Article category: Time

Critical Chain
A detailed example

Many projects from a wide cross-section of project environments suffer from poor results. Many solutions have been proposed and implemented. But since poor results persist, these solutions must be inadequate. It is easy – too easy – to believe that there can be no general answer, not only because the problems have been around for so long, but because of the wide range of settings in which they occur. A basic principle of the Theory of Constraints is that unsuccessful projects are usually the result of a relatively few core problems. "Relatively few" means a manageable number. By solving the core problems, rather than merely attacking their symptoms, we will eliminate many of the poor results from projects. Robert Newbold; Project management in the fast lane – Applying the theory of constraints, Creative Technology Laboratories, 1996.

A critical chain example

To further explore the critical chain concepts discussed in previous TOOLS AND TECHNIQUES articles, we will explore a detailed example of project planning using critical chain techniques. The example is similar to one from [1].

The project

The project brief is to develop and test a prototype receiver. The receiver will consist of a receiver module, a power supply module and a self-test module.

The work breakdown structure and related details of the project are contained in exhibit 1. Following common practice, the term "resource" is used for that individual who is responsible for performing each task. Although resources could also include equipment, raw materials and buildings, these turn out to be irrelevant to this project.

Step 1 – Develop an initial plan

Remove all safety from each task by halving the task duration. That is a good estimate of the average duration, as opposed to the worst case duration [2]. In step 4, the duration thus removed will be placed in a project buffer.

Then create an activity on the node project network diagram. Schedule each task as late as possible, whilst respecting precedence dependencies. Each task is scheduled as late as possible to minimize the project's work-in-process. A project's work-in-process consists of all project tasks started but not yet completed. Work-in-process should always be minimized:

- The more work-in-process, the longer it will take for new work to

Ad Sparrius
Ad Sparrius graduated in electronic engineering from Stellenbosch and Berkeley, California, and was awarded an MBL by UNISA. He has extensive project management experience in the high-technology industry. He consults in engineering management to a broad cross-section of South African industry. Many thousands of people have attended one or more of his courses or guest lectures.
Tel (012) 47-4318
Cell 082 574-0266
Fax (012) 47-3828

	Task	Estimated duration (weeks)	Preceded by	Resource
A	Design receiver module	4	—	John
B	Program receiver module	8	A	Thabo
C	Build receiver module	4	B	Ann
D	Design power supply module	8	—	John
E	Construct power supply module	6	D	Ann
F	Integrate receiver and power supply modules	4	C and E	John, Thabo
G	Program self-test module	4	—	Thabo
H	Construct self-test module	2	F and G	Ann
I	Integrate all modules to construct receiver	4	H	John, Thabo
J	Program acceptance test	6	—	Thabo
K	Perform acceptance test	4	I and J	John, Thabo
	Exhibit 1: Project details			

be started and eventually completed. Conversely, the less work-in-process, the faster we can react to new customer needs.

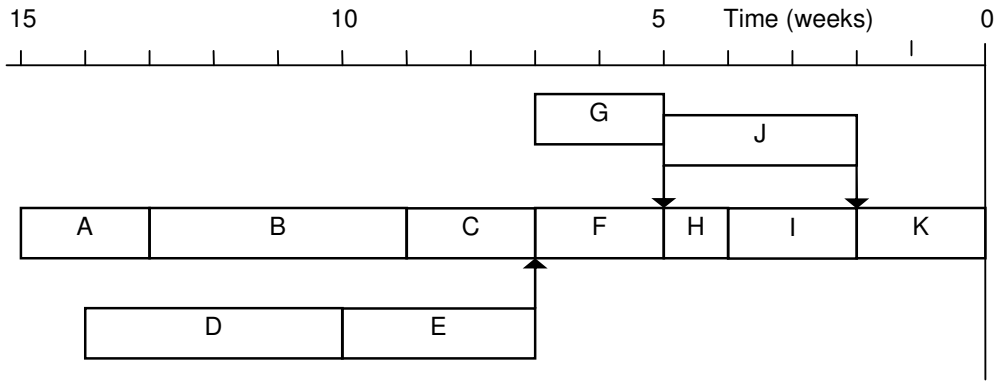
- The less work-in-process, the less time the customer has to change his mind about something.
- The initial plan is shown in exhibit 2. Note that task durations have been halved and that all tasks have been scheduled as late as possible, subject to precedence dependencies.

Step 2 – Reschedule tasks to remove all resource contention

The main problem with the initial plan is the neglect of resource dependencies. A resource dependence exists when various tasks compete for the same resource and that resource has insufficient capacity to perform all those tasks simultaneously.

Whenever resource contention occurs, some tasks will have to be scheduled earlier to remove that contention.

The technique used to eliminate resource contention is known as resource levelling or as finite capacity scheduling, and ends up with a resource constrained



Total duration = 15 weeks.
Critical path: A B C F H I K

Exhibit 2: Initial plan

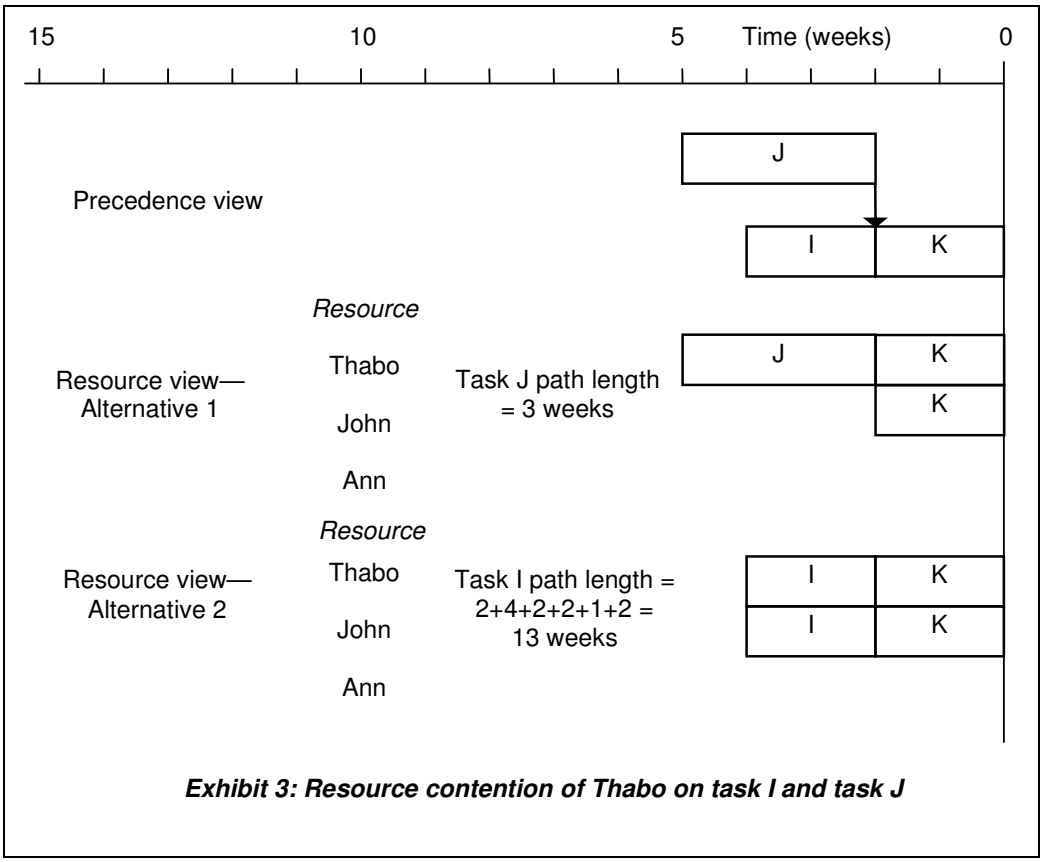


Exhibit 3: Resource contention of Thabo on task I and task J

schedule. In a resource-constrained schedule the start and end dates of each task reflect the availability of each resource.

Start at the completion date, and systematically eliminate all resource contentions by moving contending tasks earlier. Work-in-process remains minimized since tasks are still scheduled as late as possible.

The resource contention problem is illustrated in exhibit 3. Tasks I and J both compete for Thabo, but only one can be handled at a time. Which task should Thabo work on just before task K – task I or task J? Exhibit 3 illustrates a traditional precedence view as well as a resource view which shows the loading on Thabo, John and Ann. For instance, task K is performed by both Thabo and John, but for task J only Thabo is needed.

The two finite capacity alternatives are clearly shown in exhibit 3. Which alternative is best, in other words, results in least project duration?

There is no simple method to determine the best solution. A simple algorithm is the following: Calculate the length of the critical path leading up to and including each of the competing tasks. Schedule the task with the longest critical path last since it has the greater opportunity to delay the project.

In this case, there are no precedence dependencies leading up to task J hence its path length is only 3 weeks, the duration of task J. The critical path leading up to task I consists of tasks A B C F and H with total path length of 2+4+2+2+1+2 or 13 weeks. Task I has the longer path length. Schedule task I later and reschedule task J by moving it earlier.

There is no resource contention on task H. Tasks F and G compete for Thabo. The path length for task G is two weeks and for task F is 2+4+2+2 or 10 weeks. Schedule task F last and reschedule task G earlier.

Incidentally, it is pointless to argue about which finite capacity alternative is best if the difference between them is much less than the size of the project buffer to be determined in step

4. Resource levelling does not need to be optimal or best – just good enough.

The complete finite capacity schedule is shown in exhibit 4. The process of eliminating resource contention continues until all tasks are placed somewhere on the resource view; see exhibit 4. It is quite possible that some tasks are rescheduled earlier than time now, in other words finite capacity scheduling pushes some tasks too early. Since that is nonsensical, it means the end date must shift so that the earliest task starts at time now. In exhibit 4 the end date becomes 18 weeks.

An alternative to eliminating resource contention by means of resource levelling is to use additional resources. This is of course not always possible.

Step 3 – Determine the critical chain

The critical chain is that set of tasks which determines the overall duration of the project considering both precedence dependencies and resource dependencies [3]. In step 2 all tasks were scheduled as late as possible. No tasks can be postponed without delaying the completion date. The critical chain tasks are those which also cannot be started earlier, in other words they cannot be delayed nor advanced.

A glance at exhibit 4 shows that only tasks G, B and J, all performed by Thabo, may be started two weeks earlier. All other tasks are on the critical chain. In other words, the critical chain consists of tasks A D E C F H I and K; see exhibit 4.

One way of determining the critical chain is as follows: exhibit 4 shows the late start for all tasks. A similar exhibit can be constructed showing the early start for all tasks. Then calculate the difference between the early start and the late start for each task. Those tasks whose difference is zero are the critical chain.

The critical chain identified by the methods outlined above may need to be adapted. Remember, most emphasis will fall on the critical chain. Is this where we want the management focus to be?

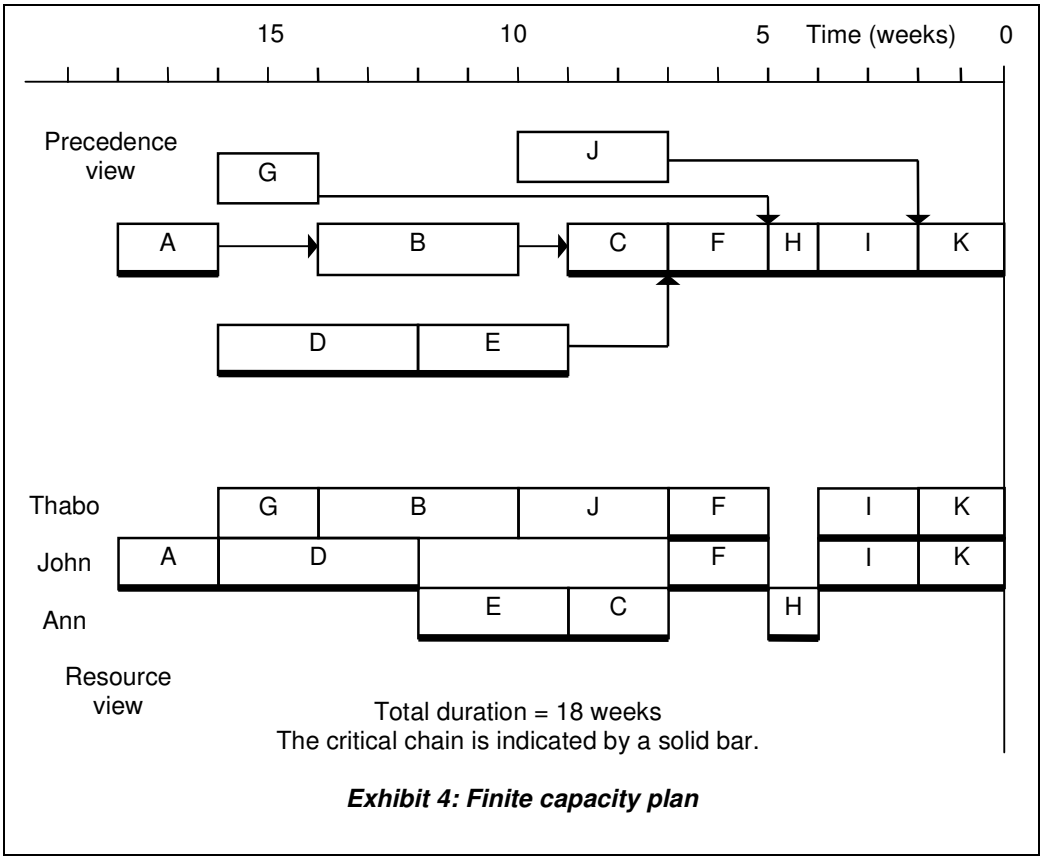


Exhibit 4: Finite capacity plan

Reasons for changing the critical chain include:

- The duration of some critical chain tasks may be shortened by allocating more resources. Shorten those tasks and return to step 2.
- The final schedule may place some high-risk tasks at the end of the schedule. Move them earlier and go back to step 2.
- Some tasks on which you believe the project's completion will depend have not been identified as part of the critical chain. Figure out why, and adjust the schedule according to what you believe is right.

In all cases, change the critical chain and return to step 2.

We now have a realistic finite capacity schedule which minimizes work-in-process. The remaining problem is that we have ignored uncertainty.

Step 4 – Identify buffer points and determine buffer sizes

The three types of buffer and their positions are shown in exhibit 5. The *project buffer* protects the completion date against duration variations of critical chain tasks [3] [4]. It is placed after the last task of the project, in other words behind task K. What should the size of the project buffer be? As a simple rule of thumb, let the project buffer size equal one-half of the duration removed from the critical chain tasks in step 1. From exhibit 1, project buffer = ½ (2+4 +3+2+2+1+2+2) = 9 weeks. The project will thus end 18 + 9 = 27 weeks after the start. The project buffer is not slack to be removed if the customer wants the project to end earlier—it is an essential part of the schedule.

A *feeding buffer* is placed wherever a non-critical chain task joins the critical chain [3] [4]. A feeding buffer protects the critical chain from disruptions on tasks feeding it, and allows critical chain tasks to start earlier if the opportunity arises. There will thus be three feeding buffers. The feeding buffer size

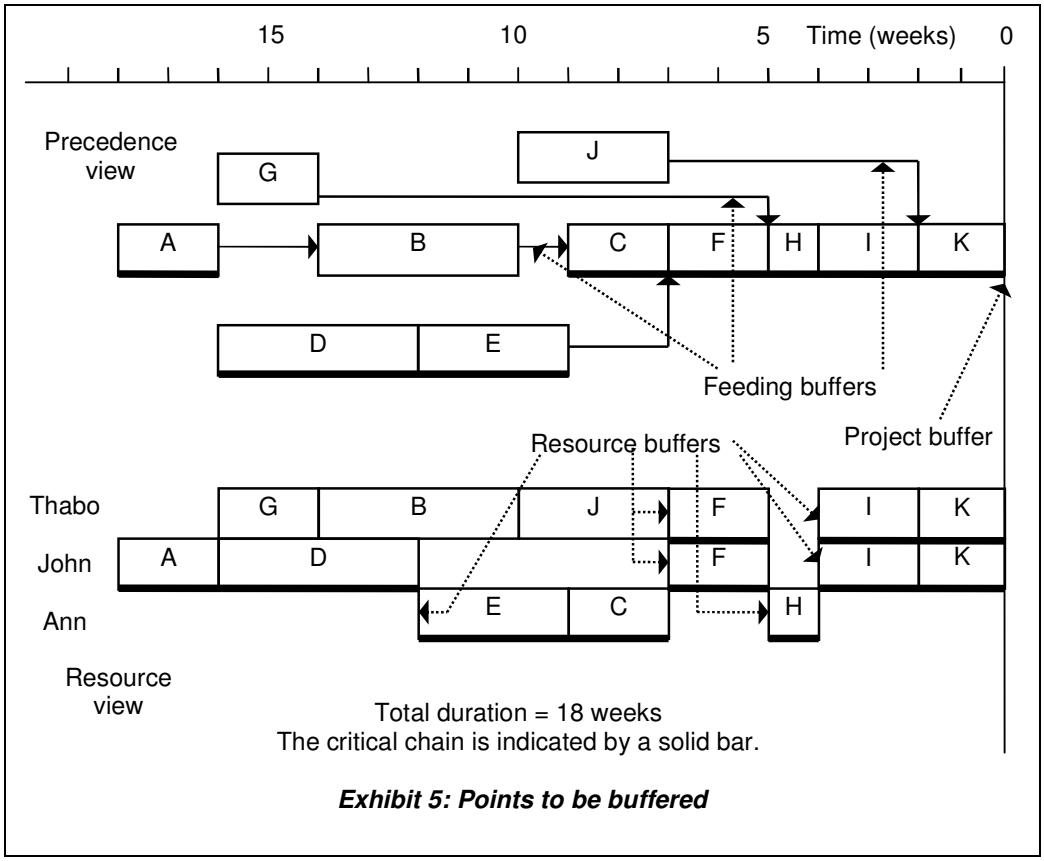


Exhibit 5: Points to be buffered

equals one-half of the savings in the path leading to the feeding buffer. From exhibits 1 and 4:

- Between tasks B and C; size = ½ (4+2) = 3 weeks.
- Between tasks G and H; size = ½ (2) = 1 week.
- Between tasks J and K; size = ½ (3) = 1½ weeks.

A *resource buffer* is a wake-up call to ensure that each resource is ready to start its critical chain task without delay [3] [4]. The resource view of exhibit 4 shows there are six resource buffers:

- Warning Thabo of tasks F and I.
- Warning John of tasks F and I.
- Warning Ann of tasks E and H.

Each resource buffer is sized at two weeks. A resource buffer does not lengthen the project duration – it is merely a countdown warning.

Sometimes there is both a feeding buffer and a resource buffer at the same place. For instance, at task H there is a feeding buffer of one week and a resource buffer of two weeks. These buffers are part of different views and have different meanings. The one week feeding buffer ensures that Ann doesn't have to wait for task G to complete before she can start task H. Also, if task F turns out to be early, task G will already be completed so Ann can immediately continue with task H. The two-week resource buffer warns Ann that task H is coming, and that whenever it arrives, she should drop everything and start with task H.

Incidentally, if a scheduling decision needs to be made and it turns out to be insignificant relative to the size of the project buffer, then it doesn't matter which decision is made. Safety is neither eliminated nor spread out throughout the project—but it is applied at key points.

Step 5 – Insert the buffers

Exhibit 6 shows the final buffered schedule. The project's completion date is the end of the project buffer. Resource buffers are shown in the resource view because they protect the critical chain from resource dependencies. Feeding buffers are shown in the precedence view since they protect the critical chain against precedence dependencies. Both types of buffers are placed flush against the critical chain tasks they protect.

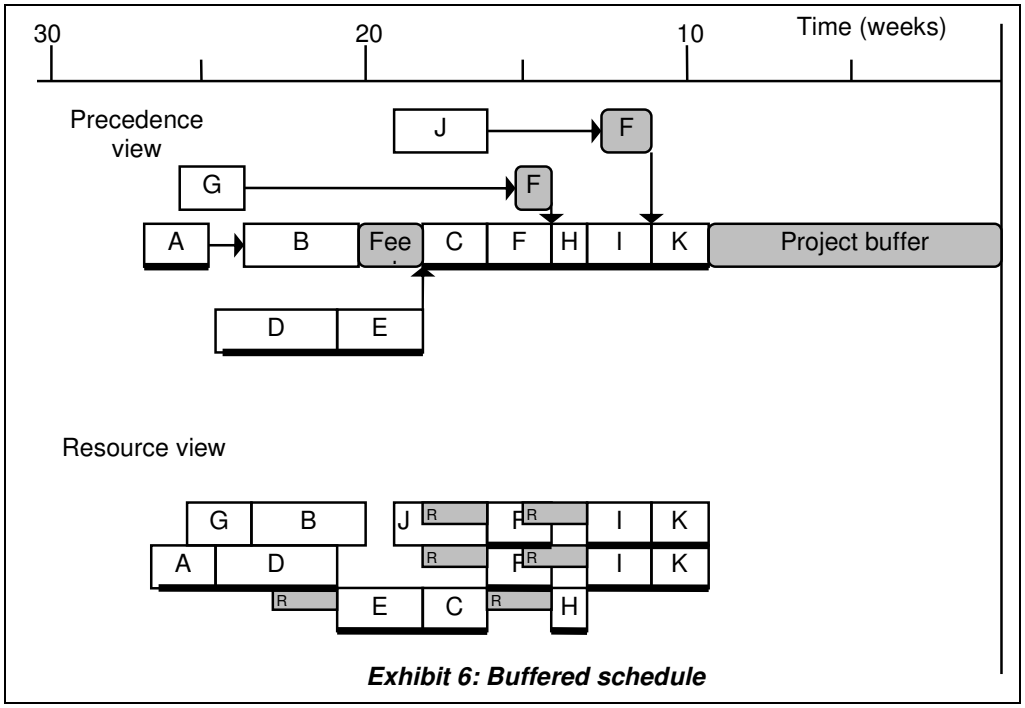


Exhibit 6: Buffered schedule

Inserting a feeding buffer between task B and task C requires that task B and task G each be moved one week earlier. What would happen if the feeding buffer between tasks B and C were longer? A feeding buffer of three weeks can easily be handled by advancing tasks B and G by another week. If the feeding buffer were to be four weeks, the only solution would be to postpone the completion date by one week. On the other hand, since the extra week is small in comparison with the project buffer of nine weeks, one could ignore it.

The final schedule

The final schedule contains no milestones, since a milestone inevitably leads to the student's syndrome [5]. Multi-tasking is absent, since the schedule indicates clear priorities between tasks [5]. For instance, when a critical chain task arrives, everything else should be dropped to complete it. In fact, the resource buffer warns the resource about the impending arrival. Always finish a task as quickly as possible.

Shortening the project

If the project needs to be speeded up, how can that be achieved? The easiest way to shorten the project seems to be a reduction in the project buffer. That should really be the last resort. The project buffer should be reduced only if there is a compelling reason to believe that a shorter project buffer will be adequate.

References

[1] Newbold; *ProChain project scheduling – Critical chain concepts*, Creative Technology Laboratories, 1997.

[2] Sparrius; *Critical Chain part 1 – Projects are padded with lots of safety duration,* PROJECTPRO, Vol. 8, No. 1, February 1998, pp 22 – 23.

[3] Sparrius; *Critical Chain part 4 – The critical chain,* PROJECTPRO, Vol. 8, No. 4, July 1998, pp 46 – 49.

[4] Sparrius; *Critical Chain part 3 – The issue of random variations and dependent tasks,* PROJECTPRO, Vol. 8, No. 3, May 1998, pp 52 – 54.

[5] Sparrius; *Critical Chain part 2 – Safety is horribly wasted,* PROJECTPRO, Vol. 8, No. 2, March 1998, pp 41 – 42.

This article was first published in ProjectPro magazine Vol 8 No 5, September 1998

Back to top
