A Note on Project Scheduling

July 24, 2003

1. Introduction

The management of large projects requires analytical tools for scheduling activities and allocating resources. This note describes a set of tools that has proven to be consistently valuable to project managers. The tools are collectively known as the Project Evaluation and Review Technique (PERT) and the Critical Path Method (CPM). PERT was developed by the U.S. Navy and its consultants for the Polaris Missile Project, while the Critical Path Method was created by DuPont and the Remington Rand Corporation for the management of large chemical plants. Applications of these tools are pervasive, from construction to software development.

This note describes the basic concepts and calculations for project scheduling with PERT/CPM. These include the construction of network diagrams, the calculation of feasible project schedules, determining the effect of uncertainty on project schedules, and adjusting schedules to conform to time and resource constraints. The tools are important for planning a project and for keeping it on track once it has begun.

Throughout this note we will refer to a particular project: the installation of an information system at a major commercial bank, InterTrust Bank. Once in place, the system will collect accounting entries generated by banking products such as corporate accounts and loans. The bank's present system is operated by a contractor, and once the new system is up and running it will save contracting costs of $3,000 per week.

Table 1 below describes the project's activities and expected durations. The system will be developed and installed by computer programmers, systems analysts, and personnel from the accounting function. The bank has sufficient programmers and accountants on staff, but the systems analysts who mediate between the 'techies' and the accountants are in short supply. Currently, only three systems analysts are available to the firm, and the number of analysts needed for each activity is listed in the table. The table also specifies *immediate predecessors*, the smallest possible list of tasks which must be completed before starting each activity.

Prepared by Robert A. Shumsky, Associate Professor of Operations Management, William E. Simon Graduate School of Business Administration. Copyright © 2003.

	Description	immediate predecessors	duration (weeks)	systems analysts
A	Specify functional and user interface features	-	4	2
B	Design and code functional component	A	4	2
C	Test and debug functional component	B	4	2
D	Internal audit of functional test	C	2	1
E	Design and code graphical user interface	A	6	1
F	Integrate functional component and interface	C,E	6	2
G	Train accounting personnel on interface	E	6	1
H	Train personnel on testbed using integrated system	F,G	4	2

Table 1: Description of InterTrust Information Systems Project

Begin by assuming that the project durations listed in Table 1 are guaranteed, so that they are not subject to randomness. In Section 5 we will consider the effect of random activity times on the project.

2. Network Diagrams and Critical Paths

The information in the table may also be represented by a network diagram in which rectangles, or *nodes*, signify activities. The relationships between activities are represented by arrows between the nodes. For the InterTrust project, activities B and E may begin only after activity A has ended, a relationship represented by:

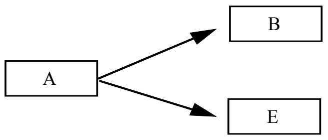

Figure 1

All arrows to a node begin at the node's immediate predecessors, indicating that the activity cannot be started until all activities prior to that node in the network are completed. Figure 2 displays the complete project network for InterTrust. Activity A must be completed before activities B and E are started. Activity F cannot be started until both activities C and E are completed.

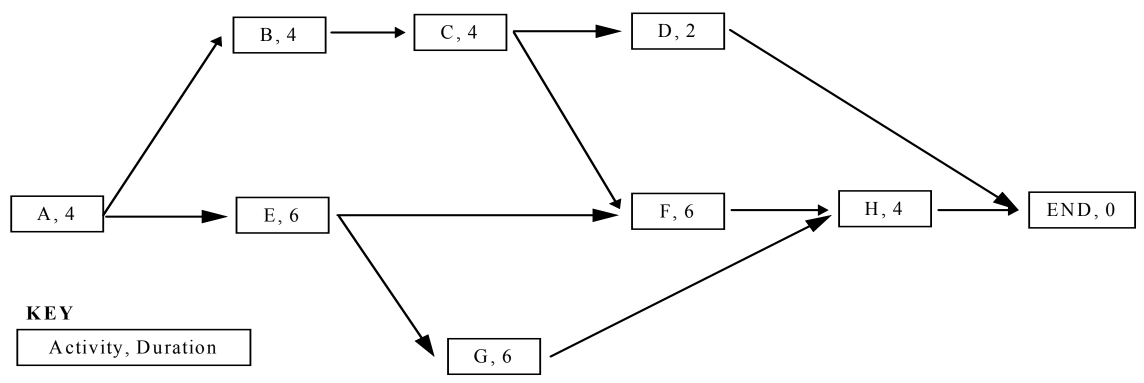

Figure 2: Project Network for InterTrust Information System Project

Since the project itself cannot be completed until all activities are complete, we have created a ‘dummy’ node, END, with immediate predecessors D and H. One might also create a dummy BEGIN node if multiple activities could start the project in parallel.

A project manager often begins with the simplest question: how long will the project last? Once the project network is drawn, the answer is also simple: the duration of the project is equal to the longest path from the beginning to the end of the network. Since all activities along this path must be completed, the duration of the project must be at least the length of the longest path. Since all other paths are shorter, the duration of the project must be the length of the longest path. For InterTrust, the longest path from ‘A’ to ‘END’ may be found easily by trying all paths (how many are there?) and choosing the longest. This path is ‘A-B-C-F-H-END’ and its duration is 4+4+4+6+4 = 22 weeks.

For this relatively simple project, there were only a few paths to compare. For larger projects with thousands of activities, finding the longest paths is difficult unless a structured method is used. The method described here is one such structured method, and along the way it derives much useful information besides the length of the project.

2.1 Earliest and Latest Start and Finish Times

For each activity we will calculate the following:

Quantity	Abbreviation	Description
Earliest Start Time	ES	Earliest time the activity may begin after allowing preceding activities to finish
Earliest Finish Time	EF	Earliest time the activity may finish after allowing preceding activities to finish

Latest Start Time	LS	Latest time the activity may begin without delaying project completion
Latest Finish Time	LF	Latest time the activity may finish without delaying project completion
Activity Slack	SLACK	Extra time an activity is allowed before it delays the project (assuming that it begins at ES). Under this definition, “slack” is sometimes also called “total slack” or “total float.”

In a network, earliest start and finish times are found by repeatedly calculating from the beginning of the project (node A) until the end (END). Use the following equations:

$$\boxed{\begin{array}{l}\text{Earliest Start time} = \text{ES} = \max[\text{EF of immediate predecessors}] \\ \text{Earliest Finish time} = \text{EF} = \text{ES} + \text{activity duration}\end{array}}$$

For InterTrust, begin with activity A at week ‘zero’. The earliest finish time for A is four weeks later, and this is the earliest start times for activities B and E. The earliest finish time for B is 4 + 4 = 8 weeks, and the earliest start time for its successor, C, is 8 weeks. Figure 3 displays the results of these calculations. One must be careful with activity F since it has two predecessors. The earliest start time for F is the later of the earliest finish times of its predecessors:

$$\begin{aligned}\text{ES for activity F} &= \max(\text{EF for activity C, EF for activity E}) \\ &= \max(12, 10) \\ &= 12 \text{ weeks}\end{aligned}$$

Similar calculations are completed for remaining activities until we reach END. We find that the project will take 22 weeks.
Latest start and finish times for each activity are found by working backwards, from the end of the project to the beginning:

$$\boxed{\begin{array}{l}\text{Latest Finish time} = \text{LF} = \min[\text{LS of immediate successors}] \\ \text{Latest Start time} = \text{LS} = \text{LF} - \text{activity duration}\end{array}}$$

For END, latest start and finish times are set equal to the earliest start and finish times since any delay to END will delay project completion. The latest finish time for activity H is the latest start time for END, 22 weeks, and the latest start time for activity H is 22 - 4 = 18 weeks. The latest finish time for D is also 22 weeks, while the LF for F is the LS for H, 18 weeks. Activity C has two immediate successors, so:

$$\begin{aligned}\text{LF for activity C} &= \min(\text{LS for activity D, LS for activity F}) \\ &= \min(20, 12) \\ &= 12 \text{ weeks}\end{aligned}$$

These calculations are repeated until latest times for activity A are found. See Figure 3 for the results.

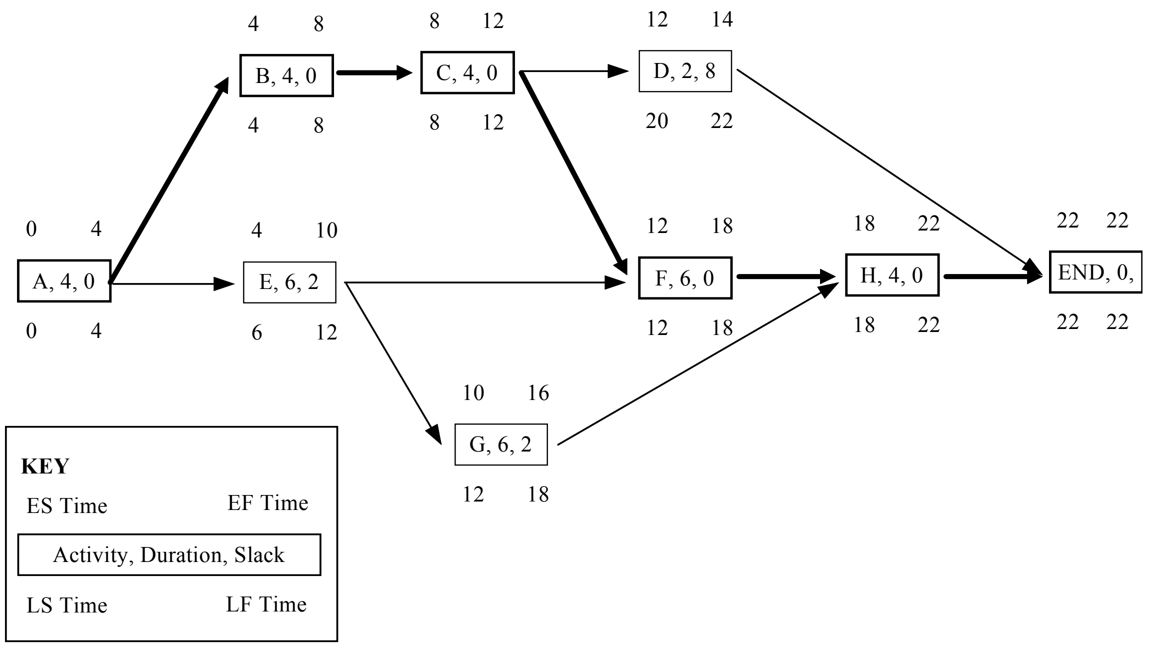

Figure 3: Start and Finish Times and Critical Path for InterTrust

2.2 Activity Slack, Critical Activities, and Critical Paths

The activity slack for each node may be easily calculated:

$$\boxed{\text{Activity Slack} = \text{SLACK} = \text{LS} - \text{ES} = \text{LF} - \text{EF}}$$

In any network, there will be activities with zero slack. Any delay to these activities will produce a delay in the completion of the project as a whole. We call these tasks *critical activities*, and a path through the network made up of critical activities is called a *critical path*. There will always be at least one critical path, and there may be more than one. All critical paths have the same length. Not surprisingly, critical paths are the longest paths through the network and the length of a critical path is equal to the duration of the project. In Figure 3, critical activities and the critical path for InterTrust are shown in bold. For this project, the critical path is A-B-C-F-H-END. As we expected, the critical path is the longest path through the network, and its duration is equal to the duration of the project.

Some activities, such as D, E, and G, have slack greater than zero. The start times of these activities may be delayed without affecting the length of the entire project. The durations of these activities may also be extended without pushing back the project completion time. However, if delays and extensions exhaust the size of the activity slack, these activities become critical activities, too.

3. Resource Constraints

An invaluable method for shortening the duration of a project is the ability to run multiple activities in parallel. For example, the InterTrust project allows both activities B and E to run in tandem from week four to week eight. If all InterTrust activities were to occur in series, the project would last 36 weeks rather than 22. A Gantt chart, such as the one shown in Figure 4, displays the degree of parallelism in the project. The chart displays the activities beginning at their earliest start times, as well as the number of analysts needed for each activity.

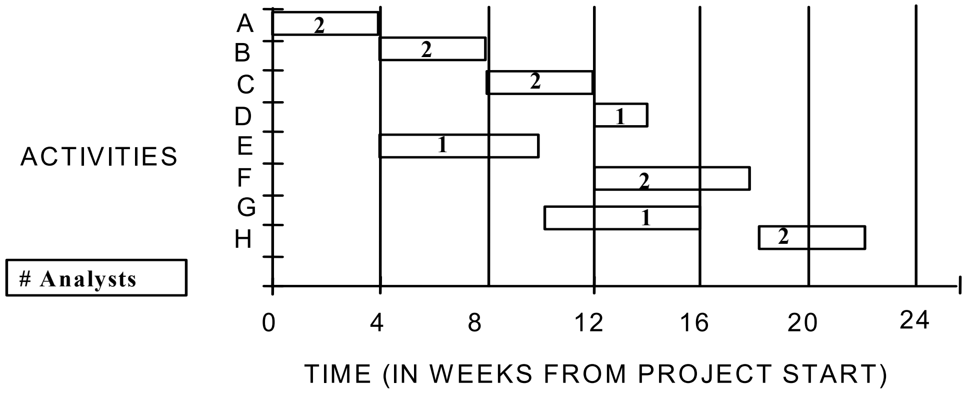

Figure 4: Gantt Chart for InterTrust with All Activities at Their Earliest Start Times

However, sufficient resources must be available for activities to be completed in parallel. If all activities in the InterTrust project were to begin on their earliest start dates, then the number of systems analysts needed varies from two to four. This is shown in Figure 5, which is derived from the resource listing in the Gantt chart. Note that the period from the end of week 12 to the end of week 14 requires four programmers: one for activity D, two for F and one for G.

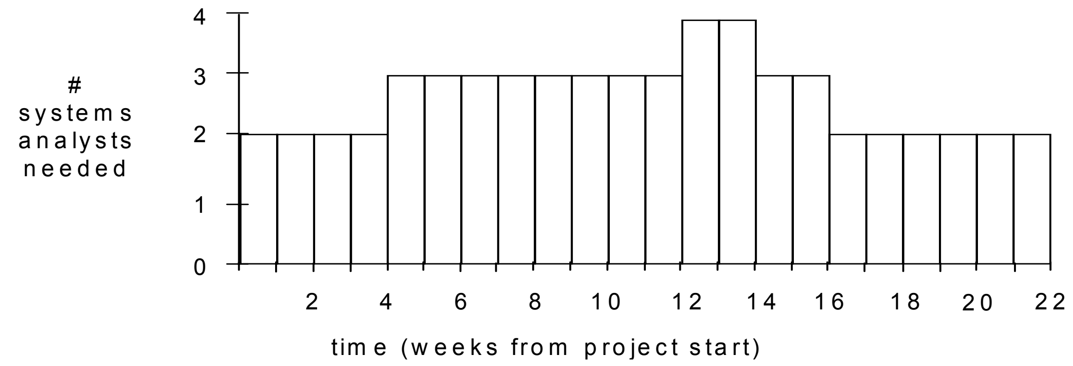

Figure 5: Number of Systems Analysts Needed if all Activities Begin on the Earliest Start Dates

Because only three systems analysts are available for the InterTrust project, it appears that the project violates the resource constraints. One solution to this problem is to purchase more resources. However, it may be possible to delay activities so that resource constraints are satisfied. In fact, if a single activity is delayed less than its slack, the project as a whole will not be delayed. Activities along the critical path cannot be delayed, but an activity with slack may be delayed without affecting the total project length.

For InterTrust, activity D has eight units of slack and it may be rescheduled to begin after 16, 17, or even 20, weeks without delaying the project. If activity D begins after week 16, only 3 systems analysts are needed (see Figure 6).

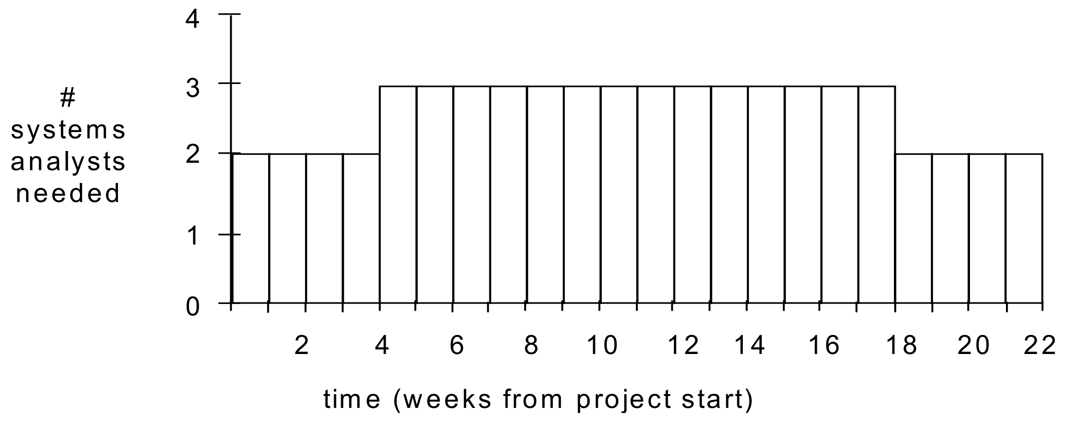

Figure 6: Number of Systems Analysts Needed if Activity D is Delayed by Six Weeks

With this small project, we could find a schedule by hand which satisfied the resource constraints. Scheduling activities so that resource utilization remains as low as possible is often called ‘smoothing’ the project. With larger projects and multiple resources, computer-based optimization tools are needed to find a smoothed schedule.

4. ‘Crashing’ a Project to Shorten Duration

For many projects there is a trade-off between project cost and project duration. When a project lags behind its schedule, extra people may be assigned to the job to speed it up. Even for an on-time project there may be opportunities to ‘crash’ the project by hiring personnel or purchasing additional equipment. A manager must asses the costs and benefits of speeding up the project.

Recall that InterTrust is spending $3,000 per week until its new information system is on-line. Suppose that there is an opportunity to hire extra programmers to work on activities B, C, D, and/or G. The available programmers have specialized skills and abilities; one programmer may be assigned to each task, and relevant data are listed in Table 2. Here we assume that hiring an extra programmer is an all-or-nothing decision, e.g., if we want to shorten the duration of activity B we must pay $2,000 for a programmer who reduces the expected activity time by 3 weeks. In other situations it may be possible to purchase crashing help 'by the week.' To find this variable cost, we would interpolate: the weekly cost of crashing activity B would be \$2,000/3 weeks = \$667/week. But, again, we now assume that the labor costs shown in Table 2 are indivisible.

On first glance, all four programmers would seem to be a good deal. Every week eliminated from the duration of the project will save \$3,000, each programmer costs less than \$3,000, and each will cut at least a week off an activity's duration.

	Cost of	Time Estimates (weeks)	
Activity	Extra Programmer (\$'000s)	Normal	Extra Programmer
B	2	4	1
C	2.5	4	1
D	2	2	1
G	1	6	1

Table 2: Data for Crashing the InterTrust Project

However, because of the dependence between activities, the benefits of the extra programmers may be limited. Typically, activities on the critical path should be crashed since these activities determine the duration of the project. But eventually, these paths are no longer critical. We are faced with the question: which programmers, if any, should be hired?

One method for evaluating opportunities to 'crash' a project is described by these steps:

A Method for Crashing a Project

Step 1: Assess the cost-effectiveness of crashing activities on the critical paths (it may be necessary to crash more than one activity to have an effect). If no set of crashes leads to a net gain, stop;
Step 2: Implement the most cost-effective crash until it is no longer cost effective or the paths involved are no longer critical;
Step 3: A crash in step (2) may create new critical paths. Revise the network and identify the new critical paths. Return to step (1).

As an example, consider the InterTrust project and the data in Table 2:

Step 1: Activities B and C are on the critical path for InterTrust. However, if either is shortened by two weeks then paths 'A-E-F-H-END' and 'A-E-G-H-END' become critical. Therefore, we save at most 2 weeks by adding a programmer to either B or C, and the additional programmer for B costs less than the additional programmer for C. Therefore, crashing activity B is the most cost-effective plan, saving $6,000 while spending $2,000.

Step 2: Crash activity B by reducing its duration from four weeks to one. Note that crashing B from four weeks to two would have had the same effect, but we did not have that option at a lower cost.

Step 3: With activity B crashed, we have the new network shown in Figure 7. There are two new critical paths.

Step 1: Activity G is the only activity on a critical path available for crashing. However, crashing activity G will not decrease the duration of the project since path 'A-E-F-H-END' would remain critical and the project duration would not dip below 20 weeks. Therefore, we are done.

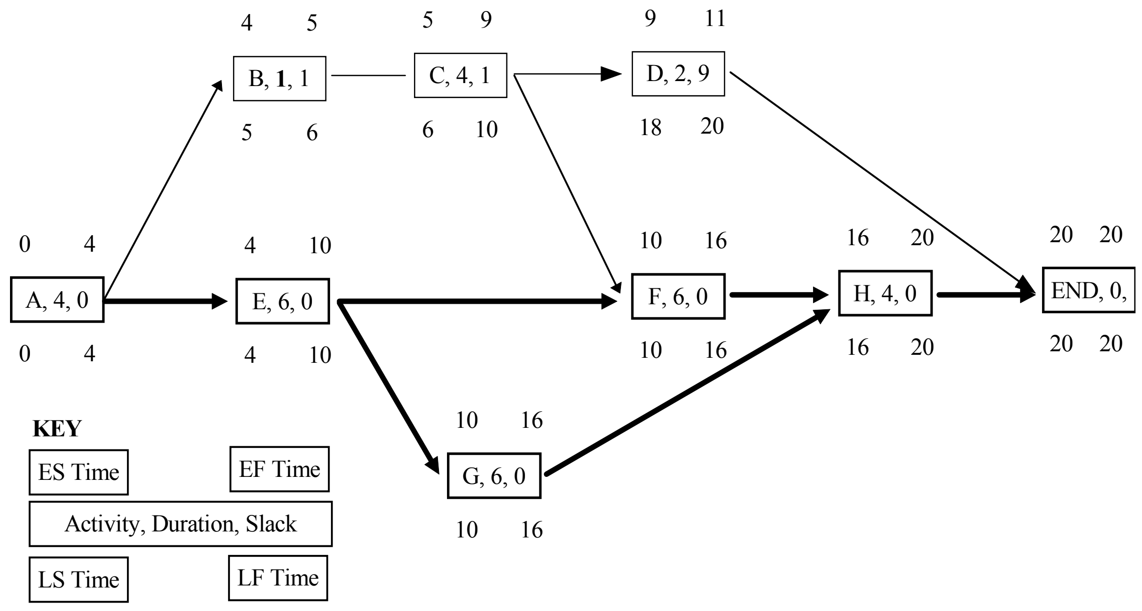

Figure 7: InterTrust with Activity 'B' Crashed by Three Weeks

Note that an extra programmer for activity G seemed like a bargain: 5 weeks saved for only $1,000. However, hiring this programmer would not have shortened the duration of the project.[1]

In general, crashing a single activity will not shorten the project unless that activity is on a unique critical path. If there are multiple critical paths, activities on all of them must be crashed to shorten the project duration.

[1] It is possible that an extra programmer for activity G would have liberated other programmers, who would then have been shifted to activities on other critical paths. You see how complicated this can get!

5. Random Activity Times

In the previous section, dependence between events limited the benefits of extra resources. This dependence also exacerbates the effect of randomness, so that variations in project times tend to *increase* the length of the total project. For InterTrust, we have assumed that activity durations are known in advance. In practice these durations will be means, medians, or some other estimate, and the true activity times will vary around these estimates.

What is the effect of this variation? We have calculated that the original (non-crashed) InterTrust project will take 22 weeks with no variation in activity times. But if activity times vary, we can expect the project to take longer than 22 weeks *even if* the expected values of individual activities are equal to the numbers listed in Table 1. In fact, for any project subject to randomness,

Expected project duration
$\geq$ Project duration calculated from expected durations of individual activities

An example should make this clearer. Suppose that the duration of activity E in the InterTrust project were uncertain, and the project manager estimated that there was a 50% chance the task would take 2 weeks and a 50% chance it would take ten weeks. Note that the expected duration is still six weeks, as listed in Table 1. Is the expected total project duration still 22 weeks?

If activity E lasts two weeks, the project remains 22 weeks long because a reduction in the duration of a non-critical path does not affect the duration of the project (see Figure 3). If activity E lasts ten weeks, then the path 'A-E-F-H-END' becomes critical, and the project duration is 24 weeks. The expected duration of the project is now (1/2)(22) + (1/2)(24) = 23 weeks. The expected duration has risen by one week, even though the expected durations of each individual activity remained the same.

When multiple activities are subject to randomness then it is very difficult to calculate the expected duration of the project as a whole. Mathematical approximations or Monte Carlo simulation may be used to find project durations, costs, and resource utilization.[2] However, the underlying insight is similar to the insights from The Goal and from queueing theory.[3,4] Deterministic calculations tell only part of the story; statistical fluctuations (what we have called variability) and dependent events conspire to lengthen the duration of the projects.

[2] A 'Monte Carlo simulation' is designed to reproduce the behavior of a system that is subject to randomness. One input to the simulation is a sequence of random variables, and the simulation's response to this random input imitates the response of the real system.

[3] E. M. Goldratt and J. Cox, The Goal: A Process of Ongoing Improvement, North River Press Publishing Corporation; 2nd Revision edition (May 1992).

[4] Queueing theory, also known as waiting line theory, helps us to make capacity decisions when demand and process variability cause congestion.