbol3 f=marker v=A;

/* PROC DTREE statements */
proc dtree stagein=stage
           probin=prob
           payoffs=payoff
           nowarning;
   evaluate;
   treeplot / graphics
              compress ybetween=1 cell
              lwidth=1 lwidthb=2 lstyleb=20
              hsymbol=2 symbolc=1
              symbold=2 symbole=3;
   quit;
```

Output 1.9.2. Decision Analysis



Project Management Systems

As illustrated in the "Data Flow" section on page 17 and the "Examples" section on page 26, the procedures of SAS/OR software, when combined with the other parts of the SAS System, provide a rich environment for developing customized project management systems. Every company has its own set of requirements for how project data should be handled and for how costs should be accounted. The CPM, GANTT, NETDRAW, and PM procedures, together with the other reporting, summarizing, charting, and plotting procedures, are the basic building blocks that can be combined in several different ways to provide the exact structure that you need. The interactive PM procedure can be used as the primary editing interface for entering all activity information for your projects. Further, the application building tools in the SAS System can be used to cement the pieces together in a menu-driven application. You can create easy-to-use applications enabling the user to enter information continually and to obtain progress reports periodically.

The Projman Application

The Projman application is a user-friendly graphical user interface for performing project management with the SAS System. Through the use of an interactive Gantt chart window provided by the PM procedure, you can easily create and manage multiple projects.

Projman is accessed by invoking the **projman** command in the SAS windowing environment or by selecting **Solutions->Analysis->Project Management** from the primary SAS menu. Projman enables you to define multiple projects, information about which are stored in a project dictionary data set. This project dictionary provides a convenient way to manage all the data sets associated with each project.

Projman also provides a variety of project reports. These reports include Gantt charts, network diagrams, calendars, and tabular listings as well as resource usage and cost reports. You can modify these reports to add your own personalized reports to the application.

For details about the Projman application, see Chapter 7, "The Projman Application."

Web-Based Scheduling Systems

The examples in this chapter describe several scenarios that illustrate the different ways in which the project management procedures can be used to define, manage, and monitor projects. As described in the previous sections, the SAS System can be used to create comprehensive Decision Support systems or project management systems, in particular, using the procedures described in this book. With the availability of SAS/IntrNet software, you can also create Web-based project management or scheduling systems where the browser is used to display schedules and resource usage information that is updated using the CPM procedure's scheduling engine.

Examples of such Web-based applications are available at SAS Institute's external Web site at the following url: http://support.sas.com/sassamples/demos/supplychain/demos. In particular, the "Enterprise-Wide Resource Management" (EWRM) demo uses several of the ideas described in this chapter and illustrated in the examples throughout this book to create an application that schedules the tasks required for the maintenance of aircraft engines at a hypothetical service facility.

Note: The EWRM Web demo is a client-server application driven from your desktop and running at SAS Institute in Cary, NC. You can access the demo from SAS Institute's Supply Chain Web site (http://support.sas.com/sassamples/demos/supplychain/demos/ewrm/ewrm_index.html). The graphs and reports in the demo have not been saved, but are calculated on demand; this means that they change dynamically as the data used to calculate them change. This demo requires Internet Explorer, version 5.0 or later.

Microsoft Project Conversion Macros

MDBTOPM and MP2KTOPM are two SAS macros that convert Microsoft® Project data to a form that is readable by the PM procedure. MDBTOPM converts Microsoft Project 98 data, and MP2KTOPM converts Microsoft Project 2000 data. The macros generate the necessary SAS data sets, determine the values of the relevant options, and invoke an instance of the PM procedure with the converted project data. For details about the macros, see the "Microsoft Project Data Conversion" section on page 722 in Chapter 6, "The PM Procedure."

References

Cohen, M. (1990), "Decision Analysis in Project Management," *PMNetwork*, IV, 3, 37–40.

Fleming, Q. W. (1988), *Cost/Schedule Control Systems Criteria: The Management Guide to C/SCSC*, Chicago: Probus Publishing Company.

Moder, J. J., Phillips, C. R., and Davis, E. W. (1983), *Project Management with CPM, PERT and Precedence Diagramming*, New York: Van Nostrand Reinhold Company.

SAS Institute Inc. (1993), *SAS/OR Software: Project Management Examples*, Cary, NC: SAS Institute Inc.

Williams, G. A. and Boyd, W. L. (1990), "Decision Support Systems and Project Management," *PMNetwork*, IV, 3, 31–36.