



Critical Chain Project Management – Completing More Projects in Less Time

Bob Sproull – John Sproull

When we were asked to write an article on Critical Chain Project Management (CCPM), as a starting point, we asked ourselves, “What would make people want to read it?” We reasoned that the appalling statistics on current project completion rates might stimulate readers, especially if they are in an organization that relies on project completions as their source of revenue. For example, in a 1994 survey (The Chaos Report) by the Standish Group¹ studying nearly 10,000 IT projects across America, the key finding was that 52% of projects cost greater than 189% of the original budget, 31% were cancelled and only 16% of the projects were completed on time and on budget. An update to the survey was completed in 2007 and 35% of the projects were completed on time, on budget and meeting user requirements. Although this is a marked improvement from the first groundbreaking report, we think it’s safe to say that these statistics still aren’t acceptable. In another study completed in Australia², construction projects completed only 1/8 of building contracts within the scheduled completion date, and the average overrun exceeded 40%. There are many other reports from numerous industry types, from all over the world, that all conclude the same thing, that project completion rates are abysmal!

So the question we pose is this -- what if we could demonstrate a method that would push your project successful completion rate from your current rate to over 90%? Would you be interested in hearing about it? We thought you would be and here is our article.

Why are projects failing?

It may be surprising, but most of the problems associated with project management’s lack of success are related to behaviors. Not just individual behaviors, but organizational behaviors as well. What we hope to clarify in this article are what these negative behaviors are, why they negatively impact projects, and what can be offered as antidotes. Let’s start this discussion with what’s happening on most projects today.

90% of the project managers around the world are using Critical Path Project Management (CPPM) and have been doing so for the last 40 years. If you ask a typical project manager about what factors delayed a completed project, most will tell you that something they didn’t expect or even had no control over cropped up in some of the tasks and delayed them and therefore delayed the project. In other words, they experienced inevitable uncertainty or Murphy’s Law. Every project from virtually every environment has uncertainty associated with it and how this uncertainty is dealt with determines the ultimate success or failure of the project. So in order for a project to be



successful, there must be a way to protect it from uncertainty. Let's take a look at how traditional project management (CPPM) protects a project from uncertainty.

CPPM uses a "fudge factor" to protect projects from inevitable uncertainty. When developing the project plan, durations for each individual task are estimated by the resources responsible for executing them and then a safety factor is added to each task by the same resource. For example, suppose the realistic estimate of time for an individual task is one week. Is one week what the resource actually tells his or her project manager? Typically the resource will add his or her own safety factor to guard against "things" that might happen that would delay completion of the task. So it's not unusual for the original one week to be quoted as two weeks. Resources react this way because they know from experience that as soon as they give the project manager an estimate, it automatically becomes a commitment! *Behavior problem number one, adding safety times to individual tasks!*

A typical project manager will add up all of the individual, inflated time estimates and then add his or her own safety factor. Why is this done? Project Managers know that at some point in the project Murphy will strike and some of the tasks will be delayed, so they too add a safety factor to protect the project from being late. Keep in mind that every resource inflates every task, so it's not uncommon for the estimated duration to be 50% greater than it takes to actually complete the task. So with all of this built-in safety, the project should be completed on time....right? So it seems, but the statistics on project completions don't bear this out. We'll explain why later.

In traditional project management how do you track progress against the completion date? The typical method involves calculating the percentage of individual tasks completed and then comparing that percentage against the due date. Sounds reasonable, but is this the right way to track progress? The problem with using percentage of tasks completed is that not all tasks have the same duration. That is, comparing a task that has an estimate of one day to a task that should take one week is not a valid comparison. Compounding this problem is the mistaken belief that the best way to ensure that a project will finish on time, is to try to make every individual task finish on time. This too sounds reasonable, but later on we'll show you why this isn't so. *Behavior problem number two, focusing on individual task completion!*

So the question remains....if individual project tasks have so much extra time embedded in them, then why are so many projects coming in late? This is partially explained by two additional and common human behaviors. Resources know that they have built "safety" into their tasks, so they often delay work on the task until much later than they had originally planned. Think back to your high school days. When you were given a due date for a paper for next Thursday, when did you start working on it? Wednesday? Eli Goldratt coined the term, *Student Syndrome*, to explain one of the reasons why the apparent built-in safety gets wasted. When the task start is delayed



and Murphy strikes, the task will be late. *Behavior problem number three, procrastination!*

Another human behavior that lengthens projects is referred to as Parkinson's Law. Resources intuitively know that if they finish a task in less time than they estimated, the next time they have the same or a similar task to complete, they will be expected to finish it early. So, to protect against this, when a task is finished early, the resource doesn't notify the project manager that it is finished until the original due date. After all, we're talking about credibility here, so to protect it, early finishes aren't reported. Parkinson's Law states that work expands to fill the available time. The key effect on projects, of these two behaviors, is that delays are passed on, but early finishes aren't. So is it any wonder that projects are late? *Behavior problem number four, not reporting early task finishes!*

While these four behaviors negatively impact project schedules, there is another behavior that is perhaps the most damaging of all. Many organizations today have multiple projects going on at the same time and it's not unusual for projects to share resources. In fact, many project managers tend to "fight over" shared resources because they believe their project is the one that has the highest priority. In many project based companies, leadership initiates projects without considering the capacity of the organization to complete the work. Leadership also mistakenly assumes that the sooner a project is initiated, the sooner it will be completed. As a result, perhaps the most devastating problem of all associated with project completion occurs..... multitasking! But wait a minute....I thought we'd all been taught for years that multitasking is a good thing? *Behavior problem number five, pushing projects into the organization without considering its capacity!*

Multitasking happens when resources are forced to work on multiple project activities at the same time. Many people believe (especially in leadership positions) that multitasking is a good thing because it increases efficiency since everyone is "busy" all of the time. But is local optimization really a good thing?

If you've ever read *The Goal* by Eli Goldratt (if you haven't, you should), you might remember how focusing on local activities actually damaged the overall system performance. *The Goal* is a business novel about a plant manager (Alex Rogo) of a failing manufacturing facility that is on the chopping block. Alex is on his way to a divisional meeting to discuss his plant's performance, when he happens on his old university physics professor, Jonah. Alex explains that he has installed a new robot in one area of his plant. He's very proud of this achievement until Jonah asks him if the robot has resulted in the shipment of more products. Alex ponders this question and responds by saying that it has improved the department's efficiency. Jonah points out that if the robot hasn't resulted in more product out the door, then it was probably not a good idea. In fact, all the robot did was create a mountain of inventory in the middle of the process. Goldratt's point is that the sum of local optimizations (i.e. maximizing local



efficiencies), does not improve the overall performance of the system. Maximizing efficiency only matters at the system constraint. The same can be said when focusing on completion of individual tasks in a project.

The negative impact of multitasking in a project management environment is much, much worse. Let's look at an example. Suppose you have three projects (Figure 1)

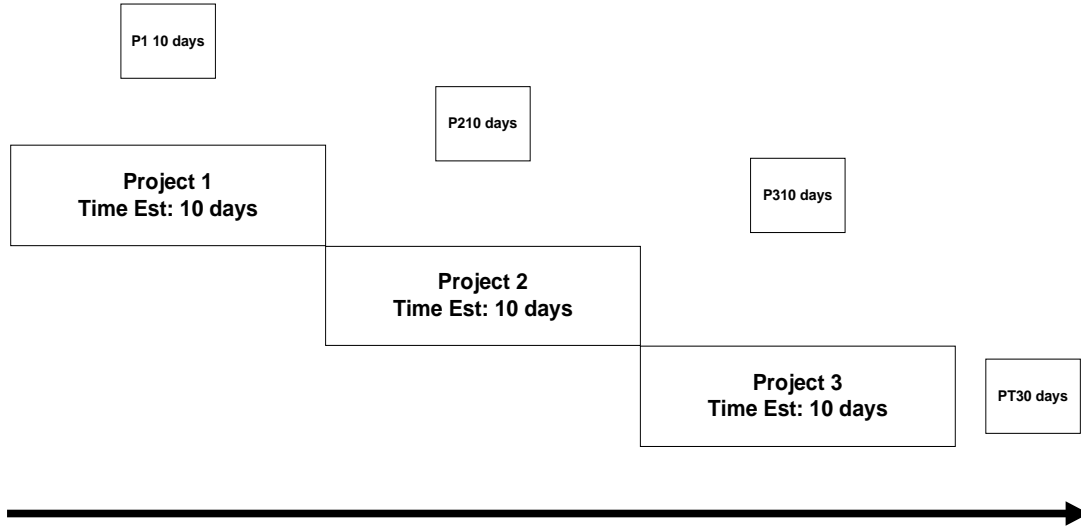


Figure 1

that you are assigned to and in each project you have estimated that you have 2 weeks (10 days) of work on each project for the tasks assigned to you. Assuming the projects finish somewhat on budget, if you started and finished Project 1 without stopping or working on any other project, it would be done in 10 days. Ten days because that's what you told everyone it would take (Parkinson's Law). Likewise for Projects 2 and 3, assuming no other interruptions, each would take 10 days to complete for a total time to complete the three projects of 30 days.

Because there are probably three different project managers, each one is most likely telling you (or maybe even screaming at you) that they need you to show progress on their project (remember, projects are typically measured by % of tasks complete). You want to satisfy all three managers, so you decide to split your time between the three projects (i.e. you multi-task). As is demonstrated in Figure 2, you start with Project 1 and work on it for 3 days. On the fourth working day, you begin Project 2 and work on it for 3 days. You repeat this sequence until all projects are completed.

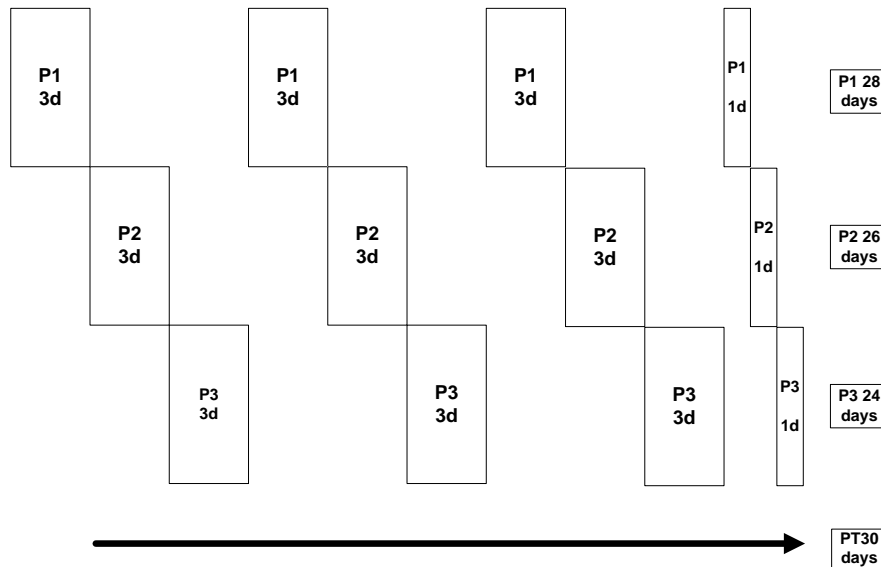


Figure 2

By multitasking, look what's happened to the time to complete each individual project. Without multitasking each project took only 10 days to complete and 30 days to complete all three. With multitasking, Project 1 took 28 days, Project 2 took 26 days and Project 3 took 24 days. Both methods completed all three projects in 30 days, but which set of results do you think your leadership would prefer? (Note: If you start timing all three projects from day 1, then the elapsed time would actually be 28, 29, and 30 days, respectively).

Now, let's further consider the scenario above. Imagine Murphy strikes, as we would expect, for at least one of these projects. Let's assume Project 3 around its 5th day is impacted by something that will cause a delay in the completion by 2 days. What does that do to my completion dates for Projects 1 and 2, which are both theoretically independent of Project 3? By multi-tasking the three projects, each project's completion becomes susceptible to all of the uncertainties of all three projects. In other words, the three projects are now dependent on each other rather than just the individual project. Due to the delay of Project 3, all of the projects are delayed by 2 days and not just Project 3.

Keep in mind, the delays we are demonstrating above are not even considering studies that indicate tasks often take 2-3 times their estimated durations when multi-tasking occurs, due to time necessary to re-acquaint ourselves with each project and task!

So let's summarize what we've learned before we move on. We've learned that there are 5 behaviors that negatively impact the on-time completions of projects as follows:

1. *Adding safety times to individual tasks.*
2. *Focusing on individual task completion*
3. *Procrastination or the Student Syndrome and Parkinson's Law.*
4. *Not reporting early task finishes*
5. *Pushing projects into the organization without considering its capacity, resulting in multi-tasking.*

Let's explore these negative behaviors in a bit more detail and determine what, if anything, can be done to counter their negative effects.

As we've seen, in CPPM task durations are inflated to protect against uncertainty. What if we could significantly reduce these imbedded safety buffers and still provide the protection that we need? In our example from Figure 1, suppose we were able to reduce the estimated duration by 50% and still protect against uncertainty. In other words, if we could complete the tasks in 5 days instead of 10 days, wouldn't this be a quantum leap in project completion time reduction?

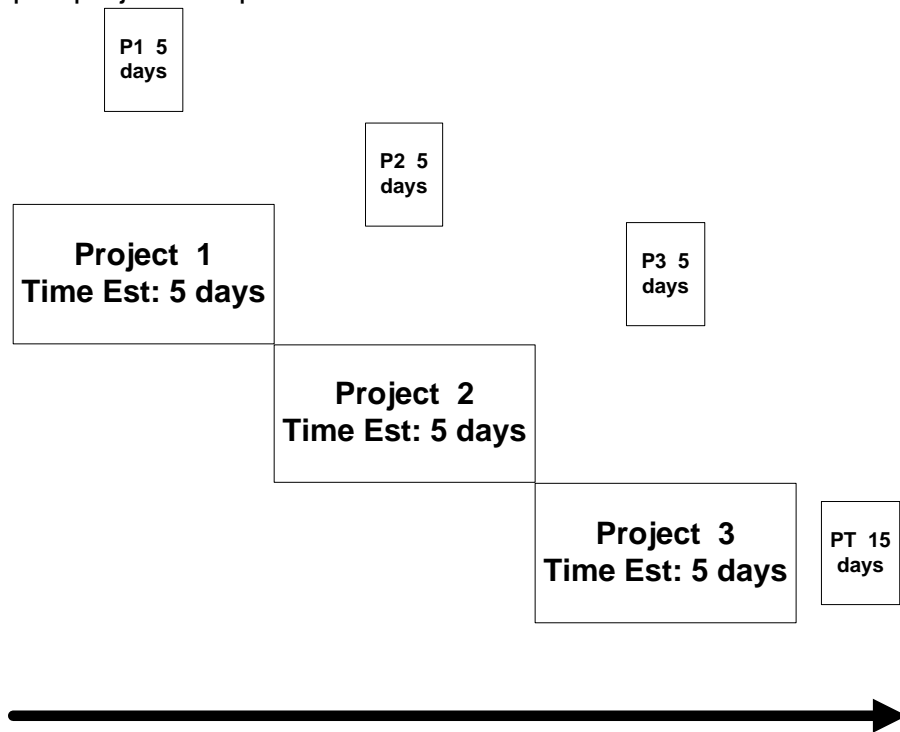


Figure 3

Figure 3 is a depiction of the reduction in durations of each project. We have just reduced the time to complete these three projects from 30 days to 15 days, but can we do this and safely guard against Murphy? The answer is yes we can! But before we explain how to do this, let's re-examine some key points from the Theory of Constraints (TOC).

TOC came on the scene in the mid-1980's through its developer, Eli Goldratt. Goldratt taught the world that every organization has at least one (and usually only one) constraint that prevents an organization from coming closer to its goal. For most companies, the goal is to make money now and in the future. In fact, Goldratt analogized this concept to the strength of a chain being dictated by its weakest link. The best way to understand TOC is to envision a simple, repeatable process as in Figure 4.

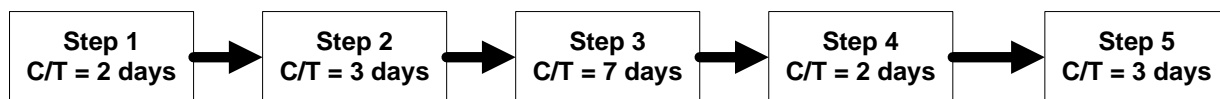


Figure 4

If the process in Figure 4 was starting for the first time, the fastest a part could be produced or a service could be delivered would be the sum of the individual cycle times or 17 days (i.e. $2+3+7+2+3 = 17$ days). Once the process is up and running, the throughput of the process is dictated by its slowest resource. In our example Step 3 is the slowest resource since it requires 7 days to complete. Therefore the maximum throughput of the process in Figure 4 is 1 unit of product (or service) every 7 days. TOC identifies Step 3 as our constraint and tells us that if we want to improve throughput, then we must focus our improvement efforts on this step.

One of the problems associated with today's Lean (and/or Six Sigma and/or Lean-Sigma) efforts is to attempt to improve all processes. In reality, however, much of these efforts do little to impact the overall throughput of processes. For example, in the above process, what if a Black Belt successfully reduces the cycle time in Step 2 from 3 days to 1 day, or a 67 % reduction. A reduction of this magnitude is clearly an improvement, right? But, what does this improvement do for the throughput of the overall process? If you answered nothing, then you would be right. The process throughput is still limited by Step 3, the constraint. This is an example of local optimization that has no effect on the total process. But what if the Black Belt had achieved this 2 day reduction in Step 3? The throughput would have improved from 1 unit every 7 days to 1 unit every 5 days! By focusing improvement resources on the process step that is limiting throughput, significant gains in throughput and profitability can be achieved. In addition, if we starve a constraint (i.e. make it wait for incoming product instead of keeping it busy all of the time), we lose throughput that can never be recovered.



There are two key points then that must be reinforced:

1. Attempts to reduce the cycle times of non-constraint process steps do nothing to improve the overall output of the total process or system. Only improvements to the constraint positively impact the output of the process.
2. The focus must be on protecting the constraint from starvation because any time lost at the constraint is lost to the entire process.

TOC's 5-step process of on-going improvement is as follows:

1. Identify the system constraint.
2. Decide how to exploit the constraint.
3. Subordinate everything else to the constraint.
4. If necessary, elevate the constraint.
5. Return to step 1, but don't let inertia create a new constraint.

Earlier we demonstrated how, by simply eliminating multitasking, significant gains can be made in project completion rates, but we still have to address the impact of the other behaviors that negatively impact on-time project completions. Remember how excess safety is embedded into traditional project management plans? Resources estimate task times and add in their own protection against disruptions caused primarily by uncertainty. Knowing that this safety exists, resources then delay starting work on their tasks until the due date is close. Even if the resources don't delay the task starts and finish early, these early finishes are not reported and passed on. Our solution to these behaviors is Critical Chain Project Management (CCPM).

While traditional CPPM relies on individual task durations as well as scheduled start and completion dates, CCPM does not. The focus is no longer on finishing individual tasks on time, but rather starting and completing these tasks as soon as possible. Like CPPM, CCPM still gathers estimates on individual tasks and identifies its own version of the Critical Path. Unlike CPPM, CCPM considers competing resources (i.e. the same resource has to work on different tasks) and makes them a part of the critical path. Let's look at an example of how CPPM and CCPM identifies the critical path.

CPPM defines the critical path as the longest path of dependent tasks within a project. That is, tasks are dependent when the completion of one task is not possible until

completion of a preceding task. The critical path is important because any delay on the critical path will delay the project correspondingly. Figure 5 is an example of a series of tasks which must be completed in a project with the critical path highlighted in grey. Traditional project management determines the critical path by looking at the task dependencies within the project. Task A2 can only be initiated after A1 is completed. Task B3 can only be performed after completion of B2 and C2 only after C1. Task D1 can only be performed after completion of A2, B3 and C2. Using CPPM the critical path would have been identified as C1-C2-D1 and the project completion estimate would have been 29 days (i.e. 8d+12d+9d).

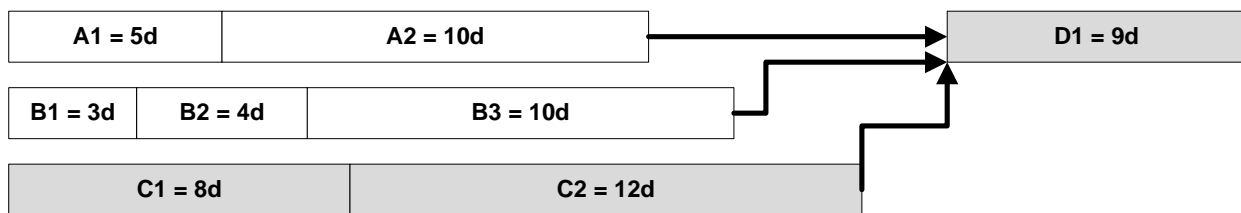


Figure 5

In addition to task dependencies there are also resource dependencies that CPPM fails to recognize. What if, in our example, tasks A2 and B3 are performed by the same resource? Is the critical path different? In Figure 6 we see the new critical path that includes a provision for resource dependencies. As you can see, the new critical path is A1-A2-B3-D1 with corresponding durations being 5d+10d+10d+9d or 34 days. So the minimum time to complete this project is now 34 days. In our opinion, the failure to consider resource dependencies is one of the key reasons why project completion rates are so terrible (another negative behavior). The simple implication of incorrectly identifying the critical path, which we will now refer to as *critical chain*, is the project team will never be able to complete their project on time without heroic efforts, adding additional resources, overtime or a combination of all three. The practical implication of incorrectly identifying the real critical path is that, unlike the critical chain, the focus will be on the wrong tasks. Recall from our earlier discussion of TOC that only improvements to our constraint will improve the output of a system. The same is true in a project environment where we must focus on the right tasks at the right time. In a project environment, the critical chain is the constraint.

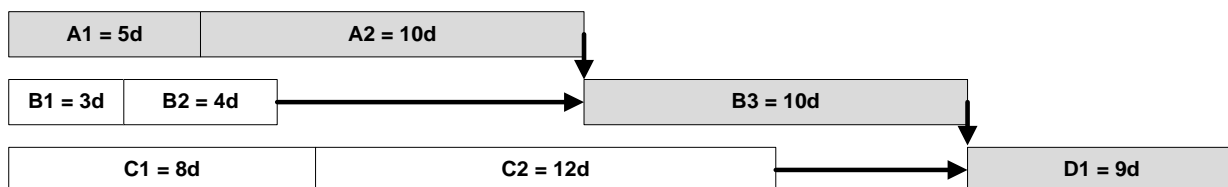


Figure 6



We said earlier that safety is embedded within each task as a way to guard against the uncertainties of Murphy. Critical Chain takes a completely different approach by assuming that Murphy's uncertainty will happen in every project. Unlike CPPM, CCPM removes these safety buffers within each task and pools them at the end of the project plan to protect the only date that really matters, the project completion date. In the example below, we have removed all of the protection from individual task estimates, which we estimate to be 50% of the original estimate. Figure 7 demonstrates the removal of this safety. So now, the length of the critical chain is no longer 34 days, but rather 17 days. Instead of just eliminating the safety buffer, we want to place it where it will do the most good.....at the end of the project to protect the due date.

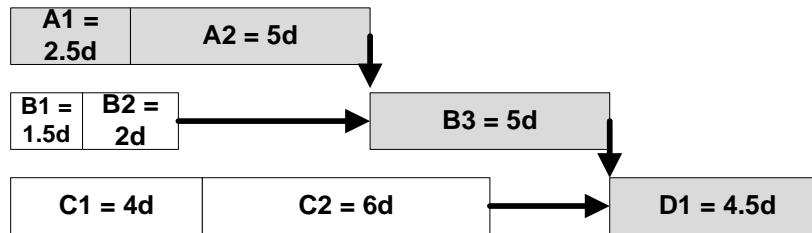


Figure 7

Figure 8 is this same process, but this time the safeties that we removed are added to the end of the project to act as a cushion to Murphy's inevitable attack. We have also added a feeding buffer of 2 days at the end of C1-C2 to assure that it is completed in time and not delay the start of D1. So the question now becomes, how do we utilize this buffer and how does it improve the on-time completion of the project?

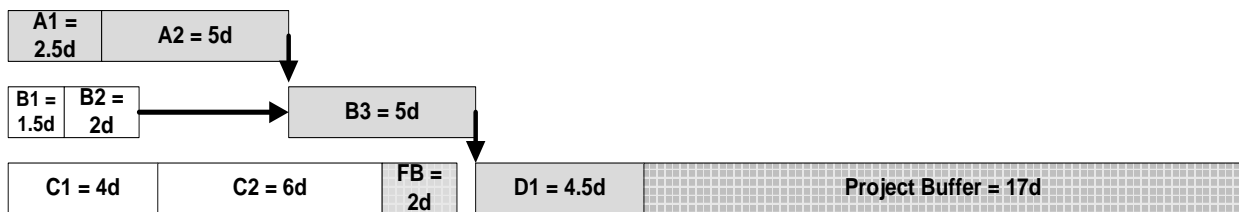


Figure 8

Suppose task A2 takes 7 days instead of the 5 days that are in the plan? In a traditional project management environment, this would be cause for panic. In a CCPM environment we simply consume two days from the project buffer and we're still on schedule. Suppose now, for task B3, we only take 3 days instead of the planned 5 days. We add the gain of 2 days back into the project buffer. In traditional CPPM, delays accumulate while any gains are lost. This is a significant difference! The project buffer protects us from delays. For non-critical chain tasks, or subordinate chains such as C1-C2 from our example, we also can add feeding buffers to assure that they are completed prior to negatively impacting/delaying the critical chain.



One of the key differences between CPPM and CCPM is what happens at the task level. In traditional project management each task has a scheduled start and completion date. CCPM eliminates the times and dates from the schedule and instead focuses on passing on tasks as soon as they are completed. This function serves to eliminate the negative effects of both the Student Syndrome and Parkinson's Law from the equation and permits on-time and early finishes for projects. In order for this to work effectively, there must be a way to alert the next resource to get ready in time. This is equivalent to a relay race where the baton is handed off from one runner to the next. Earlier, we explained that in traditional project management we track the progress of the project by calculating the percentage of individual tasks completed and then comparing that percentage against the due date. The problem with this method is that it is nearly impossible to know exactly how much time is remaining to complete the project. Using this method to track progress, many times you'll see 90% of a project completed, only to see the remaining 10% take just as long. In fact, looking at the number or percentage of tasks completed instead of how much of the critical path has been completed only serves to give a false sense of conformance to the schedule.

CCPM measures the progress of a project much differently and in so doing allows the project to make valuable use of early finishes. Critical chain uses something called a *Fever Chart* or *Buffer Management Chart* which is simply a run chart of % of Critical Chain Complete versus % of Project Buffer consumed. Figure 9 is an example of such a chart. In this chart we see that 20% of the critical chain has been completed while only 8% of the project buffer has been consumed, thus indicating that this project is actually ahead of schedule. Contrast this with Figure 10 which has completed 26% of the critical chain, but has consumed 28% of the project buffer, making it a bit behind schedule.

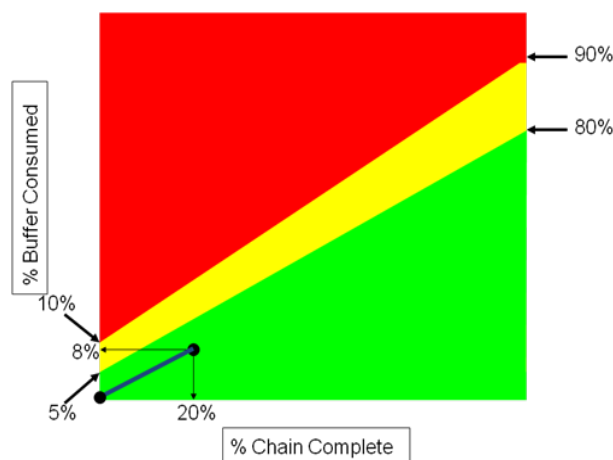


Figure 9

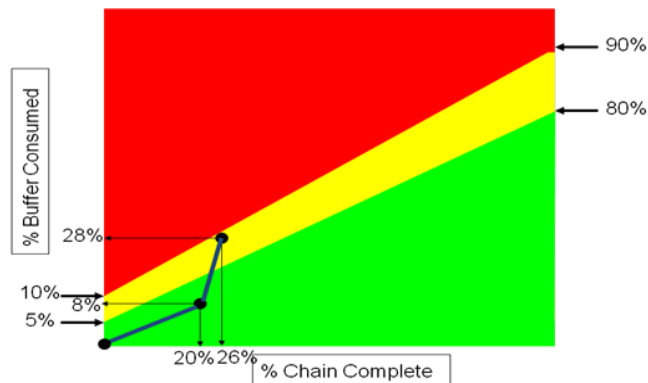


Figure 10

The green, yellow and red areas of the fever chart are visual indicators of how the project is progressing. If the data point falls within the green area of the chart, the project is progressing well and may even finish early. If the data point falls into the yellow zone, there is cause for concern and plans should be developed to take action, but not yet implemented. Vertical rises such as that demonstrated in Figure 10 indicate that buffer is being consumed at too fast a rate relative to how the project is progressing. If a data point falls into the red zone, then the plan we developed should now be executed. But keep in mind that even if the entire amount of project buffer is consumed at the completion of the project, the project is still on time and not late.

In addition to using the fever chart, we also recommend calculating a project index by dividing the % of critical chain completed into the % of the project buffer consumed. As long as this ratio is 1.0 or less, then the project will come in on-time or early. For example, from Figure 9 we calculate the index at $8\%/20\% = 0.40$ and is considered well ahead of schedule. From Figure 10 the index is $28\%/26\% = 1.08$ and is therefore a bit behind schedule. The index is a great way to track simultaneous, on-going projects.

By tracking projects in this manner, CCPM gives project managers an early warning of possible delays or problem areas within projects. In addition to monitoring the project buffer consumption that we just described, it is important to also monitor the status of activities that feed into the critical chain. For this reason it is also important to develop feeding buffers and then, like the project buffer, measure its consumption versus time remaining to complete feeder tasks. The subject of buffer management is large enough for an article on just that subject, so it won't be addressed in detail here. Suffice it to say, CCPM will provide results that are far superior to the traditional CPPM methodology.

A reasonable question might be, what kind of results could you expect to achieve if you converted from traditional CPPM to CCPM? There is no shortage of examples where organizations converting to CCPM have demonstrated anywhere from 30 to 100 % reductions in project times, which translates into greater than 90% average on-time



completion rates for organizations using this methodology. Our own experience with CCPM is right in line with these results.

While the results that can be achieved using CCPM are quite significant, are there ways to reduce project completion rates even further? We believe the answer is yes! The critical chain dictates the speed of project completion in the same way that the system constraint dictates the throughput of a manufacturing process. We know that, in the case of a manufacturing process, focusing improvement activities on the constraint reduces cycle time and increases throughput. The same principle applies to project management. The critical path is the constraint in a project, so by identifying the critical path and focusing Lean and Six Sigma efforts on it, we believe that additional time can be removed from it resulting in shorter and shorter project times.