Chapter II

Literature Review

2.1 Traditional Project Management

2.1.1 Critical Path Method (CPM)

The Critical Path Method (CPM) is a technique for determining the amount of scheduling flexibility (the amount of float) on various logical network paths in the project schedule network, and to determine the minimum total project duration. Early start and finish date are calculated by means of a forward pass, using a specified start date. Late start and finish dates are calculated by means of a backward pass, starting form a specified completion date, which sometimes is the project early finish date determined during the forward pass calculation. However, CPM requires assumption that the duration of each activity is known with certainty (Carter and Price, 2001; Hillier and Lieberman, 2005; Project Management Institute, 2004; Winston, 2004).

The CPM was developed in the 1950s by DuPont, and was first used in missile-defense construction projects. Since that time, the CPM has been adapted to other fields including hardware and software product research and development. Various computer programs are available to help project managers use the CPM.

In applying the CPM, there are several steps that can be summarized as follows:

- Define the required tasks and put them down in an ordered (sequenced) list.
- Create a flowchart or other diagram showing each task in relation to the others.
- Identify the critical and non-critical relationships (paths) among tasks, where critical path “generally, but not always, is the sequence of schedule activities that determines the duration of the project” (Project Management Institute, 2004).
- Determine the expected completion or execution time for each task.
- Locate or devise alternatives (backups) for the most critical paths.

2.1.2 Program Evaluation and Review Technique (PERT)

The Program Evaluation Review Technique (PERT) is a technique which can be used to estimate the probability that the project will be completed by given deadline (Carter and Price, 2001; Hillier and Lieberman, 2005; Winston, 2004).

PERT was developed by the U.S. Navy in the 1950s to manage the Polaris submarine missile program.

For each activity, PERT requires project manager estimate three times, namely:

- The most likely estimate denoted by m, is the most likely value for the activity's duration.
- The optimistic estimate denoted by a, is the estimate of activity's duration under the most favorable condition. (It is an estimate of the lower bound of the probability distribution.)
- The pessimistic estimate denoted by b, is the estimate of activity's duration under the least favorable condition. (It is an estimate of the upper bound of the probability distribution.)

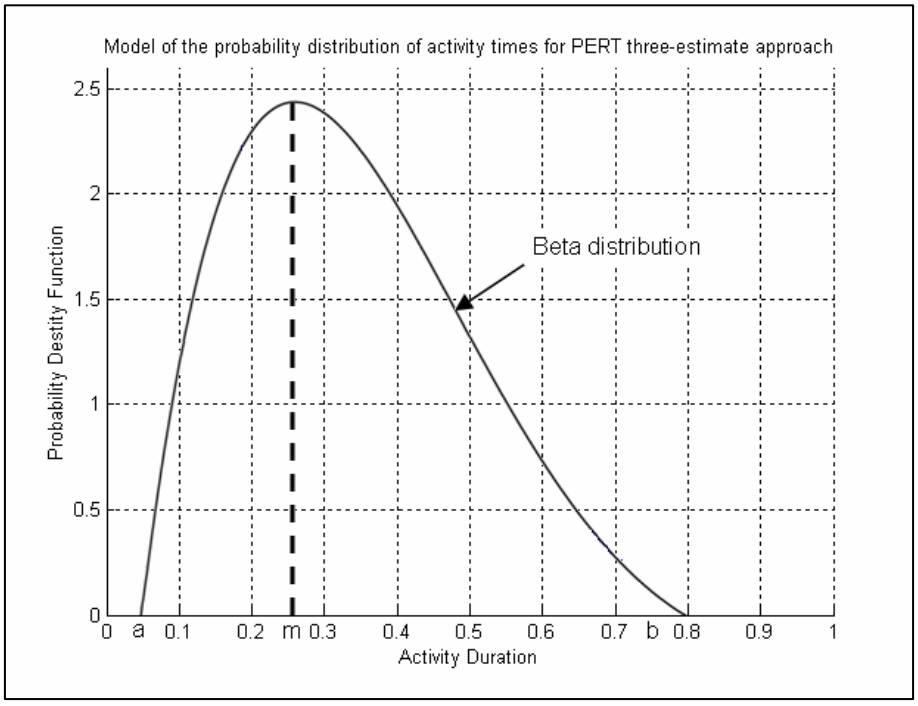

Figure 2-1: Model of the Probability distribution of activity times for the PERT three-estimate approach: m = most likely estimate, a = optimistic estimate, b = pessimistic estimate.

Furthermore, PERT requires four assumptions (Hillier and Lieberman, 2005) as follows:

- The probability distribution of each activity time is a beta distribution.
- The spread between the optimistic estimate and pessimistic estimate is 6 standard deviations.
- The durations of all activities are independent.

- The critical path always requires a longer total elapsed time than any other path.

Under these assumptions, we obtain:

- The expected value of the activity time is approximately

$$t_e = \frac{a + 4m + b}{6}.$$

- The variance of an activity time is:

$$\sigma^2 = \left(\frac{b-a}{6}\right)^2 = \frac{(b-a)^2}{36}.$$

- The expected project time is approximately the sum of the expected activity times on critical path.
- The variance of the project time is approximately the sum of the variance of the activity time on the critical path.

2.2 Undesired Effect of Traditional Project Management

2.2.1 Excessive Activity Duration Estimates

Most project managers include contingency time– difference time between a 50% probable estimate and a 90% probable, (Leach, 2004) –within each activity estimate to explain individual activity common cause variation defined that a cause that is inherent in the system (Leach,1999).

In the most project plans, we found that there are many failures to specify the probability and confidence expected for activity duration estimates. Most failure is provide a quantitative basis for the activity duration estimate (Leach, 1999).

Figure 2-2 illustrates a typical project activity performance time distribution. The solid curve shows the probability of a given time on the horizontal axis. The dotted line shows the cumulative probability of completing the activity in time less than or equal to time on the horizontal axis. We find that there is a huge difference if we add contingency to the 50% probable task estimates, as compared to adding it to the 90% probable task estimate.

Furthermore, in Leach (Leach, 1999) states that "… It should be note that the statistician does not attempt to make any verifiable prediction about one single estimate; instead, he states his prediction in terms of what is going to happen in a whole sequence of estimates made under conditions specified in the operational meaning of the estimate that he chose …"

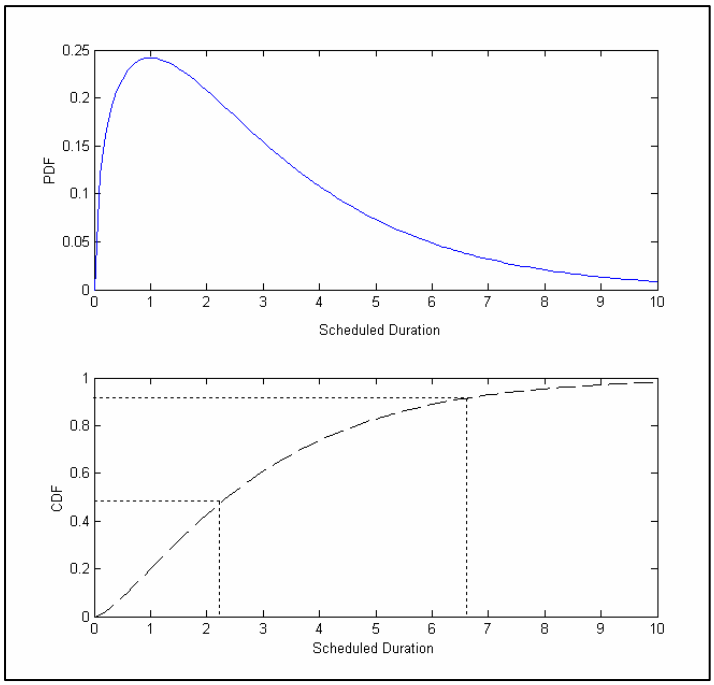

Figure 2-2: Typical Project Activity Performance Time Probability Distributions

Therefore, attempts to deal with uncertainty by including contingency in individual activity estimates are useless, and obviously extend project plan duration.

2.2.2 Student Syndrome

In Goldratt business novel, *Critical Chain* (Goldratt, 1997), he 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.

Figure 2-4 shows student syndrome. Many people do less than a third of the work on an activity during the first two-thirds of the activity duration, and the final two-thirds during the last third of the activity duration. They are more likely to find they have a problem to complete the activity in the remaining time during the last third of the scheduled activity time. If they are working above 100% capacity already to complete two-thirds of the work in one-third of the time, it is unlikely they can keep up to the activity duration. They have littlie opportunity to recover from an unexpected problem, such as computer failures.

The student syndrome tends to waste their contingency time before they start the activity, forcing them to perform most of the work in the later portion of the scheduled activity time. So, if this problem occurs, there is no time to recover.

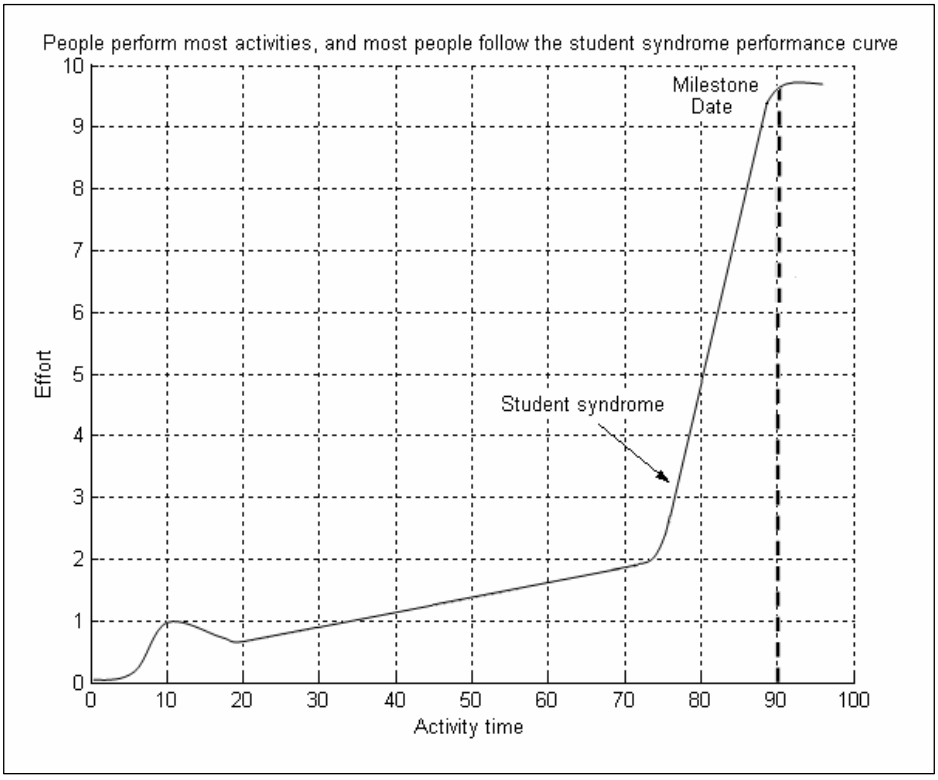

Figure 2-3: People perform most activities, and most people follow the student syndrome performance curve.

2.2.3 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.

2.2.4 Multitasking

Multitasking is the practice of working on two or more significant tasks simultaneously in a manner that is characterized by shifting back and forth between them

before finishing either. Trietsch (2005), Raz (2003) and Herroelen (2005) all agree that multitasking on project activities has a much worse impact.

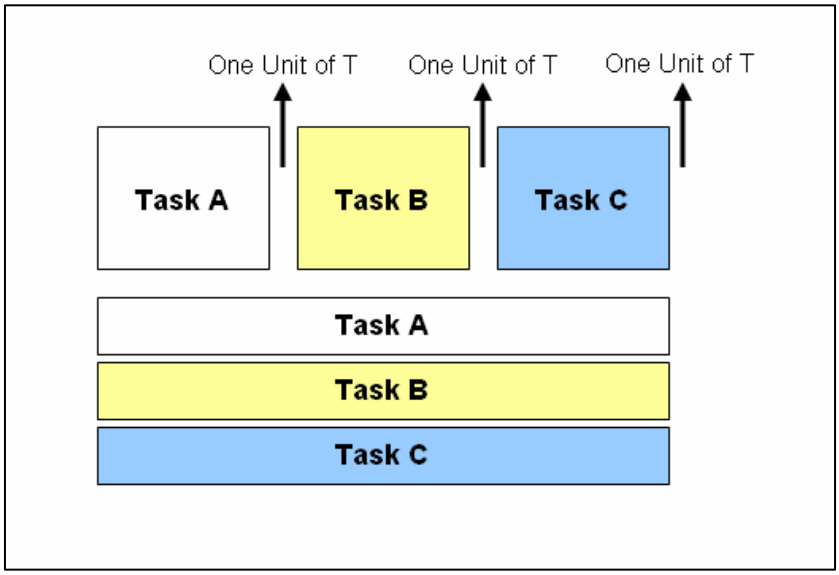

Figure 2-4: Multitasking Extend Activity Duration

From Figure 2-4, we will consider a person who has to do three one-week tasks for three different projects. If that people was allowed to work single-handedly on each one, the first task would be completed in one week. The second task would have its output at the end of second week, and the third task at the end of third week. On the contrary, supposing that that person multitasks by spending one-third time for each day on each task, we found that none of the tasks get their output until the end of the third week. All three activities have three-week duration, in the other word; each task will extend its duration.

2.2.5 No Early Finishes

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 Critical Chain delivers. Let's see how the Critical Chain Method delivers better project performance.

2.3 Critical Chain Project Management Theories

Critical Chain Project Management (CCPM) uses three theories – Theory of constraints (TOC), Common and Special Cause Variation and Statistical Laws Governing Common Cause Variation – to improve project performance. It applied these theories to eliminate aforesaid undesired effects of traditional project management which are the cause of project schedule overruns.

2.3.1 Theory of Constraints (TOC)

The goal: a process of ongoing improvement, which is a novel written by Eliyahu M. Goldratt in 1984, offers a concept of Theory of Constraints to apply in operations management. The Theory of Constraints can be summarized that "Any system must have a constraint. Otherwise, its output would increase without bound, or go to zero." (Goldratt, 2004)

The Theory of Constraints, which can apply to any physical systems, is based on five steps (Goldratt, 2004):

1. *Identify* the system's constraint which is the part of the system that constraints the objective of the system.
2. Decide how to *exploit* the system's constraint.
3. *Subordinate* everything else to the above decision.
4. *Elevate* the system's constraint, and
5. If, in the previous steps, a new constraint has been uncovered, repeat the process. Do not let inertia become the system constraint.

Step 1: The system's constraint is that part of the system that constrains the objective of the system. For money making organizations, (Goldratt ,2004) defines the objective as being to make more money, now and in the future. In production planning terms the system's constraint is the bottleneck. In Step 1 this constraint needs to be identified.

Step 2: For example, if it is a machine, maximum possible utilization needs to be achieved (because it is the objective of the system). So as to exploit the system's constraint, we may command the machine running during the lunch hour, with operators staggering their break, or reducing the number of changeovers. It will mean ensuring that there is always work for the machine to do.

Step 3: If this is the constraint, then there is no point running other machines at a higher production rate: so every other planning decision needs to be subordinated to the schedule required to keep the bottleneck machine running.

Step 4: To improve the objective the system's constraint may need to be elevated. In the case of the bottleneck machine, for instance, it might be run during an additional shift, thus increasing its output. The constraint, the capacity of the machine, has been increased, or elevated.

Step 5: With its increased capacity, the original bottleneck may no longer be constraining the system, as a result the new bottleneck needs to be identified, and the process repeated. Consequently, this is a process of continual improvement. Although this example comes from the production environment, Theory of Constraints is applicable to every system. Critical Chain Project Management seeks to explain how this approach can be applied to project management.

Remark: The difference between Step 2 and Step 4 relates to the amount of investment required, whether in terms of time, effort, money, or willingness. The difference is sometimes concisely expressed as "whatever we can do tomorrow is Step 2". The application of Step 4 may have changed the system's constraint.

2.3.2 Common and Special Cause Variation

Leach (1999) identifies two types of variation as below:

- *Common Cause Variation*: Variation within the capability of a system to repeatedly produce results.
- *Special Cause Variation*: A cause that is specific to some group of workers, or to a particular production worker, or to a specific machine, or to a specific local condition.

Generally, management's function is to improve the system while avoiding two mistakes:

- *Mistake 1*:Treating common cause variation as if it were special cause variation;
- *Mistake 2*: Treating special cause variation as if it were common cause variation.

he states that mistake 1 always decrease the performance of a system; Larry [20] gives the case of machine to explain that the machine had a feedback device attached to measure each part and to adjust the tool location based on that measure to try improve the repeatability of each part. It made the variation in parts much larger, since the measurements included the natural variation of the system to produce parts. The tool simply amplified that natural variation.

Mistake 1 relates to the measurement and control of project performance, and the decisions to take management actions based on those measurement. This phenomenon means that responding to common cause variation as if it were special cause variation will make the system performance worse. In other words, responding to small variances by making project changes decrease project performance.

In many organizations always provide an ongoing set of examples for mistake 2. Something undesirable happens, and they put in place a rule to ensure it never happens again. Finally, they end up with a number of regulations and rules, each applicable to some rare event or events not even applicable to the subject action.

All the estimates in a project plan are uncertain. Performing each of the tasks within a project plan is a single trial of a system (the project task performance system) and is unpredictable. Nevertheless, statistical techniques enable us to predict with known precision the likely results of many trials from a production system and to separate out the special cause of variation requiring corrective action. While knowledge of variation has been used to great profit in production operations, it has not been used to improve project performance. The PMBOK™ Guide and supporting literature we have examined fail to distinguish between common cause variation and special cause variation, a major oversight in the current theory.

2.3.3 Statistical Laws Governing Common Cause Variation

From the Central Limit Theorem: “if a number of independent probability distributions are summated, the variance of the sum equals the sum of the variances of the individual distribution.” Therefore, if n independent distributions with equal variance σ^2 are summated, it follows that:

$$\sigma^2_{\Sigma} = n \cdot \sigma^2 .$$
$$\sigma_{\Sigma} = \sigma\sqrt{n} .$$

where σ_{Σ} is the standard variation of the sum.

Therefore:

$$\sigma_{\Sigma} < n \cdot \sigma .$$

This illustrates the reduction in overall risk when risks are aggregated, because $\sqrt{n}$ is significantly smaller than n, the effect of aggregation of independent risks is significant. The higher the number of risks that are being aggregated, the more marked the effect. (Steyn ,2002).

The statistical method to combine variance means that we can protect a chain of activities to the same level of probability with much less total contingency time than we can protect each individual activity.

Aggregation of the contingency times dramatically reduces the overall estimated time for a chain of activities.

2.4 Critical Chain Project Management: Single Project Case

Goldratt ' s Theory of Constraints (Goldratt,2004) analysis identifies the core problem leading to the most project failure as "failure to effectively manage uncertainty". The core problem leads to five undesired effects which were previously mentioned in 2.2. The Critical Chain Project Management applies *Theory of Constraints*, *Common and Special Cause Variation* and *Statistical Laws Governing Common Cause Variation* to eliminate the causes of these effects.

2.4.1 Identify the constraint – Critical Chain

Goldratt identified the constraint of a project as the critical chain or "the sequence of dependent events that prevents the project from completing in a shorter interval. Resource dependencies determine the critical chain as much as do activity dependencies" (Goldratt, 1997).

We obviously find that the resource constraint is often a significant project constraint, Theory of constraints method of project planning always consider it. Consequently, the critical chain includes the resource dependencies that define the overall longest path (constraint) of the project.

The improvements that result from Critical Chain Project Management do not depend on having significant resource constraints or conflicts in the project. For a project without resource constraints, the critical chain is the same as initial activity path as the critical path.

With regard to the difference of project plan between critical path and critical chain is describe by PMBOK^{TM} Guide 's definition of critical path (Project Management Institute, 2004) which states that "the critical path may change during the performance of project. This occurs when other paths experience delay, and redefines the longest 'zero float' path to complete the project". Nevertheless, Tolerate states that "Critical chain does not change during project performance" (Goldratt, 1997)

To illustrate concept, consider the project shown in figure 2-5. This example of software development project considered in this section was modified form Araki (Araki, 2004). The data and network of example project are shown in table 2-1 and figure 2-5 respectively.

Table 2-1: Data for example project

Activity	Estimated duration (week)	Precedence	Resource multi tasking
A: Survey the Market	3	A < B,D,E	○
B: Design Function	4	A < B < C,F	
C: Design Hardware	3	B < C < G	
D: Order Hardware	6	A < D < G	
E: Merchandise Planning	6	A < E < J	○
F: Design Software Module	2	B < F < H	
G: Hardware Prototyping	5	C,D < G < J	●
H: Design Software Specification	2	F < H < I	
I: Coding	2	H < I < K	●
J: Produce Hardware	4	E,G < J <L	
K: Test and Debug Software	3	K > G,I	
L: QC and put on sale	2	L > J,K	

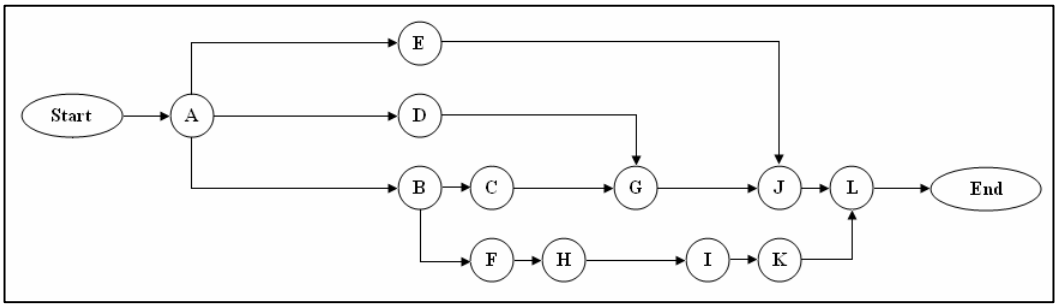

Figure 2-5: Network for example project

The results of critical path analysis are shown in Table 2-2. The path <A-B-C-G-J-L> is the longest, therefore critical path with a length of 21 weeks. However, figure 2-6 shows that CPM has not taken into account resource contention, causing resource conflict occurred in activity I and activity G.

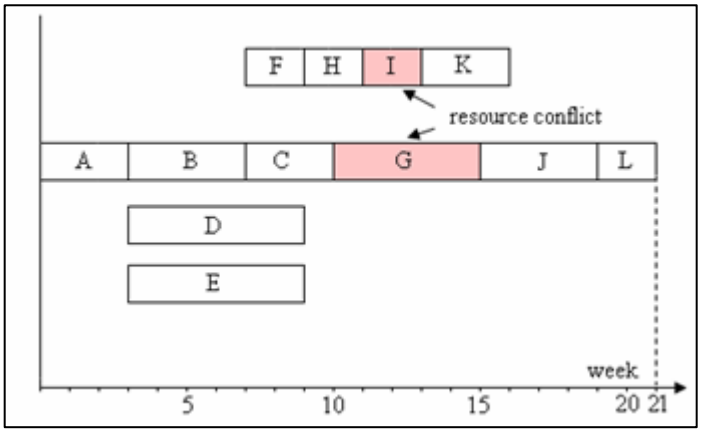

Figure 2-6: Baseline schedule using CPM

Table 1-2: Critical path analysis

Activity	Estimated duration	Earliest start	Latest start	Total slack
Start	0	0	0	0
A	3	0	0	0
B	4	3	3	0
C	3	7	7	0
D	6	3	4	1
E	6	3	9	6
F	2	7	10	3
G	5	10	10	0
H	2	9	12	3
I	2	11	14	3
J	4	15	15	0
K	3	13	16	3
L	2	19	19	0
End	0	21	21	0

For this example, we identify critical chain to be <A-B-C-G-J-L> (the critical chain is set the same as the critical path.)

2.4.2 Exploit the constraint – Project Activity Estimates

CCPM seeks to use the best estimate, or 50% probable, individual activity time estimates. First, gather the plan by using the low-risk activity estimate duration which was provided by the project resourced. Then, instruct the person who estimate duration to understand variation and critical chain method, including assurance that they will not be criticized or other aftereffect which obtains from underrunning or overrunning estimated duration. (Graham, 2000). Next, request their estimate "average" times, assuming everything went as they would hope it would, they have all inputs when they start, and they are able to devote 100% effort to activity as soon as they start. Finally, build the critical chain plan using these reduced activity duration estimates, and collect the difference (D) between the low-risk and average estimates to develop buffers.

Table 2-3: show the excluded contingency duration for project network example in Figure 2-5

Activity	Duration (weeks)	Excluded contingency duration (weeks)
A	3	2
B	4	2
C	3	2
D	6	3
E	6	3
F	2	1
G	5	3
H	2	1
I	2	1
J	4	2
K	3	1
L	2	1

2.4.3 Subordinate everything else to the above decision

2.4.3.1 Subordinate Non Critical Chain Paths

Project manager understand that early start schedules can reduce project risk by getting things done early, while late finish schedules:

- Reduce the impact of chances on work already performed.
- Delay the project cash outlay.
- Give the project a chance to focus by starting with fewer simultaneous activity chains, allowing the project team and processes to come up to speed.

Much project management guidance recommend that project managers use an early start schedule and many scheduling computer programs use early start schedule as the default. However, CCPM use “late start” for all project activities by using buffer.. Feeding buffer, which is a time buffer at the end of a project activity chain that feeds the critical chain, provides a sized buffer to protect the overall project from late completion of the feeding paths. This maximizes the advantage to the project, while ensuring project schedule protection.

2.4.3.2 Subordinate to The Constraint – Project Buffer (PB)

CCPM protects the overall project delivery time with a “project buffer” at the end of the critical chain. This exploits the statistical law of aggregation (previously mentioned in section 2.3.3) by protecting the project form common cause uncertainty of the individual activities in an activity path by using buffer at the end of the path. Buffer appears as activity in the project path, but has no work assigned to them (see Figure 2-7).

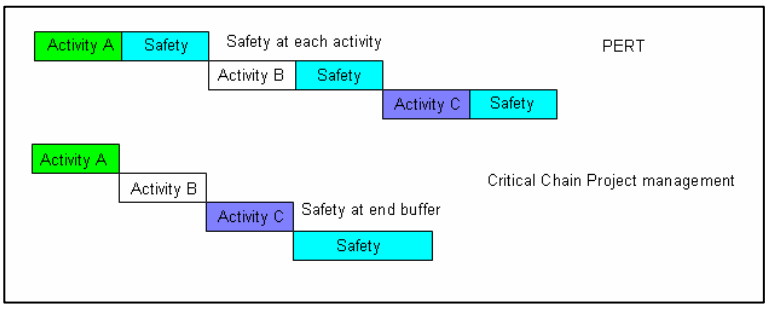

Figure 2-7 : Comparison between PERT and CCPM with regard to safety time.

2.4.3.3 Subordinate Merging Paths – Feeding Buffer (FB)

CCPM protects the critical chain from potential delays by subordinating critical chain feeding paths; placing an aggregated “Feeding Buffer” (FB) at the end of each path

that feeds the critical chain. This includes paths that merge with the critical chain at the end of the project. The Feeding Buffer provides a measurement and control mechanism to protect the critical chain. Figure 2-8 illustrates the location of Feeding Buffers.

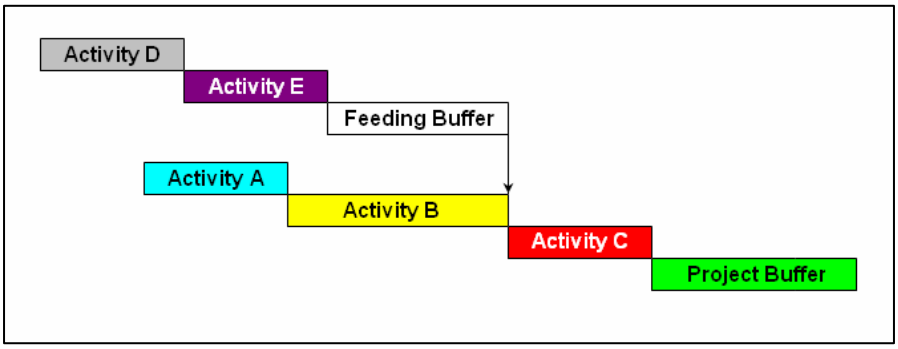

Figure 2-8: The location of Feeding Buffer

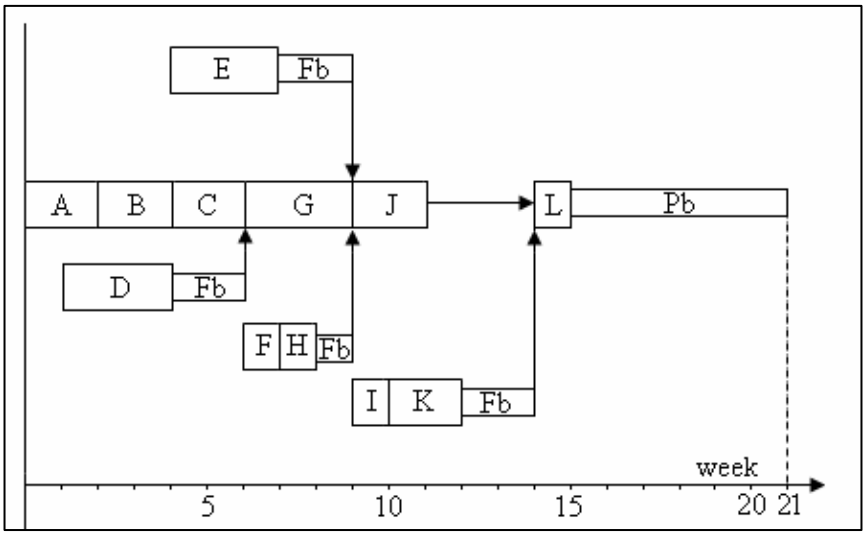

Figure 2-9: Critical Chain Representation of project network example in Figure 2-5.

Figure 2-9 illustrates Critical Chain representation of project network example in Figure 2-5, the Critical Chain consists of activity A, activity B, activity C, activity G, activity J and activity L, Feeding Buffer are added at the end of Non-Critical Chain. At the end of the Critical Chain we add a Project Buffer.

2.4.4 Roadrunner Mentality

CCPM plans only provide dated for the start of activity chains and the end of the project buffer. This enables the project team to focus on completing the project as soon as possible. For the rest of the project, the plan provides approximate start times and estimated activity duration. This helps eliminate date-driven behavior.

Critical Chain project managers do not criticize performers who overrun estimated activity durations, as long as the resource:

- Start the activity as soon as they had the input.
- Work 100 % on the activity(no multitasking)
- Pass on the activity output as soon as it is completed.

This is called “roadrunner mentality” activity performance; they expect 50% of the activities to overrun

2.4.5 Elevate Activity Performance by Eliminating Multitasking

CCPM seeks to eliminate this type of multitasking by eliciting 100% focus on the project activity at hand by all resources supporting the project.

2.4.6 Buffer management process

As activities are complete, project management ought to keep track of buffer consumption. Goldratt notes that “To keep track of buffer consumption is reckoned as measurement that will enable us to judge the impact of a local decision on the global goal” (Goldratt, 1997).

The improved measurement system for Critical Chain Project Management follows the practices established by Goldratt for production operations in his novel, *The goal: a process of ongoing improvement* (Goldratt, 2004). It uses buffers (time buffer) to measure activity chain performance. We set explicit action levels for decisions. The decision levels are in terms of the buffer size, measure in days:

- *Green (O.K.) zone* – within the first third of the buffer: No action.
- *Yellow watch-and-plan zone* – penetrate the middle third of buffer: Assess the problem and plan for action.
- *Red act zone* – penetrate the third third : Initiate action.

These measures apply to both the Project Buffer (PB) and Feeding Buffer (FB), Figure 2-10 and Figure 2-11 show an example of using the buffers (Leach, 1999; Trietsch, 2005).

Project teams monitor the Project Buffer and each Feeding Buffer at the appropriate time intervals for the project, usually weekly but at least monthly. For this tool to be fully useful, the buffer monitoring time must be at least as frequent as one third of the total buffer time. If the buffers are negative, namely latest activity on the chain is early relative to schedule date or less than one third of the total buffer late such as less than 10 days if the total buffer is 30 days, you need not take any action. If extended durations penetrate the buffer between 1/3 and 2/3, the project team should plan actions for that chain to speed up the current or future tasks and recover the buffer. If the activity performance penetrates the buffer by more than 2/3 of the buffer size, the project team should take the planned action.

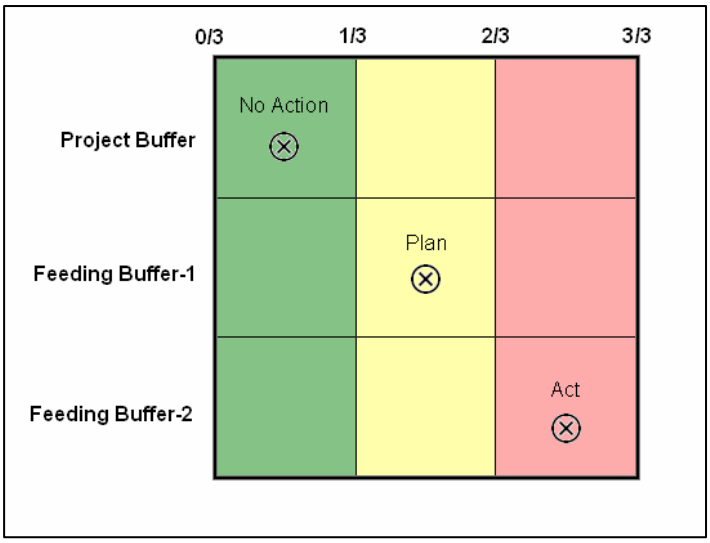

Figure 2-10: Buffer penetration provides the essential measurement for CCPM project control.

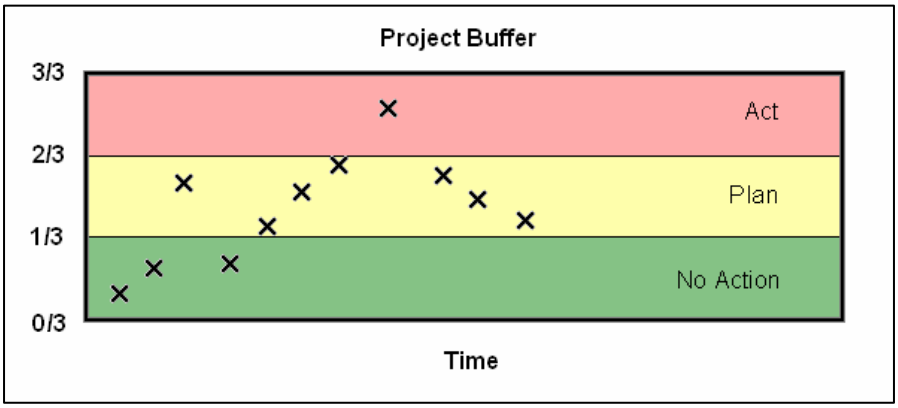

Figure 2-11: For long projects, it may prove useful to plot buffer penetration vs. time.

2.4.7 Project Buffer and Feeding Buffer sizing approaches

Because CCPM moves the contingency time which associated with the critical chain activities to the end of the critical chain in the form of a project buffer, and places feeding buffers whenever a non-critical chain activity joints to the critical chain so as to protect the critical chain from disruptions on the activities feeding it and to allow critical chain activities to start early in case things go well.

Nevertheless, the problem is what size project buffer and feeding buffers we ought to take in our project. In according to many research papers (Tukel, Rom, and Eksioglu, 2006; Herroelen and Leus, 2001; Leach, 2004; Newbold, 1998), based on a literature survey, there are two accepted buffer sizing approaches, namely, the cut and paste method (C&PM, also called 50% rule) and the root square error method (RSEM).

2.4.7.1 Cut and Paste Method (C&PM)

Because the activity's target durations are 50% confidence estimates, a project buffer of half the project duration is expected to provide sufficient protection. Also the size of a feeding buffer is usually set at half the duration of the longest non-critical chain path leading into it (Herroelen, 2001)

For example, consider feeding path in Figure 2-12(upper) with three activities. Each task's safe estimate is given as 20 days. If we cut the 50% safety from each task, then average estimate duration of each task would be 10 days. Summing up all 50% safety estimate is 30 days and half of it is 15 days. Therefore, C&PM suggests adding 15 days as feeding buffer in the end of feeding chain as illustrated in Figure 2-13(under)

The clear advantage of the C&PM procedure is its simplicity. However the procedure is linear procedure. The size of buffer that is calculated increases linearly with the length of chain with which it is associated. This might cause a two year project to have one year project buffer. Especially in a low risk project environment, this is an unnecessary protection. On the other hand, a feeding chain consisting of one task with 10 day duration will have a feeding buffer of 5 days which might not be sufficient at all. Moreover, Herroelen states that "using 50% rule for buffer sizing may lead to a serious over estimation of the required project buffer size, and consequently of the project duration" (Herroelen, 2001).

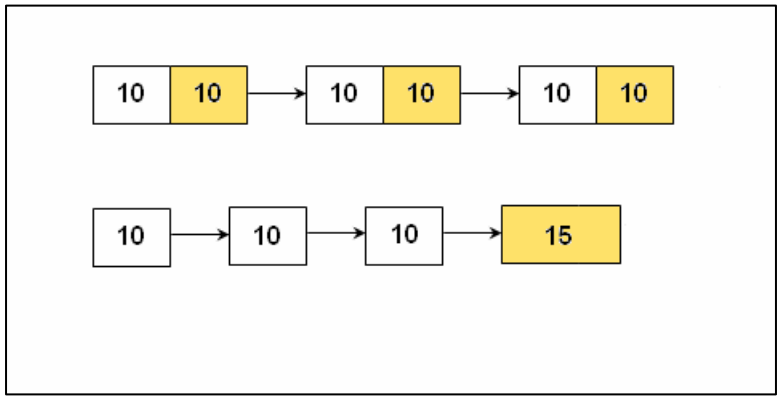

Figure 2-12: Example using Cut and Paste Method (C&PM)

Therefore, we should be careful using this method. In general, C&PM is not recommended to be used in project environments such as new product development (Tukel, Rom, and Eksioglu, 2006).

2.4.7.2 The root square error method (RSEM)

This method requires two estimates of task duration. The first estimate should be a safe estimate(S) which includes enough safety to protect against all likely sources of delays. The second estimate (A) should be one that includes no such protection, in other word, it is the average (50%) estimate of task (Tukel, Rom, and Eksioglu, 2006).

In addition, it should be assumed that: (Herroelen, 2001).

- All activities will be worked at a full level of effort.
- No interruption imposed by external factors.
- Project activities are independent.

Then the uncertainty for task duration is calculated as:

$$D_i = S_i - A_i .$$

Where D_i is uncertainty of task i, S_i is the safe estimate of task i and A_i is the average (50%) estimate of task i.

Newbold suggests that the standard deviation in the task duration is $\frac{S_i - A_i}{2}$ (Newbold, 1998). The buffer size is the two standard deviations, and then the standard deviation of the feeding chain is:

$$\sigma = \sqrt{\left(\frac{S_1 - A_1}{2}\right)^2 + \left(\frac{S_2 - A_2}{2}\right)^2 + \ldots + \left(\frac{S_n - A_n}{2}\right)^2} .$$

Where n is the number of activities in feeding chain. The assumption here is that the task completion times are independent. The buffer size is then two standard deviations:

$$2\sigma = 2 \times \sqrt{\left(\frac{S_1 - A_1}{2}\right)^2 + \left(\frac{S_2 - A_2}{2}\right)^2 + \ldots + \left(\frac{S_n - A_n}{2}\right)^2} .$$

$$= \sqrt{(S_1 - A_1)^2 + (S_2 - A_2)^2 + \ldots + (S_n - A_n)^2} .$$

It has been suggested that in cases where there are fewer than four tasks in the feeding chain, the feeding buffer should be at least equal to the longest activity in that chain (Leach,2005).

Note that in cases where the feeding chain consists of one task, say i, then the buffer size is $S_i - A_i$

Figure 2-13, using the same example as for the C&PM case, we find that the buffer size is now 17.3205.

Note that for each task i, S_i = 20 days, A_i = 10 days, and $2\sigma = \sqrt{300} \approx 17.3205$

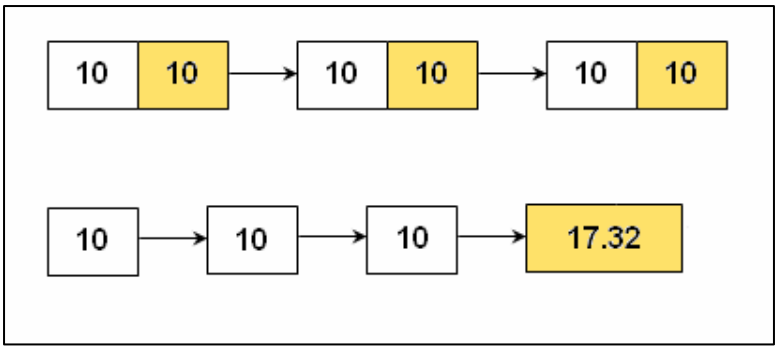

Figure 2-13: Example using the root square error method (RSEM)

The primary advantage of this method is that it allows using known task variation. The primary disadvantage is that it may lead to undersized buffers for long chains. There are two primary reasons for this. The first is underestimation of the actual task variation. The second is that this method assumes that all project variation is stochastic, and reality demonstrates that some is not. Some project variation is a result of bias: things that can make projects take longer, but not make them shorter. The path merge bias is one obvious bias in this category. The statistical basis of this method does not work for variation that includes bias.

2.4.7.3 Buffer sizing in single project case

In single project environment, we had adjusted percent of C&PM to be 40%, 30%, 20%, and 10%, in order to examine whether the due date corresponding with higher percent of C&PM is greater than due date corresponding with lower percent of C&PM. We found that when we cut and paste lower percent from each activity, it did not mean that the due date of project would be lower than cut and paste higher percent from each activity, because there are resource constraint and precedence constraint in each project.

For support aforesaid statement, we will use an example to illustrate a comparison among C&PM 50%, 40%, 30%, 20%, 10% and RSEM under single project environment.

According to the project shown in figure 2-5, CCPM disallows individual reserve, because Goldratt, E. M., (1997) accounted for such doing that in realistic project, project manager always confronts with two specific behaviors that increase the lead time of projects. They are student syndrome, the natural tendency of many people to wait until a due date is near before applying full energy to complete an activity (Leach, L.P., 2002), and Parkinson's Law, meaning that humans tend not to finish their tasks ahead of time even though they have the chance to do so (Lechler, T.G., Ronen,B., and Stohr, E. ,2005).

In the next step, the safety time that is eliminated from the critical chain activity duration by selecting aggressive duration estimates is shifted to the end of the critical chain in form of a project buffer. The excluded contingency duration by using C&PM and RSEM was shown in Table 2- 4.

In order to identify critical chain, because we know that < A-B-C-G-J-L > is critical path, the critical chain is set the same as the critical path. For example, we use the default procedure is used to 50% buffer sizing rule (C&PM), namely, to use a Project buffer of half the critical chain and use a Feeding buffer of half the duration of the non-critical chain path leading into critical chain activity. we generate three feeding buffers: a two-period feeding buffer to protect critical chain activity G form variation in activity D, a two-period feeding buffer to protect critical chain activity J form variation in activity E, a one-period feeding buffer to protect critical chain activity J form variation in path <F-H> and a two-period feeding buffer to protect critical chain activity L form variation in path <I-K>. For project buffer and feeding buffers which used C&PM 40%, 30%, 20%, 10% and RSEM are shown in table 2-5.

For C&PM 50%, in order to protect the over all schedule form the variation, a six-period project buffer is inserted that lets the project due date equal to 21.

Table 2-4: Excluded contingency duration by using C&PM and RSEM

Activity	Excluded contingency duration by using C&PM & RSEM (weeks)					
	C&PM50%	C&PM40%	C&PM30%	C&PM20%	C&PM10%	RSEM
A	2	2	2	2	3	2
B	2	2	3	3	4	2
C	2	2	2	2	3	2
D	3	4	4	5	5	3
E	3	4	4	5	5	3
F	1	1	1	2	2	1
G	3	3	4	4	5	3
H	1	1	1	2	2	1
I	1	1	1	2	2	1
J	2	2	3	3	4	2
K	2	2	2	2	3	2
L	1	1	1	2	2	1

Table 2-5: project buffer and feeding buffers which used C&PM 50%, 40%, 30%, 20%, 10% and RSEM

Buffer sizing method	Project buffer (weeks)	Feeding Buffers (weeks)			
		D=>G	E=>J	<F-H>=>J	<I-K>=>L
C&PM 50%	6	2	2	1	2
C&PM 40%	5	2	2	1	1
C&PM 30%	5	1	1	1	1
C&PM 20%	3	1	1	<F-H-I-K>=>J ; Fb = 2 *	
C&PM 10%	2	1	1	1	1
RSEM	5	3	3	<F-H-I-K>=>J ; Fb = 3 *	
* Because G and I do not occur resource conflict					

Table 2-6: comparison among project due date corresponding with C&PM 50%, 40%, 30%, 20%, 10% and RSEM

Project due date corresponding to each buffer sizing method (weeks)					
RSEM	**C&PM10%**	**C&PM20%**	**C&PM30%**	**C&PM40%**	**C&PM50%**
17	25	19	21	19	21

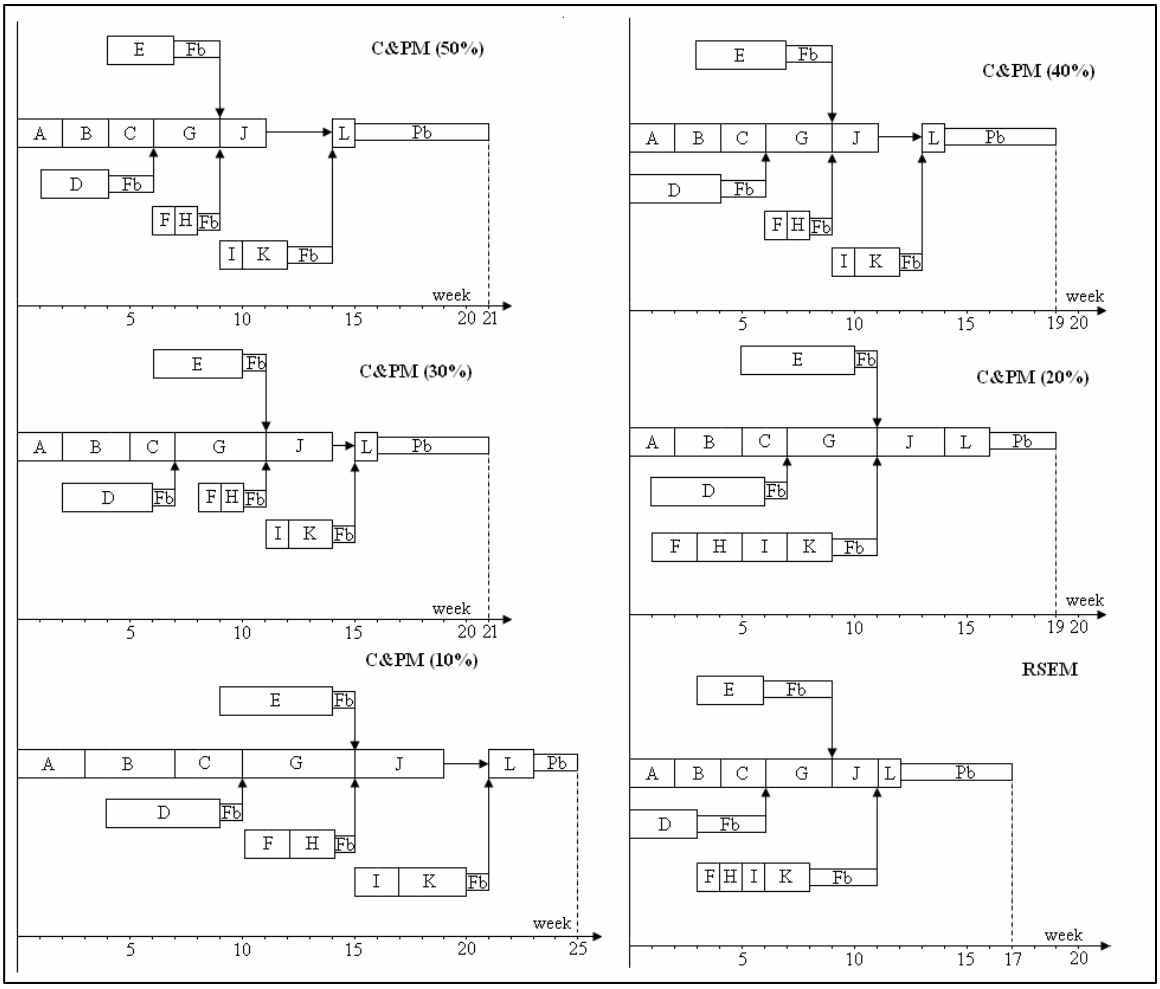

Figure 2-15: Gantt chart showed a comparison among project due date corresponding with C&PM 50%, 40%, 30%, 20%, 10% and RSEM

According to Table 2-6, we can conclude that for this case, the comparison among project duration corresponding with C&PM 50% , C&PM 40%, C&PM 30%, C&PM 20%, C&PM 10% and RSEM could be shown as below:

C&PM 10% > C&PM 30%=C&PM 50% > C&PM 20%=C&PM 40% > RSEM

Therefore, we can proof that when we cut and paste lower percent from each activity, it did not mean that the due date of project would be lower than cut and paste higher percent from each activity; moreover, in this case RSEM is the most effective buffer sizing.

2.5 Critical Chain Project Management: Multi Project Case

2.5.1 The impact of multitasking

Remarkable as it may seem, traditional project management regularly fails to include explicit account of resource conflicts in its schedules. It is extremely common, for example, to find workers implicitly assigned to tasks on two, three or more projects simultaneously. Let us consider the implications of this.

Figure 2-16 shows schedules for two high-priority projects, which have been constructed on the implicit assumption that there are no resource restrictions, and hence no resource conflicts. Assuming each task takes 15 days to perform both projects will complete in around 45 days.

Now, in this organization there are three distinct resources, which are specialists in tasks A, B and C respectively. However, since it is not possible to literally work on two things simultaneously, the resources have to switch from one project to another – to multitask. The outcome is shown in Figure 2-17.

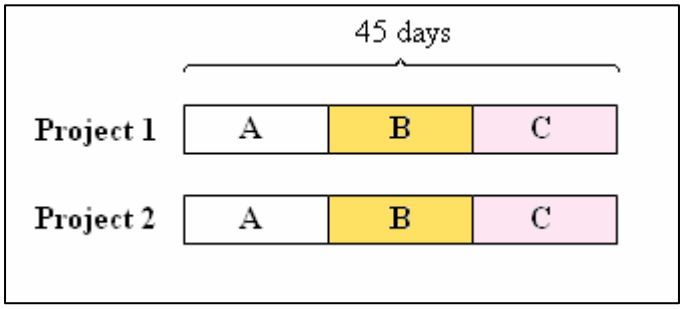

Figure 2-16: Two project schedules, no resource conflicts

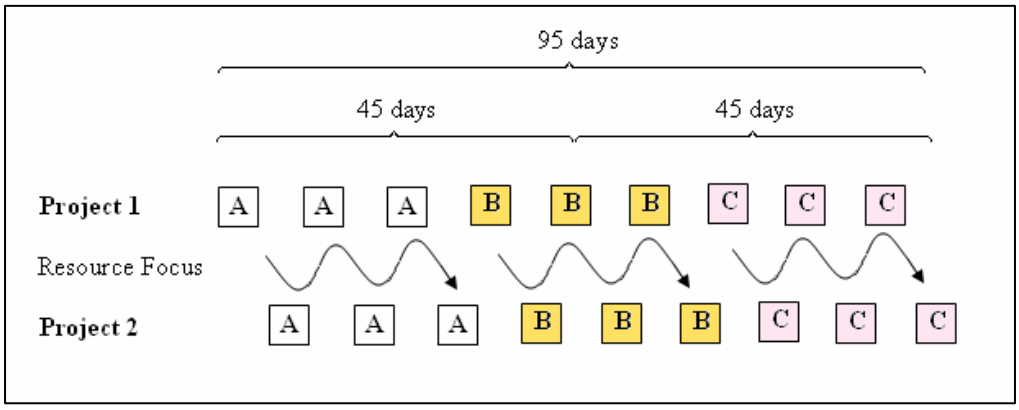

Figure 2-17: Two project schedules, limited resource, multi-tasking

Firstly, notice how the duration of each project has grown significantly. Project 1 now takes 90 days and Project 2 slightly longer. Secondly, it would probably be correct

to assume that the picture is actually overoptimistic. Real people do not operate like computers. When a person switches from one task to another he requires a degree of ramp-up time. This can be anything from a few minutes to several hours depending on the nature of the task and the personality.

The typical reaction from project managers when a project appears to be taking twice as long as originally envisaged is to cry I need more resources Under these circumstances project workers, too, find more substance with which to justify their pessimistic task duration estimates

In an alternative arrangement, shown in Figure 2-18, we have staggered the starts of the projects until resources are available to work on them. Consequently, Project 1 is delivered in 45 days and Project 2 in 60 days. Without the addition of any new resources, both projects are delivered significantly earlier than in the previous setup.

Yet this is not the only advantage: Firstly, resources are fully concentrated on the tasks they must perform. No ramp-up penalty is incurred because of context switching. Quality, too, will be positively affected by this. Secondly, from a financial perspective, work-in-process is significantly reduced – less capital per project is tied up in the system. Return on Investment is up. Thirdly, again highly significant in fast-moving environments like software development, requirements have less time to change, which means less work will have to be redone. Fourthly, the principal cause of student syndrome has been quashed because a worker, once assigned to a task, views this task as the number one priority. And finally, project workers who are not required to multitask sense (correctly) that they actually have a realistic chance of delivering a result according to the original estimate. And given that context switching is no longer necessary means, two major sources of stress have also been dealt with. The result is a happier, healthier staff.

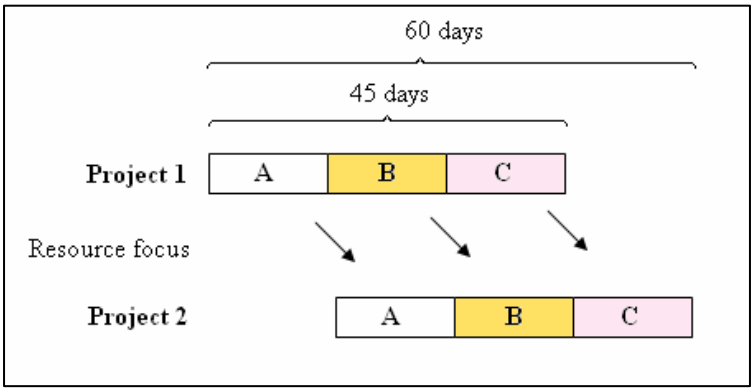

Figure 2-18: Two project schedules, limited resource, no multi-tasking

All this adds up to an almost outrageous increase in productivity, which is achieved simply by scheduling tasks for when resources are available. Ironically, executives generally perceive multitasking as something positive, because it appears that

resources are being fully utilized. They are following the gross misnomer that maximizing utilization is a good thing.

2.5.2 Resource leveling

We have now seen that a description of task dependencies alone does not constitute a plan. A schedule based on critical chain can only be finalized once resources have been assigned to tasks. And only once this happens can we assign task start dates. Figure 2-19 presents excerpts from two projects prior to resource assignment.

Clearly there is a resource conflict. Chris will be unable to work on tasks 1 and 4 simultaneously. He could, of course, multitask, but we have seen where this leads us. So let us adjust the schedule to take this into account.

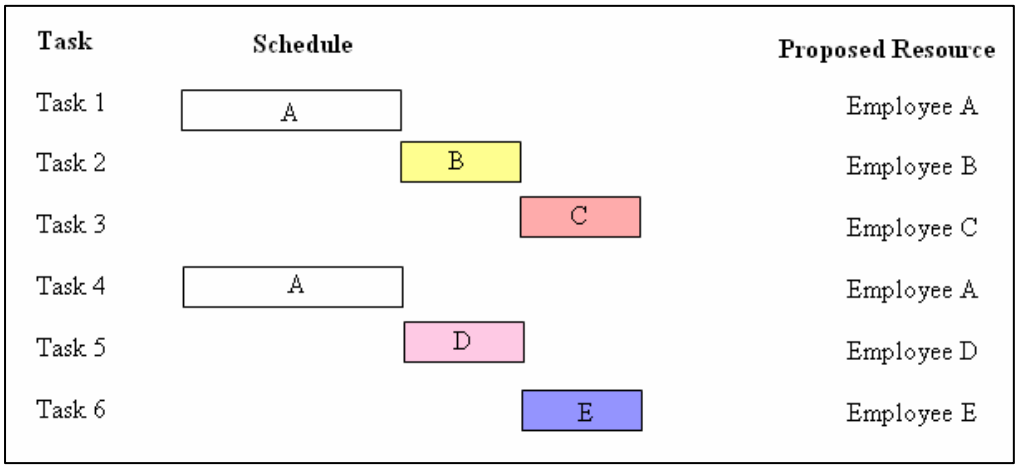

Figure 2-19: A single-project schedule with a proposal for resource assignments

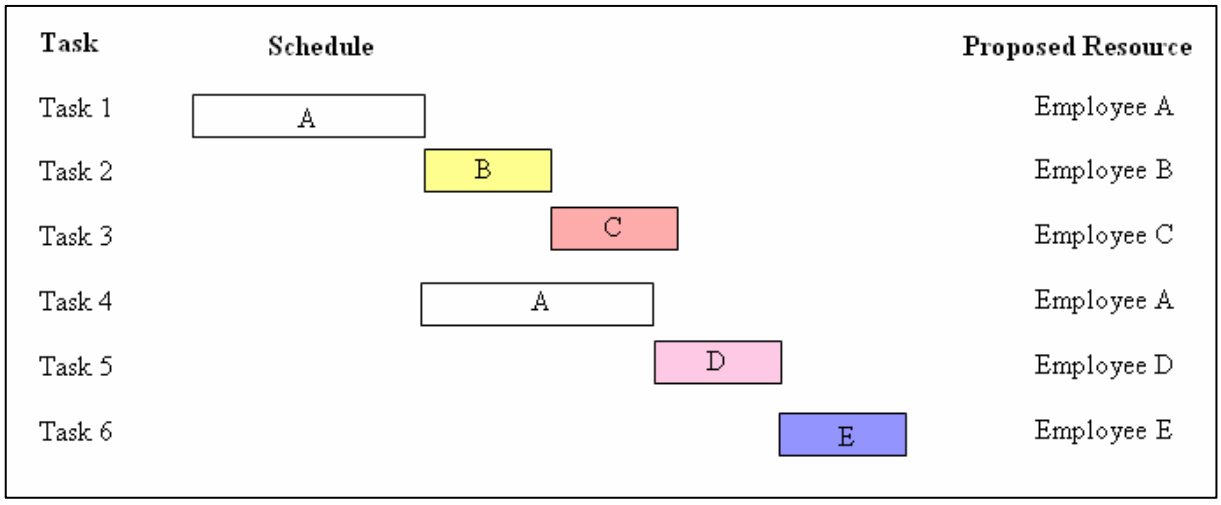

Figure 2-20: A single-project schedule after resource-leveling

Figure 2-20 demonstrates the result of this process, which is called resource-leveling. Although there is no algorithm for determining an "optimal" resource-leveled schedule, project management software can assist in this process.

2.5.3 Resource buffering

In critical chain project management, task start dates are highly flexible in nature. Flexible start dates are necessary because of variation: some tasks will be delivered earlier, some later than originally estimated.

So although resource-leveling removes conflicts from the plan, conflicts can still occur because of variation. There is, of course, no way to predict in advance when this will occur. But what we can do, at least, is build in a warning for the PM that a potential resource conflict exists – one which will cause delay if it occurs.

Just such a warning is shown in Figure 2-21. A resource buffer entitled "RB: Emp A", which is specific to the Employee A, indicates that should a delay occur in task 1, he will not be able to begin task 4. In this way the PM knows precisely when she should inquire about the status of task 1. If task 1 looks like it is going to exceed the original estimate, the appropriate action can be taken.

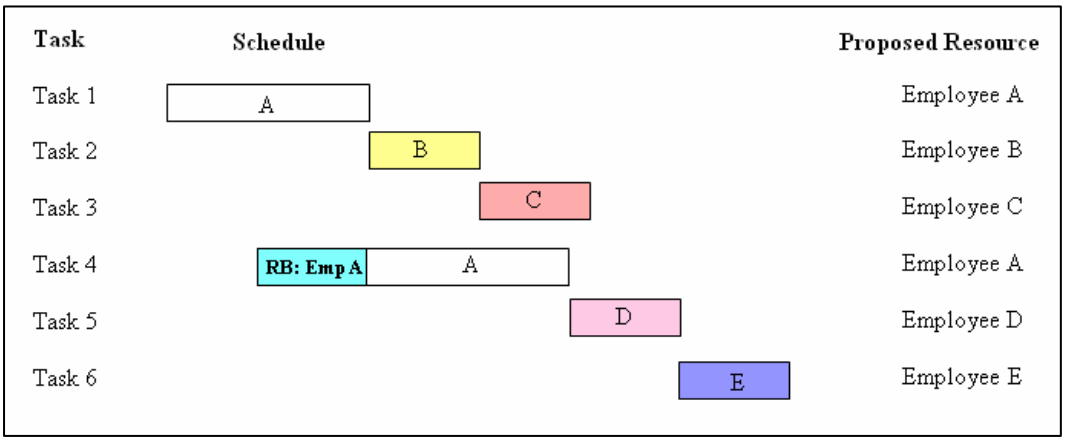

Figure 2-21: A resource buffer warns the PM of a potential resource conflict

2.5.4 Multi-project critical chain methodology

What we have learned so far greatly increases our chances of delivering individual projects in a competitive and timely manner. Thousands of projects around the world already bear testament to this. However, for the technique to work in a multi-project environment, the program manager must take resources into consideration, which are shared across projects.

Fortunately, it is not necessary to consider each and every possible resource conflict, which in any case would be an impractical undertaking in most organizations.

Instead it is the job of the program manager (or project management office) to identify the one shared resource, which has the smallest capacity compared to other shared resources. This resource is termed the *capacity-constraint resource (CCR)*, as it determines the number of projects which can be executed by the organization in a given timeframe.

Once the CCR has been identified a centralized schedule for its usage is drawn up based on project prioritization. Typically the highest priority project gets to use it first, the second highest second etc. Individual project plans are developed around to the availability of the CCR. From here on the capacity constraint is termed drum resource because it determines the rhythm – the beat – with which the organization executes projects. A simple example of this is presented in Figure 2-22. In this case the CCR can serve only one project at a time and projects 1, 2 and 3 and staggered in respect of this.

Since the drum resource is – by definition – a constraint, to obtain the highest possible throughput of projects in an organization over time, the drum resource must be kept constantly busy. And whilst critical chains are protected from common cause variation using project, feeding and resource buffers, the drum is protected from common-cause variation by means of a *capacity-constraint buffer* (also shown in Figure 2-22). In addition, risk management must ensure that the drum is not subject to outages because of special causes. To increase the number of projects which can be performed by the organization over time, we turn to the five focusing steps. If we succeed in elevating the drum resource so it is no longer a capacity constraint, the next capacity constraint acquires the role of drum, and individual project plans are rescheduled according to this resource's availability.

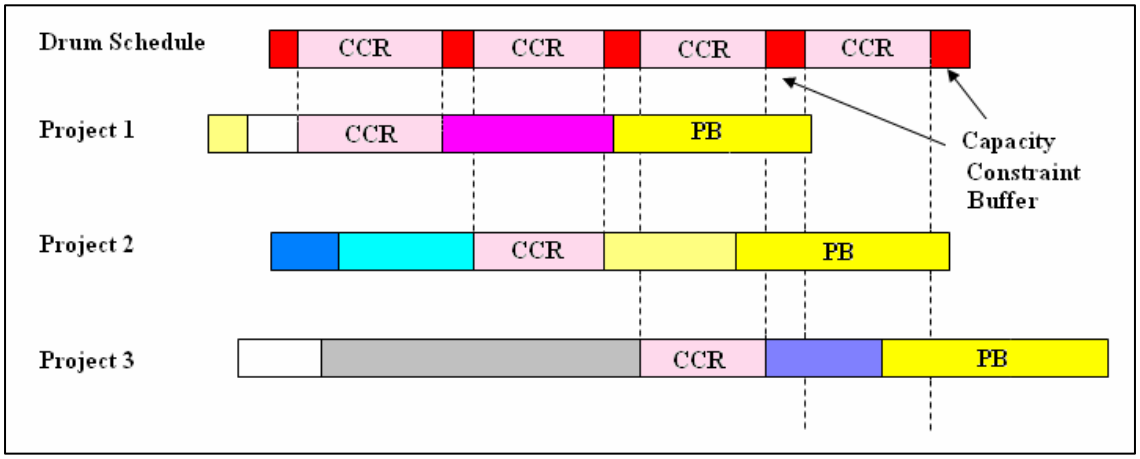

Figure 2-22: A Multi-project critical chain scenario

2.5.5 Exploit Multi Project Resource Allocation

Resource managers prioritize resource allocation across project according to the importance of activities to the projects. They give priority to (in order):

- Critical chain activities over non-critical chain activities.
- Activities of projects with the highest level of Project Buffer utilization, namely, the least slack time.
- Activities in project chains with the highest Feeding Buffer consumption.

Penetration of Feeding Buffer and Project Buffer resolves remaining resource contentions (Leach, 1999).

2.5.6 Buffer sizing in multi project case

Supposing that the example of master project has three subprojects and the precedence was determined as figure 2-23.

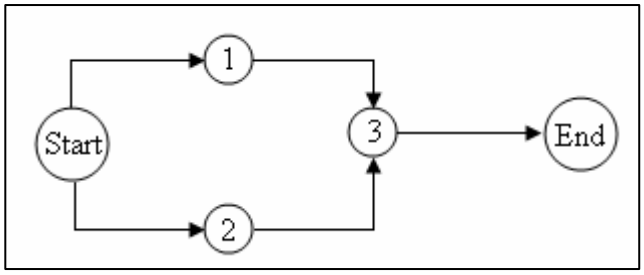

Figure 2-23: Network for the example of master project

Subproject 1, 2 and 3 are identical. Moreover, the precedence of subproject is represented as figure 2-24. The estimated duration of activity is shown above the corresponding node.

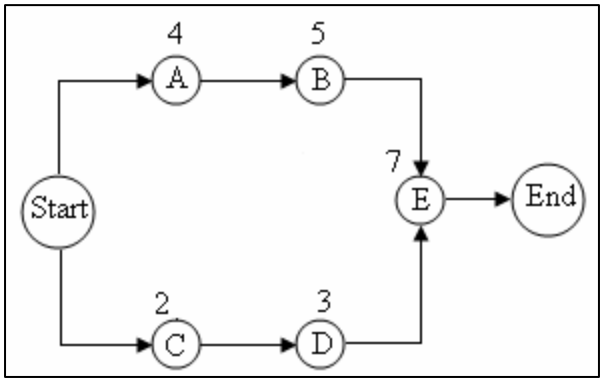

Figure 2-24: Net Network for each subproject

Supposing Activity B and activity D use the same resource and such resource has only one unit. Furthermore, the resource supplying activity B and activity D is the “capacity constraint resource”.

Considering each subproject: According to figure 2-24, we know that <A-B-E> is critical path; the critical chain is set the same as the critical path. In the next step,

eliminate the safety time from each activity. The excluded contingency duration by C&PM and RSEM were shown in table 2-8.

Table 2-7: Estimated duration of each activity in subproject

Activity	Estimated duration
A	4
B	5
C	2
D	3
E	7

Table 2-8: The excluded contingency duration by C&PM and RSEM

Activity	Excluded contingency duration by using C&PM & RSEM					
	C&PM50%	C&PM40%	C&PM30%	C&PM20%	C&PM10%	RSEM
A	2	2	3	3	4	2
B	3	3	4	4	5	3
C	1	1	1	2	2	1
D	2	2	2	2	3	2
E	4	4	5	6	7	4

For example, we demonstrate buffer sizing calculation by C&PM 50%, namely, to use a project buffer of half the critical chain, the feeding buffer of half the duration of the non-critical chain path leading into critical chain activity, and capacity constraint buffer of half the summation of duration activity B and D (because the resource supplying activity B and activity D is the capacity constraint resource).

In each subproject, we generate two feeding buffers: a one-period feeding buffer to protect critical chain activity B form variation in activity C and a one-period feeding buffer to protect critical chain activity E form variation in activity D. In order to protect each subproject from the variation, a five-period project buffer is inserted. For project buffer and feeding buffers calculated by C&PM 40%, 30%, 20%, 10% and RSEM are shown in table 2-9

Table 2-9: The project buffer and feeding buffers calculated by C&PM and RSEM

Buffer sizing method	Project buffer	Feeding Buffers	
		C=>B	D=>E
C&PM 50%	5	1	1
C&PM 40%	4	1	1
C&PM 30%	4	1	1
C&PM 20%	3	1	1
C&PM 10%	2	1	1
RSEM	5	1	1

Because the capacity constraint resource is supplying activity B and activity D in each subproject, therefore, a 3-period capacity constraint buffer is inserted between subproject 1 and subproject 2.

According master project precedence shown in figure 2-23, we can calculate master project duration corresponding to C&PM 50% buffer sizing method (Gantt chart is shown in Figure 2-25). The master project duration corresponding to C&PM 50% buffer sizing method is 43. For the master project duration calculated by other buffer sizing method is shown in table 2-10.

Table 2-10: Master project due date corresponding to each buffer sizing method

Master project due date corresponding to each buffer sizing method					
RSEM	**C&PM10%**	**C&PM20%**	**C&PM30%**	**C&PM40%**	**C&PM50%**
44	54	46	47	40	43

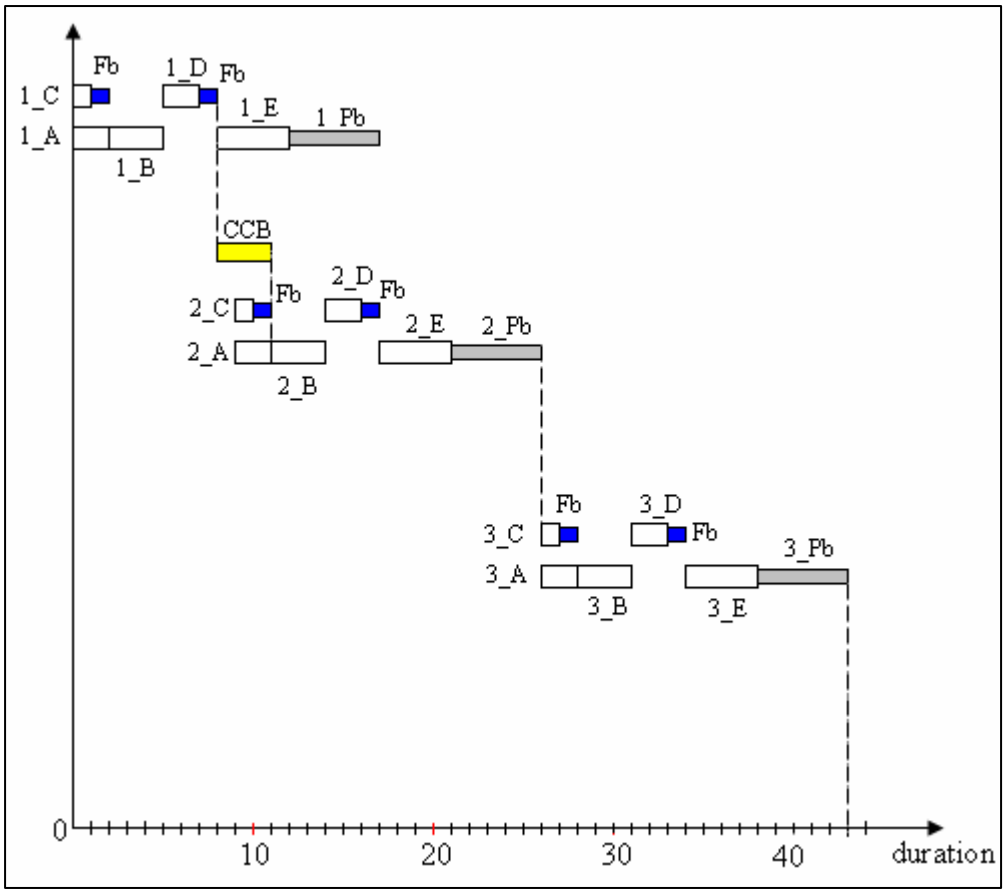

Figure 2-25: Gantt chart showing master project duration corresponding to C&PM 50%