Kumpania CONSULTING

Practitioner papers

INTRODUCTION TO CRITICAL PATH ANALYSIS

Roderic Gray

1991,1998

www.rodericgray.com

Critical path analysis (CPA) is one of the basic techniques of project management. It allows planners to establish when work packages can take place, and what the effect on the overall project will be if something happens to affect the schedule. Because it places activities in time it is also used as the basis for planning budgets and for scheduling resource requirements. Even in fairly straightforward projects the technique can be extremely useful and for complex projects it is almost indispensable and is the basis for a range of project control operations including budgeting and performance measurement. The basic technique is simple and, once mastered, allows for more sophisticated factors to be handled fairly painlessly, such as complex relationships between work packages, the effects of non-working days and the need to schedule work to make optimum use of resources.

In practice, the calculations are usually done by computer, and some ~ but by no means all ~ software packages can also produce good clear graphical representations of the information which can be very helpful in communicating plans to other people. Using a computer package without a good understanding of what the machine is doing with the source data can be risky, though, so familiarisation with the techniques is worth the effort it demands. It will also enable the project planner to get the inputs right, without which there's little chance that the outputs will be much use!

Building-Blocks

Before a critical path analysis can be performed it is necessary to identify all the work packages that go to make up the whole project. ("Work packages" may also be called tasks or activities - the terminology varies but whatever term is used it means a separate, identifiable piece of work which is a constituent part of the overall project). There are several techniques for doing this, of which the Work Breakdown Structure, or WBS, is probably the best-known and most widely used. Work packages are the basic building-blocks of a project and it is worth spending time to identify and refine their definitions.

The most commonly-used format for displaying work packages and their inter-relationships is to draw boxes to represent the work packages and then link them together to show their relationships. This is known as a Precedence Network and is the format used by most modern project planning computer software. It is a particularly clear way of representing a project and tends to focus attention on work packages one at a time. The essential elements of a work package are its:

- Description
- Duration
- Timing
- Responsibility
- Resources

Work Package Description: A summary in a few words of the task to be carried out. What this covers will vary with the "level" of the plan - it could be a synopsis of an entire subsidiary project, or a very precise and strictly limited activity.

Work Package Duration: Once a work package has been defined it should be possible to estimate how long it is likely to take to carry it out. There are three basic methods of estimating a work package's duration:

a) comparison with similar tasks performed in the past (but, was the previous performance level satisfactory?, are the two tasks really comparable in every relevant way?).

b) analysis of the activity and estimation of the durations for the component elements. This involves planning at a much lower level than required for the overall project plan, but the components can be "re-assembled" once a total duration has been estimated. If it proves impossible to estimate the duration of every part of the activity, at least the overall inaccuracy will be limited.

c) guesswork. (some guesswork may be involved for the first two methods as well).

The estimate of duration should usually be the "best guess" or most likely figure. Sometimes it will be best to play safe and use the most pessimistic estimate, but if this is done too often the overall duration of the project will stretch out towards infinity!

The duration of a work package is the time required for carrying it out. It has nothing to do with the time available for doing it (see the remarks about "float", below).

Work Package Timing: The nature of some tasks will dictate their timing (in real time, ie, against the calendar), but the timing of most work packages should be determined solely by their relationships with other work packages or events. Arbitrarily fixing the timing of work packages without a pressing reason causes problems which can be difficult to track down. Even where a timing may need to be fixed it is usually best first to use the Critical Path Analysis methods described below to find out when it logically *can* happen and then make any changes necessary if it turns out that the "natural" timing is not what's wanted.

Work Package Responsibility: This means identifying someone who will be responsible *to the Project Manager* (an important point) for ensuring that the work package is carried out to its specified time, cost and quality. Finding someone to take on this responsibility is possibly the most crucial task a project manager has to carry out and is the key to successful project implementation.

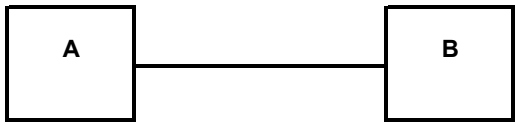
Work Package Resources: These are the resources, people, materials, money, etc., which will be required to complete the work package. Usually, they are identified on a hypothetical basis to begin with - "what resources would have to be provided in order to carry out this activity?". Once this has been done the availability of resources is considered and any conflicts are dealt with as necessary. In practice, of course, the two processes can't be separated as neatly as this and resource requirements are assessed with at least some idea of the kind, quality and quantity of resources that might be available. The second stage, comparing availability with requirements, is then more a matter of fine tuning.

Because work packages will eventually be set against time-scales, attaching resource requirements to individual work packages enables quite accurate forecasts of outgoing cash-flow to be made.

Logical Relationships

Once the component parts of the project – the work packages – have been defined, the ways in which they relate to each other can be described. There are four ways in which one work package can relate to another, described below. (In each case work package **A** "drives", or governs, the relationship).

1 A Finish to Start relationship



In this case work package **A** must be completed before work package **B** can begin. This is by far the most common relationship, following normal thought-patterns ("once A is done we can get on with B").

2 A Finish to Finish relationship

Here, work package **B** cannot finish until work package **A** is complete. This is fairly unusual but can be a useful way to describe what happens when, for example, **B** depends on closure reports or documentation produced in **A**.

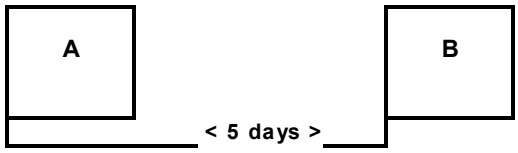
3 A Start to Start relationship

Work package **B** can only start once work package **A** has started.

4 A Start to Finish relationship

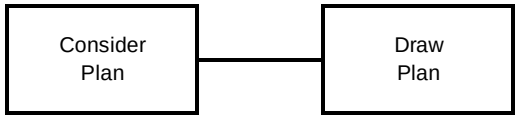
Work package B cannot finish until work package A has started.

In any of these relationships it's fairly common to impose a delay, or *lag*. For example. a delay of 5 days on a Start to Start relationship:

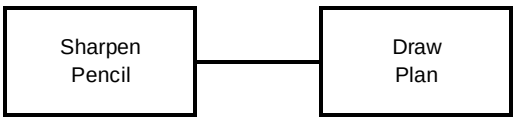


would mean that work package **B** can only start 5 days after work package **A** has started. Equivalent delays can be applied to any of the four relationship types. This can be a useful way of representing time when nothing (apparently) is happening, such as the gap between placing an order and receiving delivery of the goods. In a situation like this a delay is used as an alternative to specifying another (and fairly useless) work package. It is also possible to make delays negative.

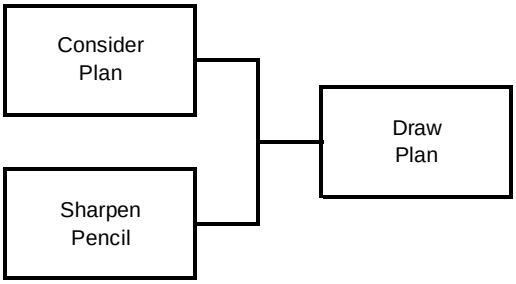
Specifying the relationships between work packages is simple in principle but it can be quite hard to do in practice with real tasks in a real project. Care taken at this stage, and involvement of the work package managers, really pays off if it avoids errors which can affect the entire project plan. The most common traps in defining relationships are: 1) to miss an important relationship and 2) to define bogus relationships. Of the two traps the second is the more common and the hardest to eliminate. A trivial example illustrates the point:



There is an obvious Finish to Start relationship between thinking about the plan and drawing it on paper. There is also an obvious similar relationship between sharpening your pencil and doing the drawing:



There isn't, however, any logical relationship at all between considering the plan and sharpening the pencil. The fact that you may habitually do the three activities in a particular order is not in itself a sound reason, in planning terms, for defining a logical relationship between them. In the above case the true logic produces the following complex relationship:



Doing the sums

If there seems to have been a great deal of preparation before reaching the point of performing some arithmetic then this reflects the way critical path analysis should be done in practice. Performing calculations on casually-produced source data results in misleading answers. This is the biggest risk attached to the use of so-called "project management" computer software, which rapidly performs sophisticated calculations but doesn't have any means of checking whether the source material is sensible.

There's another, rather subtle but none the less invaluable benefit from this painstaking preliminary work, which is that everyone involved in it now understands the logic behind the schedule, and the need for/relevance of each work package. This can quite literally make the difference between success and failure.

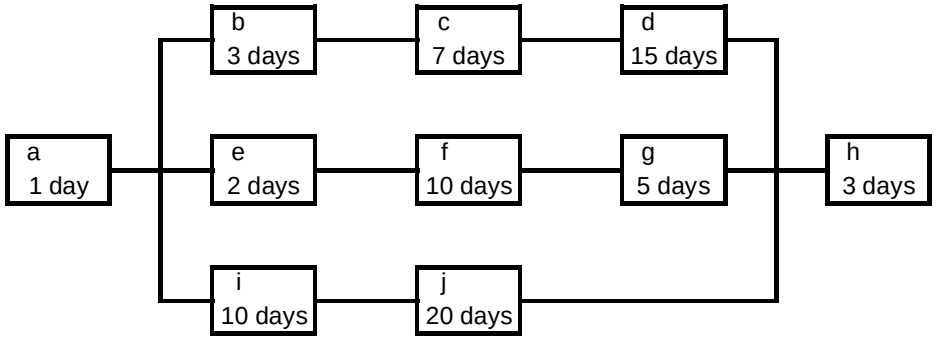
The calculations to be performed in CPA are basically quite simple, but complications can be added-in to deal with special situations.

Step 1

Draw the "Network" of work packages, showing their durations and their logical relationships.

In the example below all the relationships are simple, Finish to Start with no delays imposed. Also no allowance will be made for weekends or other non-working days.

Step 1 - Draw the network



Step 2

Assuming that work starts at the beginning of day 1, add the durations of the work packages together, starting from the left and working along each "path" of related work packages separately:

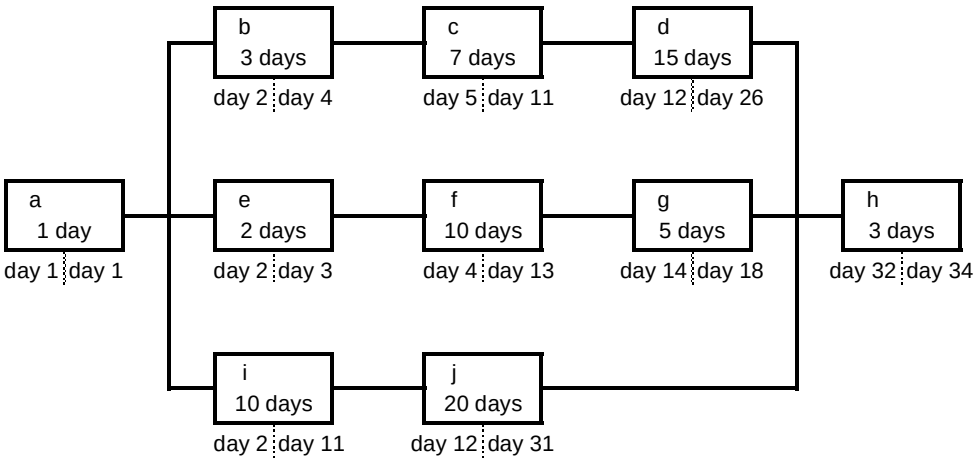
a + b + c + d + h
a + e + f + g + h
a + i + j + h

Work package **a** starts at the beginning of day 1 and takes 1 day, so it should finish at the end of day 1. This means that work package **b** can start at the beginning of day 2. It takes 3 days, so it should finish at the end of day 4.

This means that **c** can start at the beginning of day 5, and so on. Similarly, work package **e** also follows **a**, so **e** can start at the beginning of day 2 as well. **e** takes 2 days so should be finished at the end of day 3. This means that **f** can start at the beginning of day

The only complication so far is that the start date of work package **h** keeps changing because the three "paths" that lead up to it are of different lengths. Obviously, the start date of **h** will be governed by the longest of the "paths" leading to it. The starting and finishing dates of each work package are shown underneath the work package "box" on the diagram below:

Step 2a - Left to Right



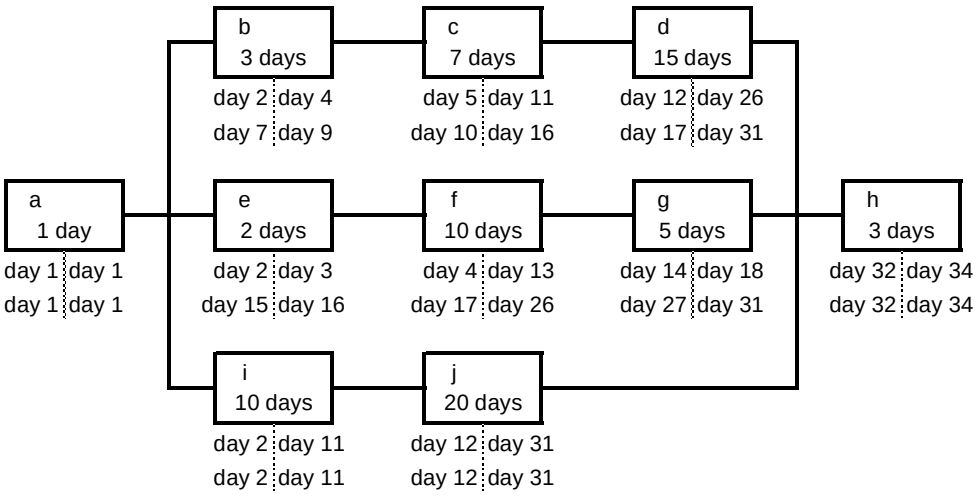
This left-to-right calculation, called the "forward pass", has shown the earliest date on which each work package can start and the earliest date on which each can be completed. Since this produces the finishing date for the last work package in the network, it also shows the earliest date by which the whole project could be completed.

A second calculation shows the latest date by which each work package must start if this project completion date is to be met. This is done by working from right to left, known as the backward pass, subtracting work package durations from finishing dates.

Thus, work package **h** finishes at the end of day 34 and takes 3 days, so it needs to start at the beginning of day 32 (we already knew that). This means that work package **g** must finish by the end of day 31. As **g** takes 5 days it must start at the beginning of day 27. Therefore work package **f** must be finished by the end of day 26 and since **f** takes 10 days it must start at the beginning of day 17 at the latest. Similarly, work package e which takes 2 days must be completed by the end of day 16 and must start at the beginning of day 15. These dates are clearly different from those produced by the "forward pass".

A similar pattern emerges along the **h**-**d**-**c**-**b**-**a** "path", with a different set of dates resulting from the "backward pass". However, along the **h**-**j**-**i**-**a** "path", which has the longest overall duration, the two separate calculations produce the same dates.

Step 2b - Right to Left



The dates for work package **a** are determined by those of **h**-**j**-**i**, because that is the "path" with the longest overall duration.

The difference between the "forward pass" and "backward pass" dates is known as *float*. The float of work package **g**, for example, is 31-18 (or 27-14) = 13 days. This means that **g** could be delayed or over-run by up to 13 days without delaying the completion of the project.

The same would apply to **e** or **f**. However, 13 days is the *total* amount of float available to activities **e**, **f** and **g**. If **e** is delayed by 7 days, and **f** then takes 6 days longer than estimated,**g** has got to go according to plan or the project will over-run.

Because this float is shared between several work packages it should accurately be called *interfering float.* Float which can all be used up on one work package without impacting on another (eg, where there is only one work package on a particular "path" through the network) would be called *free float*. Both kinds of float added together are a work package's *total float*. In the simple network above there isn't any *free float*, so *total float* and *interfering float* are the same.

Work packages which have no float, in other words, ones which must start and finish on particular dates or else the whole project will be delayed, are called *critical*. The "path" through the network which is made up of "critical" work packages is called the *critical path*. The term "critical" in this context has nothing to do with the relative importance of the task. A work package may absolutely vital to the project's success but still not be "critical". The definition of "critical work package" as "one having zero

float", is the usual convention and should be the default definition. It is sometimes redefined as, for example, "having less than 5 days' float" (or 2, or 10) and some computer software provides for this.

Critical path analysis is one of the basic techniques of project management and is the basis for a range of project control operations including budgeting and performance measurement. The basic technique is simple and, once mastered, allows such complications as complex relationships between work packages, the effects of non-working days and the need to schedule work to make optimum use of resources to be handled fairly painlessly.

In practice, the calculations are usually done by computer, and most software packages will also produce good clear graphical representations of the information which can be very helpful in communicating plans to other people. Using a computer package without a good understanding of what the machine is doing with the source data can be risky, though, so familiarisation with the techniques is worth the effort it demands.

Familiarity with the mechanisms and process of critical path analysis, even if a computer is always used to perform the mathematics, can lead to faster, more accurate project planning, and to a much deeper understanding by all the participants of the details of their specific project. A little time spent on learning about this invaluable tool now will pay dividends when future projects are being planned and implemented.