Chapter 3: Project Estimating Techniques and Tools

Overview

The IT project manager must correctly estimate the following factors to achieve a successful, cost−effective IT project: cost, level of efforts, work breakdown structure (WBS), scheduling, staffing, milestones, budget, and IT system development and maintenance phases. Many techniques and tools assist the manager in such estimation. Each of the following tools is explained in detail in this chapter:

Cost−estimating tools:

- The constructive cost model (COCOMO) predicts the effort and duration of a project.
- The function points (FP) method measures the system development productivity.
- An earned value (EV) tool keeps a record of tasks and activities.

Scheduling techniques:

- The Gantt chart shows activities, their duration, and their interrelationships.
- The program evaluation and review technique (PERT) chart identifies the relationship of many steps involved in complex systems. PERT is a special application of network analysis for estimating and controlling the time required for activities and work packages about which little information is available from experience.
- A critical path involves activities through the critical path method (CPM) network. The CPM is a tool designed to reduce the length of time to complete a project.
- Microsoft Project 98 software is an automated tool that meets most of the IT manager's requirements in managing a project.

Cost−Estimating Tools

Constructive Cost Model

Barry Boehm developed the COCOMO in 1981 at the University of Southern California (USC). The COCOMO predicts the effort involved in and duration of a project. The COCOMO allows the project manager to estimate the cost, effort, and schedule when planning a new system software development activity according to practices that were commonly used in the 1970s and 1980s. It exists in three forms, each one offering detail and accuracy the further along one is in the project planning and design process. Listed by increasing fidelity, these forms are basic, intermediate, and detailed COCOMO. However, USC has only implemented the intermediate form in a calibrated software tool.

The implemented tool provides cost, effort, and schedule point estimates. It also allows a planner to easily perform 'what if' scenario exploration by quickly demonstrating the effect that adjustment of requirements, resources, and staffing may have on predicted costs and schedules (e.g., for risk management or job bidding purposes). The COCOMO needs the following parameters to compute the cost estimation of an IT project:

1. ***Three classes of COCOMO projects***

    a. ***Organic mode projects.*** These are projects in which relatively small teams are working in a familiar environment developing well−understood applications. Communication overhead is

low, team members know what they are doing, and they can quickly proceed with the job.

b. ***Semi−detached mode projects.*** This mode represents an intermediate stage between organic mode projects and embedded mode projects. In semi−detached mode projects, the project team may be made up of experienced and inexperienced members. Team members may have limited experience of related systems and may be unfamiliar with some but not all aspects of the system being developed.

c. ***Embedded mode projects.*** Embedded mode projects are concerned with developing a system that is part of a strongly coupled complex of hardware, software, regulations, and operational procedures. Requirements are modified to get around software problems that are usually impractical, and validation and costs are high. Because of the diverse nature of embedded mode projects, project team members often have little experience in the application being developed.

2. ***Projected length of the IT project software program.*** This length is represented in thousands of lines of code (KLOC). For example, 75 represents a program with 75 KLOC.
3. ***Subcycles.*** This number is the total number of subcycles. The maintenance subcycle is subcycle 1; therefore if the manager only wants the maintenance subcycle, he or she enters 1. Otherwise he or she enters the total number of subcycles, including maintenance. On the next level of input, the manager is prompted for more information about each subcycle.
4. ***Skill levels.*** This number is the total number of different skill levels of personnel working on the project. On the next level of input, the manager is prompted for more information about each skill level.
5. ***Manpower.*** For each skill level, the manager enters the number of people and the percentage of time that each skill level spends on each of the subcycles. For example, if five people from skill level 1 spend 50% of their time on subcycle 1, then the entries for subcycle 1 are 5 and 50.
6. ***Personnel information.*** For each skill level, the manager enters the job title. For the salary field the manager enters the average annual salary for the skill level. The percent raise field represents the percentage that the skill level gets for a raise. As with other fields, it is in whole values (e.g., 12% = 12). The raise cycle is the number of months between each raise.
7. ***Subcycle information.*** The manager enters the subcycle name and the percentage of time (in whole value) of the entire software development life cycle that is spent on this subcycle. All of the subcycle percentages must add up to 100%. The manager enters the overhead cost for this subcycle not associated with salary expense.
8. ***Maintenance information.*** The subcycle name is automatically entered. The manager enters the duration in months and the overhead costs not related to salary expense.
9. ***Error message.*** Valid values must be input for all fields. Title fields (e.g., job title, subcycle name) can be made up of any ASCII characters; all other fields must be numeric values.
10. ***Report output***

    a. ***Total cost***. The total cost reflects the overhead costs and the personnel (salary) costs for all of the subcycles, including the maintenance cycle.

    b. ***Total duration.*** The total cost reflects the total amount of time (in months) needed to complete the project, including all of the subcycles and the maintenance cycle.

    c. ***Subcycle cost.*** This value is the amount of money spent on employee salaries and overhead for a given subcycle.

    d. ***Subcycle duration***. The duration of a subcycle is the amount of time (in months) spent working on a given subcycle.

    A simplified version of COCOMO is as follows:

    MM = C (KDSI)**λ

MM = man−month, C = constant, K = thousand, DSI = delivered source instructions

The intermediate model of COCOMO consists of factors that affect the IT project cost and are called ***cost drivers***. The following are some of these factors:

- Hardware compatibility
- The availability of computer−aided software engineering (CASE) and modern tools
- Staffing (domain experiences and training)
- IT project environment
- Management planning

The detailed model of COCOMO consists of the phases of software development, including all factors of the intermediate model. The phases depend on the standard and methodology to be used in the software development and maintenance life cycle. The major phases are stated as follows:

- System requirement analysis and design
- Software requirement analysis
- Software design
- Coding and unit testing
- Integration and testing
- System integration and testing

The manager estimates and applies the cost drivers to each phase separately. Ideally, the COCOMO estimated effort, MM(est), should match with the actual effort, MM(act), but this rarely happens. There is always a degree of error in computing practical applications. The percentage of error can be calculated using the following equation:

Percentage of error = [MM(est)−MM(act)]/MM(act)

The following are some of the effects of underestimating on a project (i.e., MM[est] is less than MM[act]):

- Understaffing
- Slipping schedules and milestones
- Rising costs
- Low morale

However, the following are some of the results of overestimating (i.e., MM[est] is greater than MM[act]):

- High costs
- Low productivity
- Professionals picking up 'slack habits'

The suggested solution is to compute a magnitude of relative error (MRE), which provides an absolute value:

$$MRE = \frac{MM(est) - MM(act)}{MM(act)}$$

A perfect score on the MRE test is an error percentage of zero.

COCOMO Example

A project is estimated to be equivalent in an effort to develop 33.2 KLOC of high−level language code. Let the project be of type organic (i.e., a simple application), and let all adjustment factors be nominal (i.e., the cost drivers or attributes are 'typical' of the environment for which the model was derived). Then use basic COCOMO as follows (COCOMO does not account for the concept phase costs and for early requirements, or about 6% to 10% of the front−end cost):

Effort = $2.4(\text{KLOC})^{**1.05} = 2.4(33.2)^{1.05} = 95$ person−months

Duration = $2.5(\text{Effort})^{**0.38} = 2.5(95)^{0.38} = 14.1$ months

Full−time personnel = Effort/Duration = 95 person−months/14.1 months = 7 persons

The planner uses customized cost−drivers (weights) and financial and other constraints.

COCOMO II

COCOMO II is currently being developed to include modern approaches such as reuse, commercial items, object−oriented methods, and component−based system software development. COCOMO II is a model that allows the planner to estimate the cost, effort, and schedule when planning a new system software development activity. It consists of three submodels, each of which offers increased fidelity the further along one is in the project planning and design process. Listed in increasing fidelity, these submodels are the application composition, early design, and postarchitecture models. Until recently, only postarchitecture, the most detailed submodel, had been implemented in a calibrated software tool.

The new model (composed of all three submodels) was initially given the name COCOMO 2.0. However, after some confusion in how to designate subsequent releases of the software implementation of the new model, the name was permanently changed to COCOMO II. To further avoid confusion, the original COCOMO model was also redesignated COCOMO 81. All references to COCOMO found in books and literature published before 1995 refer to what is now called COCOMO 81. Most references to COCOMO published from 1995 onward refer to what is now called COCOMO II.

If the planner, while examining a reference, is still unsure as to which model is being discussed, a few clues are apparent:

- If the terms−basic, intermediate, or detailed (for model names) and organic, semidetached, or embedded (for development models)−are used in the context of discussing COCOMO, then the model being discussed is COCOMO 81.
- If the model names mentioned are application composition, early design, or postarchitecture, or if there is mention of scale factors, precedentedness, development flexibility, architecture/risk resolution, team cohesion, or process maturity, then the model being discussed is COCOMO II.

The major features of COCOMO II are as follows:

- Model structure, which consists of three models, assumes progress through a spiral−type development to solidify requirements, to solidify the architecture, and to reduce risk.

Exponent = Variable established based on rating of the following five scale factors:

PREC = Precedentedness

FLEX = Development flexibility

RESL = Architecture or risk resolution

TEAM = Team cohesion

PMAT = Process maturity

- Size of object points, FPs, or source lines of code (SLOC)
- Cost drivers (the following 17 drivers must be rated):

RELY = Reliability

DATA = Database size

CPLX = Complexity

RUSE = Required reusability

DOCU = Documentation

TIME = Execution time constant

STOR = Main storage constant

PVOL = Platform volatility

ACAP = Analyst capability

PCAP = Programmer capability

AEXP = Application experience

PEXP = Platform experience

LTEX = Language and tool experience

PCON = Personnel continuity

TOOL = Use of software tools

SITE = Multisite development

SCED = Required schedule

Function Points Method

Allan Albrecht developed the FP method to measure IT project system software development productivity. The FP method is an alternative to estimating the LOC. The following are advantages of FP estimation:

- Helps the planner estimate LOC early in the life cycle
- Helps the planner estimate level of effort (LOE) easily

The steps involved in counting FPs are as follows:

- Count the user functions.

  External input types

  External output types

  User interaction

  Logical internal file types

  External interface file types

  External inquiry types
- Adjust for processing complexity.

The effort required to provide a given level of functionality varies depending on the environment. Once the planner computes the FPs, he or she can use them to compare the proposed IT project with past IT projects in terms of size.

Albrecht identifies a list of 14 processing complexity characteristics that are rated on a scale of 0 to 5, ranging from no influence to strong influence. The next step is to sum all the processing complexity points assigned. The number (n) is then multiplied by 0.01 and added to 0.65 to obtain weight:

PCA = 0.65 + 0.01(n)

where PCA = processing complexity adjustment (varies between 0.65 and 1.35).

Complexity factors vary between 0 and 5. This factor is then used in the final equation:

FP = FC(PCA)

where FC = function counts (previously computed).

The result is that the FPs can vary +/235% from the original FCs, which are modified by complexity of the IT project. The FPs can be used to estimate LOC depending on the average number of LOC per FP for a given language. The FPs are subjective (i.e., they depend on the estimator). They cannot be counted automatically.

COCOMO and the Function Points Method

Two misconceptions about COCOMO, SLOC, and FPs are as follows:

1. COCOMO does not support the use of FPs. FP versions of COCOMO have been available since 1987. COCOMO II supports the use of either FPs or SLOC. In both cases, this is done via 'backfiring' tables of SLOC per FP for source languages at different levels.
2. It is irresponsible to use SLOC as a general productivity metric, but it is not irresponsible to use FP as a general sizing parameter for estimation. This breaks down into two cases:

   a. If an organization uses different language levels to develop software, it is irresponsible to use SLOC as a productivity metric because higher productivity and SLOC are present at higher language levels. However, it is also irresponsible to use FP as a general sizing metric for

estimation because pure FP will give the same cost, schedule, or quality estimate for a program with the same functionality developed using different language levels, which is clearly wrong. To get responsible results in this case, FP−based estimation models need to use some form of backfiring to account for the difference in language level.

b. If an organization always uses the same programming language (level), it is responsible to use pure FP as a sizing metric for estimation. However, it is also responsible to use SLOC as a productivity metric (sunset.usc.edu).

Earned Value Tool

EV is an objective measurement of how much work has been accomplished on an IT project. *EV, performance measurement, management by objectives,* and *cost schedule control systems* are synonymous terms. EV improves on the 'normally used' spending plan concept (budget versus actual incurred cost) by requiring the work−in−progress to be quantified. Using the EV process, the IT project manager can readily compare how much work has actually been completed against the amount of work planned to be accomplished. EV requires that the project managers plan, budget, and schedule the authorized work scope in a time−phased plan. The time−phased plan is the incremental 'planned value' culminating into a performance measurement baseline. As work is accomplished, it is 'earned' using the same selected budget term. EV compared with planned value provides a work accomplished against a plan. A variance to the plan is noted as a schedule or cost deviation.

An EV method helps the project manager keep a record of tasks and activities plotted by sequence in time against actual versus planned completion. EV is computed as follows:

EV = ATD/EAC

ATD = actual to date and EAC = estimate at completion.

This represents the percentage of completion of a project in terms of effort accomplished as a portion of the total effort required.

Normally the established accounting system provides accumulation of actual cost for the project. The actual cost is compared with the EV to the indicated overrun or underrun condition. Planned value, EV, and actual cost data provide an objective measurement of performance, enabling trend analysis and evaluation of cost estimates at completion within multiple levels of the project.

An IT project manager should apply to every project where the owners of the final product wish to ensure that the expended resources were used efficiently. On major projects the application of good project management tools will aid in the selection of the right course when managers need to make financial and time allocation decisions.

Earned Value Management

Earned value management (EVM) is a management technique that relates resource planning to schedules and technical performance requirements. All work is planned, budgeted, and scheduled in time−phased planned value increments, constituting a cost and schedule measurement baseline. EVM is a valuable tool for identifying performance trends and variances from the management plan.

An EVM system has two major objectives:

- To encourage contractors (IT system developers) to use effective internal cost and schedule management control systems

- To permit the customer (IT managers) to be able to rely on timely data produced by those systems for determining product−oriented contract status

The baseline plan in Table 3−1 shows that 6 work units (A−F) would be completed at a cost of $100 for the period covered by this report.

Table 3−1: Baseline Plan Work Units

	Planned value ($)
A	10
B	15
C	10
D	25
E	20
F	20
Total	**100**

Schedule Variance

As work is performed, it is 'earned' on the same basis as it was planned, in dollars or other quantifiable units such as labor hours. Planned value compared with EV measures the dollar volume of work planned versus the equivalent dollar volume of work accomplished. Any difference is called a *schedule variance*. In contrast to what was planned, Table 3−2 shows that work unit D was not completed and work unit F was never started, or $35 of the planned work was not accomplished. As a result, the schedule variance shows that 35% of the work planned for this period was not done.

Table 3−2: Schedule Variance Work Units

	Planned value ($)	Earned value ($)	Schedule variance
A	10	10	0
B	15	15	0
C	10	10	0
D	25	10	−15
E	20	20	0
F	20	−	20
Total		**100**	**65− 35 = − 35%**

Cost Variance

EV compared with the actual cost incurred (from contractor accounting systems) for the work performed provides an objective measure of planned and actual cost. Any difference is called a cost variance. A negative variance means that more money was spent for the work accomplished than was planned. Table 3−3 shows the calculation of cost variance. The work performed was planned to cost $65 and actually cost $91. The cost variance is 40%.

Table 3−3: Cost Variance Work Units

	Earned value ($)	Actual cost ($)	Cost variance
A	10	9	0
B	15	22	0
C	10	8	0
D	10	30	−15
E	20	22	0
F	–	–	−20
Total	**65**	**91**	**−26 = − 240%**

Spend Comparison

The typical spend comparison approach, whereby contractors report actual expenditures against planned expenditures, is not related to the work that was accomplished. Table 3−4 shows a simple comparison of planned and actual spending, which is unrelated to work performed and therefore not a useful comparison. The fact that the total amount spent was $9 less than planned for this period is not useful without the comparisons with work accomplished.

Table 3−4: Spend Comparison Approach Work Units

	Planned spend ($)	Actual spend ($)	Variance
A	10	9	1
B	15	22	− 7
C	10	8	2
D	24	30	− 5
E	20	22	−2
F	20	–	20
Total	**100**	**91**	**9 = 9%**

Use of Earned Value Data

The benefits to project management of the EVM approach come from the disciplined planning conducted and the availability of metrics, which show real variances from the plan to generate necessary corrective actions (evms.dcmdw.dla.mil/index.htm).

Other Cost Estimation Tools

Other cost estimation tools are available in the industry. Some of these, like SECOMO and REVIC, are specialized toward unique methodology and language. ESTIMATICS is a proprietary product of Howard Rubin, and SLIM is a proprietary product of Larry Putnam.

Scheduling Techniques

The best known tools for scheduling and timing projects are the Gantt chart, the PERT chart, and the critical path method (CPM).

Gantt Chart

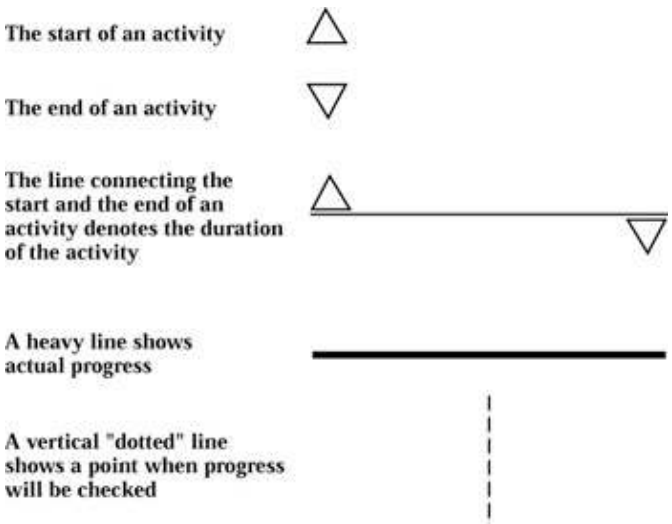
The Gantt chart, created around 1900 by Henry L. Gantt, is often used for scheduling. The Gantt chart shows activities, their duration, and the interrelationships among them. It is commonly known as the bar chart. The following are advantages of using the Gantt chart:

- It helps the manager plan well.
- It is easy to produce.
- The manager can compare work planned versus work accomplished at a glance.
- It is easy to understand.
- It provides a large amount of information on a single piece of paper.

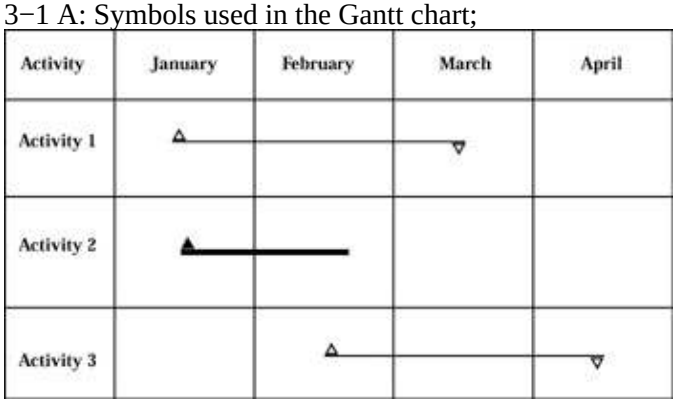
The following are disadvantages of using the Gantt chart:

- It is difficult to depict interrelationships.
- It is weak in forecasting.
- It is inflexible.

The symbols used in the Gantt chart are shown in Figure 3−1, *A*, and a sample Gantt chart is shown in Figure 3−1, *B*.



3−1 A: Symbols used in the Gantt chart;



3−1 B: Gantt chart showing project planning versus actual activity

PERT Chart

A PERT chart identifies the relationships among the many steps involved in complex projects. PERT was developed in 1959 in connection with the Polaris weapon system. The PERT chart is a special application of network analysis for estimating and controlling the time required for activities and work packages about which little information from experience is available.

A PERT network analysis consists of the following activities:

- Specifying the tasks to be completed
- Sequencing and interrelating work packages
- Setting up the network
- Scheduling

In PERT charts the events are connected by activities. Since events are end items (resulting products), they take no time, money, or resources themselves. However, activities require time, money, and resources. A sample PERT chart is shown in Figure 3−2.

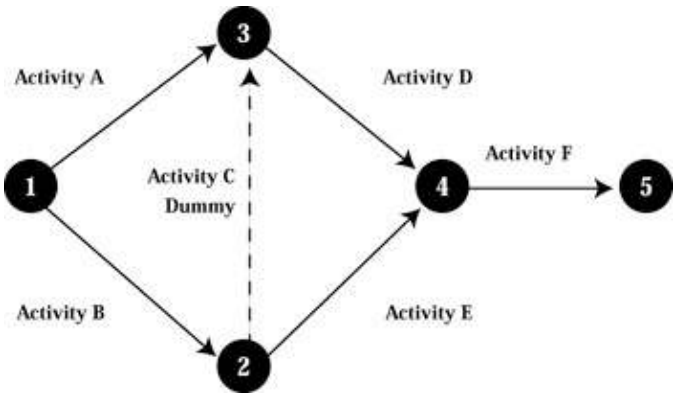


Figure 3−2: A typical PERT network showing activities that must be performed and their sequence. Note that the dummy activity has no duration.

The following are guidelines for drawing a PERT chart:

1. A person familiar with and committed to the project's objectives and requirements should develop the chart.
2. The developer starts with the ultimate targeted objective and proceeds backward to the beginning event.
3. The developer decides what activities must be completed before an event is completed.
4. An activity cannot begin until the event or end item preceding it has been completed.
5. The developer sets up two or more items and associated activities that can be accomplished concurrently in parallel paths.
6. The developer identifies a critical path to discover the longest cumulative time needed to accomplish end items and their associated activities.
7. The developer identifies a slack path to discover the shortest cumulative time needed to accomplish end items and their associated activities.

The following is a description of some of the conventions accepted by those who develop PERT charts:

- An activity is a time−consuming effort required to complete the project. It is represented by an arrow drawn from left to right. The arrows are not drawn to a time scale. Numbers above the arrows as shown in Figure 3−3, *A* indicate passage of time. By convention, an activity is designed by the node

numbers at the beginning and end of the activity.

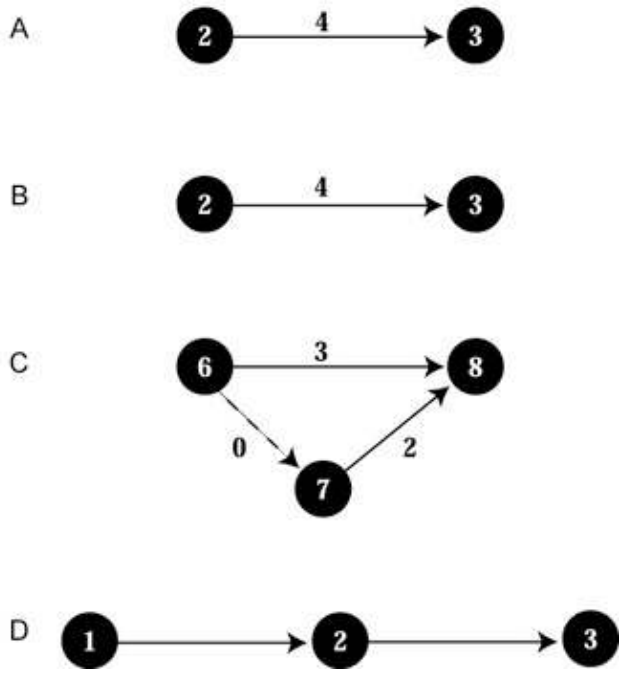


Figure 3−3: **A**, Activity; **B**, Event; **C**, Dummy; **D**, Restrictions

- An event is a particular instant in time that marks the beginning or end of an activity. Events do not require resources or time. A circle, called a *node*, containing a number, as shown in Figure 3−3, *B*, represents an event. An event is considered accomplished only after all the activities leading to it have been completed.
- A dummy is used to represent a restraint relationship that requires no time. It is indicated by a dotted line from the prerequisite activity to the beginning of the restricted activity, as shown in Figure 3−3, *C*. It is treated as an activity with zero time.
- A restriction is a prerequisite relationship that establishes the sequence of activities as illustrated in Figure 3−3, *D*. Activity 1−2 is prerequisite to activity 2−3. A prerequisite activity immediately precedes the activity being considered. Activity 2−3 is postrequisite to activity 1−2. A postrequisite activity immediately follows the activity being considered. Figure 3−4, *A*, shows that activities 2−5, 5−7, 2−4, 4−6, 3−7, and 6−7 are descendants of activity 1−2. A descendent is an activity restricted by the activity being considered. In Figure 3−4, *B*, activities 1−2 and 2−5 are antecedents of activity 5−7. An antecedent is an activity that must precede the activity being considered.

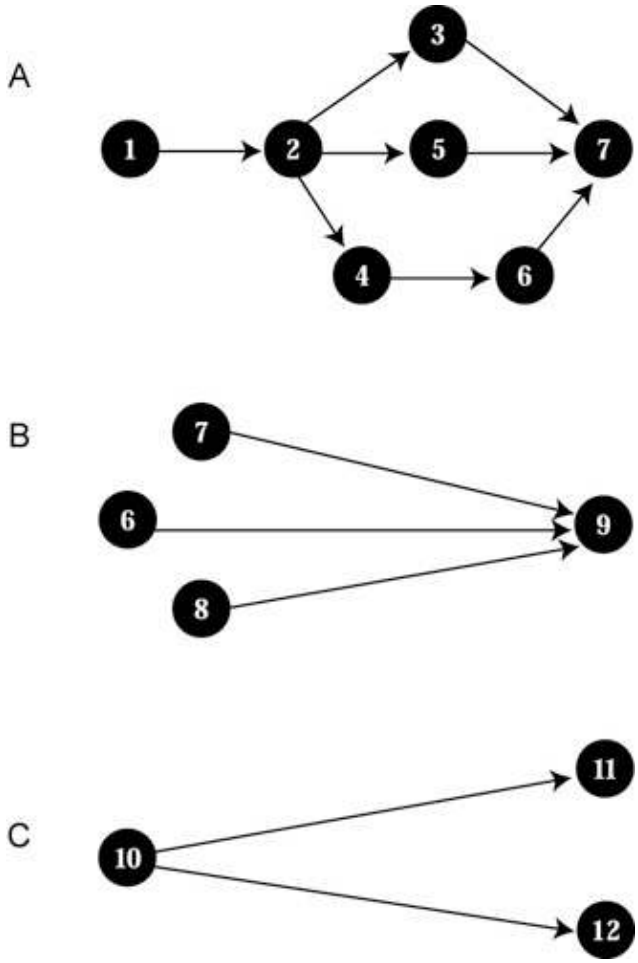


Figure 3−4: **A**, Another example of restrictions; **B**, Merge; **C**, Burst

- A merge exists when two or more activities converge at a single event. All activities merging at an event restrict all activities beginning at that event as shown in Figure 3−4, *C*. The event is not considered to occur until all the activities merging at the event have occurred.
- A burst exists when two or more activities begin at a single event as shown in Figure 3−4, *C*. All activities have a common prerequisite. Merges and bursts occur when concurrency exists.
- An arrow network is a graphic representation of the activities and events needed to complete a project. The network shows the logical relationships existing among the various activities as shown in Figure 3−5. Conventionally, networks are drawn from left to right. No two activities may have the same starting and ending events; use of dummies eliminates the problem. It is customary to number the events in such a way that the event at the right of the activity arrow has a higher number than the event at the left of the arrow. The lengths of the activity arrows have no meaning and may vary to provide clarity to the network. Crossovers are permitted but should be avoided wherever possible. Loops are not permitted (i.e., no path subsequent to an event may lead to a prior event).

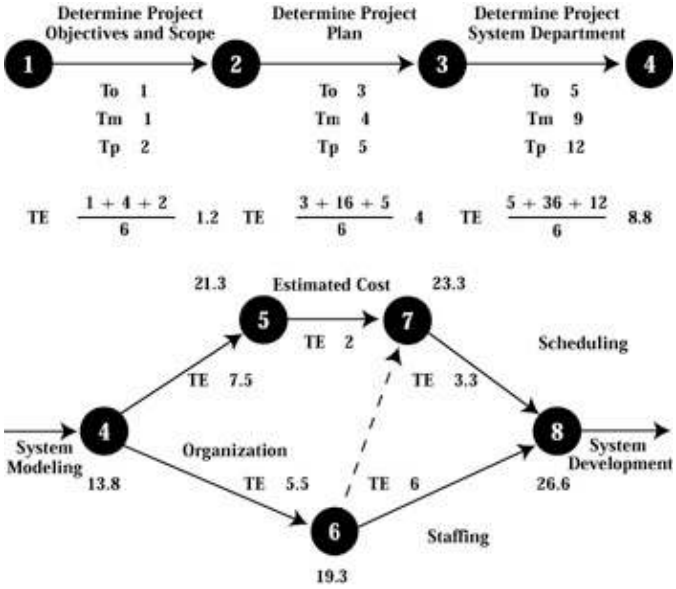


Figure 3−5: A PERT network

- For PERT activities, time expected equals three times the average estimate. Estimates are required for all the activities of the network to estimate the total time:

$$\text{Time expected (TE)} = \frac{To + 4Tm + Tp}{6}$$

To = optimistic time (an estimate of the minimum time an activity will take if unusually favorable conditions are experienced), Tm = most likely time (an estimate of the normal time an activity would take if it was repeated an indefinite number of times under identical conditions), and Tp = pessimistic time (an estimate of the maximum time an activity will take if unusually unfavorable conditions are experienced).

- The manager finally calculates an array of estimates regarding different durations for the project.

The following are advantages of using PERT charts:

- The use of the three time estimates makes it more valid than any other technique.
- It is easy to calculate the effect of delays.
- PERT charts are strong in the system development and maintenance phases of scheduling.
- The manager can see the activities that must be completed before end objectives can be met.
- They are good for simulating the influence of various resource allocations on the schedule.
- They are strong in forecasting whether the manager will accomplish future events on schedule.
- They predict slippage and their effects early enough for the manager to take remedial action.
- Activities are clearly identified, and elapsed times can be obtained as needed.
- It is easy to adjust time estimates when changes occur.
- It is easy to plan and schedule work.

The following are disadvantages of using PERT charts:

- They are complex.
- Estimation of activity times is time consuming.
- They are expensive.

Critical Path Method

The critical path is the longest path through the CPM networks. Morgan R. Walker of Dupont and James S. Kelly of Remington Rand developed the CPM in 1957. The CPM predicts the duration of the project. When a project is delayed, this is the critical path through which to go. All other paths are called *noncritical paths, slack paths,* or ***float paths.*** These paths are the difference in duration between two nodes.

Nodes are the places where two or more activities are starting or ending (see circle 3 in Figure 3−2). The activity is a distinct part of a project. In Figure 3−2 the project has six activities:

1. Activity A: 1 to 3
2. Activity B: 1 to 2
3. Activity C: 2 to 3
4. Activity D: 3 to 4
5. Activity E: 2 to 4
6. Activity F: 4 to 5

An event is a specific point in time and is shown by a circle. This is the start or completion of an activity. A dummy activity is an activity that has no duration. It is used only to show the precedence of activities and is shown as a dotted line. In Figure 3−2, activity C from 2 to 3 is a dummy activity.

The following are advantages of using CPM:

- It is good for single−shot activities.
- It is excellent for simulating alternatives plans.
- It forecasts strongly so that future events can be accomplished on schedule.
- It allows for clear identification of activities.
- It allows for easy estimation of time.
- It is easy to change the graphics to reflect program changes.

The following are disadvantages of using CPM:

- It does not provide a formula to estimate time to completion.
- It is easily prone to error.
- It is weak in the system software development phases.
- It is not good for scheduling.
- It is a complicated tool for planning and status reporting.

Case Study 3−1

The Modern Project Management Corporation (MPMC) decided to reengineer an existing IT system. They need new hardware, including laptop and desktop computers, networking, and Internet access. They also need the necessary software. You have been selected as manager to plan this project. The following are suggestions on how to begin:

1. Compile the data in tabular form as shown in Table 3−5.

Table 3−5: Case Study Date

Activity	Predecessor	Successor	Duration (days)
A. Plan project	None	B, C, D	5
B. Acquire hardware	A	E	10
C. Select location	A	E	6
D. Determine needed amenities			
♦ Electricity ♦ Telephone ♦ Workstation ♦ Software ♦ CASE tools ♦ Office furniture	A	F	7
E.Install hardware	B, C	F	4
F.Provide amenities	D, E	G	2
G.Start functional			
♦ Early start (ES) ♦ Late start (LS) ♦ Early finish (EF) ♦ Late finish (LF)	F	None	2

2. Draw the CPM network and determine the early start (ES) and early finish (EF) for each activity, moving from the start to the end of the project, as shown in Figure 3−6. Place the number zero in the ES position of the first activity. The EF of the activity is equal to its ES plus the duration of the activity. Place in the ES position of each successor activity the EF of its predecessor. If an activity has two or more predecessors, use the largest EF. Continue until the EF of the last activity of the project has been calculated.

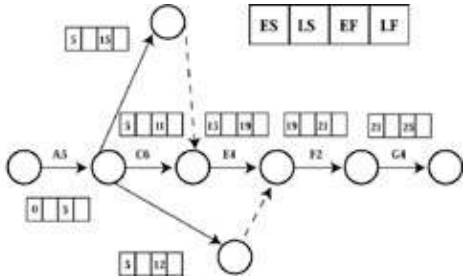


Figure 3−6: Project activities

3. Determine the late start (LS) and late finish (LF) of each activity, moving from the end to the beginning of the project, as shown in Figure 3−6. Place in the LF position of the last activity the number that is in its EF position. The LS of the activity is equal to its LF minus the duration of the activity. Place in the LF position of each predecessor activity the LS number of its successor. If an activity has two or more successors, use the smallest LS. Continue this process until the LS of the first activity of the project is determined.
4. Determine the LS and LF of each activity, moving from the end to the beginning of the project, as shown in Figure 3−7. Place in the LF position of the last activity the number that is in its EF position. The LS of the activity is equal to its LF minus the duration of the activity. Place in the LF position of each predecessor activity the LS number of its successor. If an activity has two or more successors, use the smallest LS. Continue this process until the LS of the first activity of the project is determined.

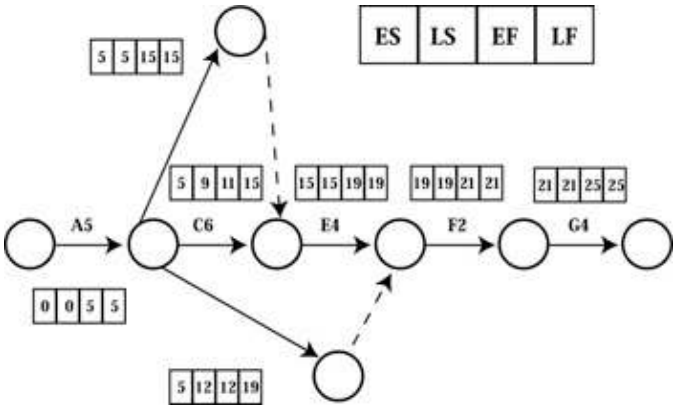


Figure 3−7: Project activities CPM

5. Determine the longest path through the CPM network, or the critical path: ABEFG = 25 days
6. Draw the Gantt chart for the project as shown in Figure 3−8.

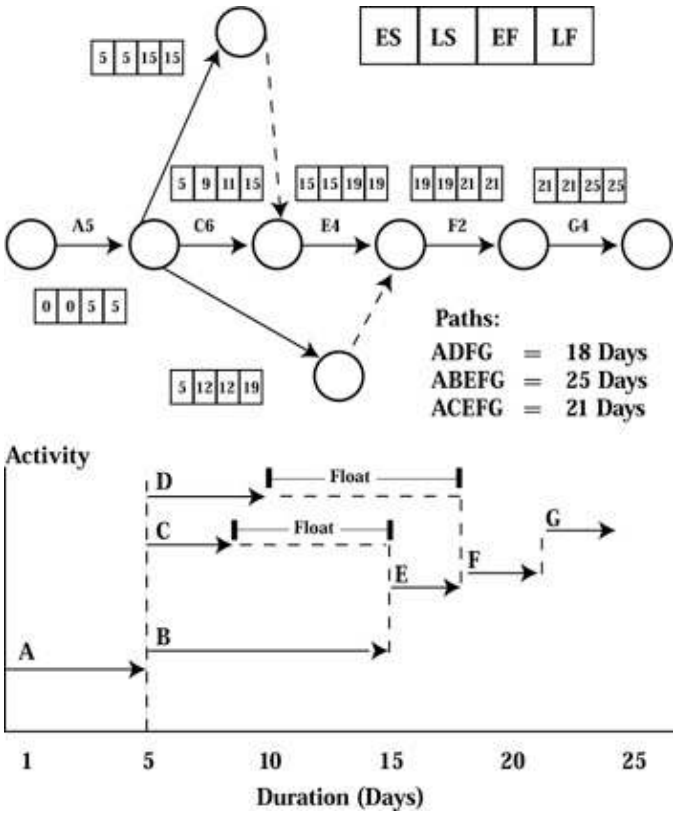


Figure 3−8: Project activities and Gantt chart