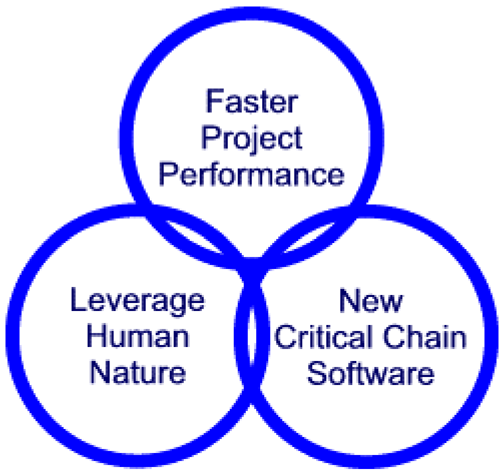
Critical Chain Concepts

Introduction

Traditional project management concepts have been around for over thirty years. If you look at the impetus for their development in the 1950s, you find that early studies noted that for Department of Defense projects, cost and time overruns were often two to three times the initial estimates and that project durations were frequently 40 to 50 percent greater than the original estimates. Similar studies of commercial projects noted cost and duration estimates overran by 70 and 40 percent respectively. Critical Path-based project management was introduced as a cure for these problems with a goal of delivering projects within the original cost and time estimates. Today, Critical Path project management is a significant industry. The discipline is used throughout the world based upon techniques defined in the 1950s and 1960s.



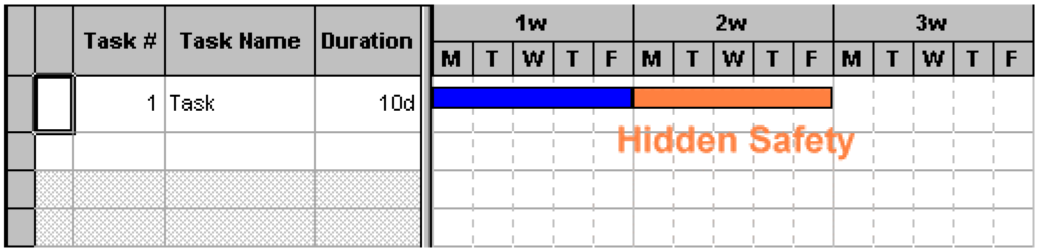
In 1997, Dr. Eliyahu Goldratt introduced the first significant new approach to project management in over thirty years with the publication of his best selling business novel, *Critical Chain*. The genius of Goldratt's approach resides in his development of a new paradigm that addresses, for the first time, both the human side and the algorithmic methodology side of project management in a unified discipline. Based upon Goldratt's break-through unified discipline, Critical Chain project management completes projects in significantly shorter time than traditional Critical Path project management techniques. Importantly, Critical Chain project management is also simpler to use and requires less work for the project team in both the planning and tracking phases of projects.

Critical Chain: the Human Side

Dr. Goldratt recognized that people plan and execute projects not computer programs. His Critical Chain methodology is based upon great insight into human nature and what happens when a project management discipline is applied to people. Goldratt tells us that, unless we are careful, we often get the opposite of what we intended. Let's look at the some of these unintended consequences.

Estimating

When you are asked to estimate a task, ask yourself if this example sounds familiar. You think about the task and the effort and decide that you can do the task in 5 days. Then, you think a little bit more. There may be something unfamiliar in the task. You worry about the effect of unplanned work interruptions. Finally, you want to make sure that you won’t be late on your estimate because you don’t want negative attention. Based on all this uncertainty, you announce that you can do the task in 10 days.



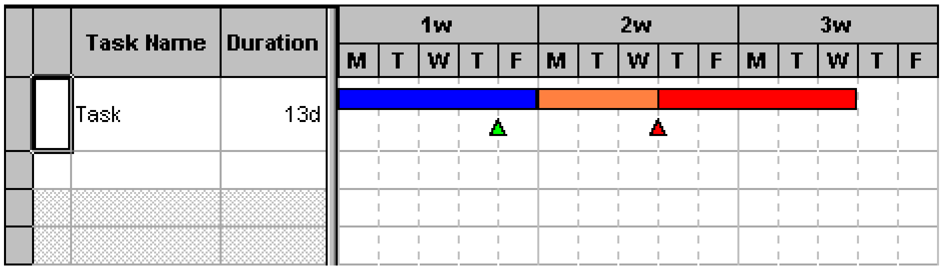
Task #	Task Name	Duration
1	Task	10d

As the above illustration shows, you have hidden 5 days of safety in your 10-day estimate. We say that the safety is hidden because the task is entered in the project as a 10-day task; the 5-day safety is your private contingency factor. It’s important to note that the establishment of a safety factor in a task estimate isn’t wrong. It’s a perfectly reasonable thing do to considering the factors involved and the project management environment in which you work. After all, you don’t want to be the one that misses a task due date.

Given a project that is composed of tasks with hidden safety, let’s take a look at what happens when the task is actually performed.

Student Syndrome

In his business novel, *Critical Chain*, Goldratt tells the story of what happens when a professor gives a class assignment that is due in two weeks. The students complain that the assignment is tough and will require more time. The professor agrees and gives them additional time. Later, when the students look back on how they actually performed the assignment with this additional time, they note that they all thought they had plenty of time, with safety, to do the assignment so they put off starting until the last minute anyway. Let’s look at how this student syndrome can affect your task, and the whole project.



Task Name	Duration
Task	13d

Given the student syndrome, you put off really getting to work until fifth day of the task as noted by the green milestone in the above illustration. This start should be ok because you have adequate safety in your estimate. Unfortunately, on the Thursday noted by the red milestone, you encounter an unexpected problem with the work. Suddenly, you realize that your safety is gone, and that you will overrun your estimate no matter how hard you work. You spend the next five days working as fast as you can in the red zone, with an overrun of 30% of your original estimate

Like the students in *Critical Chain*, you objectively look back on the task. You note that you wasted four days of safety in your slow start on the task. You also note that the problem appeared when you were at the 80% point of your original estimate and that there was simply not enough safety left to recover.

This simple single task example is not unusual. It happens over and over again in the completion of a project. We are all human, and when we establish a task schedule with a hidden safety margin, most of us naturally fall into the student syndrome.

Parkinson's Law

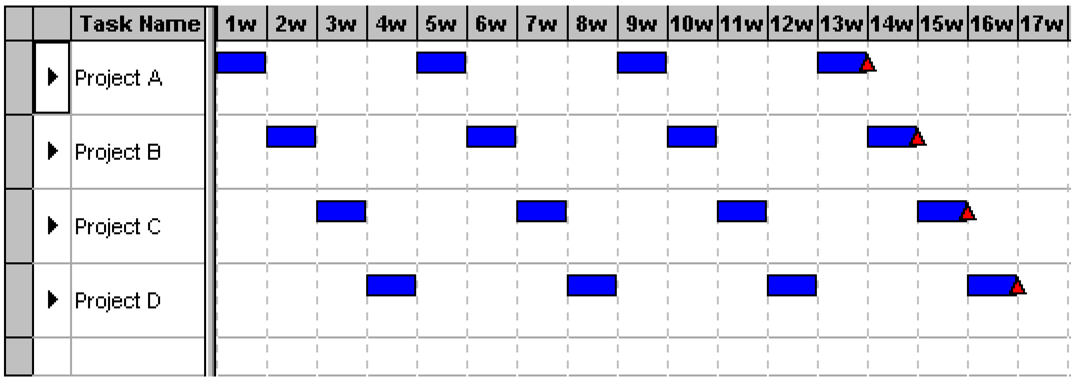
Work expands to fit the allotted time. Most of us have heard about Parkinson's Law and seen it in action on projects. If a task is estimated to take 10 days, it usually doesn't take less. This adjustment of effort to fill the allotted time can come in a number of ways. Software projects often exhibit a tendency towards creeping elegance when the developers sense that they have more time than actually necessary on a task. In other cases, people will simply adjust the level of effort to keep busy for the entire task schedule. As we will discuss later, traditional project environments stress not being late, but they do not promote being early. This environment encourages hidden safety, the student syndrome, and Parkinson's Law effects.

Multi-tasking

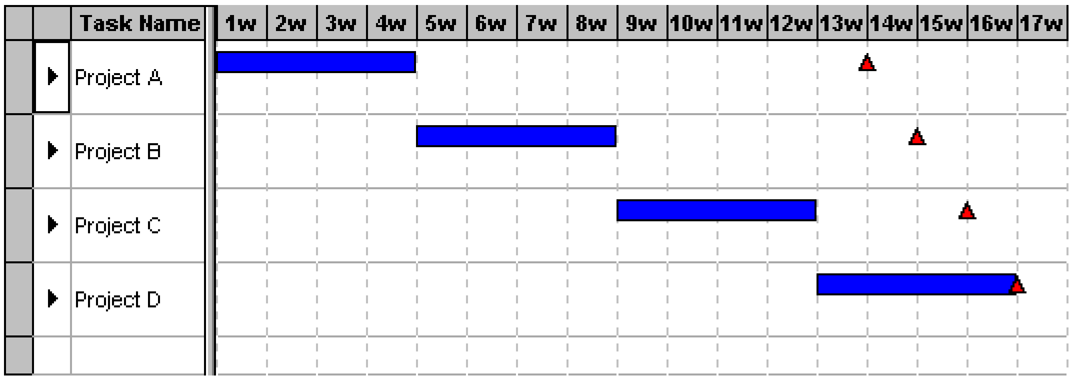
Most of us work in a multi-project environment. We all have experiences of having to stop working on one task so that progress can be accomplished on another task in another project. Often, we wonder if all this jumping around makes sense because it comes with the penalties of reduced focus and loss of efficiency. However, there is a reason for this multi-tasking environment.

Project managers are responsible to a customer for successful completion of a project. These customers can be internal or external to an organization. Customers have a tendency to be demanding. They think that their project is the highest priority and they want to see frequent progress on their project. Resources tend to migrate between projects in response to the latest, loudest customer demand in an attempt to keep as many customers satisfied as possible. This focus on showing progress on as many active projects as possible is the major cause of multi-tasking. As we will see, this focus is to the detriment of the overall project thru-put of the organization.

Let's consider the bad effects of multi-tasking in a simple multi-project example. Assume we have four projects, A, B, C, and D each of which is estimated to take four weeks to complete. Our project environment is one of organized chaos. Resources migrate from one project to the next to show as much simultaneous progress as possible to the project customers. To keep this example simple, lets assume resources work one week on each project and then migrate to the next project. In this environment, the projects are accomplished in intermittent spurts as shown in the illustration. The completion date of each project is noted with a red milestone. Note that this example assumes zero efficiency loss due to changing tasks so it minimizes the real-world bad effects of multi-tasking.



Now, let's assume we get organized with the simple goal of doing work based upon which projects are most important to your organization. This is an important change, we are moving from organized chaos based upon sub-optimized micro-level decisions to an optimized organization based upon macro-level decisions. For our example, let's assume the project priority, from highest to lowest is, A, B, C, and D. By eliminating multi-tasking and executing our projects by priority, we get the results illustrated below.



Note how the lowest priority project, D, is still accomplished on the same date as the multi-tasking example as depicted by the red milestone. However, let's look at our highest priority project, A. Project A is done nine weeks sooner; that's a 225 % improvement. Projects B and C also are done in much less time than in the multi-tasking environment. The message is clear, if you eliminate multi-tasking and make resource allocation decisions based upon project priority, you get better performance on your projects.

The elimination of multi-tasking also applies within a single project. The demanding customers can be work package managers who demand progress from limited resources. If the resources are allocated to silence the squeaky wheels, the project can suffer unnecessary delays as tasks are performed in an un-optimum sequence. Later, we shall see how the Critical Chain method gives us a simple method for eliminating this intra-project multi-tasking with clear, concise rules for which work should be done first.

No Early Finishes

Did you ever notice that tasks seem to either finish on time, or late, but rarely early? We have already discussed how the student syndrome and Parkinson's Law contribute to this common outcome, but there is another factor. Our project management methods, including rewards and punishments, rarely reward early finishes. In fact, they often punish early finishes. Let's next discuss a number of reasons why this is so.

If you finish a task earlier than planned, you might be accused of sandbagging your estimates instead of being rewarded for completing ahead of schedule. In this environment, you worry about your future estimates being cut based upon history so you quietly enjoy the lull that your hidden early completion gives you, and officially finish on schedule. In this case you will probably get accolades for good estimating even though you know you could have finished earlier.

If you finish early and announce your results, you then encounter the next problem. The task that is dependent upon your completion might not be able to start early because the required resources are off doing something else. Remember, the project schedule gave a clear start date for the following task and the resources were allocated elsewhere based upon this schedule.

When you integrate student syndrome, Parkinson's Law, with the likelihood of no early finishes, you get the following result. Traditional project management methods loose the effects of early finishes and only propagate late finishes in the schedule. In other words, the best they can do is to finish on time, and the likelihood of that happening is small. To get a different result, to get faster time to market, you need a different approach, and that is what Goldratt's Critical Chain delivers. Let's see how the Critical Chain Method delivers better project performance.

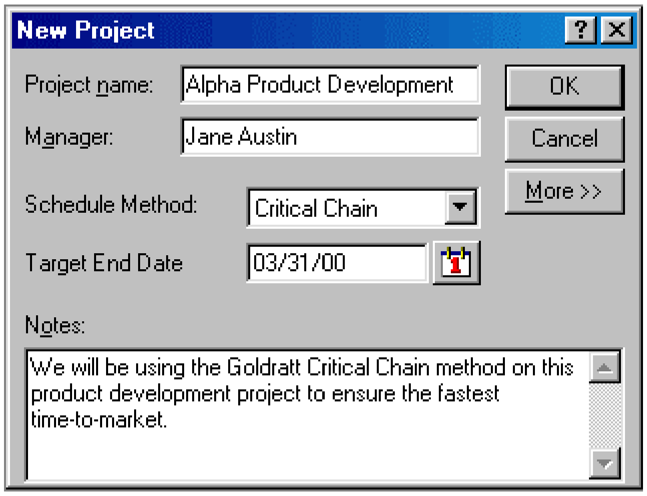
Critical Chain Method

We will discuss Critical Chain Project Management (CCPM) as it applies to the two phases known to every project manager, planning and tracking. As we shall learn, Critical Chain scheduling operates differently in these two phases and therefore Scitor PS8 provides two modes: Planning and Tracking to support this different operation. Let's start at the start with Planning Mode.

Planning Mode

Scheduling Backwards

In Critical Chain Planning mode, you develop a plan backwards in time from a target end date for your project. This focus on completion date is natural. When you get assigned a new project you are usually told when the results are needed as opposed to the project start date. It's your job to determine when you have to start to meet the target end


New Project

Project name: Alpha Product Development

Manager: Jane Austin

Schedule Method: Critical Chain

Target End Date 03/31/00

Notes:

We will be using the Goldratt Critical Chain method on this product development project to ensure the fastest time-to-market.

date. PS8 gives you this information when you develop your plan.

Let’s talks a little about planning backwards. This doesn’t mean that you have to think backwards. Instead, it means that as you define your project with tasks, durations, and dependencies, PS8 schedules your tasks backwards in time from your defined end date. When you are done with your plan, the calculated project start date tells you the latest date you can start that will still meet the target end date.

Now, there is a school of thought that also advocates that you develop your project plan by actually thinking backwards from your project objectives using a network drawing as the medium for defining the tasks and their relationships. This method existed well before Goldratt invented Critical Chain and is not fundamental to the CCPM method. The approach is based upon the reasoning that working backwards is counterintuitive and therefore you are less likely to create unnecessary task dependencies based upon past practices. Additionally, because you start with the objectives and work back, you are also less likely to add tasks that don’t add value to the objectives. However, this backward thinking approach is, as it was designed to be, counterintuitive. If you find it difficult to think this way, and many people do, don’t do it.

As-Late-As-Possible Scheduling

In traditional Critical Path scheduling, your tasks are scheduled as-soon-as-possible (ASAP) from the project start date. This scheduling places work as close as possible to the front of your schedule. In Critical Chain planning, your tasks are scheduled as-late-as-possible (ALAP) based upon the target end date. This as-late-as-possible scheduling places work as close as possible to the end of your schedule. There are many benefits to delaying project work as-late-as-possible, and one drawback.

Delaying work as late as possible has many benefits. Using a production analogy, you are minimizing work-in-progress (WIP) and not incurring costs earlier than necessary. From the project manager’s viewpoint, there is better focus at the critical start of the project because there simply aren’t as many tasks scheduled to start. Importantly, in complex, knowledge work, your knowledge increases the further you go into the project. By scheduling tasks as-late-as-possible, you are capitalizing on this increasing knowledge and will significantly minimize the need for re-work.

The single drawback is directly related to the scheduling of all work as-late-as-possible. In traditional Critical Path terminology, this means that all tasks are critical once you are in tracking mode. An increase in duration of any task will push out the project end date by the increased amount. Fortunately, Goldratt has a simple, elegant solution to this problem. In Critical Chain planning, as you will read later, you insert buffers at key points in the project plan that will act as shock absorbers to protect the project end date against task duration increases. With the buffer approach, you get the benefits of as-late-as-possible scheduling with adequate protection against uncertainty.

Task Estimating

Critical Chain task estimating requires a change in individual and organizational behavior to be effective. You want to remove the hidden safety in the task durations (see the

Human Side section for a discussion of Hidden Safety). Now, because this safety is hidden, you have to establish an organizational culture that removes the fear of exposing this safety and removing it from task estimates.

First, you have to get everyone to understand that you are not removing safety and throwing it away. Instead, as we shall see later, we are pooling this removed safety as a project resource as opposed to a hidden task-level resource.

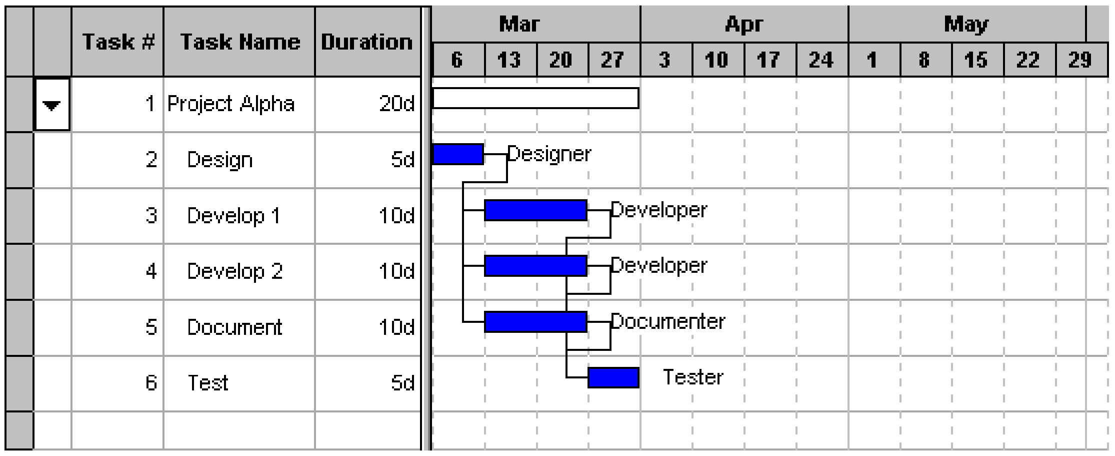
Next, you have to get management and the project team to embrace uncertainty rather than think that they can defeat it with better estimating techniques. When you remove the safety from a task, you must accept the fact that this task has a good probability, say 50%, of exceeding the estimate. This is not bad; it is normal. You can't have an environment where task actuals are analyzed against baseline estimates, and reported and treated as problems. If you do, then the team will adjust their behavior to the measurements, and will put lots of deeply hidden safety back in their next estimates thereby defeating the method.

Given an agreement by all involved to accept the new Critical Chain paradigm, you finally come to the "how" of estimating. There are two basic approaches. One involves the development of a single estimated duration for each task. The second approach requires the development of two duration estimates for each task. PS8 supports both methods; however, the single duration method is what Goldratt originally proposed. We will use it in this discussion because it is simple and effective.

The goal is to get a task estimate that has a 50% chance of being met. This means that there is also a 50% chance that the task will take longer than the estimate. And, of course, there is some chance that the task will take shorter than the estimate. Now, most people have trouble thinking in terms of probability of outcome when it comes to task estimating. Instead of asking for a 50% estimate, you should ask the team to provide estimates that are based upon a number of positive assumptions as follows.

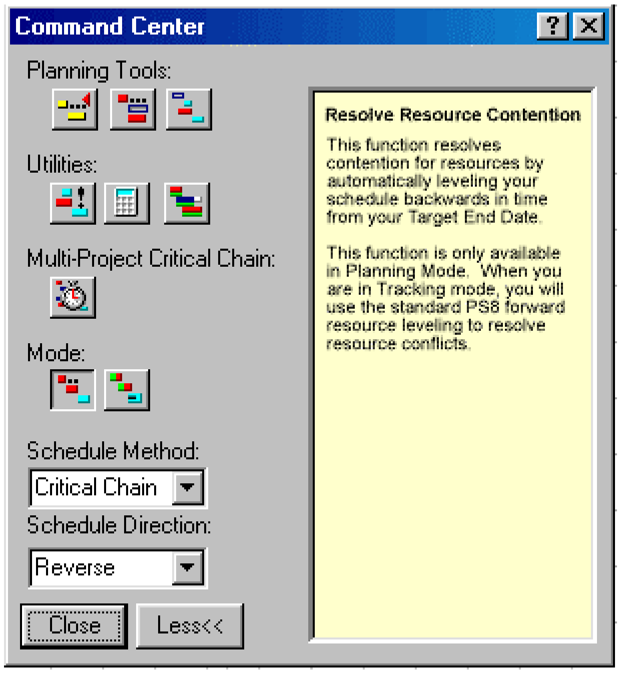
When deriving an estimate you should assume that all the material and information needed for the task is on hand. You should assume that you are able to focus on the task without any interruptions. Finally, and most importantly, you should assume that there won't be any surprises that cause additional work. If you use these assumptions, you will be off to a good start in deriving a 50% probability estimate without worrying about probabilities.

We will be using a simple project example for the remainder of this section based as illustrated below. In this illustration you see the result of planning a simple Critical Chain project backwards from a Target End Date of March 31, 2000. Note that the estimates were derived with the above assumptions. All tasks are scheduled as-late-as-possible. Given this schedule let's move on to the next step in Critical Chain planning, Resolving Resource Contention.



Task #	Task Name	Duration
1	Project Alpha	20d
2	Design	5d
3	Develop 1	10d
4	Develop 2	10d
5	Document	10d
6	Test	5d

Resolving Resource Conflicts



Command Center

Resolve Resource Contention

This function resolves contention for resources by automatically leveling your schedule backwards in time from your Target End Date.

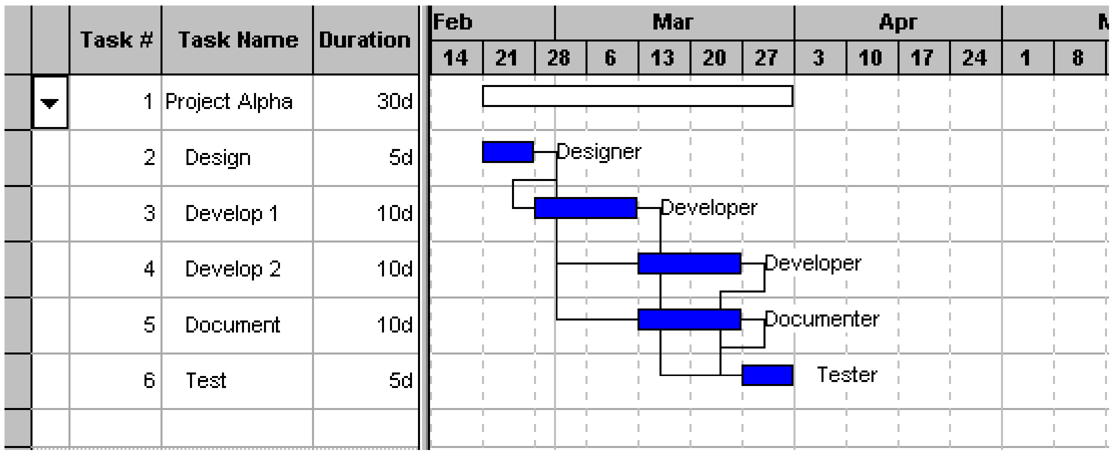
This function is only available in Planning Mode. When you are in Tracking mode, you will use the standard PS8 forward resource leveling to resolve resource conflicts.

Schedule Method: Critical Chain

Schedule Direction: Reverse

Our project has a conflict between the use of the Developer resource on the Develop 1 and Develop 2 tasks (the resource used is shown to the right of the task bar in the Gantt Chart). We need to resolve this conflict before moving on to identifying the Critical Chain. With PS8, you simply use the Resolve Resource Contention function that is available in the Tools Menu under Critical Chain, or in the Critical Chain Command Center. The Critical Chain Command Center brings all of the PS8 Critical Chain functions into one place with a tool icon for each function. Additionally, a full description of the functions and their use is provided in a help pane that is sensitive to which icon you have the mouse pointer hovering over. This PS8 Command Center approach takes the mystery out of Critical Chain and puts all of its power at your fingertips.

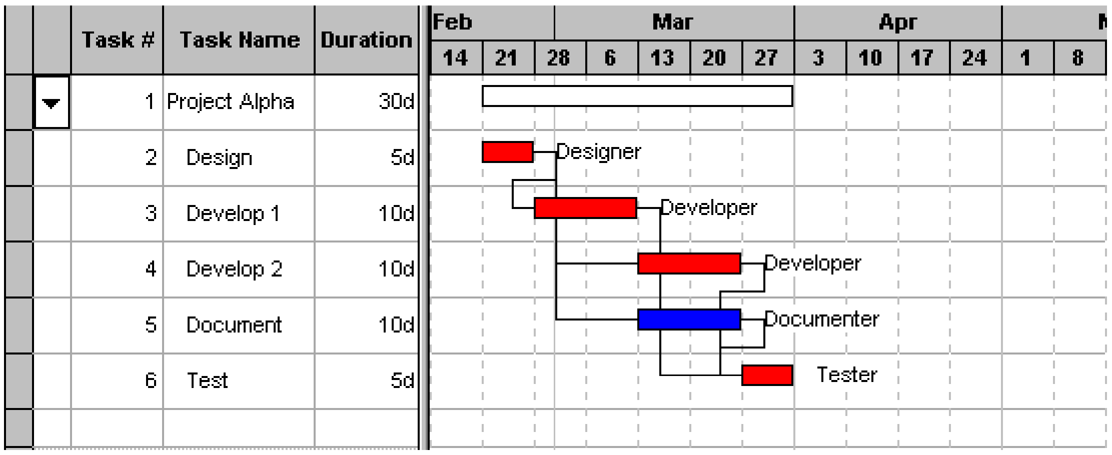
As noted in the Command Center help pane, the Resolve Resource Contention function performs automatic resource leveling backwards in time from your Target End Date. The following illustration shows our example after resolving resource contention. Note how the conflict on the Developer resource has been resolved by moving the Develop 1 task earlier in time. Also note how this leveling was accomplished backwards from the March 31, 2000 Target End Date. Now, we are ready to identify the critical chain.



Task #	Task Name	Duration
1	Project Alpha	30d
2	Design	5d
3	Develop 1	10d
4	Develop 2	10d
5	Document	10d
6	Test	5d

Identifying the Critical Chain

Goldratt defines the Critical Chain as the longest chain of tasks that consider both task dependencies and resource dependencies. This is different from the definition of the Critical Path, which is defined as the longest chain of tasks based upon task dependencies. This is a subtle, but important difference. Critical Chain recognizes that a delay in resource availability can delay a schedule just as a delay in dependent tasks. PS8 provides a function for finding the Critical Chain based upon both task and resource dependencies. When you use this Identify Critical Chain function on our example, you get the results illustrated below. Note that the Critical Chain is clearly displayed in red. We are now ready to calculate and insert buffers in our schedule.



Task #	Task Name	Duration
1	Project Alpha	30d
2	Design	5d
3	Develop 1	10d
4	Develop 2	10d
5	Document	10d
6	Test	5d

Inserting Buffers

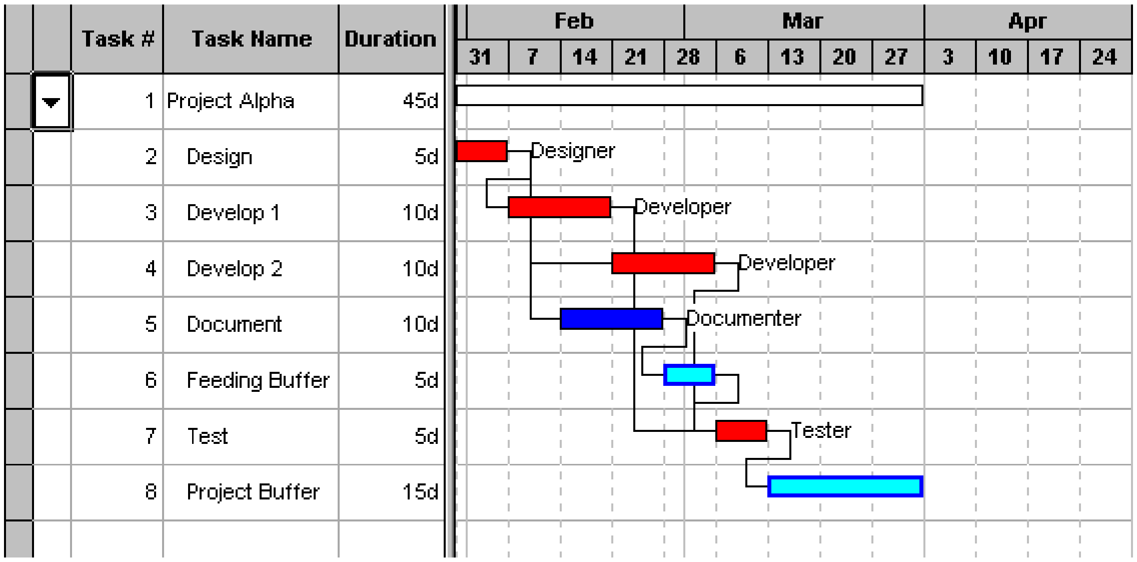
We have effectively removed the safety from our tasks based upon our estimating technique. Now we are going to form a pool of this safety and place it in shock-absorber buffers at key points in our project. These buffers automatically contract when they are pushed by overrunning tasks and absorb the overruns without affecting the Target End Date. When inserting buffers we need to determine the size of the buffer. Let's start with the simplest buffer, the Project Buffer.

The Project Buffer protects the target end date against overruns in Critical Chain tasks. It is placed at the end of the project after the last Critical Chain task. For our example, we will size this project buffer at 50% of the length of the Critical Chain tasks. In effect we have reduced the total safety that would have been hidden in the individual Critical Chain tasks and placed the reduced amount in the Project Buffer. This reduction in total safety is based upon the same theory of pooled risk that is used in the insurance business.

The Project Buffer protects us from overruns on the Critical Chain. However, the Critical Chain is exposed to overruns from non-Critical Chain tasks that link to the Critical Chain. In our example, the Document task is a non-Critical Chain task that feeds into the Critical Chain Test task. The Document task has been scheduled as-late-as-possible so any increase in it's duration during tracking, will effect the Critical Chain and the Project Buffer. Goldratt protects the Critical Chain against overruns on these Feeding Chains by inserting a Feeding Buffer at the point where the Feeding Chain intersects with the Critical Chain. For our example, we will size the Feeding Buffer at 50% of the length of the feeding chain.

When we use PS8's Insert Buffer function, the buffers are automatically inserted and any resource contention caused by the buffer insertion is resolved. We get a complete Critical Chain project as illustrated below. Note the light blue Feeding Buffer and Project Buffer that have been inserted into our project. These buffers protect our Target End Date from any increases in task durations.

Now, we are ready to implement our Critical Chain project by going into Tracking Mode with PS8. Before we do this, it is important to note that Critical Chain project, as planned with buffers, is already 25% shorter than the equivalent Critical Path project.



Task #	Task Name	Duration
1	Project Alpha	45d
2	Design	5d
3	Develop 1	10d
4	Develop 2	10d
5	Document	10d
6	Feeding Buffer	5d
7	Test	5d
8	Project Buffer	15d

Tracking Mode

You go from Planning Mode to Tracking Mode with a single click of a button in the PS8 Command Center. Remember, in Planning Mode, you have been developing a project schedule backwards in time from your Target End Date. Now, in Tracking Mode, you will be tracking your project schedule forward in time with the goal of meeting your project Target End Date.

In Tracking Mode, PS8 functions exactly as it does in the traditional Critical Path method with one exception. The project end date will not change until your Project Buffer has been completely absorbed by task overruns. As a Critical Chain Project Manager, it’s your job to manage the work to protect the buffers so that you finish earlier than the project Target End Date. As you might expect, Critical Chain has a unique approach to managing the project work, which we will discuss next.

Relay Race Approach

In traditional Critical Path project management, the schedule is the vehicle for clocking the work. Every task has a published start and finish date. Resources have access to their list of tasks with scheduled start and finish dates. As the schedule is updated, the task dates move later in time to reflect slippages. While this concentration on task scheduled start and finish dates seems logical, it does not promote speed-to-market driven project performance. In fact, it ensures that early finishes are lost, and only late finishes accumulate in the schedule. Let’s look at a relay race analogy to understand this problem.

The goal of a relay race is to win. Each runner runs his or her leg as fast as possible and hands the baton off to the next runner who is ready to run as soon as the preceding runner is finished. The competitive excitement of the race encourages the runners to do their best and beat their best times. Because runners capitalize on an early finish of the preceding runner, a fast leg can offset a slow leg to the benefit of the team.

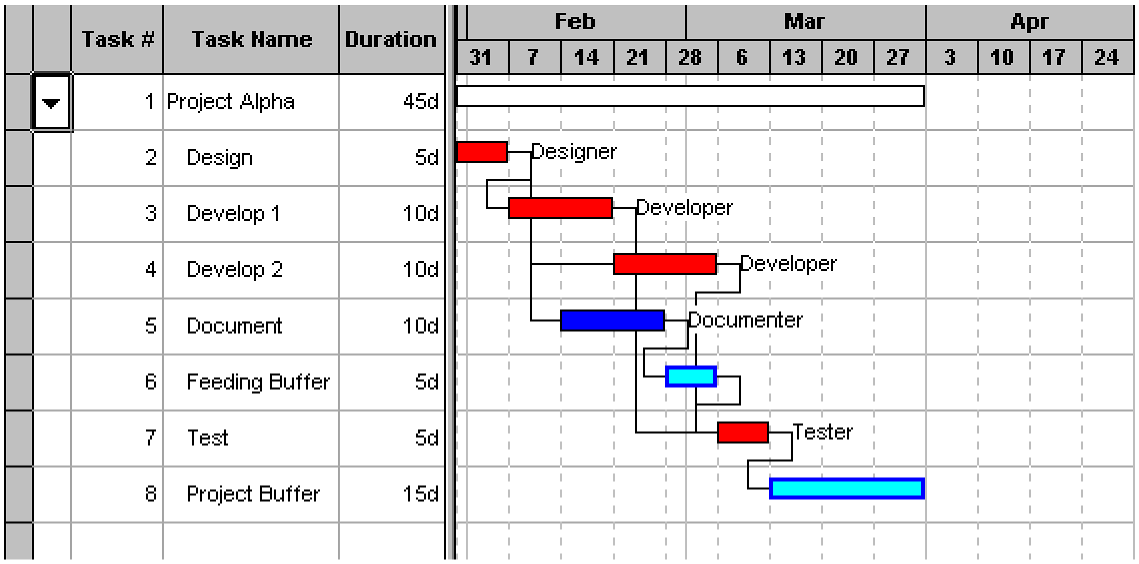
You want to manage your Critical Chain project using this relay race approach. This means that when one task is getting close to completion, you must have the next task’s resource on the track and ready to go as-soon-as-possible after the preceding task completes.

This relay race approach means that you must get your team to de-emphasize the task scheduled start and finish dates and concentrate, instead, on triggering their preparation and start on the preceding task’s progress. Importantly, once a task is started, the resources work as fast as possible towards completion without clocking themselves to the scheduled finish date.

PS8 provides a Resource Alert report to help you in this relay-race management approach based upon Goldratt’s concept of a resource buffer. The Resource Alert report, published on your project Intranet, gives each resource a heads-up alert that they should start getting ready to work on a new task before the scheduled start date of the task.

In the development of our Critical Chain schedule, all tasks were scheduled as-late-as-possible with buffers inserted to protect the Target End Date. As you apply the relay-race approach to this schedule, it means that a chain of tasks should start no earlier than scheduled, but once started, you must finish the chain as fast as possible.

Looking at our simple Alpha project example below, we have two chains. We have the Critical Chain displayed in red beginning with the Design task, and we have a single feeding chain consisting of the Document task shown in blue. We would start a relay race on the Critical Chain tasks beginning with the Design task as soon as the project starts. We would start a relay race on the single feeding chain with the start of the Document task on February 14.



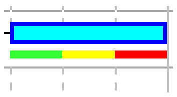
Task #	Task Name	Duration
1	Project Alpha	45d
2	Design	5d
3	Develop 1	10d
4	Develop 2	10d
5	Document	10d
6	Feeding Buffer	5d
7	Test	5d
8	Project Buffer	15d

Schedule Updating

As in traditional project management, you update your schedule on a periodic basis by entering the completed duration on your tasks and updating the remaining duration with an estimate of the work needed to finish the task. Because you recognize that estimates are uncertain, you don't worry when a particular task overruns its estimate. Instead, you take a big picture approach and simply watch the effect of many tasks on your buffers.

Buffer Management

Buffer management is the key to tracking project performance in Critical Chain project management. Again, because you recognize that task estimates are uncertain, you don't try to apply exact measurement techniques such as variance analysis and earned value to your tasks. Instead, you watch your buffers and act depending upon how much of the buffer is penetrated by task schedule changes.



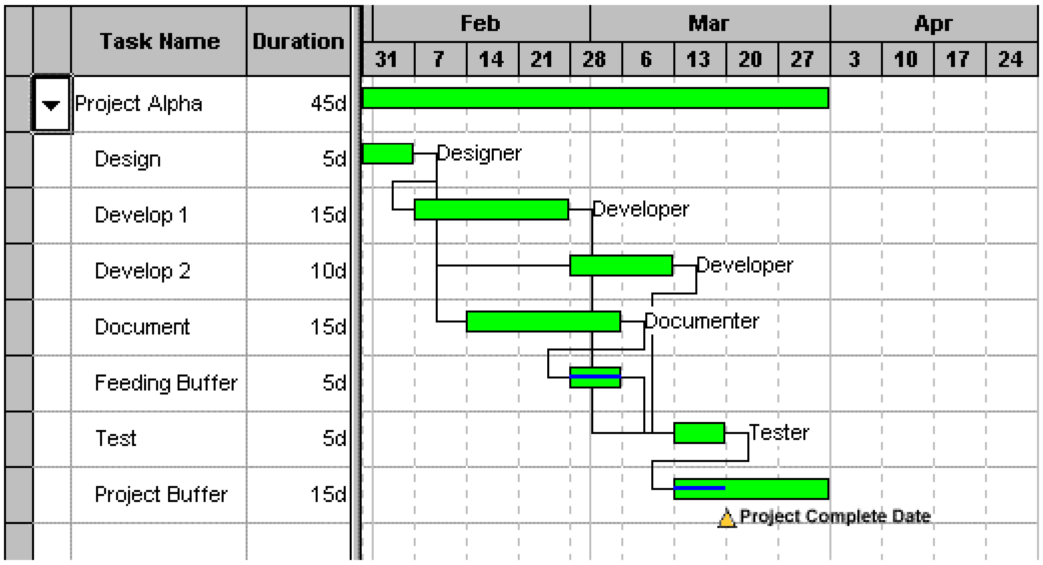
Treat the buffer as if it were divided into three equally sized regions. The first third is the green zone, the second third is the yellow zone, and the final third is the red zone. If the penetration is in the green zone, no action is required. If the penetration enters the yellow zone, then you should assess the problem and think about possible courses of action. If the penetration enters the red zone, then you should act. Your action plans should involve ways to finish uncompleted tasks in the chain earlier or ways to accelerate future work in the chain to bring the buffer penetration back out of the red zone.

Resource Allocation Decisions

When faced with a resource conflict during the project, we all understand that the allocation decision should be based on applying the resource to the highest priority task. However, highest priority is often in the eyes of the beholder. When task mangers are competing for a resource, each manager considers their task to be of the highest priority. Critical Chain offers a simple, consistent way to resolving this conflict. When faced with a resource allocation decision, allocate the resource to the task with the goal of minimizing buffer penetration. With PS8 this is a snap. You can easily try different alternatives, look at the buffer report, and use the unlimited undo/redo to back out of each alternative. This approach solves two problems. First, it takes politics and persuasion out of the allocation decision process. Second, it optimizes the use of resources towards the true highest priority goal: finishing the project early.

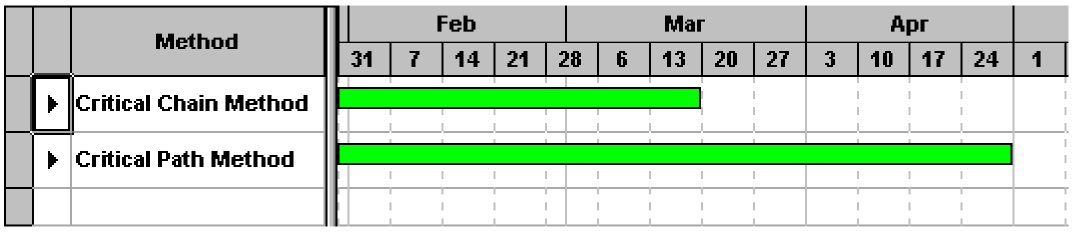
Finish Early

By using the Critical Chain project management method, you should have a goal of not just finishing on time, but of finishing early as depicted below. Note how the buffers absorbed a 50% increase in the duration of the Develop 1 and Document tasks while the project still finished early with two thirds of the project buffer remaining.



Task Name	Duration
Project Alpha	45d
Design	5d
Develop 1	15d
Develop 2	10d
Document	15d
Feeding Buffer	5d
Test	5d
Project Buffer	15d

To demonstrate the effectiveness of the Critical Chain method on project performance, let’s look at a simulation of the above project using Scitor Process. Scitor Process is a powerful tool for defining and simulating any process. We built two models of our sample project. The first model used traditional project management methods. Importantly, it modeled the effects of student syndrome, procrastination, and not capitalizing on early finishes. The second model used the Critical Chain method including the relay race management technique. We ran 100 iterations of each model with the results illustrated below. The Critical Chain model finished, on average, about 45% faster than the Critical Path model.



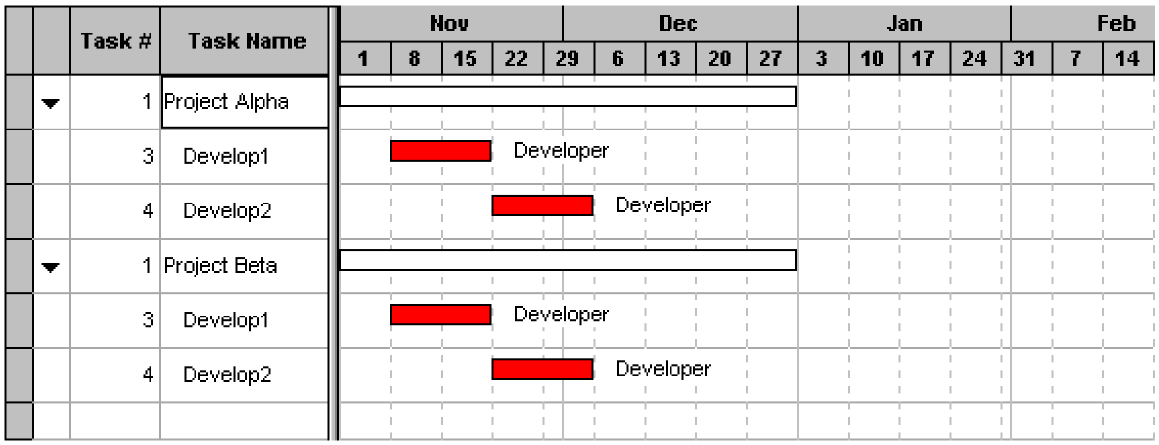
Multi-project Critical Chain

The Critical Chain approach to multi-project management is based upon the goal of maximizing the total project throughput of the organization based upon project priorities within the constraints of key resources. The implementation is simple. You want to introduce new projects into your multi-project mix at a point that avoids conflict with key resources allocated to existing or higher priority projects. This staggering of projects based upon key resource constraints is known as project synchronization. You are, in effect, synchronizing your project schedules based upon key resource constraints.

Goldratt uses the term “drum resource” to refer to the key, critical resource around which projects will be synchronized. This term was first used in Goldratt’s early production optimization work where he developed the successful “drum-buffer-rope” approach. In production systems, the drum is the constraint of the system. This constraint sets the pace of the whole production system just as a drum is used to set the marching pace of a troop. In the project management arena, the drum resource is usually the resource that is always in the most demand or is so limited by capital costs that it is the bottleneck around which schedules are drawn. So far we have been referring to the drum resource in the singular. You can have more than one drum resource.

You need to synchronize projects around resource constraints because if you don’t you will introduce chaotic multi-tasking into our environment with the results described earlier in the Human Side discussion. This chaotic multi-tasking environment is detrimental to priority –based project throughput as well as team morale.

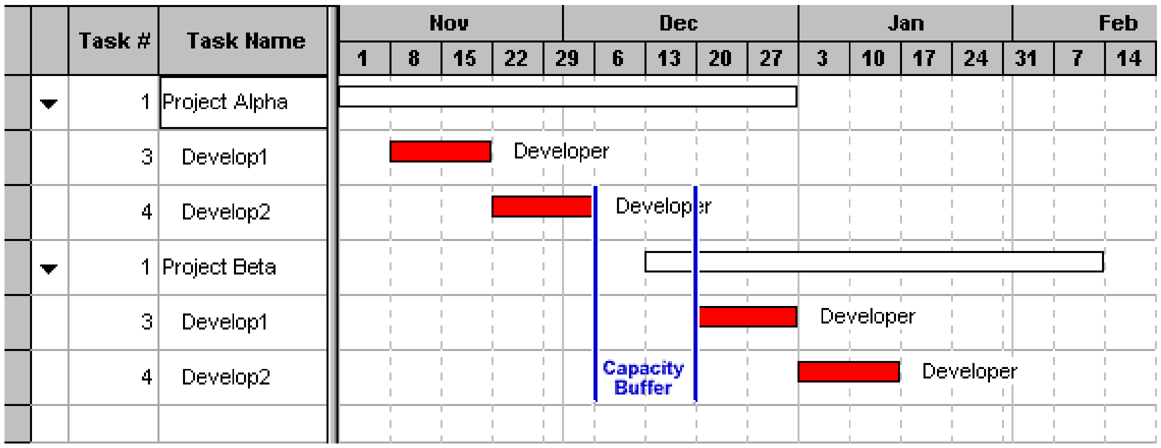
Using PS8’s Project Synchronization function, you have full control over your project mix. You can set project priorities and even fix certain projects so they won’t move during synchronization. The unlimited undo/redo facility allows you to try something, look at the results, and then back-out to try something different. Let’s look at a simple example of project synchronization as illustrated below.



Task #	Task Name
1	Project Alpha
3	Develop1
4	Develop2
1	Project Beta
3	Develop1
4	Develop2

The illustration shows two projects, Alpha and Beta. For simplicity, the projects are identical copies of the example we have been using throughout the section. PS8's task filtering capability has been used to focus on tasks that use the drum resource which is the Developer resource in our example.

When we invoke the Project Synchronization function in PS8, we get a dialog that shows us a list of all resources and a list of drum resources. We can change the drum resource selection from this dialog. PS8 helps us in picking the drum resource by giving us an option to sort all the resources based upon the amount of cross-project use in the project mix. Additionally, we can define the parameters that PS8 will use to calculate the length of the Capacity Buffer that will be inserted during synchronization. Let's look at the results of synchronizing our projects to understand the concept of a Capacity Buffer using the following illustration.



Task #	Task Name
1	Project Alpha
3	Develop1
4	Develop2
1	Project Beta
3	Develop1
4	Develop2

Project Beta has been moved later in time based upon the constraints of the Developer drum resource. A Capacity Buffer has been inserted between the last use of the Developer drum resource in Project Alpha and the first use of the Developer resource in Project Beta. Now, this Capacity Buffer isn't a buffer in the normal sense defined by our Feeding and Project Buffers. Instead, it is simply a gap that has been placed in the use of the Developer resource across projects. This gap acts as a protection for Project Beta against changes in Project Alpha that delay the availability of the Developer resource for Project Beta.

Critical Chain Budgeting

If your project environment requires the development of a cost and effort budget for either submission in a bid or for internal controls and reporting, PS8 provides you with the tools for defining the budget. Your final Critical Chain plan is based upon estimates that have about a 50% chance of accomplishment for any one task. You need to adjust the effort and cost elements of this plan to obtain a budget with an adequate safety margin.

Using the Save Critical Chain Budget function, you specify the margin parameters and PS8 automatically increases the task durations, with a corresponding increase in cost and effort, reschedules the project based upon the increased durations, saves the entire profile in one of the five available baselines, and then restores the project back to its original state. This baseline provides a detailed time profile of efforts and costs that occur over the approximate duration of the project including the buffer times. If you compare the base plan costs with the budget costs using cumulative curves, you will see that the budget is higher, and occurs over a longer time period than base plan.

Summary

Critical Chain project management provides the following major benefits to your project organization:

- Projects will be completed faster

- The project team's morale and effectiveness will improve because they will be operating in an environment that is comfortable with uncertainty and that avoids micro management.

- Project managers, resource managers, and executives will have a simple, highly effective macro-level method for evaluating project performance and making resource decisions using green-yellow-red buffer management.

- Executives will have an effective tool for making decisions on projects based upon project priority and organizational capacity using the project synchronization capabilities.

To achieve the above benefits, you need to establish a total project environment that integrates both the human behavioral elements and the methods into an effective operating unit. PS8 makes the implementation of the methods easy with its seamless integration of Critical Chain functionality into the best software foundation in the industry. The human side requires everyone, from management to the project team, to understand and buy-in to the concepts. Scitor offers a comprehensive range of training and consulting services to help you achieve a successful Critical Chain project management operation.