Critical Chain as an Extension to CPM

A key innovation of critical chain scheduling can readily be added to the critical path method, and improved by using estimates of optimistic, most likely and pessimistic durations.

Vincent McGevna, PMP

Critical chain project management has been presented as an alternative to the critical path method (CPM). However, key concepts of the critical chain methodology are readily used as an extension to CPM. A critical chain schedule recognizes that task duration estimates are not exact, but have a probability distribution around a most likely value. When the duration of each task is chosen from this distribution to assure a high probability of success, then the total schedule will be longer than necessary. If, for example the estimates are set for a 90 percent probability of completion on time, then it would be expected that 90 percent of the tasks should finish in less than the allotted time, while only 10 percent of the tasks will take longer than planned. However, because of Parkinson's Law, activities tend to fill the available time, and the student syndrome, waiting until the last possible minute to start a task, then the tasks that should finish early, finish on time or late, while those expected to be late are late. The result is that the whole project is late. To resolve this dilemma, the duration for each task in the schedule is set with a value such that there is a 50 percent probability of it being completed within the allotted time. This creates tension in the schedule which helps to significantly reduce time lost due to Parkinson's Law and the student syndrome. However, the probability of completing the project within schedule can be very low. Therefore, the critical chain schedule adds strategically placed buffers to provide contingency to assure a high probability of completing the schedule in the allotted time.

There is a methodology for generating critical chain schedules, and it is described in a book by Leach [*Critical Chain Project Management,* Artech House, 2000] and another by Newbold [*Project Management in the Fast Lane,* St. Lucie Press. 2000]. However, assumptions that form the basis for the methodology are inconsistent with this author's experience on engineering projects. A key assumption is that CPM schedules are generated without considering resource dependencies. It is this author's experience that for medium to large engineering projects, resources assignments and related task dependencies are addressed early in the planning, and throughout schedule creation. Thus when the critical path is identified, it coincides with what is called the critical chain. A bigger problem than ignoring resource dependencies is failing to perform the up front planning, which includes an overall development strategy or game plan. The critical chain load leveling algorithm cannot compensate for this. With a work breakdown structure (WBS) which identifies the components to be developed, and a development strategy to effectively create and integrate those components, then, as will be shown, the critical chain buffers can be directly included as part of the normal creation of the project schedule.

Another basic assumption is that developers knowingly overestimate task durations to prevent looking bad. Therefore, the estimates created by a developer are cut in half to arrive at the 50 percent estimates. This may be valid in some project environments, but it is certainly not an appropriate generalization. Again, this author has observed that engineers tend to be incurable optimists. While some will overestimate task duration, most underestimate because they do not allow sufficient time for debugging or for handling the myriad of problems that are inevitable on a development project. A more appropriate approach under these circumstances is to use three point estimates and the statistics derived from them. How to use the three point estimates in the context of critical chain will be presented to give other project managers an alternative when halving of estimates is not justifiable.

Schedule Buffers

While a complex project may have many parallel paths, the use of buffers can be illustrated with a simple project. Consider a project having two subsystems, A and B, each of which must be developed, and the two integrated into a functional system. Subsystem A requires six tasks, 1 through 6, subsystem B requires four tasks, 7 through 10, and there is a task for integration and test, task 11. Also, Subsystem A is on the critical path. Exhibit 1 shows a network diagram for this project.

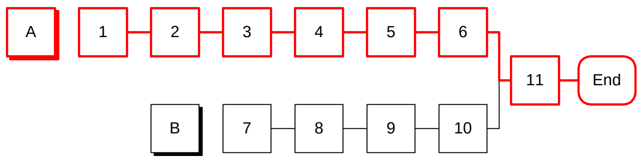

Exhibit 1. Network diagram for a simple project.

The critical chain methodology adds three types of buffers to the schedule: feeding, project and resource buffers. A feeding buffer is added as contingency for tasks not on the critical path, to assure that work needed for the critical path is available when needed, while the project buffer provides contingency at the very end of a project. Resource buffers are added as wakeup calls to alert resources to be ready to work on critical chain tasks. The network diagram for the example schedule can be modified to create a critical chain schedule by adding a feeding buffer following task 10, a project buffer following task 11, and, assuming that an external resource is required for integration and test, a resource buffer feeding into task 11. The updated network diagram is now shown in Exhibit 2.

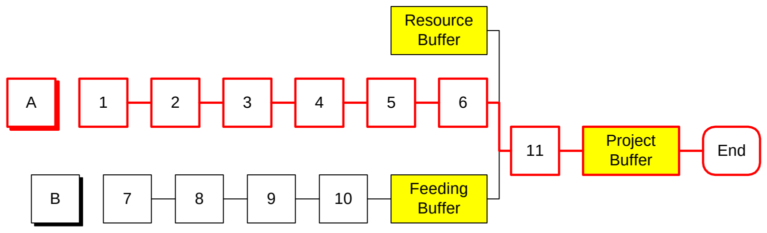

Exhibit 2. Network diagram with critical chain buffers added.

When there are multiple paths in a schedule, more than one may be critical, and many of the others may not be far off the critical path. Adding feeding buffers in all but a single critical path will likely create a new critical path. Also, in this example, an external resource is required for integration and test, but since task 1 through task 6 are sized for 50 percent probability of completion on time, the start of integration and test will likely be late. This will impact the resource buffer, which is keyed off when task 11 begins. Since paths in a carefully planned schedule generally represents a string of coherent tasks focused on a specific deliverable, it is a much simpler task to insert a feeding buffer at the end of the chain of tasks for each deliverable when the schedule is created, and not worry about which is the critical path. Then, when the tasks and buffer are properly sized, the schedule will provide a reasonable estimate of when each deliverable will be available. For the example schedule, a feeding buffer is added following task 6, as shown in Exhibit 3. Now, the original project buffer is split between the new feeding buffer and the project buffer, with a large fraction of the original buffer duration going into the new feeding buffer. If the integration is not too complex, then the project buffer may be eliminated; else it may provide contingency for integration and test.

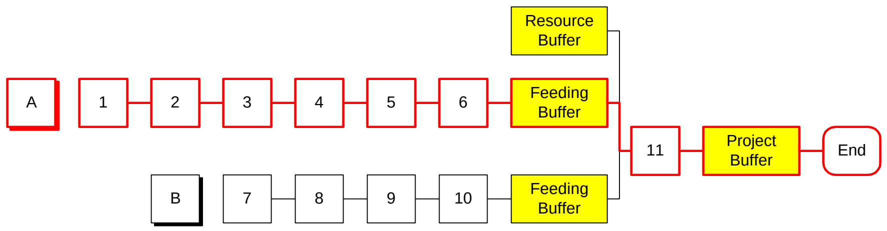

Exhibit 3. Network diagram with feeding buffers in all paths.

Task and Buffer Sizing

When estimating the duration for a task that has never been performed before, a most likely estimate is usually made based on characteristics of the task, such as similarity to previous tasks. There will always be some uncertainty in this number resulting in a probability distribution of possible values. Recognizing this, Project Managers may require developers to provide three estimates for each task: a most likely, an optimistic and a pessimistic estimate, to characterize the distribution. Then the mean is calculated as a weighted average of these three values and used as the task duration. For the critical chain methodology, the task duration should be set to the median of the distribution. However, the median is generally very close to the mean, and when the distribution is symmetrical, that is the most likely estimate is centered between the optimistic and pessimistic estimates, the two are equal. Thus, when the mean is used without any buffers, the probability of completing one path on time is 50 percent, and with two parallel critical paths, the probability is reduced to 25 percent. Additional paths on or near critical will reduce this even further. No wonder so many projects are late.

Experienced project managers build contingency into their schedules, which is frequently based on a percentage of each path or the overall schedule duration. However, critical chain provides buffers to build in the necessary contingency, and now the question is how to size the buffers. The most likely, optimistic and pessimistic estimates can be used to calculate the standard deviation of each task. For a sequence of tasks on a path, the standard deviation of the path is the square root of the sum of of the squares of each individual standard deviation:

$$(std.dev.)_{path} = \sqrt{\sum (std.dev.)^2_{task}}$$

An example of making these calculations is provided in Section 11.2.2 of the 1996 PMBOK® (PMI, 1996) along with the equations for calculating the mean and standard deviation for two commonly used distributions. Assuming the path duration is normally distributed since it is the sum of random task estimates, the probability of completing the work with a buffer set equal to the standard deviation of the path is approximately 84 percent. The calculation assumes that the tasks in the path are sequential, and that there is no parallelism. Since each path generally represent the development of a major deliverable this is a reasonable assumption. Also, small amounts of parallelism will have less impact on the buffer size than the errors in the estimates used.

With an 84 percent probability of success, a Project Manager should be able to make this 100 percent most of the time by managing the development[1]. However, for a really complex schedule, the Project Manager's efficiency will be reduced and more contingency will be required. This means enlarging the buffers. Enlarging a buffer to 1.65 standard deviations increases the probability of success to 95 percent, and two standard deviations makes it 98 percent.

[1] Management activities typically include risk management, maintaining focus, eliminating unnecessary functionality, facilitating, providing needed resources, establishing good communications, etc

When a project has significant integration and test or other activities at the end, it is desirable to have a project buffer. The integration and test activities can be broken into discrete tasks, and the buffer size calculated in the same manner as the feeding buffers.

While the feeding and project buffers are sized based on uncertainty in the tasks, the resource buffer size is arbitrarily set depending on the resource that is added to the schedule, whether it be a person or a piece of equipment.

Creating Buffers in Microsoft Project®

Once the buffers are set in the schedule, management of the schedule requires monitoring the size of each buffer for indications that the buffer will be used up prior to the completion of its associated delivery. Leach describes one approach for managing the buffers, that is realizing when to react to changes in the buffer size, which is based on control chart methodology. Regardless of the process for managing buffer size, when actuals are added to the schedule, the buffer size needs to be automatically updated to track the changes. Currently, Microsoft Project does not provide a task type that would change automatically. However, it is possible to simulate the desired behavior. When a path has been selected, then:

- Add a task for the buffer is at the end of the path
- Add a milestone after the buffer
- Perform the task and buffer size calculations and update all durations, including the buffer
- Constrain the task type of the milestone to *Must Finish On*
  (This is on the Advanced tab of the Task Information window)
- Change the buffer to a milestone (set its duration to 0 days)
- Constrain the buffer task type to *As Soon As Possible*.

The slack for the buffer will now be the buffer size, and any updates to tasks which are predecessors to the buffer will change its slack. Note that this is using slack because it gives the correct number, it is not equating the buffer to slack in the schedule. The initial buffer size can be copied to one of the ten additional duration fields (Duration1 to Duration10) for a quick comparison.

Considerations When Using Critical Chain

The purpose of the foregoing methodology is to produce a schedule which provides a reasonable estimate of completion, minimizes the total development duration, and to which the Project Manager can commit and be assured of a high likelihood of success. It needs to be understood that the use of statistics can produce a reasonable single point estimate only when there are a large number of tasks with relatively small, random variations. For some projects, such as when pushing the state-of-the-art or pushing performance limits, there are major tasks within the project which can be open ended, where the pessimistic estimate is significantly greater than the most likely, and all too often, the most likely estimate represents the “right answer”. While such tasks can be modeled by an open ended log-normal distribution, using a log-normal distribution to generate a single best estimate is likely to lead to great disappointment. The fact is, in these situations a single, statistically generated estimate is meaningless, and this needs to be highlighted early in the planning. Those few open ended tasks require risk management and a development strategy. The development strategy frequently involves iterations to minimizes the impact to all other tasks, and to produce functional interim deliverables. With a solid plan of action, a log-normal distribution can then be used in Monte Carlo simulations to calculate a range of possible values.

The methodology also assumes that the paths are not coupled. Frequently, interim deliverables are needed by other developers, and there is a significant number of links between the paths. These links may be weak, in that work may be started on the successor task before the predecessor task is complete, or have sufficient slack to have minimal impact on the analysis. However, the presence of strong coupling among multiple paths could alter the statistical analysis, requiring more contingency than planned. Rather than perform more sophisticated analysis that will push out the schedule, the presence of coupling between the paths should be identified and managed as risk during project execution.

Finally, critical chain focuses on only one area of the triple constraint: schedule. It can definitely help a Project Manager create a viable, minimized schedule with a high probability of success. However, the information can be even more useful when coupled with scope and cost. Assumptions about scope and cost should always be identified along with the three point estimates. Frequently, when there are open ended tasks, the uncertainty can be significantly reduced by a small cost increase or a change in scope. This holistic approach will provide more meaningful discussions when finalizing the actual scope of the project, and have an even bigger impact on the likelihood of success.

*****

Vincent McGevna, PMP has over 16 years of Engineering Project Management and Systems Engineering experience which includes Automatic Test Equipment, State-of-the-Art Computer Telephone, Flight and Ground Computer Systems for Space Shuttle Life Science Experiments, and a Safety Assessment System for a Nuclear Power Plant. He can be reached at vmcgevna@yahoo.com.