RDM

Relationship Diagramming Method

A Thesis

Submitted to the Faculty

of

Drexel University

by

Fredric Leigh Plotnick, Esq., P.E.

in partial fulfillment of the

requirements for the degree

of

Doctor of Philosophy

August 2008

Table of Contents

List of Figures

Abstract

RDM Relationship Diagramming Method
Fredric Leigh Plotnick, Esq., P.E.
Joseph P. Martin, Ph.D., P.E., Supervising Professor

The Critical Path Method of Planning and Scheduling (CPM) was developed in the mid to late 1950s to improve upon the Gantt Chart or bar-chart, by using the recently developed digital computer to perform much of the tedium of the process. The methodology developed was therefore largely dependent upon and constrained by the limits of digital computers of that era, which initially lacked what is now called "random access memory." The enhancement of adding RAM in the mid 1960s led to a variant in the original methodology. To make the distinction the original method was now named the Arrow Diagramming Method (ADM) variant, while the newer variant was named Precedence Diagramming Method (PDM.) However, due to the still limited computers of the 1960s, the methodology utilized for PDM was less rigorous than that theorized by leaders in the field. Commercialization of personal computers in the 1980s exacerbated these issues, as initial models had less capability than the large mainframe computers for which most software had previously been developed.

In considering the industry application of CPM, a side effect of the dispersion and convenience of personal computing was that simplified software was mass marketed. Many individuals with lesser training and experience with CPM now have access to sophisticated scheduling software. Such individuals may unwittingly misuse the software and possibly obtain erroneous output.

The further advancements in power and memory of desktop computers to the present day has largely been used to provide superior graphics, but not to resolve these issues, nor to provide additional functionality beyond that developed in the 1950s. The purpose of this research has been to revisit the simplifications of the developers of the 1950s, 1960s and 1980s, and then to develop a more robust and accurate methodology utilizing the capabilities of modern computers. This led to development of a third variant, named RDM (or Relationship Diagramming Method.) This dissertation discusses the problems related to the initial development of CPM, limitations of the ADM and PDM variant, and how these have been addressed using the new RDM variant.

Chapter 1 – History of Infrastructure Project Scheduling

The Critical Path Method of Planning and Scheduling (CPM) was developed in the mid to late 1950s to improve upon the Gantt Chart or bar-chart, by using the recently developed digital computer to perform much of the tedium of the process. This initial work was developed by James E. Kelley, Jr. and Morgan Walker for DuPont, and in an effort to reduce the time and effort required to schedule plant turnarounds. A similar and parallel effort, known as Performance Evaluation Review Technique (PERT,) was developed by the United States Navy to assist in the development of the Polaris Missile System. A key difference between the two systems was the respective focus upon activities (or defined scopes of work) and events (or defined deliverables.) Both systems recognized the importance of events, either as the start and finish of an activity or as independent entities, and also focused upon the relationships between the respective activities and events.

An example of the difference between CPM and PERT is illustrated below:

- CPM may use three activities, "Form Wall," "Rebar Wall," "Pour Wall," (or perhaps a one combined activity "Form/Rebar/Pour Wall,") following an activity "Cure Foundation." Each activity will be assigned a duration, and possibly specified resources.
- PERT has only an event "Wall Erected" following an event "Foundations Poured." A clearly estimated duration (or range of Optimistic, Most Likely, and Pessimistic durations) is provided between the two events, with no detail of "how to get from one to the next," nor any suggestion of the resources that may be needed to accomplish such a task or tasks.

Another example graphically illustrates the difference in Figure #1 (next page,)

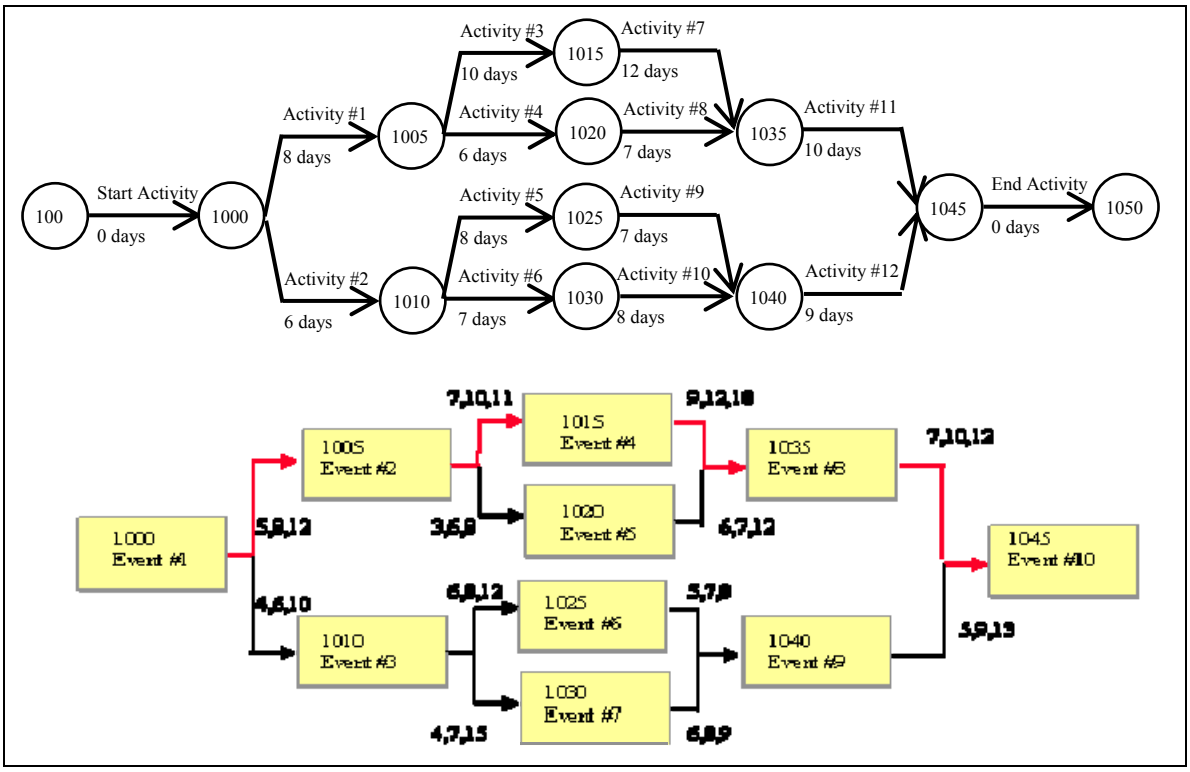


Figure #1 – Graphic Illustration of CPM (ADM variant) versus PERT

The methodologies developed were largely dependent upon and constrained by the limits of digital computers of that era, which initially lacked what is now called "random access memory." This is somewhat illustrated by the graphic Figure #2 taken from Practical PERT ("A Step by Step Guide") by BJ Hansen (1964, America House.) This figure also illustrates many of the challenges of early users of these techniques. While a bar-chart is a simple task to assemble and interpret, only those properly trained could develop a logic network in the special format required for data entry to a computer, and the output also required specialist review for interpretation and reporting to the other members of the project team, these being the ultimate decision makers. The professional result was that users of the early computerized schedule methods were able and required to use judgment to compensate for the methodology limitations.

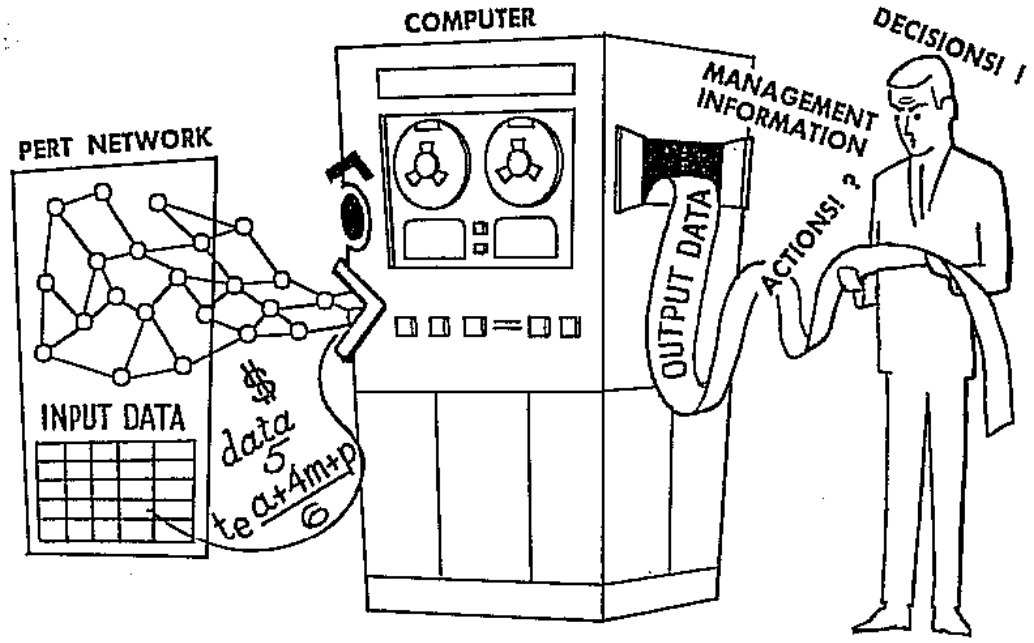


Fig. 1—*Computer Processing Pert*

Figure #2 – Graphic depiction of early computers utilizing "linear access memory"

The enhancement of adding "RAM" in the mid 1960s led to a variant in the original methodology, distinguishing the original method, now named the Arrow Diagramming Method (ADM) variant, from the newer variant, named Precedence Diagramming Method (PDM.) However, the implementation of PDM was done in a manner which diminished the recognition of events, even to the point that a Milestone event in PDM must be coded as a zero duration activity. This and other historical information provided in this section, is perhaps best discussed in the text, CPM in Construction Management, James J. O'Brien, McGraw-Hill, 1965, and as updated through the Sixth Edition, now O'Brien and Plotnick, 2005. Further citation to this source, as well as interviews with Mr. O'Brien and the personal experiences of the author of this dissertation, have been omitted for purposes of better readability.

Recent developments relating to the field of CPM have drawn attention to the current focus on activities and the lack of focus upon the relationships between these activities that was the hallmark of the original ADM and PERT methodologies. An article in Engineering News Record (ENR) (written by Richard Korman and Stephen Daniels, 26MAY03) brought many of these issues to the fore and set off a heated debate amongst professionals in the field of scheduling. In the article, "four experts lamented the state of scheduling, … [and] widespread abuses of powerful software to produce badly flawed or deliberately deceptive schedules that look good but lack mathematical coherence or common sense about the way the industry works. The result is confusion, delayed projects and lawsuits."

Responding to the ENR article just cited, the co-founder of CPM, James E. Kelley, Jr., writes:

> Your cover story, Critics Can't Find the Logic in Many of Today's CPM Schedules, paints a disheartening picture of the current state of CPM schedules (ENR 5/26 p. 30). It's only 46 years since Morgan Walker and I first worked out CPM for duPont and yet project people are still falling into some of the same scheduling traps warned against during CPM's childhood. The use of features like "leads and lags," "multiple calendars" and "assigned constraints" do provide some levels of schedule flexibility. In practice, their use too often leads to inconsistent schedules and misleading views of project condition.
>
> In the early days of CPM, computing capability was at a premium. Rooting out inconsistencies in scheduling data had to be left completely to the planner. In practice, this meant deliberately limiting the use of the "flexibility" features. Today, the desktop computer I'm using to compose this letter has far more capability than the UNIVAC we used for our first CPM calculations. Thus, there is no reason why the computer cannot be programmed to tell me that my scheduling input is inconsistent and why.

Mathematics and Computer Issues

Modern scheduling started in the early 1900s with the management studies of Fredrick Taylor and Henry Gantt. (This is attributed by several sources to an article, *Work, Wages and Profit*, Gantt, *The Engineering Magazine*, New York, 1910.) The Gantt chart, or bar chart, familiar to industrial and infrastructure engineers, provides an easy graphical depiction of the work to be performed. However, it is a static view. Because relationships between the listed activities are not defined, even a minor change order may require preparation of a new bar chart to account for such change. Even determining the impact of changing the duration of one activity of the bar chart requires a great deal of work.

Figure #3 below illustrates this point. A Gantt Chart for a small project initially envisions the activity of "Excavate for Sewer" to require 10 working days. Based upon a 05JUN00 start, the overall project is predicted to be capable of completion by 04SEP00. It is possible that such calculation, if performed by hand, may take less effort than to enter such information into a electronic format for calculation by a computer. However, if it is later anticipated that the excavation activity will require 20 working days, it requires much less effort to recalculate the impact of this change by computer than by manual efforts. As noted in Figure #3, the impact of the 10 day change is only 5 days.

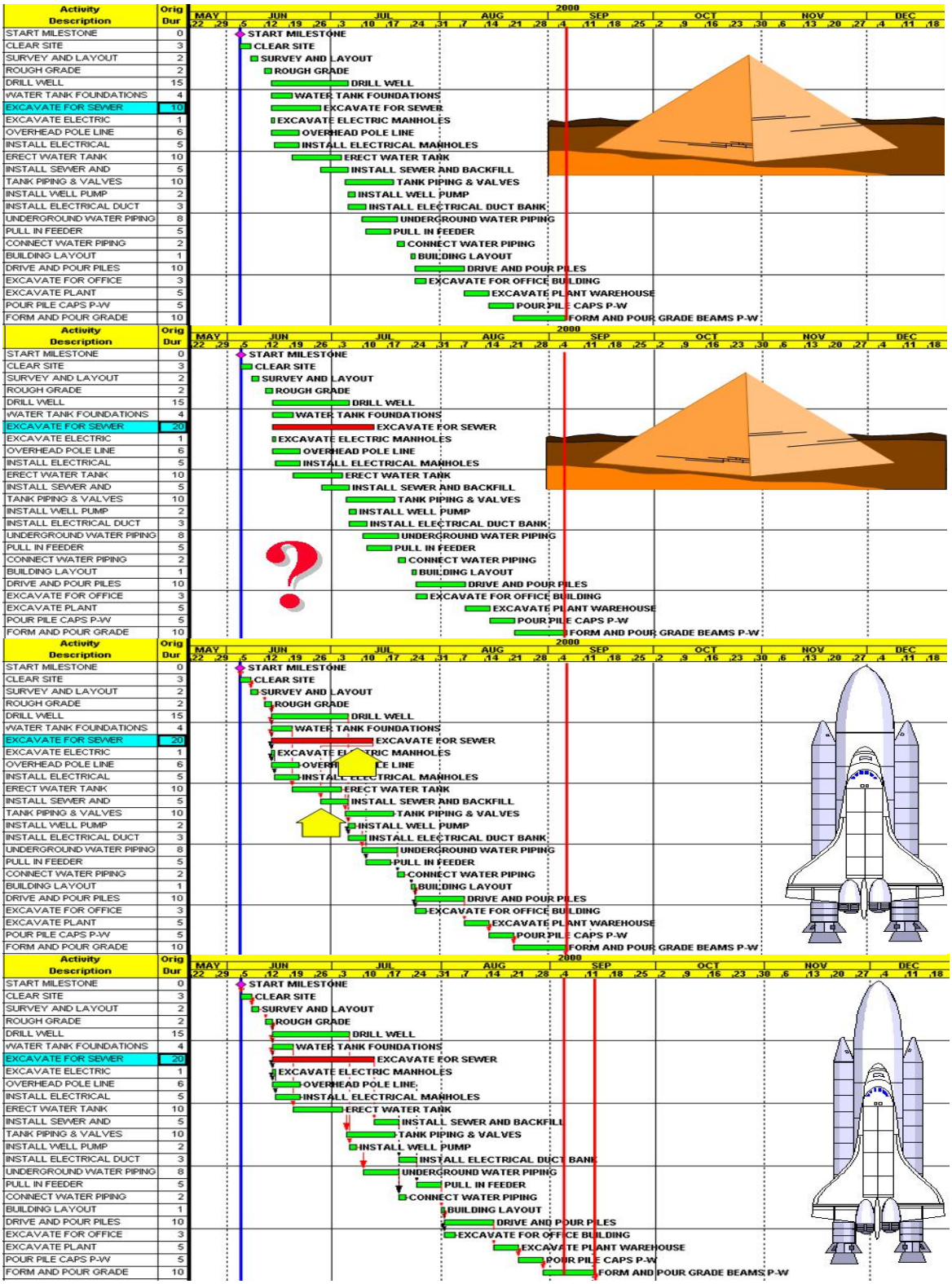


Figure #3 – CPM automates the revision of the bar-chart to incorporate change

In 1956, James Kelley and Morgan Walker developed the Critical Path Method of schedule analysis, or CPM. A documented test pitting a traditional "bar-chart" scheduling group against a separate group given but a 40 hour course on the new CPM technique was performed by DuPont under the direction of Kelley and Walker in 1957. The results of this study catapulted the use of the new technique both within and outside of DuPont.

A similar methodology, based upon the same mathematics, was developed by the Navy and is known as PERT. Both methodologies used relatively primitive computers having only linear memory provided by reel to reel tape. In 1964, more powerful computers with random access memory were used, and these were capable of handling both an activity file (including the duration of the activity) and precedence file (including the restraint from one activity to the next) concurrently. Using such a computer, IBM implemented the PDM concept that had been discussed for several years in non-computer form.

An example of the two variants of CPM, now named ADM and PDM, is provided in Figure #4 below. A small logic network for planning and scheduling of a building foundation is shown in ADM (top) and PDM (bottom.) The ADM logic network shows event nodes connected by activities (description of work above line, duration below) and pure logic restraints (lines not bearing activity description data.) The PDM logic network places the activity description and duration within a box with such boxes then being connected by pure logic restraints. Note the absence of event nodes in PDM.

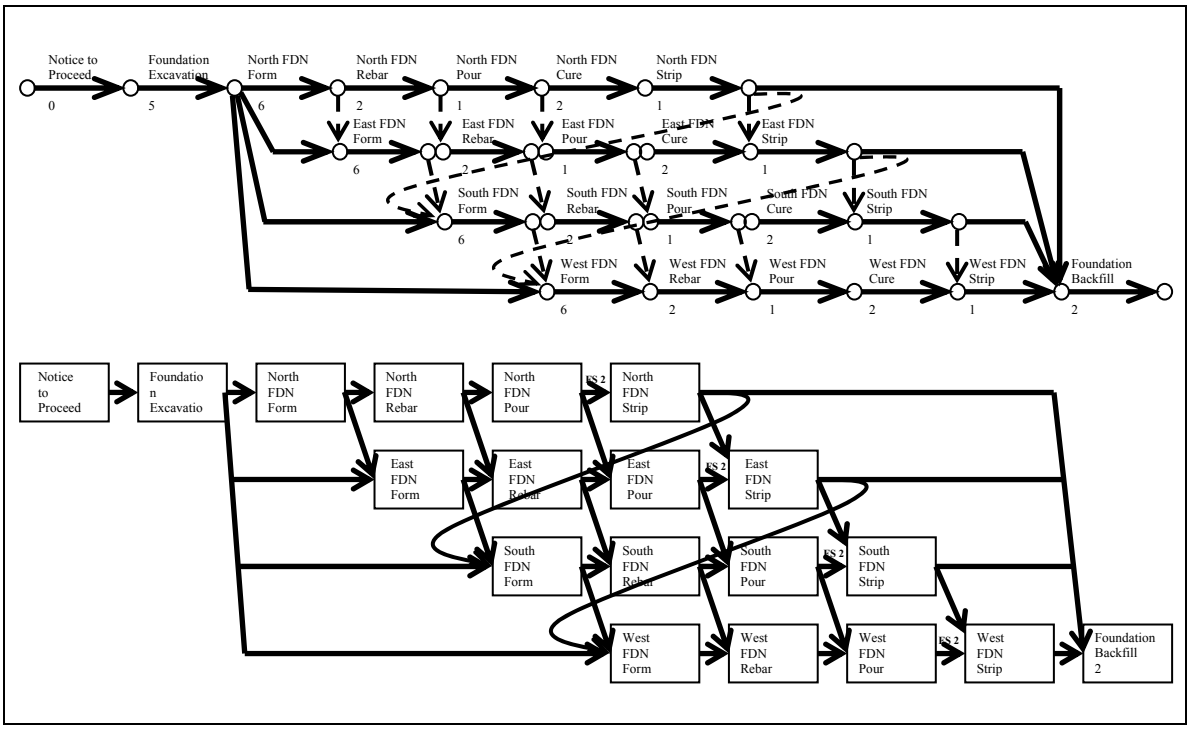


Figure #4 – Examples of an ADM and PDM logic network for a typical foundation

Efforts to commercialize these methods, such as by the consulting firm then formed by Kelley and Walker, had difficulties convincing industrial companies other than DuPont to use their services. The focus shifted to market the concept of CPM to construction professionals. This was greatly aided by James J. O'Brien's classic text, CPM in Construction Management, first published in 1964 and now in its 6th edition. Acceptance by this market has moved CPM planning and scheduling from academia to practice and made CPM planning and scheduling a requirement for the management of construction, ranging from large scale infrastructure projects to more modest projects such as home improvements. Indeed, as provided in the introduction to the 6th edition:

> The Critical Path Method (CPM) was developed specifically for the planning of construction. The choice was fortuitous, since construction accounts for more than 10 percent of the annual gross national product. Almost every activity and every person is affected to some degree by new construction or the need for it …. The construction industry is a heterogeneous mix of companies ranging in size from the large operations to one-person operations. No matter the size, construction

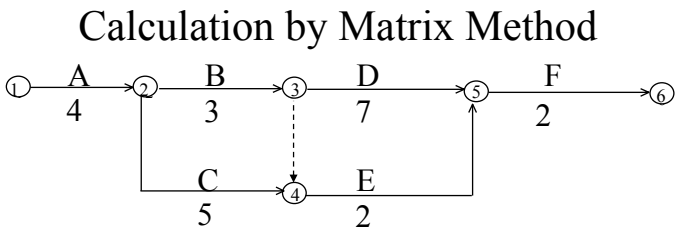
> companies face similar situations and, to some degree, similar pressures. Many factors, such as weather, unions, accidents, capital demands, and work loads, are either beyond individual control or difficult to control. New problems in project approvals due to increased public awareness include pollution and ecological controls. CPM does not offer clairvoyance, but it does assemble all the information to the project managing team.

And so, today, the management of a construction project is typically under better control than that of many other fields, as indicated in the body of the text.

The concept of the initial ADM variant of CPM appears to be intuitive, but rational use requires a deeper understanding of mathematics. Each activity resides somewhere between two points of time – its early start and late finish. Each point in time occurs after the completion of one or more activities – other than the first point in time – and prior to the start of one or more activities – other than the last point in time. While the computer calculates one time-early for each point in time during the forward pass, and one time-late for each point in time during the backward pass, secondary calculations convert the time-early to the early start of each activity emanating from that point – plus activity duration to calculate the early finish. Similarly, the point's time-late attribute is converted to the late finish of each activity terminating at that point – and then subtracting the duration to calculate the late start of the activity. The total float attribute is then calculated by subtracting the early finish from the late finish – or subtracting the early start from the late start, since addition is commutative.

Figure #5 illustrates this by showing how CPM was initially taught and how calculation of project duration and various activity attributes were calculated. Information about each activity or pure logic restraint between events were transcribed to a matrix for

further calculation. (Note the duration of activity A being copied to the matrix cell of *i*=1 and *j*=2.) The manual calculation then added the duration for *j*=2 to the initial "time early date" (TE_i) defined at project start as of zero on the line for *i*=1, and entered as the TE_i for line *i*=2. This is followed by adding the duration for *j*=3 to the TE_i for line *i*=2 (or 4 plus 3) and entering such as the TE_i f for line *i*=3. Further detail of this tedious method may be found in CPM in Construction Management or other texts.

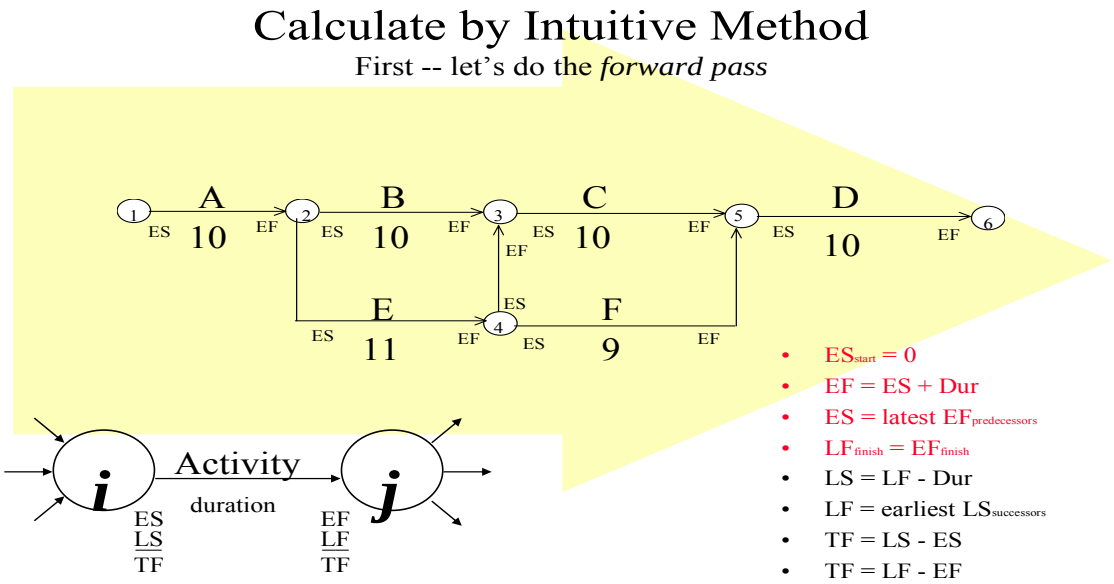


i/j	1	2	3	4	5	6	TE***i***	max(TE***i*** +D***ij)***		***i-j***	dur	ES	EF	LS	LF	TF
1	---	4					0	defined as 0		1-2	4	0	4		4	
2	---	---	3	5			4	0+4		2-3	3	4			7	
3	---	---	---	0	7		7	4+3		2-4	5	4			12	
4	---	---	---	---	2		9	4+5 or 7+0		3-4	0	7			12	
5	---	---	---	---	---	2	14	7+7 or 9+2		3-5	7	7			14	
6	---	---	---	---	---	---	16	14+2		4-5	2	9			14	
TL***j***	0	4	7	12	14	16				5-6	2	14			16	
min(TL***j***-D***ij***)	4-4	12-5 or 7-3	14-7 or 12-0	14-2	16-2	defined as TE***i***						TE***i***	ES + D		TL***j***	

Figure #5 – Calculation of TE*i* and TL*j* – then ES, LF – then EF, LS – then TF

TE*i* – Time Early for the *i*-node
TL*j* – Time Late for the *j*-node
ES – Early Start of an activity
EF – Early Finish of an activity
LS – Late Start of an activity
LF – Late Finish of an activity
TF – Total Float of an activity

Modern software tends to skip the step of calculation of the early and late times for the event nodes (and later morphing to PDM which ignores nodes entirely) but the step is still implicit in the "intuitive" method of calculation. Figure #6 illustrates this "intuitive" method by skipping the steps of calculating a TE_i and TE_j for each event node, and rather



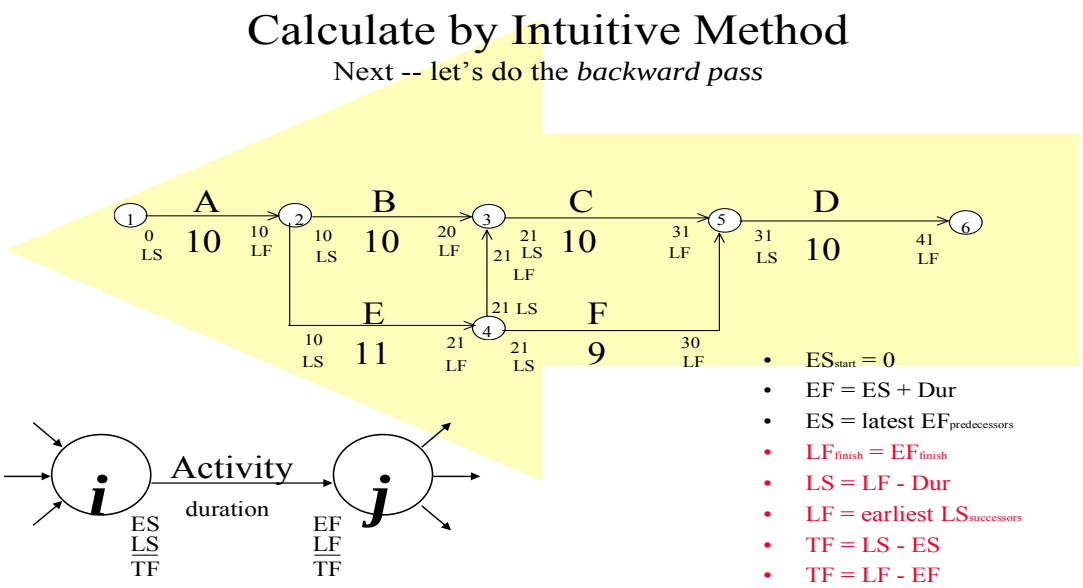


Figure #6 – Modern software skips the step of calculation of TE*i* and TL*j*

immediately calculating the activity attributes of early start (ES) and early finish (EF) in a "forward pass," setting the late finish of the last activity as equal to the early finish (thus

making the earliest completion "critical,") then calculating the late finish (LF) and late start (LS) in a "backward pass," and finally calculating the total float (TF.)

PERT is based upon the same mathematics. Clearly defined events (nodes) are separated by logic representing vaguely defined work of vaguely defined duration. (In PERT, only the event deliverables are defined, the work required to get from one to the next is not stated and must be inferred. Similarly, the time required to get from one event deliverable to another has a greater variability since the work required to get from one to the next has not been clearly defined.) Each event – or point in time – occurs after one or more of these logic lines of estimated duration – or vectors – other than the first point in time – and prior to one or more vectors emanating from that event – other than the last point in time. Because the format deals with events and not activities, true PERT does not have early or late starts or finishes. There exists only a time-early attribute of when the event MAY first occur and a time-late attribute of when the event MUST occur if completion of the project is not to be delayed. True PERT only calculates TE*i* and TL*j*.

A question arises whether PDM is more or less simple than the original, or ADM, variant of CPM. On the surface, PDM appears more simple. Event nodes are absorbed into the activity. The additional memory and power of the new breed of computers dramatically changed the algorithms for solving the scheduling problem, eventually going to direct calculation of the early start (ES,) early finish (EF,) late start (LS,) and late finish (LF) for each activity. The new capability to run restraints from the "start" of an activity, or to the "finish" of an activity did have the downside of creating the need for a "start float"

and "finish float" and "most critical float." (If the finish of an activity is determined by a finish-to-finish restraint from an activity finishing later than the date calculated by the early start plus duration (EF > ES + Dur,) then the late finish of the activity less the early finish will yield a different result than the early finish less the early start (LF-EF ≠ LS-ES.) However, the input is simple, and the output is easy to read, and if the mathematics is a little vague, the user still has an answer that works. This is illustrated in Figure #7 which depicts how the total float calculated from EF-ES may differ from that calculated from LF-LS. Figure #7 also illustrates how casual users of PDM may envision how the events nodes of ADM are addressed by PDM. This belief is incorrect, as is discussed below.

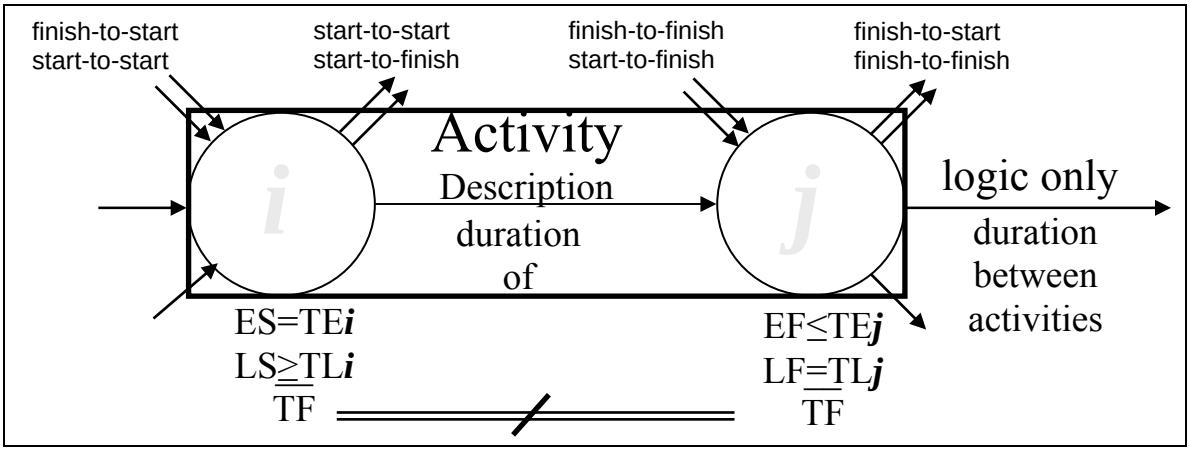


Figure #7 – Assumed Simplicity of PDM

The elimination of visible event nodes (or "points in time") has other consequences. The old rule (for ADM) stated "each point in time occurs after the completion of one or more activities and prior to the start of one or more activities." The new rule (for PDM) states "each activity starts after one or more other activities and prior to one or more other activities." The first such consequence to be examined focuses upon the additional feature of "Start-to-Start" and "Finish-to-Finish" restraints that is the hallmark of PDM

(rather than the sole "Finish-to-Start" restraint permitted by ADM) creates several other issues.

Figure #8 illustrates how an activity having only a Start-to-Start successor, or only a Finish-to-Finish predecessor, can compromise the validity of a logic network. If the only restraint emanating from an activity is a start-to-start restraint, that activity need never finish – completion of the activity is dissociated from the logic network. If the only restraint terminating at an activity is a finish-to-finish restraint, the start of that activity may presumably be before "And then there was Light."

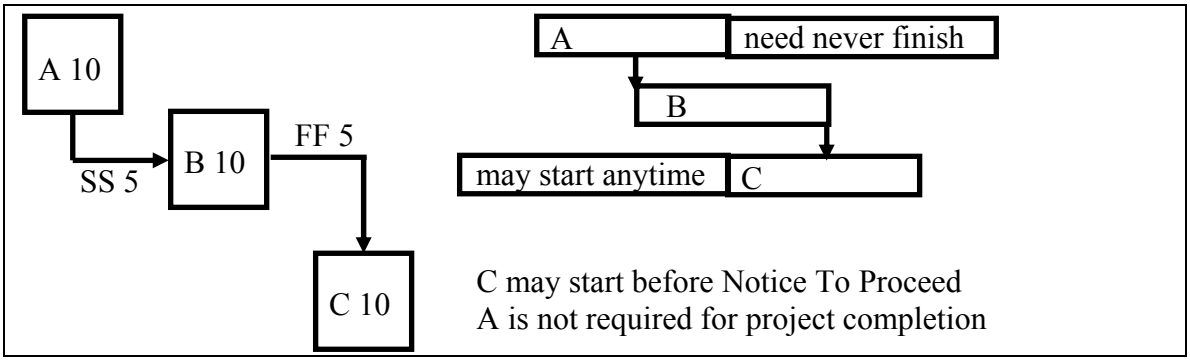


Figure #8 – Hidden "Open Ends" created by PDM logic

The new rule should have stated, "each activity may start after the completion of one or more other activities and each activity finish must be prior to the start of one or more other activities."

The consequences of this type of error in logic are not inconsequential. It is not uncommon for an activity, such as A in the example above, to be reported as started and then face some issue causing a delay in completion. The failure of the CPM to forecast an impact to activity B, or for that matter any other activity of the project, may certainly

reduce the effectiveness of the project management team using this tool. Diagnostic reports of most major software systems that perform CPM, as tested by this author, fail to note this type of "open end" in a logic network. The diagnostic notes merely that activity A has a successor and activity C has a predecessor, but fails to check whether the finish of A has a successor or the start of C has a predecessor. Thus enforcement of this proper rule is left to the manual tedium of the scheduler, reviewer or other user.

The non-traditional start-to-start and finish-to-finish restraints create another mathematical issue that is typically ignored. If the start and finish of an activity are separately restrained, it is possible that the finish of an activity is later than the start plus the activity duration. This is illustrated in Figure #9 which depicts the start of an activity B being restrained until 5 days after the start of activity A, and the finish of B being restrained until 5 days after the finish of A.

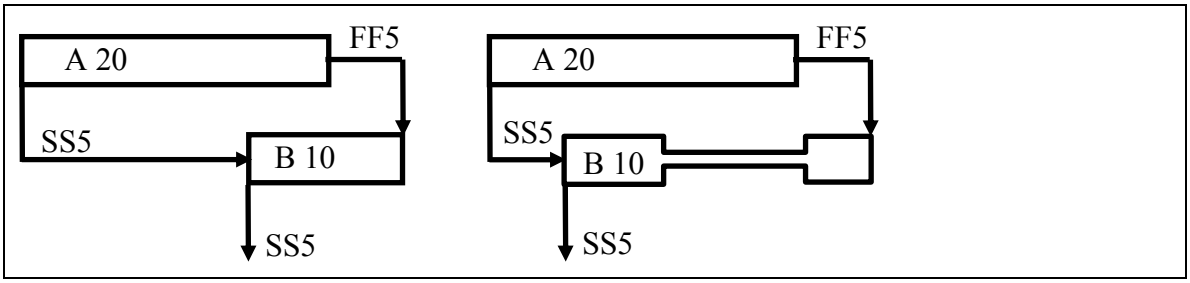


Figure #9 – PDM issue of interruptible activities

Two interpretations of these instructions are possible, either allowing B to start but then wait until the completion of A until continuing to completion, and delaying the start of B so that once resources have been mobilized, they can be used without interruption. This is the default for most commercial scheduling software. If the activity is interruptible, a whole host of additional questions arise, such as into how many segments may the

activity be split, the minimum period of a segment of work, and most importantly how much of the activity may not be performed until the predecessor is complete.

On the other hand, if the activity is to be performed in a continuous flow of work, this may have serious consequences to the start of other activities tied to the start of the referenced activity. For example, again referring to Figure #9, a hypothetical activity C (not shown but following the SS5 restraint flowing from B,) may start 5 days after the start of B. In the right example, this will be 10 days (5 plus 5) after the start of A. In the left example, this will be 20 days after the start of A as the start of B is deferred in the name of avoiding interrupting performance of B. Some software vendors provide only the left example option, some provide both. Finally, the software vendor must balance the need to make this an activity by activity determination of several questions for each activity with the need for ease of use by the end user.

Thinking in terms of activities, rather than events (or points in time) may lead to other problems, such as the non-existent loop error illustrated in Figure #10, where the start of activity “drywall” is followed by activity “electrical rough-in” which is followed by finish of activity “drywall.” Traditional PDM, which has abolished the concept of “events,” reads the logic as “Drywall to Electrical to Drywall.” Traditional ADM would break the “Drywall” activity into three components, “start drywall,” this to be followed by the “electrical rough-in” and “continue drywall,” with these two activities to be followed by “complete drywall.” It was the difficulties in reporting progress for three (or more) “sub-activities” of the one physical activity of “drywall” that led to the prevalence

of PDM over ADM. However, in this example, software supporting PDM fails. The proposed RDM model, advanced later in this paper, reestablishes events and reads the logic as "point 'A' of partially performed Drywall to Electrical to point 'B' of concluding portion of Drywall."

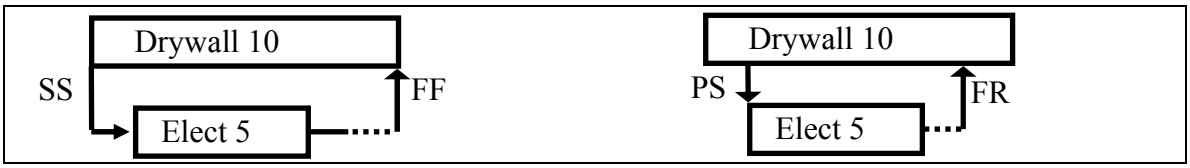


Figure #10 – PDM false loop error report versus RDM accepted logic

The most serious issue may be distinguishing measuring passage of time from the reported start of an activity as compared to measuring progress towards reducing the remaining duration of the activity to zero. If work, in the real world, progresses exactly as planned, this does not present a problem. However, as illustrated in Figure #11, if work does not progress as planned, it becomes extremely important to know whether an activity B should start seven days after a 10 day activity A has started, or an activity B should start after 70% or 7 days of a 10 days activity A have been performed.

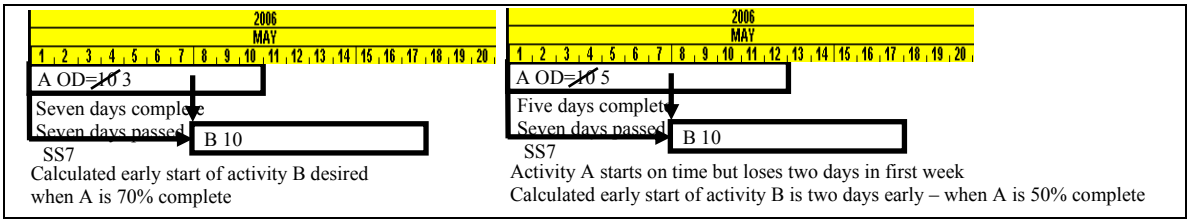


Figure #11 – Misinterpretation of a Start-to-Start relationship after an Update

If, for example, progress falls "behind schedule" and after 7 days from the start of activity A only 5 days of work have been performed, one would expect the start (and finish) of a successor activity B to be delayed by 2 days. However, most modern CPM software programs do not distinguish between whether 7 days of work has actually been performed - leaving 3 days remaining duration, and 5 days of work actually performed -

leaving 5 days remaining duration. The calculated dates for performance of activity B are the same in each instance.

The reason this is so serious is that it highlights that the PDM activity not only has a hidden ***i*** and ***j*** node, or start and finish events, but also may have additional events internal to the activity. This is illustrated in Figure #12. Like the mathematics behind the ADM variant of CPM or PERT, these hidden points in time each have attributes of time-early, time-late as well as total float and others. Without recognition of these internal events, the user is forced to say a restraint is start-to-start when possibly the user actually meant something else.

Once recognizing these internal events, the user can say what is actually meant. Start B 7 days after A has started. Start B when 7 days of A are complete. Start B when 70% of A is complete. Start B when 35cy of the 50cy of A have been excavated. And so forth.

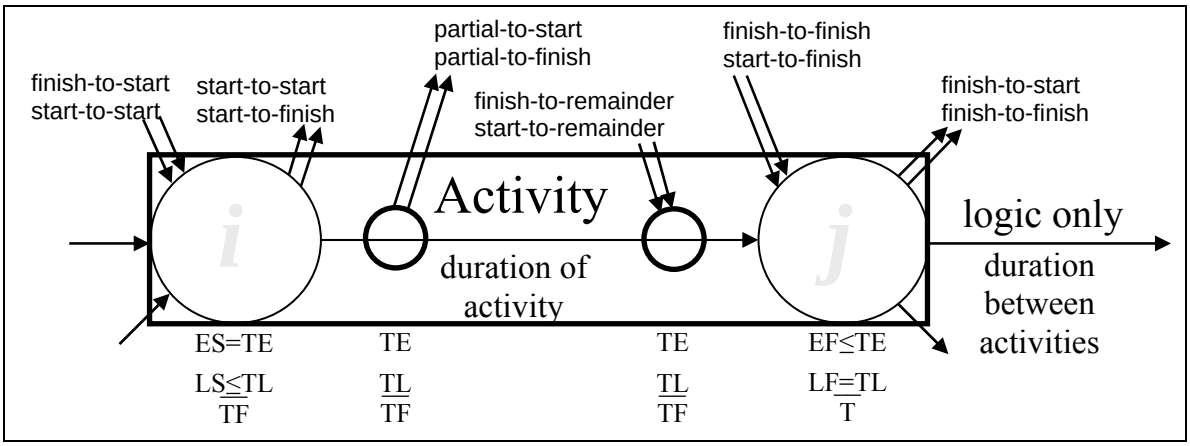


Figure #12 – Actual complexity of PDM

Human Interface Issues

An entirely separate issue pertains to the operational aspects of usage of any of the methodologies discussed. Although the general concept was taught to many individuals in the engineering and construction fields for the purpose of "selling" the methodology, actual implementation was usually left to specialists, generically known as "schedulers." Many of the issues previously discussed were known to these "specialists" who would conduct interviews with project managers and other team members, and then convert the information obtained to hand-drawn logic networks, followed by calculation by hand or computer. Experienced schedulers would then routinely adjust the logic networks or interpret the calculations to take into account the issues discussed above.

Thus the field of scheduling evolved into a "priesthood" that used both the generally available basic instructions as well as a verbal tradition of "tips and tricks" to make the calculations relevant to the real world. This is similar to other fields of engineering where the choice of "factor of safety" is left to the discretion of an *experienced* professional engineer. This is perhaps a key reason why the right to take the professional engineering exam is predicated upon not only study but a number of years of "apprenticeship."

The advent of the PC may have provided the tools to a great number of new end users, but did not pass on these verbal traditions. Shortcuts and special software features designed to simplify the planning and scheduling process, in appropriate situations, are now made available to any individual who has a PC. The results are predictable.

How do project managers like to schedule a project? Each starts with a preconceived basket of resources – crew, equipment, whatever – and schedules – or bar charts – the movement of that resource. Only rudimentary consideration is given to checking if the infrastructure needed to use these resources is in place, since the project manager instinctively knows that, for example, the foundation crew will move fast enough to be ahead of the crew erecting the prefab walls. However, the point of CPM is to record this instinctive (or experiential) knowledge to permit automated rescheduling when unexpected events occur.

"Priests" understood these issues and, while building a logic network, would periodically review the restraints between activities to be assured that (1) each activity had all required physical precedent logic and (2) multiple sets of resource restraints did not creep in as the network was being developed. Even here there were issues. The early CPMs were typically run on large mainframe computers via timeshare services. The formula for pricing these services included a charge for each activity or restraint. Although the charge for each restraint was perhaps only a penny or two, with networks of hundreds or thousands of activities and restraints, there was pressure to remove "redundant" physical restraints, although these still may be noted in the scheduler's notes.

These notes, incorporated in a full, annotated logic network diagram, were typically redrafted (by hand) to mylar and submitted (via reproducible sepia) along with the computer printouts and the most important item, the narrative. The narrative would then

explain the “input” or logic and interpret the “output” or calculations for the non-specialist engineer’s review. Somewhere, in the mid-1980s, as specialists were replaced by project managers, armed with PCs, preparing Microsoft Project or Primavera schedules, the practice of enforcing the submission of these pure logic networks was dropped, leading to removal of the requirement from typical specifications.

That the requirements to provide pure logic diagrams were dropped is somewhat understandable. These typically are only read by scheduling specialists. The new software generated bar-charts that could be read by any engineer, inspector or contractor. Any additional information buried in the computer reports could be explained, perhaps by a specialist or perhaps not, in the narrative. But, even where a specialist may provide the narrative, only that information deemed useful to the engineer need be discussed. Certainly, information questioning the accuracy of the entire procedure need not be mentioned.

For one of the “dirty secrets” of CPM is that the entire methodology, as typically presented to management, is flawed – CPM does not accurately predict the total length of a project nor does it predict the probable date of performance of any one activity of the project. This was known to the original creators and promulgators of the methodology, who understood that the computers of the day were not powerful enough to properly address the issue and thus instead chose to do so by providing a factor of safety (or contingency) to be determined by experience.

Quoting from page 168 of the first edition of CPM in Construction Management, (and as written by the sole author of that edition, James, J. O'Brien,)

> There is a definite tendency for the actual completion date to exceed the first CPM end date. It is, then, reasonable to allow for some contingency between the CPM end and the actual desired completion dates. There is no definite answer on how much contingency to allow for, because it will vary with the specific circumstances of the project. However, if you need a 12-month period for completion of the project, set your CPM goal at about 11 months, and so forth.

As explained by this author in the 6th edition of the text,

> A more mathematical approach to this issue is to look at the consequences of merge bias upon the amount of contingency required. Assume that each "Most Likely" duration given is an estimate somewhere between a "best case" or Optimistic duration and "worst case" or Pessimistic duration. We would expect that the "worst case" estimates are further from the "Most Likely" than the "best case" estimates. Let us assume only a slight skewing of this nature, such as the "best case" is 15 percent better than the "Most Likely," but the "worst case" duration is 20 percent greater than the "Most Likely."
>
> Now consider the John Doe project. If a subset of activities are extracted such that there is only one linear chain of activities forming the network, and the durations are randomized to between 15 percent less than estimated to 20 percent more than estimated, it would be expected that this skewing to the high side would tend to make the project take longer than the simple CPM calculation (Figure 22.10.1).
>
> And in fact, due simply to this skewing between "best case" and "worst case," there is only a 14 percent chance of completion by the 21DEC04 date calculated by the CPM algorithm. Similarly, there is only a 50 percent chance of completing by 28DEC04 (1 week late), a 95 percent chance of completing by 07JAN05, and a 99 percent chance of completing by 12JAN05 (3 weeks late). Thus, even for the simplest logic network of a single linear path, there is a 5 percent likelihood that this 10-month project will run over by more than 12 work or 17 calendar days (Figure 22.10.2). (*See* Figure #13 for the two figures within this quotation.)

The full John Doe project logic generates a 50 percent likelihood that the project will extend beyond 25JAN05, or more than one month for this ten month project. However, imagine the reaction of the owner to a schedule submittal that indicates a significant probability of such a 10% overrun. The owner would typically desire a 100 percent

assurance of timely completion (this calculated to approximately 07FEB05) while the knowledgeable engineer may understand that a 90 percent or even 80 percent certainty should be the maximum pursued from a cost benefit standpoint. The unknowledgeable engineer is now only angered that the previously accepted schedule submittal is shown to be "inaccurate."

Activity ID	Activity Description	Orig Dur	Total Float	Early Start	Early Finish
0	START MILESTONE	0	0	01MAR04	
100	CLEAR SITE	3	0	01MAR04	03MAR04
1	SURVEY AND LAYOUT	2	0	04MAR04	05MAR04
2	ROUGH GRADE	2	0	08MAR04	09MAR04
3	DRILL WELL	15	0	10MAR04	30MAR04
4	INSTALL WELL PUMP	2	0	31MAR04	01APR04
5	UNDERGROUND WATER PIPING	8	0	02APR04	13APR04
8	CONNECT WATER PIPING	2	0	14APR04	15APR04
13	BUILDING LAYOUT	1	0	16APR04	16APR04
14	DRIVE AND POUR PILES	10	0	19APR04	30APR04
15	EXCAVATE PLANT	5	0	03MAY04	07MAY04
16	POUR PILE CAPS P-W	5	0	10MAY04	14MAY04
17	FORM AND POUR GRADE	10	0	17MAY04	28MAY04
18	BACKFILL AND COMPACT P-W	3	0	31MAY04	02JUN04
19	UNDERSLAB PLUMBING P-W	5	0	03JUN04	09JUN04
20	UNDERSLAB CONDUIT P-W	5	0	10JUN04	16JUN04
22	FORM AND POUR SLABS P-W	10	0	17JUN04	30JUN04
29	ERECT STRUCTURAL STEEL	10	0	01JUL04	14JUL04
30	PLUMB AND BOLT STEEL P-W	5	0	15JUL04	21JUL04
31	ERECT CRANEWAY AND	5	0	22JUL04	28JUL04
33	ERECT BAR JOISTS P-W	3	0	29JUL04	02AUG04
34	ERECT ROOF PLANKS P-W	3	0	03AUG04	05AUG04
35	ERECT SIDING P-W	10	0	06AUG04	19AUG04
300	SET ELECTRICAL LOAD	2	0	20AUG04	23AUG04
38	INSTALL POWER CONDUIT P-W	20	0	24AUG04	20SEP04
43	INSTALL BRANCH CONDUIT	15	0	21SEP04	11OCT04
49	PULL WIRE P-W	15	0	12OCT04	01NOV04
50	INSTALL PANEL INTERNALS	5	0	02NOV04	08NOV04
54	TERMINATE WIRE P-W	10	0	09NOV04	22NOV04
55	RING OUT P-W	10	0	23NOV04	06DEC04
56	ENERGIZE POWER	1	0	07DEC04	07DEC04
415	FINE GRADE	5	0	08DEC04	14DEC04
94	SEED AND PLANT	5	0	15DEC04	21DEC04
80	END OF NETWORK MILESTONE	0	0		21DEC04

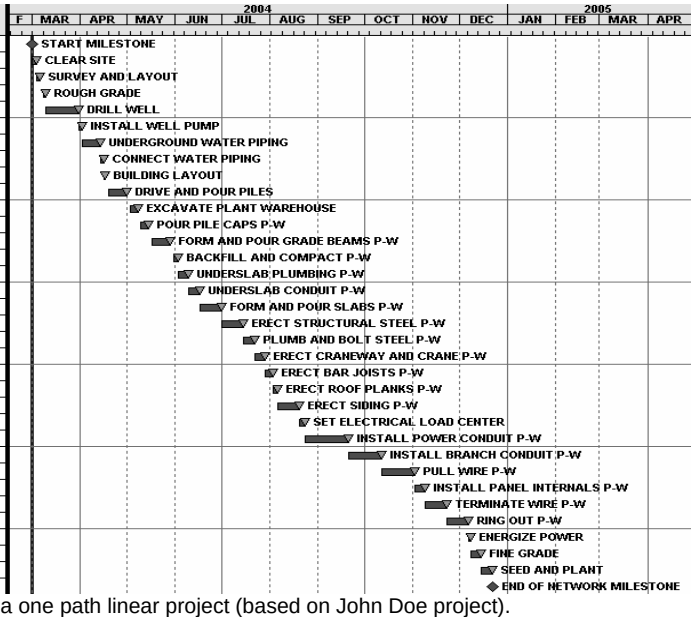


Figure 22.10.1 Critical Path for a one path linear project (based on John Doe project).

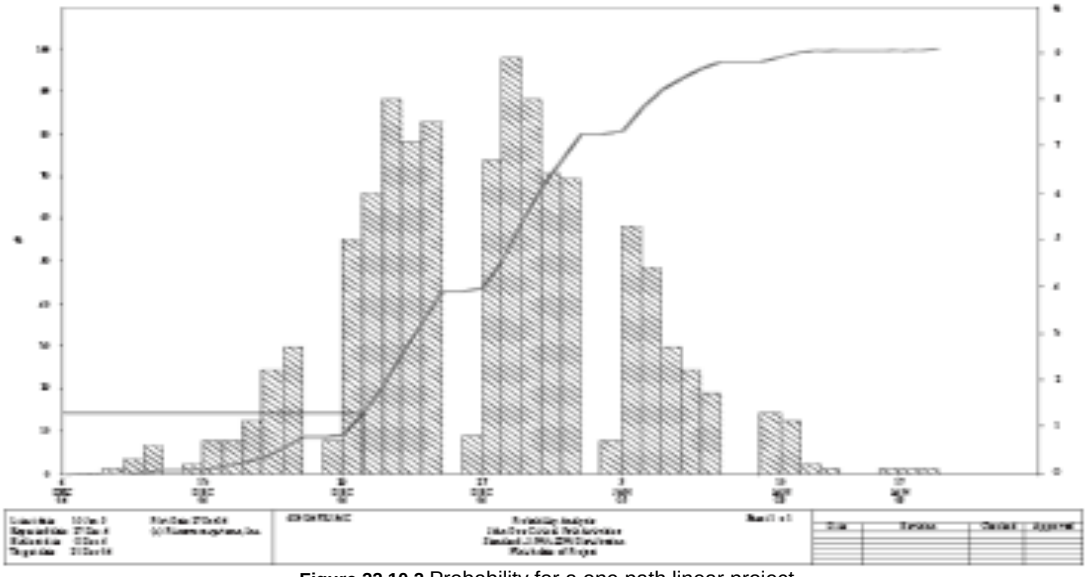


Figure 22.10.2 Probability for a one path linear project.

Figure #13 – Figures 22.10.1 and 22.10.2 of CPM in Construction Management, 6th Ed.

The concept of Merge Bias is best illustrated in Figure #14. While a list of numbers, such as a bid estimate, with a known tolerance will still tend to total toward the mean, with each high measurement balanced by a low measurement, the same is not true for a schedule. In a schedule, if one path experiences a high measurement and the other a low measurement, the high measurement counts. If both are high, the higher will count. Only if both are low measurements will the total be lower to balance against the many high totals. Thus, in the example of Figure #14, the four cost items of a bid estimate still tends to total 40, but the four activity items of the schedule will trend to total 31⅔ rather than 30.

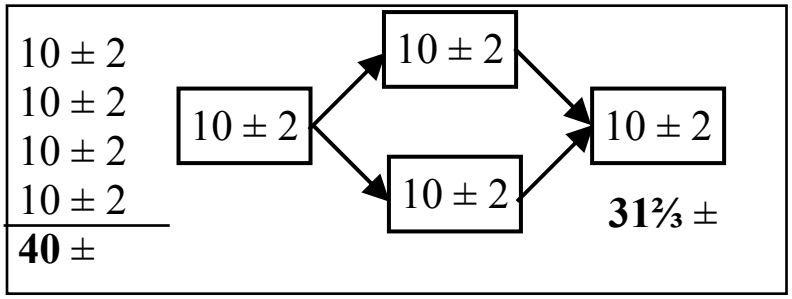


Figure #14 – Illustration of Merge Bias

This problem is then exacerbated by unknowledgeable engineers who specify that the contractor provide a schedule that "uses all the time allotted by the contract." By deliberately requiring the contractor to defer work near the end of the schedule, this requirement all but guarantees that the contractor will finish the project late. This, if properly illustrated in a Court of Law, may thus relieve the contractor for being late and even be the basis for the contractor to claim compensation for such late completion.

These complaints of many practitioners of CPM scheduling were raised in the 2003 article in ENR. The maths of the 50s were watered down to power of the 50s computers.

Introduction of PCs provided tools (but not skills) to the masses. More powerful PCs tend to used by software vendors to provide more features and benefits - more glitz – rather than to improve the accuracy of the output, even if users understand that the CPM calculated completion date may not be truly accurate.

These more powerful features include graphical user interfaces (GUIs) and WYSIWYG input/output representations that favor bar chart style scheduling rather than true CPM. "Wizards" within the software further reduce the need for skills. In conclusion, the question is raised whether PDM and modern software has thrown away the proper use of logic networks – the baby – with the effort to keep the logic clean – the bath water? As the ENR article asks, "Where is the logic?"

As discussed in the ENR article, in CPM in Construction Management, and elsewhere, the proper procedure in preparing a CPM is to first plan the job – what MUST be done before the next step. This may be performed in ADM or PDM. Then – and only then – proper procedure suggests the practitioner to schedule the plan based upon limited resources. Unfortunately, when this information (planning logic and scheduling logic) is entered into the computer, the distinction between the two logics is lost. If there is a need to revisit the logics, such as for a change or delay, much of this work must be performed again.

To summarize the problem:

- The addition in the new PDM format of logic other than the traditional "finish-to-start" was compromised by the elimination of the event nodes of the ADM format.

- The addition of the finish-to-finish restraint created the prospect that the span of time from early start to early finish may be greater than early start plus duration (EF > ES + Dur.) This raises an unanswered question whether performance of the scope of the activity may be interrupted or the start of work must be deferred for continuous performance.
- Another issue of duration, not discussed above, is whether for statusing current progress for an update requires a need to measure performance (such as for erecting masonry) or merely the entry of a start date (such as for curing of concrete.)
- Another issue of duration, not discussed above, is whether the calculation should expect that an activity started out-of-sequence (prior to when the logic anticipates) may continue to be performed or must stop and await completion of the stated predecessor(s.)
- Restraints between activities required by physics (gravity) or contract (per a specified sequence) are not distinguishable from those which are added at the discretion of the project team (such as to reduce the number of concurrent resources to be used.)
- The definition of the added start-to-start and finish-to-finish restraints of PDM are often incomplete and a subset of all of the possible restraints that may be actually requested by practitioners.
- The introduction of computers that are both faster and have more memory (both RAM and on disk) may provide additional data output to be calculated or inferred from the input already provided. The quality and quantity of outputs should not be limited to that which could be provided of the computers of the 1950s and 1960s.

Chapter 2 – Introduction to RDM

The introduction of RDM, or the Relationship Diagramming Method variant of CPM, represents the first major reformulation of the use of a logic network to plan-then-schedule a project since the inception of CPM and PERT in 1956/1958, other than the addition of PDM (or the Precedence Diagramming Method variant) in 1964. (PDM had been theorized in the 1950's along with the original variant of CPM, now known as ADM or the Arrow Diagramming Method variant, but as previously noted, early computers limited to "linear access memory" could not readily support software for practical use. The addition of random access memory (RAM) in the 1960's greatly alleviated the difficulty of flipping between an activity/duration file and predecessor/successor file required for solving the PDM algorithm.) The key aspects of a RDM system include the ability to:

- Identify nodes representing events or points of time at each point where restraints emerge or converge. Such nodes will be similar to the ***i***-node of ADM but are established not for the purpose of data entry (as in ADM,) but rather for identifying points where a definable scope of work (comprising all or part of an activity) has been completed, points of merge bias (where several restraints or logic lines come together) and the "mini-milestones" that these points represent.
- Identify the rationale or reason for each restraint, both by a code and description thereof. A physical restraint (*e.g.* "erect the walls before the roof") is the most obvious example. Other types of restraint may include resources including crew, equipment, reusable forms and others all already part of the thinking of the team preparing the CPM. However, by expanding the recording of assumptions behind the plan used to

prepare the schedule, additional power may be gained such as (1) automated guarantees that each activity in the network is preceded by a physical restraint, (2) permitting "what-if" analyses of the impact of limiting or not limiting crews by various craft, reusable forms or other specific resources, (3) sorting and selecting by reason for relationship and (4) providing an automated guide to areas of possible corrective action when various events threaten to delay or disrupt timely completion of a project.

- Expand the types of relationships between activities to account for how people actually plan their actions, rather than to match the options set by software designers. For example, few people would say "Bob is starting a 30-day activity next week and Mary will start her activity 15 days after Bob has started without regard to how much progress Bob has made." Rather, more people will say, "Bob is starting a 30-day activity next week and Mary will start her activity when Bob is 50% complete." Thus, if the scope of Bob's activity changes or if his productivity is other than expected, there will be an automatic change to the lag between the start of work for Bob and Mary.
- Provide the same level of control over lag durations (between activities) as is (1) provided for activities, such as choice of calendar, and (2) range of duration for those systems that support PERT and SPERT style calculations.
- Expand sort and select capabilities to the text of the activity description and various activity codes of the predecessors and successors of an activity. For example, a selection may highlight each instance where work by the mechanical subcontractor is immediately followed by work by the electrical subcontractor.

These aspects may be generalized as the five classes of new coding required of RDM, as will be discussed in the following subchapters:

1. Event Codes,
2. Duration Codes,
3. Reason/Why Codes,
4. Expanded Restraint or Lead/Lag Codes and
5. Relationship Codes.

Event Codes

Event codes may be used to carry three classes of information. The first is merely descriptive and includes a unique identifier and description text or title field. These are similar to those that describe an activity. However, events codes carry information related to point-in-time events (or milestones) as to be compared to activity codes for span-of-time activities. Previous users of PERT will appreciate this subtle difference.

In the absence of a PERT-style description denoting a distinct milestone, the event description may be merely descriptive of the merge of logic leading to the event and dispersal of logic from the event. Thus a computer generated description may read "Event requiring completion of activities A and B before starting C," or "Event requiring completion of D and 50% of E before starting F" or "Event requiring completion of G before completing the last two days of H." Users of traditional CPM will recognize the second example as including a start-to-start restraint and the third as including a finish-to-finish restraint. The true advantage of this description field may be realized where the description is expanded to explain "which" 50% of E is required for F, such as "installation of windows may occur when masonry reaches mid-height;" or the scope of the "last two days of H" that cannot be complete until completion of G, such as "parapet masonry cannot be complete until roof flashing is installed."

Neither the traditional ADM nor PDM models truly describe these situations. Using ADM, activity E could be broken into E1 and E2, but the masons performing the activity would not consider such as two separate activities and may have difficulty

using and updating a schedule with such an artificial differentiation. Nor would the masons desire to stop and return to complete the parapets – thus masonry of this one story structure is one activity with coordination to and from others. Using PDM, one may maintain that masonry is one coordinated activity, but then lose the knowledge and the cues relating to the coordination involved. An "SS50%" or "start-to-start at 50%" restraint does not accurately describe to the masons what is needed by the window installer; an "FF2" or "finish-to-finish with 2 days lag" restraint does not tell the masons or roofer the manner in which their work must be coordinated.

The second class of codes is user defined coding similar to the user defined activity codes provided in various software products to allow organization and summarization by such codes. These may be of use in categorization of milestones and "mini-milestones," especially where a project includes a number of ultimate deliverables, such as the erection of a residential condominium high-rise.

The third class is code fields that are to be calculated (or counted) by the RDM software implementation. This may include counting the number of predecessor restraints and successor restraints to an event. Such a function could be duplicated in part for an ADM or PDM system, but with limitations. In that the "events" embedded within PDM "activities" do not distinguish whether such events are at the ends (start or finish) of an activity or somewhere within an activity, the analysis value of such an exercise is reduced. Thus, a typical use may be listing all activities subject to an finish-to-finish-with-lag restraint noting that these require special coordination to avoid stopping work

(and the mini-demobilization and remobilization that would then occur) if the "FF" predecessor is not complete on time.

In the RDM model, the count of predecessor (and successors) may further distinguish predecessors that are based upon a physical restraint from those based upon resource restraints. Yet another count may distinguish between exclusive predecessors (that is a predecessor that is the sole successor of an activity) from non-exclusive predecessors (where the predecessor activity may have multiple successors.)

The uses of such a metric may include provision of a "physically open end" diagnostic to alert the preparer or reviewer that the basic rule of all CPM logic networks, being the start of each activity (other than the first) must be preceded by a physical restraint. A similar use may be to alert the preparer or reviewer that a specific resource restraint, say a carpenter crew, precedes an activity but that two carpenter crew restraints succeed that activity. This is a common error that occurs when a crew is reassigned during the planning/scheduling exercise and the Scheduler forgets to delete the old reference. However, it is believed that the most important aspect of the counting of these new attributes will be in the formulation of more powerful leveling and smoothing algorithms.

Duration Codes

The concept of duration may appear simple, but it is not. RDM, matching the possibilities of the real world that have traditionally been ignored or "fudged," requires at least three separate duration codes: progress v clock, retained logic v progress override, and continuous v interruptible.

The first issue is of definition. In CPM and PERT, duration may be a measure of the performance of work of an activity or it may be a measure of the necessary passage of time between events. It is neither a measure of the span of time during which an activity may be performed, nor a measure of the span of time between events. Technically, the two mentioned spans may be defined as duration plus total float; and thus, by definition, duration is less than or equal to such spans.

For example, an "activity" may be estimated to require a duration of one week. If the "activity" is "form/rebar/pour a foundation," the duration is a measure of the work that must be performed, and measurement of the actual work in place will be the basis for reporting progress (either by stating a remaining duration of work left to complete or by stating a percent of work complete.) If the "activity" is "cure," the duration is a measure of the time required from the event initiating this "activity" until the event indicating the "activity" is complete and successor activities may start. While one may calculate the status of "curing" once started (with the assistance of a calendar or clock,) such need not (and probably cannot) be measured.

The distinction between these two types of duration may, at first, appear inconsequential. However, if the type of duration can be recorded as a "duration code," as well as the duration, the necessity during an update of counting the days since concrete was poured, or since a shop drawing was submitted (where the owner has a stipulated duration for review) may be avoided. Note here that there may also be the need for a code to "stop the clock" at one (or other) unit of time for a manual confirmation of completion during an update (in this case that the shop drawing has been approved.) A more serious consideration of this distinction will be further discussed below under the heading of Expanded Restraint or Lead/Lag Codes. This first type of duration code may be considered as measurement by Performance or by Clock (P/C.)

A second duration code considers when the progress measurement or clock calculation is to start. In an initial schedule, this issue will not arise as there is only one logic. Similarly, in an update, if an activity has not yet started, the issue will not arise. However, in an update, if an activity has started but not yet finished, and if the activity has started "out-of-sequence" (or prior to the finish of preceding activities or other predecessor logic,) there are then at least two choices as to when to continue the calculation of duration. These have been traditionally noted as "retained logic" and "progress override" (R/P) The difference between these two options is illustrated in Figure #15. A third, perhaps improved, choice will be discussed in Chapter 3.

The choice of Retained Logic indicates that the logic is sound and that any work that happened to be performed out-of-sequence from that logic is an aberration and may not continue until all predecessor activities to this started activity have been finished. Thus, although one may "patch and paint" the portion of drywall already installed (and presumably done while waiting for additional deliveries of drywall material so that such activity may be finished,) further efforts to "patch and paint" must be deferred until all (or at least more) of the drywall has been installed. Note that the calculation assumes all work on the activity started "out-of-sequence" will stop immediately and that the resources (crew) assigned to this activity will be deployed elsewhere until the predecessor activities are 100% finished.

The choice of Progress Override indicates that an activity has been able to start prior to the finish of the predecessor logic and that it is therefore likely that the resources assigned (crew) may continue on such work – and subsequent work – without the need to finish that predecessor logic. An example is where a restraint between activities is based solely upon allocation of resources, such as where the logic sequences the work of a painter from one room to another, and a second painter is hired and begins work on Room #2 before Room #1 is finished. One would expect that the second painter could continue work on the day after the CPM update even though Room #1 is not yet painted. In fact, this restraint could then be ignored in all future updates – while the painting of Room #1 may be required before electrical and HVAC trim and possibly many other activities, the path to project completion will not run from painting Room #1 to painting Room #2.

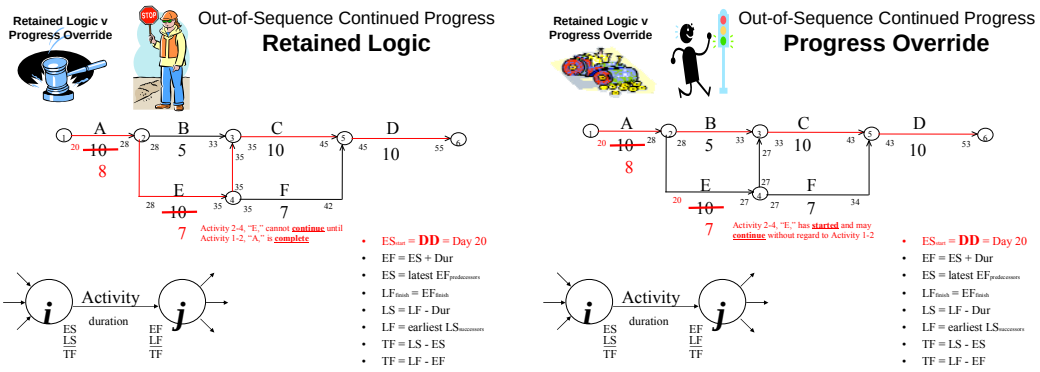


Figure #15 - Retained Logic versus Progress Override

The third duration code involves whether the project manager would plan to allow or to avoid allowing work of a specific activity to start if it appears that it cannot be finished in a continuous effort. Typically, a project manager would not plan for a painter to begin painting a room with the knowledge that the painter would need to stop, demobilize, remobilize after several days, and finish the effort of painting that one room. On the other hand, the project manager may have less concern with such "start and stop" performance for an entire building, planning for the painter to paint on any day that a room is available. (The painting subcontractor, on the other hand, would like a clear run once deploying resources to this project.) In fact, should some of the rooms of the building require substantial "furnishing" subsequent to painting, it may be absolutely necessary for the painter to paint those rooms when first available, notwithstanding that the painter may then be demobilized while other rooms are prepared for paint. The duration may thus be coded "interruptible" or "continuous" (I/C.) The duration code is thus going to be (at least) a three part code: P/C, R/P, I/C. Additional possibilities will be discussed in Chapter 3.

Reason/Why Codes

The Reason and Why codes to be associated with a restraint, and associated fields for the recording of a description and further "free form" textual information about that restraint, are the key to distinguishing whether the restraint is an artifact of planning or scheduling of the activities in a logic network. The primary choices for Reason are either "P" for "physical," or "R" for "resource." Generally, a physical Reason indicates the rather limited choices associated with planning (with infinite resources) while a resource Reason indicates the greater freedom of choice associated with the scheduling or fielding of limited resources amongst the many choices provided by the physical plan.

The Why code further describes the Reason code and answers the question of "why" the specific Reason has been chosen. Thus, the "why" of a designation of "physical" Reason may often be "gravity" – that the scope of work of the successor activity to the restraint physically sits upon or on top of the completed scope of work of the predecessor activity. Another "physical" Reason may be that the specification requires a contractor to complete A before starting B – the contractual dictate having the same authority over the contractor as the forces of nature.

The Why code associated with a Reason code of "resource" – both associated with the restraint between two activities – may mimic the resource codes and dictionary associated with those activities, or may be of a more "freeform" nature, simply noting the unit of labor, equipment, material or other limited resource.

The Reason/Why code combination can provide several enhancements over traditional CPM planning and scheduling. It is axiomatic amongst professional (planners and) schedulers that each activity ***must*** be preceded by the physical completion of another activity (other than the first activity of a project) and, in turn, ***must*** be physically completed before being succeeded by another activity (other than the last activity of a project.) Software performing the procedures of traditional CPM, either ADM, PERT or PDM, cannot distinguish between whether restraint to the predecessor (or successor) of an activity is based upon this physical rationale or is simply a statement that the project manager chooses to sequence one before the other. If the latter, the sequence may even be reversed and still be valid. Obviously, a professional scheduler should and usually will be doing this check manually, but this places the onus of checking each restraint upon both the initial scheduler and reviewer. Thus, as noted above, one of the first benefits and return upon the investment of the additional effort to record the Reason/Why will be an improved software diagnostic that may check that each activity is linked to a predecessor and successor by physical restraints.

The second benefit and return upon investment is the similar check upon "resource" restraints noted previously. As the project manager may wrestle over how to deploy limited resources by adopting one sequence and then another, a common error is to deploy the same resource twice. This is usually manifest by initially having a resource, such as a carpenter crew, move from activity A to B and then to C, then deciding to rather send from B to D while forgetting to delete the restraint from B to C. By labeling

the restraints as Reason “resource” and Why “carpenter crew,” software can be developed to catch this common error.

Moving beyond these minimal benefits, labeling of the Reason/Why for each restraint provides the manual or automated ability to run various “what if” scenario by selectively suppressing (or temporarily deleting) specified classes of restraints. For example, if the project manager, having built a CPM logic based upon limited formwork, desires to quantitatively determine how much time might be saved by deleting the restraints based upon such formwork limitations, software may be developed to schedule the logic while ignoring all such restraints. Similarly, if a contractor or owner desires to see what is the impact of a mandated sequence of construction upon project completion, this may be accomplish by suppression of the Reason/Why code of “P”/”SPEC” while running the scheduling algorithm. Looking at expediting the project from an even more aggressive approach, the project team may code some of the “P” restraints as being susceptible to being circumvented by the use of falsework (at a cost.) The project team could then quantitatively determine the savings in time based upon this investment.

Other Reason/Why codes, of a special nature, may include a “J” or “just-in-time” form of physical Reason, and an “L” or “leveled” form of resource Reason. These will be further discussed in Chapter 3.

Expanded Restraint or Lead/Lag Codes

To some extent, the inclusion in RDM of expanded restraint or lead/lag codes is merely the resurrection of "lost art." Several of the mainframe CPM programs of the 1960s and 1970s had a means to distinguish between lags measuring the passage of time from a reported start of an activity and lags measuring the performance of work (that is the original estimate of duration less the reported remaining duration) of an activity. For example, the MSCS software program (McAuto Schedule and Control System by McDonald Douglas) distinguished between "Start-to-Start" and "Begin-to-Begin" restraints. Similarly, MSCS could distinguish between "Finish-to-Finish" and "End-to-End" restraints.

Unfortunately, not even the sales or training departments of McAuto truly understood these distinctions. Thus, while this author was attending a training class at McAuto's headquarters in St. Louis, the distinction of "Start-to-Start" and "Begin-to-Begin" was explained as "two groups of software programmers developed competing code and management decided to not snub either group." After all, both styles of lag calculate the same results – unless, during an update an activity might start and then proceed faster than estimated, or start and then stall. As neither of these examples was in the materials provided to the sales and training departments, their conclusion that the two styles of lag were merely the result of a "compromise amongst the programmers" was probably logical if not correct. (Further "independent" review of the actual algorithms used did reveal the true distinction. This was then confirmed after such was forwarded to the programming staff.)

The loss of the ability to distinguish “passage of time” versus “progress of performance” styles of lag in the next generation of CPM software developed for the new microcomputers (such as the IBM PC) was thus probably a combination of this general misunderstanding of the need for such, the memory (RAM and disk storage) limitations of the early PCs and the recognition that the additional complexity and training to understand the choice between “Start-to-Start” and “Begin-to-Begin” might appear daunting to prospective purchasers of such software.

Several of the software offerings currently marketed do have remnants of the “lost art” of distinguishing between “passage of time” versus “progress of performance” styles of lag. OpenPlan by Welcome (recently purchased by Deltek) has the ability to choose “passage of time” by setting the lag in days from its reported start or “progress of performance” by setting the lag in terms of “percent complete.” Of course, if the project manager should desire to say “electrical rough-in may start when two days of installation of studs and drywall have been complete” rather than “electrical rough-in may start two days after installation of studs and drywall have started,” the project manager will be required to run a side calculation (2 days / total estimated duration) to determine the proper percentage to enter. OpenPlan also does not distinguish between the “Finish-to-Finish” and “End-to-End” types of lag.

Another way to distinguish between “passage of time” versus “progress of performance” is by the use of a “Finish-to-Start” restraint coupled with the use of

"negative lag." This, in fact, is the preferred means to model overlapping activities by Microsoft's Project software. While the concept of a "negative lag" may appear to be mathematically challenged and of little value other than "to move bars on the bar chart," the term actually converts into a form of "Start-to-Start" lag based upon release of the successor activity when the remaining duration of the predecessor is reduced to the absolute of the "negative lag" value. Thus, once again, a "priest" of CPM may use the tools currently available to "fudge" the desired results, but the common user may not necessarily have the knowledge to do so. Figure #16 illustrates how a finish-to-start restraint with negative lag is equivalent to a Start-to-Start restraint. However, one should also note that should activity A stall when 71% complete, the use of the FS-3 lag still creates a "hidden open end" as discussed previously. The "finish-to-start with negative lag" restraint is, in fact, a start-to-start restraint. The difference is that the SS+7 restraint measures days since the reported start, while the FS-3 restraint measures the declining remaining duration, releasing activity B when this reaches 3 working days. Since this measurement is of performance of activity A, rather than merely the passage of time since its reported start, this target could just as easily be stated as permitting the start of B when only 30% of A remains to be performed, or when 70% of A has been performed.

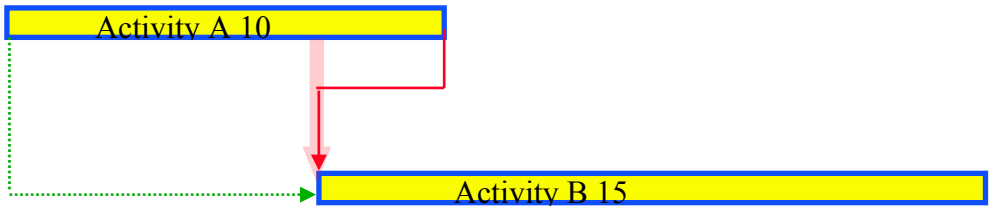


Figure #16 – SS+7 v FS-3 lead/lag codes

RDM envisions a full return of the distinction between "passage of time" versus "progress of performance" styles of lag. Thus, the selection of restraint is expanded beyond the current "Finish-to-Start," "Start-to-Start," "Finish-to-Finish" and "Start-to-Finish" choices. A full selection of the types of restraints between activities that are supported by RDM may include:

- Finish-to-Start
- Start-to-Start
- Finish-to-Finish
- Start-to-Finish
- Contiguous
- Concurrent
- Coordinated
- Progressed-to-Start
- Finish-to-Remainder

The Finish-to-Start restraint, the sole restraint supported by the original "ADM" CPM algorithm, is not as subject to multiple interpretations as the newer "PDM" (and now "RDM") types of restraints. A duration (lag) assigned to a RDM "FS" restraint will be a continuous clocked duration (p/C,) (C/i,) with its own assigned calendar and choice of out-of-sequence logic algorithm (R/P.) Note that if the retained logic option is chosen, and the preceding activity is reported finished out-of-sequence, the clock on the restraint will not start until all predecessors to that activity are reported finished. For that reason, the default setting for the "FS" restraint should be to (r/P) Progress Override.

The RDM "SS" Start-to-Start restraint, or perhaps Start-to-Start-by-Clock restraint, is equivalent to the Start-to-Start restraint used by most current CPM software products. It may be distinguished from such by the proper treatment of this restraint as carrying distinct logic between two events, namely the start of one activity to the start of another activity. Thus, the duration of this restraint is provided equal treatment as the duration of

an activity, requiring its own calendar and duration codes. By definition, if a duration is assigned to an SS restraint, it will be a continuous clocked duration. However, whether the clock is to start with the early or actual start of the preceding activity must be defined by the user. The set of choices is thus (p/C), (R/P), (i/C.)

The RDM "PS" Progressed-to-Start restraint, or perhaps Start-to-Start-by-Performance restraint, is more firmly linked to its predecessor activity. The start of this restraint is an event, not at the start or finish of the predecessor activity, but rather an event embedded within an activity and representing completion of a definable portion of that activity. Because it is intimately associated with the preceding activity, the PS restraint duration (lag) does not have its own calendar, but rather must use the calendar associated with the preceding activity. Likewise, the duration does not have independent duration codes, but rather will be defined as measuring progress by performance (P/c) and otherwise set the same as the preceding activity (R/P,) (I/C.)

Because the "PS" restraint is intimately associated with its preceding activity, it may be expressed in several manners, similar to verbal notation used to express the relationship. A restraint may be expressed as "start B when 7 days of 10 day duration activity A are complete." Or, "start B when 70% of activity A is complete." Or, "start B when remaining duration of A has been reduced to 3 days," or "start B when only 30% of A remains to be performed." If the quantity of work associated with activity A is known, one may even say, "start B when 35 of 50 units of A have been performed" or "start B when only 15 of 50 units of activity A remain to be performed." Coding or data entry of

the specified lag duration may be “7,” “70P,” 70%,” “3R,” “30PR,” 30%R,” “35/50” or “15/50R” for the preceding example, mimicking the verbal instruction of the person responsible for overseeing the work performed. .

The RDM “FF” Finish-to-Finish restraint, or perhaps Finish-to-Finish-by-Clock restraint, is equivalent to the Finish-to-Finish restraint used by most current CPM software products. This is illustrated in Figure #17. It may be distinguished from such by the proper treatment of this restraint as carrying distinct logic between two events, namely the finish of one activity to the finish of another activity. A duration assigned to a FF restraint will be a continuous clocked duration (p/C,) (C/i,) with its own assigned calendar and choice of out-of-sequence logic algorithm (R/P.) Note that if the retained logic option is chosen and the successor activity (say “C”) is finished prior to the finish of the predecessor activity (say “A”) plus lag duration, the successor restraint to the successor activity will not be able to start until the clocked time units of the lag duration have passed.

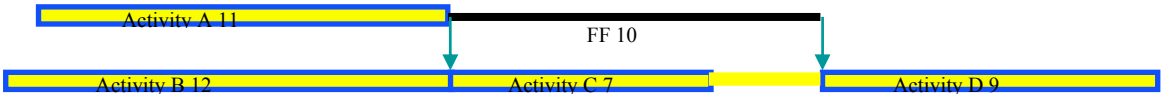


Figure #17 – C starts after A and B complete, D starts after C and 7 days after A complete

The “FR” Finish-to-Remainder restraint, or perhaps Finish-to-Finish-by-Performance restraint, may be more difficult to envision than its cousin, the “PS” Progressed-to-Start restraint, but careful consideration of the nomenclature should resolve such concerns. The main difference of the FR restraint is that it is intimately associated with the

successor activity rather than the predecessor activity to the restraint. The issue raised is what portion of C may not continue to completion until activity A is 100% complete. Assuming C to be of 10 days duration, a FR7 restraint from A would indicate that 3 days of C may be performed, perhaps starting after the completion of B, but that the last 7 days may not begin until A is complete. Thus, the calendar for the lag duration of this FR restraint must be the same as that of activity C, as will the lag duration codes for out-of-sequence (R/P) and interuptability (I/C.) The duration will, of course, be based upon the restrained future performance of activity C and not on a clocked number of time units to the projected completion of activity C (P/c.) If, notwithstanding the stated restraint, C should be 50% performed (or any percent greater than 30%) when A is finished, C would then be permitted to continue without interruption.

The FR restraint may be expressed as "finish last 7 time units (days) of C when A is finished," "finish last 70% of C when A is finished," "finish beyond the first 3 days of progress of C when A is finished," finish beyond the first 30% of progress of C when A is finished," "finish last 35 of 50 units of C when A is finished" or "finish beyond the first 15 of 50 units of progress of C when A is finished." These expressions of duration may be coded as "7," "70P," "70%," "3P," "30PP," "30%P," "35/50," or "15/50P."

The "SF" Start-to-Finish restraint, or perhaps Start-to-Finish-by-Clock restraint, is equivalent to the Finish-to-Start restraint used by most current CPM software products. The duration is measured by clocked units of time (p/C) and based upon the separate calendar of the restraint. By definition, such duration time is continuous (i/C.) If the

start of the preceding activity starts out-of-sequence, such duration may be defined to proceed upon the original retained logic or to "progress" override such logic and begin the clock upon the reported start of the preceding activity (P/R.) Notice that the definition of the SF restraint is flawed as it does not also address the issue of an out-of-sequence finish of the successor activity as has been part of the FF restraint.

The reason for that flaw is also why it is difficult to provide a proper "performance" based "SF" restraint – that is the need for two separate lags for such a restraint. Whether such a restraint is called (and coded) as "Progressed-to-Remainder" ("PR") or any other option ("PP," "RP" or "RR,") there exists the need for a lag measuring the days of progress of A performed before beginning the lag measuring the last remaining days of C to be performed. Although a double lag convention may be possible, the difficulties of implementing such in an easily understood consumer software product suggest staying with only one lag per restraint. This is illustrated in Figure #18.

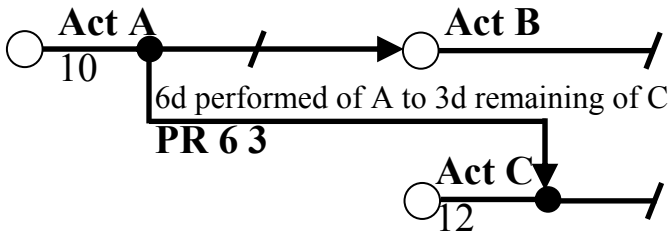


Figure #18 – The need for two lags in an RDM "SF" restraint

Insofar as scheduling professionals are generally distrustful of the use of a "SF" restraint, the need for a "PR" restraint may also be limited. A work-around, illustrated in Figure #19, is to utilize the ability of RDM to support true independent events. Rather than use a "PR" restraint requiring two lags, a scheduler may use a "PS" restraint with one lag

from an activity to an independent event and then a “FR” restraint with a second lag from that event to a successor activity.

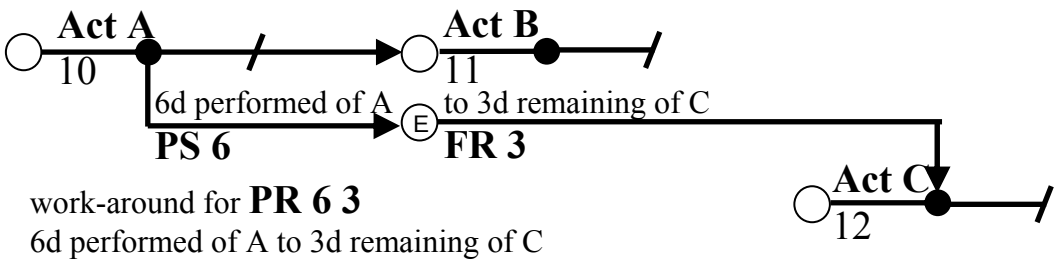


Figure #19 – Work-around for “PR” lag by means of independent Event “E”

The Contiguous, Concurrent and Coordinated restraint types are discussed in Chapter 3.

Relationship Codes

While the event, duration, reason/why and expanded lead/lag codes all involve additional recording of the information thought (but perhaps not expressed) by project managers and their team members in crafting a CPM (or any project plan or schedule,) the relationship code is one that is generated during the calculation (by hand or computer) of a CPM. The purpose of the relationship code is to ascertain or calculate the relationships between the predecessor and successor activities (and/or events) of a restraint. Any similarity or difference between the predecessor and successor may be noted and reported. The quantum or quality of differences may also be noted and reported. Actions, either manually or via the computer software implementation, may be performed based upon the noted similarities and differences.

A simple use for a relationship code may be to highlight whenever a user defined activity code, such as code for which subcontractor is performing work, changes. Since the prime contractor is responsible for coordination of (or between) subcontractors, but not responsible for the internal coordination of a subcontractor's scope of work, these are the "handoff's" which must be carefully watched. This is perhaps similar to a game of football – it is rare that a turnover will occur while running with the ball – it is more common to have a turnover while passing the ball. Since the typical bar-chart display of a schedule does not provide an easy view of the relationships with other activities, it is all the more important that those restraints revealing such "handoffs" be highlighted.

Handoffs between entities that have previously encountered problems, such as perhaps that between the mechanical and electrical subcontractors, may be highlighted as calling for a 1-day coordination period (on a 5-day/week calendar.) A more advanced use may be to assure that there exists at least a 2-day lag (on a 5-day/week calendar) when a crew moves from one location on a jobsite to another in order to account for the necessary mini-demobilization and remobilization, tearing down and rebuilding of scaffolding, etc.

Another use is as a means to root out possibly miscoded "P" physical restraints may be where relationships indicate the same resource usage but differing locations or structures. Similarly, any serious change in location for a "physical" type of restraint should be flagged for further review. In fact, this author does not believe it is today possible to dream of all the uses that a new generation of schedulers may make of the implementation of such a relationship coding algorithm.

Chapter 3 – Additional Enhancements and Extensions

The introduction of several new code fields relating to events, duration, the reason/why a restraint has been used, the type of restraint between activities (and/or events) and the relationship between the activities (and/or events) preceding and succeeding a restraint, creates the opportunity for further expansion of the ability of software to enhance a CPM. Several of these enhancements could, conceivably, be incorporated into existing PDM software implementations.

Modified Progress Override

The first "something new under the sun" enhancement to be discussed is based upon a reappraisal of the role of the Retained Logic v Progress Override options for activities and Actual Start v Early Start options for Start-to-Start restraints when preceding activities are started out-of-sequence. The basis of the Retained Logic and Early Start options is that should work on an activity be started out-of-sequence, further work on that activity (or time clocked since the reported start of that activity) should be suspended until all predecessor work has been finished.

The basis of the Progress Override and Actual Start options is that should work on an activity be started out-of-sequence, further work on that activity (or time clocked since the reported start of that activity) should be thus be permitted to continue.

In the real world, it will be a rare occurrence that the crew that was working on an activity on Thursday afternoon, the day before the DataDate of an update, will not be able to continue work on Friday morning. There may be some question as to whether the crew can finish the started activity prior to (others) finishing the predecessor scope, and there may be some question as to whether subsequent activities requiring 100% completion of this current activity can start, but the general rule is to schedule work allowing this crew to continue its work-in-progress. However, if this and then subsequent activities are presumed to be able to finish without completion of the preceding activities, the current options supported by software do not check to see if such activities are ever finished. Thus, further progress on such activities is ignored, the late finish date for such activities defaults to project completion, and the Late Finish Float (equal to LF - EF) is calculated as an unreasonably large number. (This last factoid is a little understood means to locate such "orphaned" activities and a real use for Late Total Float over that of "Most Critical Total Float.")

Suggested by this author in 2003 (at the PMI College of Scheduling convention in Montreal) is a third option, called the "Modified Progress Override" option. This option permits work started out-of-sequence to continue out-of-sequence even to a point of a reported actual finish (and thus zero remaining duration,) but does not permit release of unstarted successor restraints until all predecessor logic is finished (or technically, the calculated early finish is less than the DataDate of the schedule update.) This is illustrated in Figure #20.

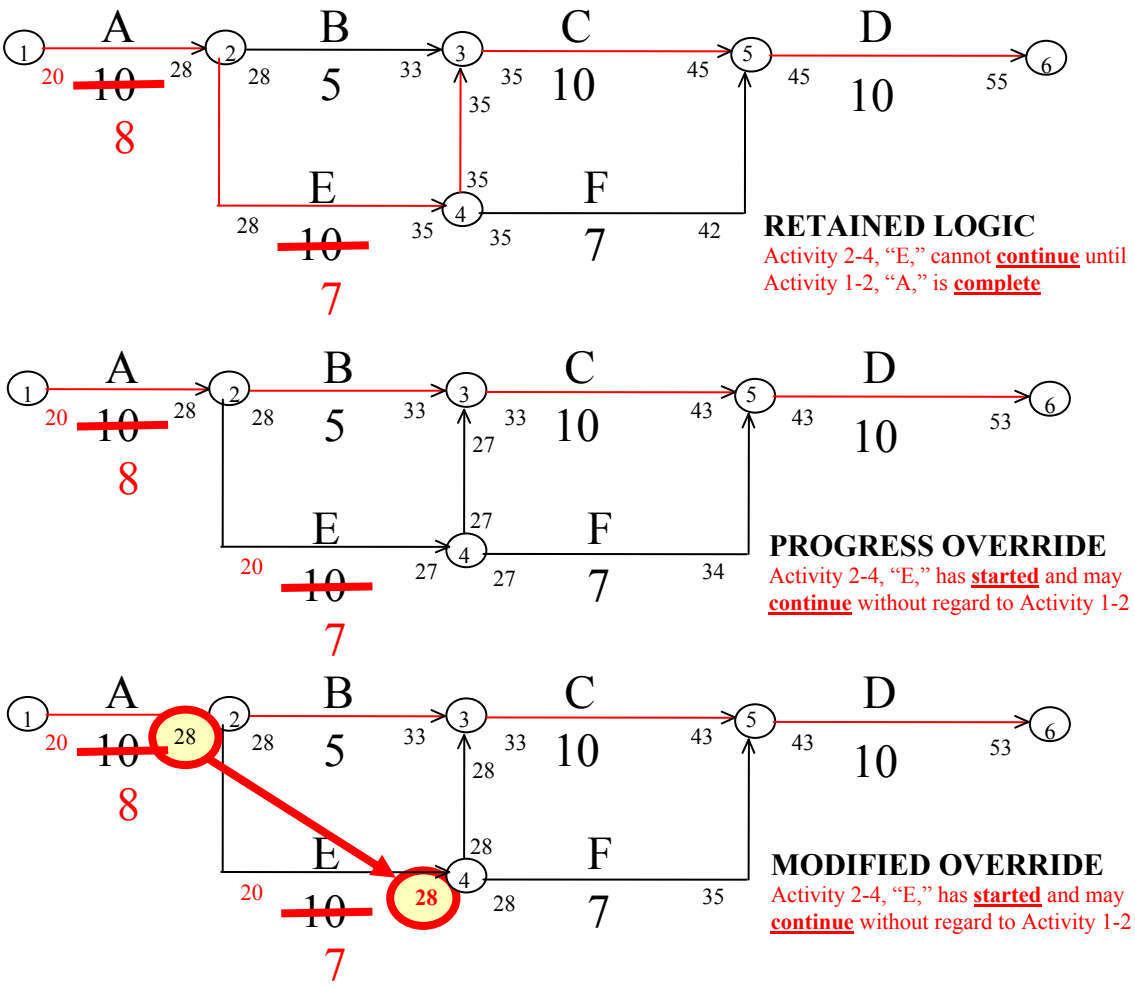


Figure #20 – Retained Logic v Progress Override v Modified Progress Override

The addition of this option modifies the prior material to the extent that wherever the choice was presented for Retained Logic v Progress Override (P/R,) there now exists a third option (M/R/P.) The practical effect of choice of this option upon an activity duration will be to convert an incoming "FS" lag to a "FF" lag. The choice of this option upon a restraint duration (lag) will be to "run the clock" from the reported finish of the preceding activity, but not release the successor activity until all preceding logic is reported finished.

Just-in-Time Restraint and Attributes

The inclusion of the Reason/Why code supports choices of “P” for “physical” and “R” for “resource.” Additional, specialized code choices may also be considered and included. The first specialized choice is a variant of the “P” code, that being the “J” or “just-in-time” code choice.

The “J” or “just-in-time” code is truly “something new under the sun” and may be used to calculate an entirely new class of attributes for activities (events and restraints) of a CPM relating to the intelligent use of float. This code indicates that the predecessor activity string should be subordinate to other strings of activities leading to the successor activity of this restraint. An example may be a string of procurement activities for a pump that is to be placed upon a poured foundation. While the use of float leading up to the installation of the foundation and subsequent rigging of the pump may be permissible in a specific logic network, jobsite economics (if not the legal theory of the proper use of float) would frown upon use of such float by a leisurely pace of the procurement string.

The question is thus, “what is the latest date that the pump should be delivered to not disrupt jobsite progress?” Traditionally this may be calculated by determining the free float of the last activity in the pump procurement string (typically “Delivery,”) and manually adding that number of work days (rather than calendar days) to the early finish date of that activity. Similarly, to determine the latest “just-in-time” start and finish dates for each activity in the procurement chain, the free float attribute of the last activity in the string must be manually added to each start and finish date of such activities. Moreover,

if the “delivery” activity has more than one successor, such as “delivery” leading to both the rigging of such equipment and on-site testing of such equipment, it is likely that the “delivery” activity will have a free float calculated as zero, thus further exacerbating the ability to answer the desired question, “when is the pump needed on the site?”

Free float is, therefore, deceptive because it shows a zero value for the parallel path with the lowest total float and also for any series of initial activities whose early finishes are not dependent to another chain. In some cases, it will equal the total float value where re-entering a critical path string of activities. It may be less than total float, but it will never be more. Many programs still print out free float even though it is virtually never used. Other times, the program may continue to generate free float but the printout is blanked off by request.

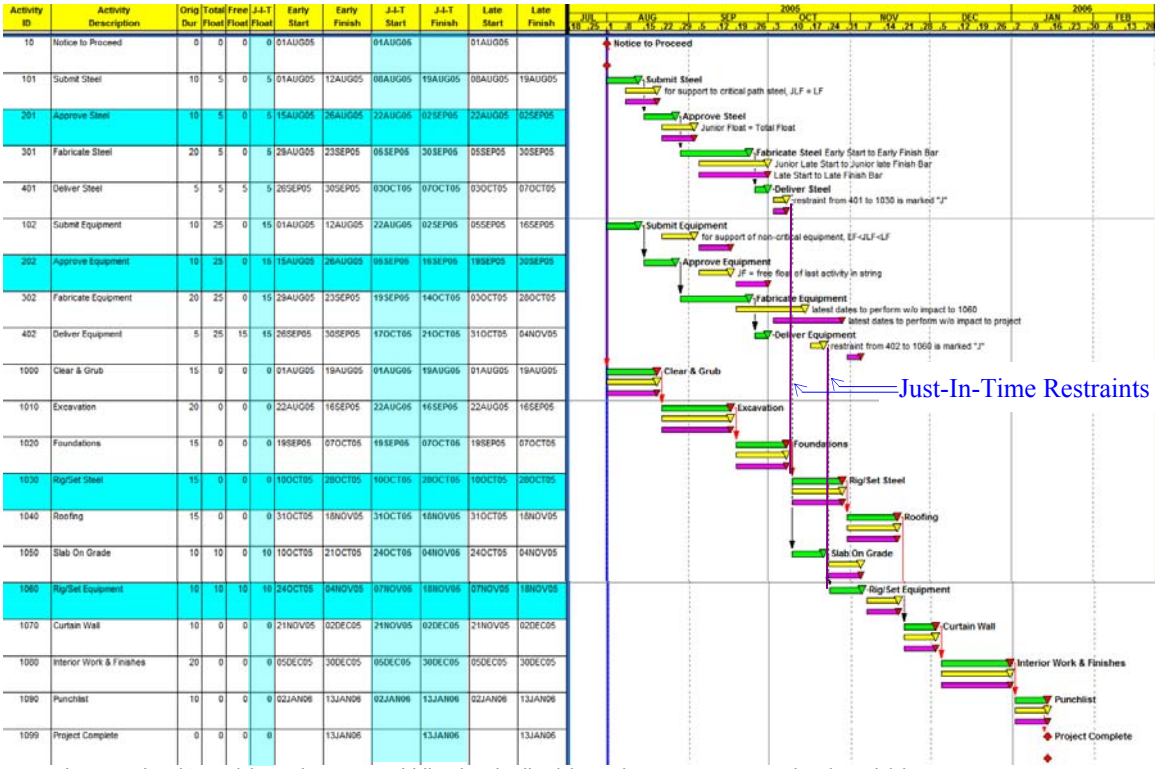
When a column listing Free Float is included in a report, it is usually to note the amount of slippage permitted for delivery of fabricated materials that will not delay the early start of a subsequent erection or installation activity. But as noted above, the calculated attribute is misleading as project personnel would desire similar information relating to the submittal, approval and fabrication activities preceding the delivery activity. Instead, each of these preceding activities has a calculated Free Float of zero because their successors each have but one predecessor. Free Float can only exist where a successor activity has more than one predecessor – it is the consequence of the merger of multiple paths of logic.

With the introduction of the “just-in-time” restraint code come the ability to support the new attributes of “just-in-time-late-finish,” “just-in-time-late-start” and “just-in-time-total-float.” These may be expressed as “JLF,” “JLS” and “JTF.” The “JLS” attribute describes, for each activity, the latest date that this “support activity” must be started in order not to delay the early start of the activity following the next “J” restraint. Similarly,

the “JLF” attribute describes the latest date that the activity must be finished in order not to delay the early start of the activity following the next “J” restraint. The “JTF” attribute measures the number of days (in the calendar of the activity) between the early start and “just-in-time” start, or early finish and “just-in-time” finish, or more critical of the two calculations. Note that if the “supported” activity has positive total float, the JTF will be greater than the TF for all predecessors prior to the “J” restraint. If the “supported” activity is critical, the JTF will equal the TF for these “supporting” activities.

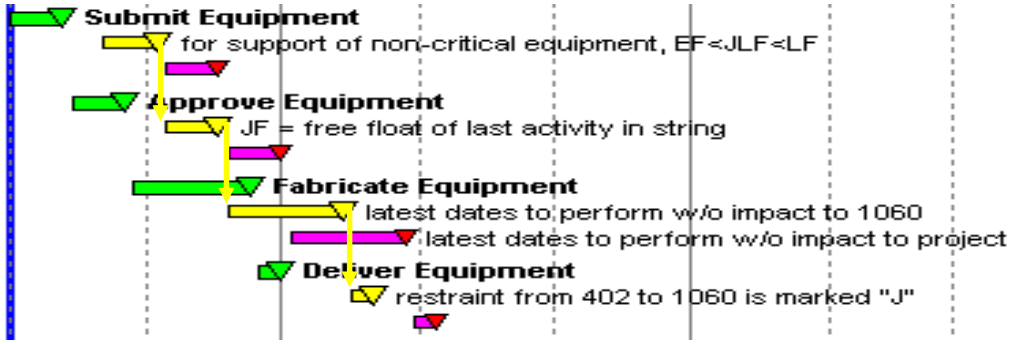
The relative positioning of “early dates,” “just-in-time dates,” and “late dates” as are generally depicted, and a detail thereof, are illustrated in Figures #21 and #22. The sequence of “Clear & Grub,” “Excavation,” “Foundations,” “Rig/Set Steel,” “Roofing,” “Curtain Walls,” “Interior Works and Finishes,” and “Punchlist” are all upon the critical path (having a total float of zero – see column #4.) The three bars for “early dates,” “just-in-time dates” and “late dates” for these activities overlap. Activities “Slab On Grade” and “Rig/Set Equipment” have 10 days total float, and the bars representing “just-in-time dates” and “late dates” overlap and are both 10 days later than that for the “early dates.” Activities for the procurement, fabrication and delivery of steel have 5 days total float, and because these are required for the critical activity of “Rig/Set Steel,” they also have 5 days just-in-time float. However, activities for the procurement, fabrication and delivery of equipment are required for “Rig/Set Equipment,” which is not a critical activity, these activities have 25 days total float but only 15 days just-in-time float. Notice also that the last activity in each of these procurement strings, the “delivery”

activity, has a free float equal to the just-in-time float. However, the free float for those activities leading to these "delivery" activities are calculated as zero.



Legend – Top Line (Green) is Early Dates, Middle Line (Yellow) is Junior Dates, Bottom Line (Purple) is Late Dates

Figure #21 – Use of "J" restraint and illustration of "JLS," "JLF" and "JTF" attributes



Legend – Top Line (Green) is Early Dates, Middle Line (Yellow) is Junior Dates, Bottom Line (Purple) is Late Dates

Figure #22 – Detail of use of "J" restraint and "JLS," "JLF" and "JTF" attributes

Calculated "L" Leveling Reason Code

The second specialized choice is a variant of the "R" code, that being the "L" or "leveled" code. This code is not entered by user choice, rather it is a report of a restraint being added by the leveling routine (or series of algorithms) of the CPM software. The addition of this code field is part of an effort to significantly enhance the functionality of a leveling routine (or collection of routines.) The first step in improvement of a leveling routine is to "unlevel" the existing plan. Typically, as a logic network is prepared, the project team (or team leader, project manager or superintendent) will be applying some degree of leveling of limited resources by means of use of "R" or "resource" restraints. Thus, one crew may be sent from foundation "A" to foundation "B" rather than simply noting that both foundation "A" and "B" may start after a common predecessor – and may be performed concurrently or in any order depending upon the amount of available float. If the project manager has allocated work amongst one or two crews, it is likely that there will be no need for computer aided leveling.

However, if the project manager has allocated work amongst eight or more crews, it is possible that a better allocation of such scope amongst the available resources can be made. However, if as typically done, the leveling routine starts with the preconceived allocations of the project manager are not as likely to be improved by a leveling algorithm. Thus, as noted above, the first step towards an intelligent leveling routine is to delete (or temporarily suppress) those logic restraints which are solely carrying "R" or "resource" logic for the resource being leveled. This functionality should be easily added once logic restraints are Reason/Why coded.

One additional step may be the need to check with the Scheduler whether specific restraints to be suppressed carry both physical and resource logic. This possible issue should be reduced in part by a diagnostic confirming that each activity has a physical predecessor (and successor) and by a check of the relationship codes to determine whether the resource is moving between or remaining within code values (such as location.) Once again, a diagnostic can highlight those restraints that are to suppressed and which may be also carrying "P" or "physical" logic.

The proposed "L" or "leveling" code value for Reason is to be generated during the next step of the leveling routine. When the leveling routine determines that there are more activities requiring a specific resource than are available, the algorithm chooses which activities are to proceed and which must be deferred. There is no need for the Scheduler or project manager to determine at this time which crew will work on which of these activities; if such a determination is important, there should be separate resource codes for each crew and each activity should be coded to require only the crew desired.

However, where an activity must be deferred due to limited resources, there will be a point in time when another activity releases that resource and it becomes available for the deferred activity. A full implementation of RDM provides that the enhanced leveling routine will then generate an "L" or "leveling" restraint between the activity releasing the resource and the activity acquiring the resource. This information will then be available for the project team and project manager's review. If the computer generated "leveled"

allocation of the resource (crew) is deemed unacceptable, the Scheduler may at that time override with the additional logic which had not previously been communicated by that project manager.

Contiguous Lead/Lag Code

The RDM system may also include a specialized variant of the "FS" restraint code, to be designated as a "CT" or Contiguous lead/lag code. This variant draws the finish of the predecessor activity up to the start of the successor activity. If the predecessor activity is set for a contiguous (or continuous) duration, and is not interruptible, the CT code will likewise draw up the start of the activity. The concept is graphically depicted in Figure #23. Obviously, activity B of the graphic would need to be restrained by other logic, thus drawing up Activity A to its start. An example may be the advance notice to an inspector prior to anticipated performance of an activity B.

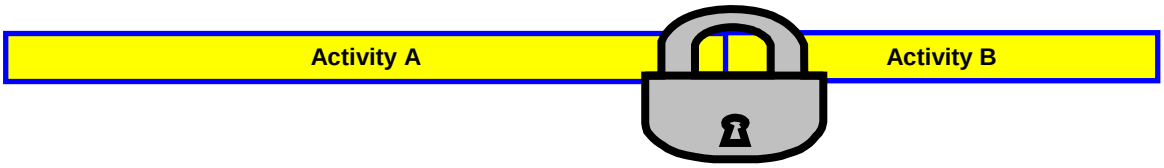


Figure #23 – Illustration of the Contiguous Lead/Lag Code

So far, this could currently be accomplished by placing a Zero Free Float (ZFF) constraint upon activity A. However, the comparison ends here. By stipulating a lag (duration between activities) with the CT lead/lag code, one may create the equivalent of a user controlled free float constraint. Thus, if it were desired to send the required 14 day "notice to inspector" activity a few days in advance (so as to not lose the opportunity if work in progress is completed faster than expected,) the use may connect activity A to activity B via a "CT 2" lead/lag code.

An additional case in which the ZFF code would not be usable is where activity A is followed by both activity B and an activity C which may, in fact, be driven by A. For

example, the start of work by the inspector (which is to be deferred until needed by activity B) may kick off an entire string of unrelated work that was to be deferred until the inspector arrived. The general point may be that the “CT” lead/lag code, like the “J” reason/why code, acts upon the restraint between two activities rather than upon a specified activity and all restraints there flowing from or to.

Concurrent Lead/Lag Code

Another specialty lead/lag code may be designated as a “CC” or Concurrent lead/lag code. In many situations, two activities appear to be intertwined. One example may be erecting a MSE wall and backfilling during placement. Another example may be pouring a concrete slab with embedded electrical conduit. In each case, the proper logical means to depict the combination is by only one activity. However, the needs for separate rollups by subcontractors and the general desire for such a splitting of this ONE activity indicate a need for this type of restraint code.

Providing such a code is therefore a placebo, allowing practitioners to say what they desire to say without compromising the mathematical logic. The calculation logic is, to some extent, based upon two-way communication of data. The duration of the predecessor activity to the “CC” restraint is the larger of that of the predecessor or successor, since the two are to be performed “concurrently.” Neither activity may start (and both are calculated to start) or continue until all predecessors of both are “satisfied.” However, each may have separate successors which may be released when actual dates are reported for the start or finish of the separate activities. Some means of internal checking will be required to avoid an illogical use of the code such as may occur if the “concurrent” activities have differing contiguous and interruptible duration code settings.

Coordinated Lead/Lag Code

The MSCS (McAuto Schedule Control System) mainframe CPM software was the gold standard in the 1960s, 1970s and early 1980s prior to the ascendancy of the personal computer. As noted, this program did distinguish between a "start-to-start" and "begin-to-begin" restraint and between a "finish-to-finish" and "end-to-end" restraint. As discussed, the loss of these distinctions was probably a combination of the limitations of the early personal computer hardware (CPU, RAM and disk memory) and the difficulty of explaining the fine distinctions between the logic types in a software product intended, not for scheduling specialists or gurus, but rather for non-specialist contractors and perhaps non-specialist engineers and inspectors.

Another one of the "lost" restraints was the "Z" restraint which was a combination of "S" and "F" restraints with a common lag, and (when used properly) for activities with the same duration. (Use of the code with activities of differing duration was accepted by the software, but may cause unforeseen results.) This type of restraint was found to be very useful among many users such as highway contractors where there may be a string of activities such as grading, stoning, BCBC paving, BBC paving and BWC paving each separated by 500 linear feet or one day. While the logic is easily duplicated by using matching "SS" and "FF" codes (each having the 1 day lag,) the advantage of the "Z" code was the lack of need to carefully check that each "SS" code was in fact matched with a "FF" code.

In preparing a methodology that truly meets the human needs of potential users as well as reintroducing and enforcing mathematical and technical rigor, it is suggested that this popular code be reinstated as part of the RDM methodology. Whether such should be coded “ZZ” or “DD” (for duplicate) or some other designation may be left for that vendor who is first-to-market with a workable RDM product.

In that activities that would be subject to the use of the “DD” code tend to be strings which may be treated in a production line manner, additional study may be called for to determine further use in manufacturing settings.

Chapter 4 – Where to Next

The development of the CPM and PERT methodologies was a true exercise of engineering. The developers understood the limitations of the mathematics and the limitations of the tools available at that time and place. The existence of merge bias in the logic models resulted in the need to accept a "good" answer, plus a empirically chosen factor of safety, rather than a "correct" answer. The limitations of limited CPU power and only linear access memory also resulted in a procedure that required a large effort in converting the logic model describing the project to a format that could be calculated by the computers of the day. Even more damaging, the severe lack of memory resulted in a procedure that deleted recording the crucial step between planning (where A must be done before B) and scheduling (where A is selected to be done before B.) But the combination of imperfect mathematics and imperfect tools for calculation were overcome to provide a workable solution, and one that greatly increased coordination and productivity on infrastructure construction and other projects.

Subsequent development has been less of an exercise in engineering and more of an exercise in commercialization. The greater power of the computers made available in and since the 1960s was not truly harnessed to obtain a better, or more correct answer. Instead, that power (and the power of ever improving computers to this day) was directed more to reduce the effort required to encode the logic model and to publish the calculated results to all stakeholders of the project. In addition, higher computer capabilities have been used to reduce the level of effort in acquiring accurate input to the logic model or in interpretation of the calculated results. As a result, users may dispense with the logic

model and merely print the preconceived “results” desired by that user. Since these “override” features of commercial software are not generally known, engineers and other reviewers of software files and printouts may erroneously believe such to be calculated, rather than the opinion of the provider of such.

The newly suggested protocol of Relationship Diagramming Method (RDM) is proffered to assist in resolving these issues. By segregating the logic used for planning from that used for scheduling, RDM allows for validation of the CPM as being first based upon a plan. Further coding of the “why” for each restraint provides a first step towards validation of the scheduling aspect of the CPM. Provision of a greater number of options in describing what is meant by users in defining activity durations and the restraints between activities may add complexity; but if handled properly the added complexity will be outweighed by simplifying the user’s need to convert what is meant to a more limited number of options. Including the ability to intelligently “read” user defined coding (currently provided solely for the purpose of tabular and graphic reporting) allows for additional provision of validation and computer assisted scheduling.

The amount of additional information that must be elicited from members of the project team in preparation of a logic model is truly minimal. Most (if not all) of the questions to be asked for implementation of RDM are the same as for a properly prepared CPM using either the ADM, PERT or PDM formats. The amount of collected information to be recorded is slightly greater – the scheduler must record (and later key enter to the computer) not only the existence of a restraint but the reason for the restraint. The

scheduler must also record precisely what is provided as a duration of an activity (calendar, measurement by performance or clock, interruptible or continuous, out-of-sequence performance subject to retained logic or progress override or modified progress override) or the duration between activities (with similar attributes independent of or linked to the predecessor or successor activity.)

The quantity and validity of calculated data from the RDM model should be well worth the additional effort of data recordation and entry. The calculation of additional attributes of Just-in-Time-Late-Start, Just-in-Time-Late- Finish and Just-in-Time-Total-Float provide useful information which is not quite provided by the traditional attribute of Free Float. Returning to Figure #21 (page 57,) the designation of the restraints between activities 401 to 1030, and 402 to 1060, as "predecessor desired to be performed Just-in-Time" has provided all project personnel with additional and useful information.

Compare the usefulness of the calculated Free Float attribute (column 5) which merely provides that the "Delivery" activity may be deferred by the printed number of days beyond the printed early start date, to the information provided in columns 6, 9 and 10 (JTF, JLS, JLF.) For the contractor to highlight that late approval of submittals for these procurement items will have a disruptive impact to the project (as to be compared to causing a delay to the project,) one must note the Free Float of the Delivery activity, and add that number of days (using the proper calendar adjustments for weekends and holidays) to the early start of "Approval" activities 201 and 202 respectively. While delay in review of the steel submittal beyond 22AUG05, or equipment submittal beyond

19SEP05, will cause delay to the project, note also that delay in review of the equipment submittal will cause disruption (and additional cost) to the contractor's proposed work flow.

For an example of how RDM may assist a preparer or reviewer to verify the validity of a CPM logic network (and the schedule calculated therefrom,) review Figures #24 through #35. Figure #24 illustrates the thought pattern typical of many project professionals providing information for preparation of a bar-chart or CPM. Several craft crews are sequenced to perform a specified scope of work. The additional logic required to provide a proper CPM is shown in light blue. Figure #25 illustrates the bar-chart desired by such an individual.

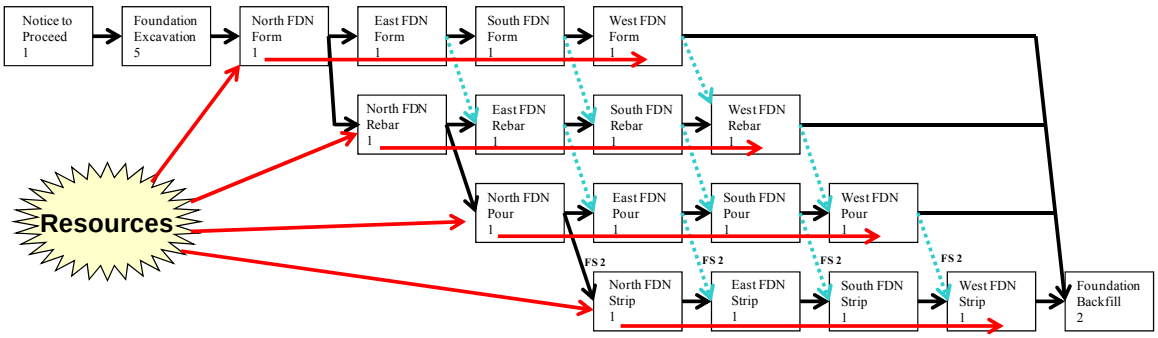


Figure #24 – Logic as Provided by Superintendent Deploying Work Crews

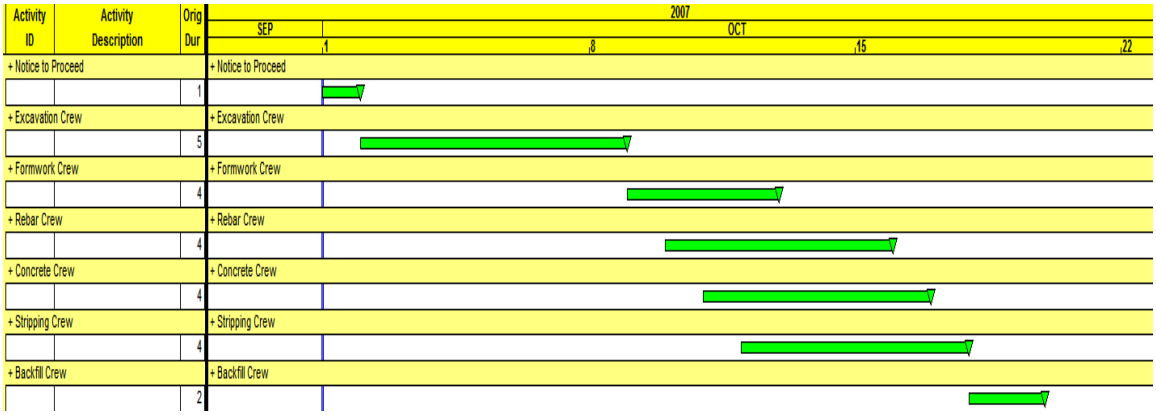
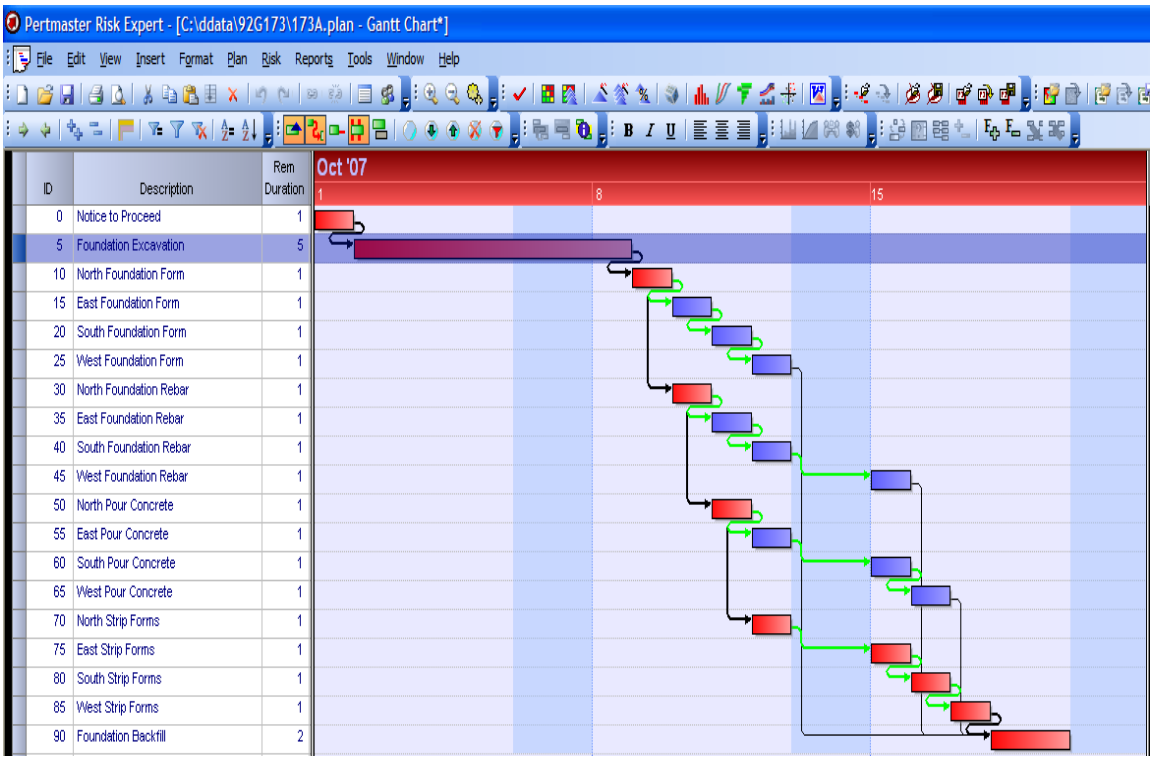


Figure #25 – Bar-chart as Desired by Superintendent Deploying Work Crews

The detail of such a bar-chart, showing how this may be keyed to popular CPM software, is provided in Figure #26. Note how the detail in this example accurately portrays the timing of all component activities, thus concrete is always shown (for any specific location) to be poured only after the forms and rebar for that foundation have been placed. The logic to and from each activity is also shown in this standard graphic. Note that the logic shown by the light blue arrows in Figure #23 is not included here.

Figure #26 – Detail of Logic Supporting Bar-chart Desired by Superintendent



Legend – Black Arrow = Physical Restraint, Green Arrow = Resource Restraint, Red Bar = Critical Activity

Figure #27 – Detail of Logic Supporting Bar-chart with Added RDM Data

In Figure #27, this information is copied to software that records RDM data, (Primavera Pertmaster version 8.2,) and restraints are designated as Physical (Link Category Type #1) as black arrows between activities or Resource (Link Category Type #3) as green arrows. (Red bars in the graphic have been calculated as critical activities, blue bars are non-critical activities.) This additional information may be entered manually, or may be calculated by an automated means by setting the link between two activities as "Resource Based" whenever the activity before and after share the same craft or crew but are to be performed in differing locations.

```
Primavera Scheduling and Leveling Calculations -- Scheduling Report Page: 1
 This Primavera software is registered to EnProMaC.
 Start of schedule for project 173A.
 Serial number...16660017
 User name FRED    .

 Open end listing -- Scheduling Report Page: 2
 ----------------
 Activity          0  has no predecessors
                                       Activity         90 has no successors

 Scheduling Statistics for Project 173A:
 Schedule calculation mode - Retained logic
 Schedule calculation mode - Contiguous activities
 Float calculation mode    - Use finish dates
 SS relationships          - Use early start of predecessor

       Number of activities.................       19
       Number of activities in longest path..      10
       Started activities...................        0
       Completed activities.................        0
       Number of relationships..............       21
       Percent complete.....................      0.0

       Data date............................  01OCT07
       Start date...........................  01OCT07
       Imposed finish date..................
       Latest calculated early finish.......  19OCT07
```

Figure #28 – Primavera P3 Schedule Checker and Open End Diagnostic

Returning to our question of "how RDM may assist a preparer or reviewer to verify the validity of a CPM logic network," we may now compare the standard schedule checker diagnostic, and its output, as provided in Figures #28, #29 and #30, to one that harnesses the additional features of RDM. A proper network usually will have only two open ends,

at the start and finish of the project. Figure #28 provides the standard diagnostic which indicates that the logic network in the prior figures indeed have only two open ends. Figure #29 is a screen shot of the Pertmaster schedule checker diagnostic dialog box requesting a standard "open end" review. Figure #30 provides the same information as Figure #28, that being reporting of two open ends.

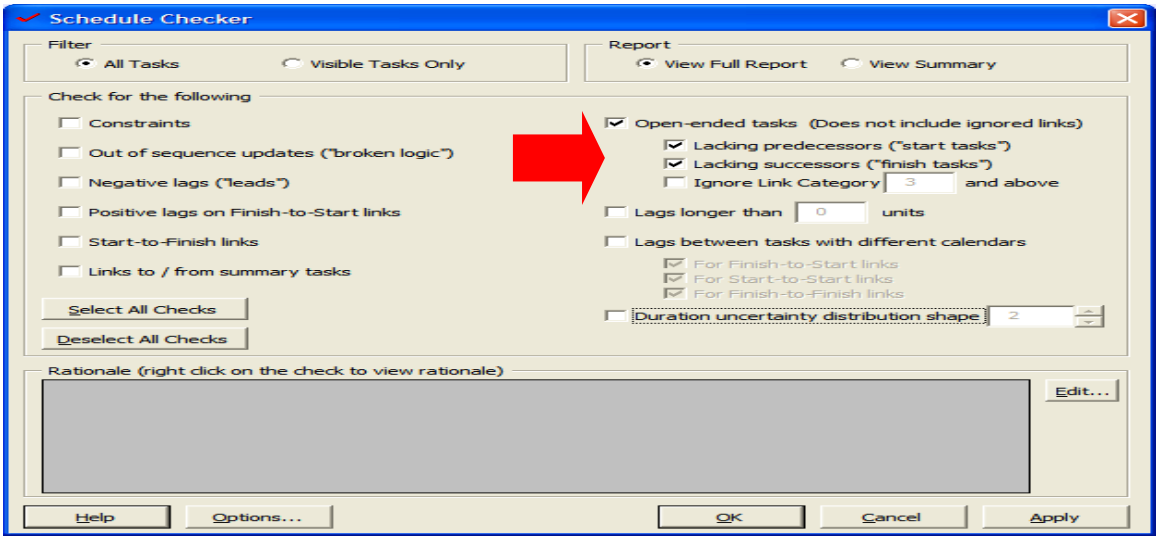


Figure #29 – Pertmaster Schedule Checker Dialog Box set for Standard Open End Test

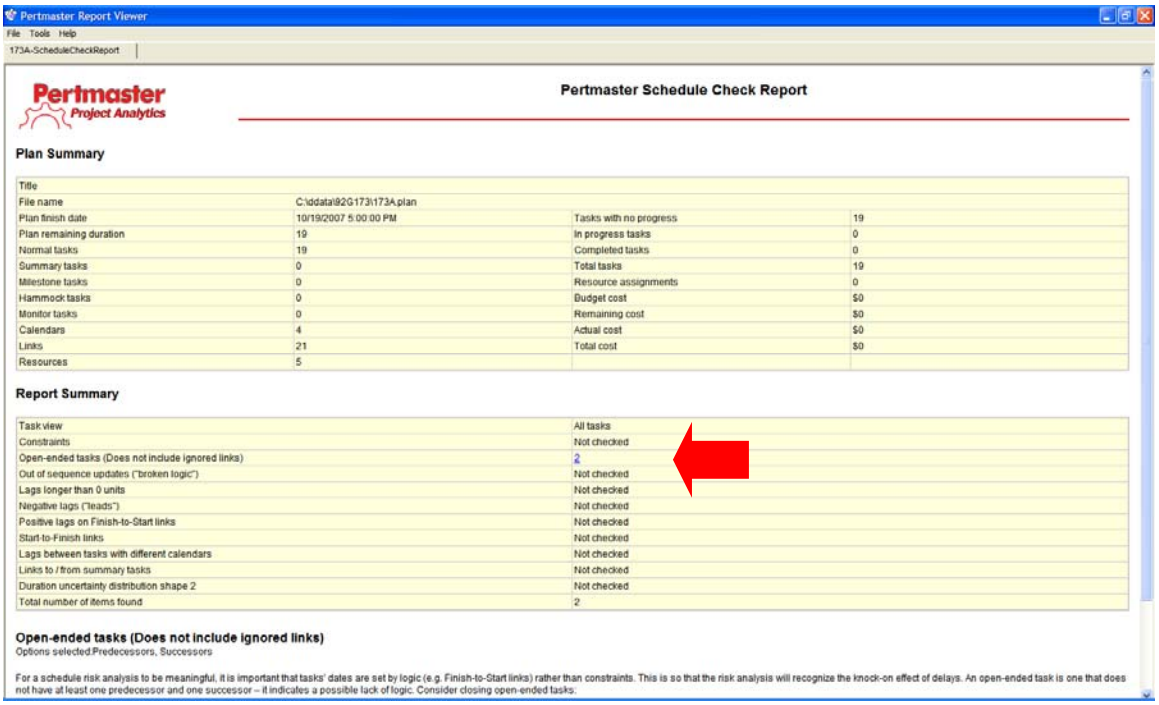


Figure #30 – Pertmaster Schedule Checker (Standard) Indicates Only 2 Open Ends

But, as noted on page 38 above, "each activity ***must*** be preceded by the physical completion of another activity (other than the first activity of a project) and, in turn, ***must*** be physically completed before being succeeded by another activity (other than the last activity of a project.)" And therefore, in Figure #31, the box "Ignore Link Category 3 and above" is checked, and an additional check is made that each activity physically require another, and be required for another.

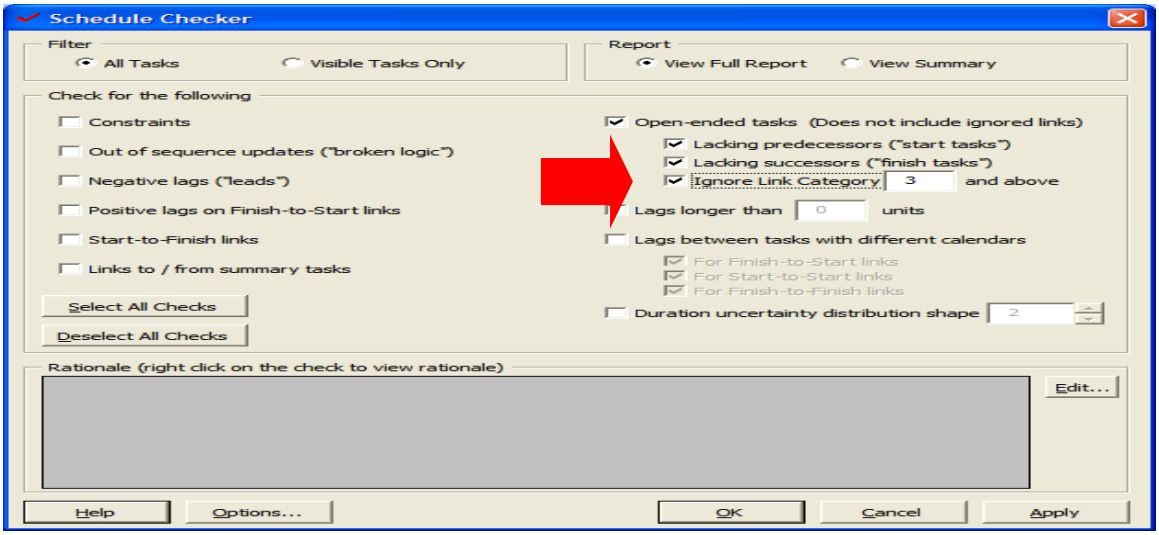


Figure #31 – Pertmaster Schedule Checker Dialog Box set for RDM Open End Test

The analysis result is provided in Figure #32, which indicates twenty-three open ends for this network of only nineteen activities. (The count indicates the number of activities not having a physical predecessor plus the number not having a physical successor; a truly "orphaned" activity will thus be counted twice.) The detail of these open ends are available for the reviewing network preparer or reviewing engineer and are provided here in Figure #33. Note in Figure #33 that activity #40 has neither a physical predecessor nor physical successor. This is further indicated in Figure #34 which indicates that the sole predecessor to activity #40 is activity #35 via a Link Category Type #3 (or Resource) restraint, and the sole physical successor is activity #45, also via a Resource restraint.

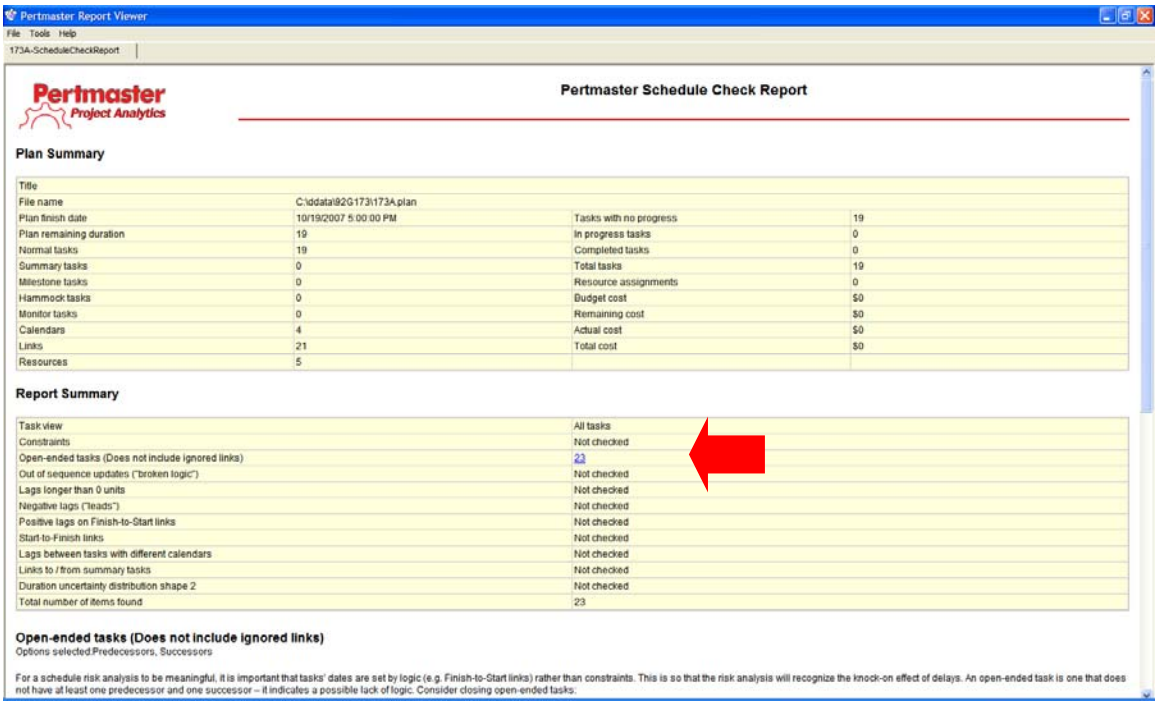


Figure #32 – Pertmaster Schedule Checker (RDM) Indicates 23 Open Ends

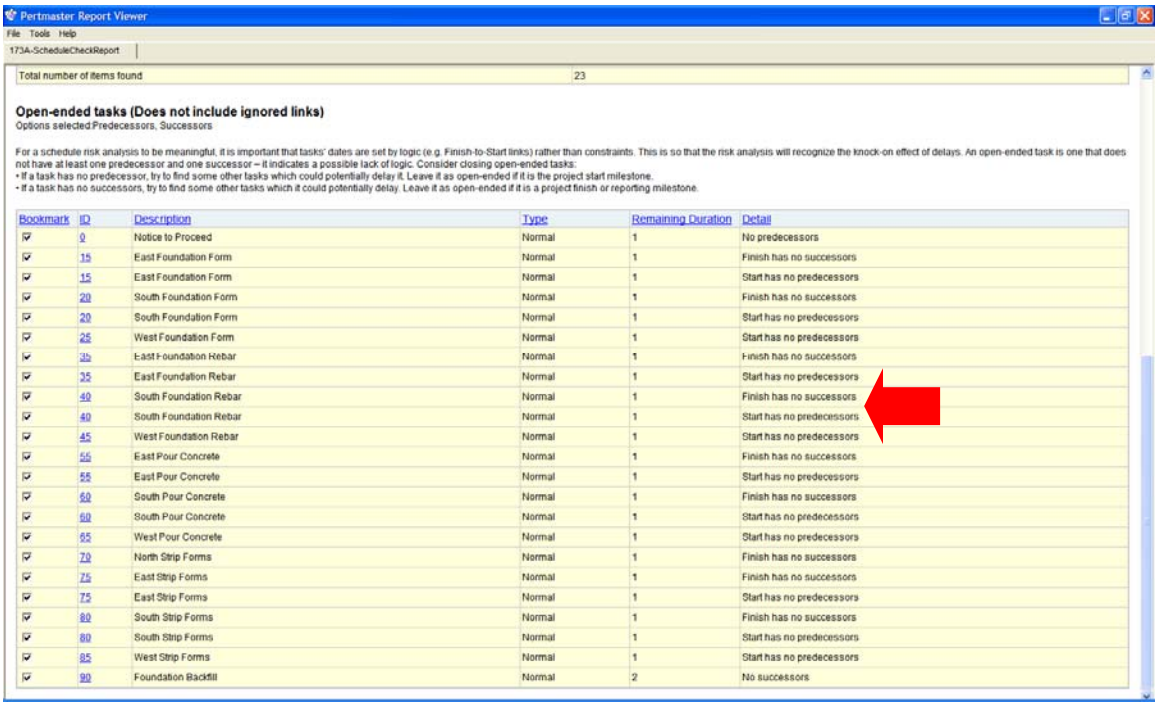


Figure #33 – Detail of Pertmaster Schedule Checker (RDM) Indicating 23 Open Ends

The means to correct these "open ends" is also stated at the top of the Pertmaster detail diagnostic, reproduced in Figure #35 for clarity. While it may be possible for a Resident Engineer or other reviewer to note the absence of the "light blue arrow logic" of Figure

#24 in Figure #26, a network of only nineteen activities, one may only imagine the level of effort to locate all "open ends" in a network of 1,000 or 10,000 activities. Thus, it may be fairly stated that this feature of RDM provides a real benefit.

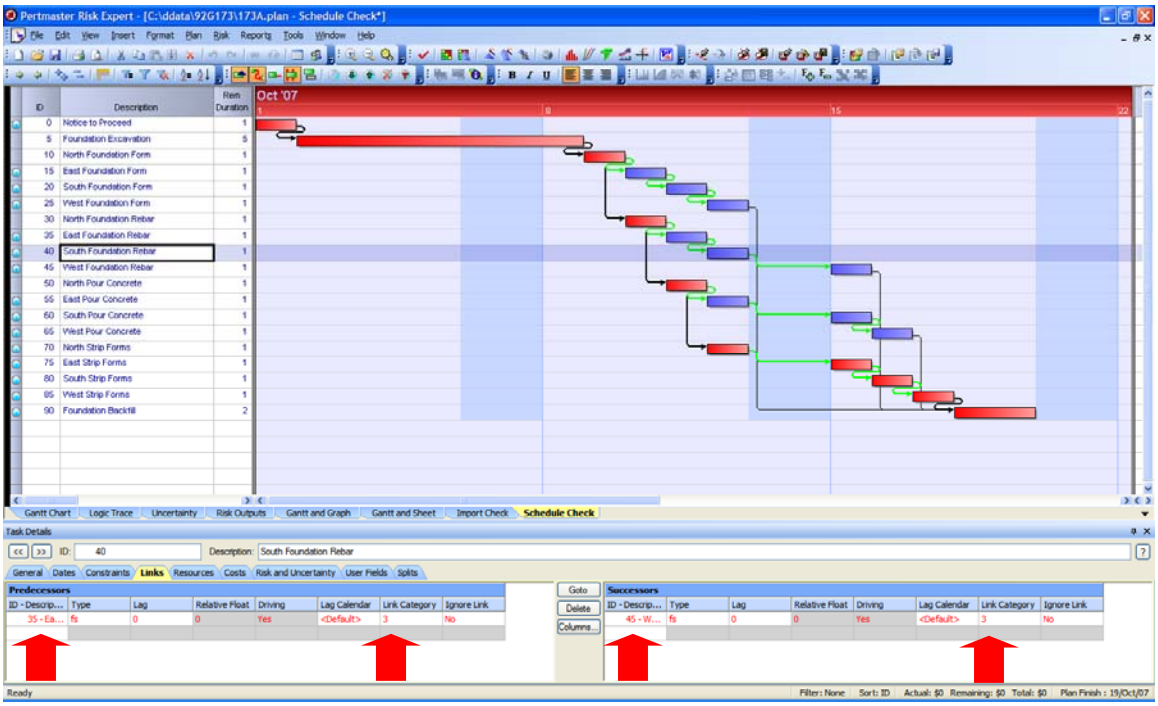


Figure #34 – Detail of Links To and From Activity 45 Indicates All are Resource Based

Open-ended tasks (Does not include ignored links)
Options selected: Predecessors, Successors

For a schedule risk analysis to be meaningful, it is important that tasks' dates are set by logic (e.g. Finish-to-Start links) rather than constraints. This is so that the risk analysis will recognize the knock-on effect of delays. An open-ended task is one that does not have at least one predecessor and one successor – it indicates a possible lack of logic. Consider closing open-ended tasks:
• If a task has no predecessor, try to find some other tasks which could potentially delay it. Leave it as open-ended if it is the project start milestone.
• If a task has no successors, try to find some other tasks which it could potentially delay. Leave it as open-ended if it is a project finish or reporting milestone.

Figure #35 – Detail of Pertmaster Schedule Checker Report

The addition of the Contiguous, Concurrent and Coordinated Lead/Lag codes provide a means for scheduling personnel to provide tabular and graphic reports in formats desired by senior project team members without the need to degrade the value of the CPM logic for prosecution of the project schedule. The ability of the computer to recognize and flag

relationships between activities should also contribute to CPM that are both technically better and more useful to members of the project team. But perhaps at some point in the future, it will be suggested that the most important benefit to RDM is ability to use the additional data recorded or calculated in pursuit of a more powerful methodology for leveling and smoothing of resources – otherwise to be called true computer aided scheduling of the project team plan.

RDM by itself may be used to improve the validity and usefulness of a CPM. But to truly answer the issues of industry critics and to fully utilize the power of computers of the 21st century require several other important tools. As noted above, the completion date for a project calculated by CPM is mathematically guaranteed to be overly optimistic. RDM does not solve this issue. This is best answered by inclusion of a Monte Carlo risk analysis step as part of the validation and reporting of the CPM calculation, as may be inferred from Figure #14 (page 24.)

The same methods used to fully implement risk analysis may also be used for forecasting and dynamic adjustments to the CPM. This may be dome by expanding the types of duration to include a Trend Duration ("TD") based upon an adjustably damped comparison between original and actual durations classified by similar work scopes. A separate SPERT style calculation could then be run based upon both the original and trend durations.

While discussing Monte Carlo or SPERT style calculations, another limitation of CPM should be considered. CPM does not have an ".OR." statement. Activity A may be followed by "B .AND. C" but not "B .OR. C." Therefore, a robust system should wrap in the power of GERT types of relationships including (1) B or C to follow A but not both at once, (2) logical loops to cover test failure, corrective action and retesting and (3) choice of action based upon progress or status of other activities within the logic network..

Finally, neither the Retained Logic nor Progress Override algorithms for out-of-sequence progress provide an accurate answer to the problem presented. The first result is not realistic – once work has started it can usually continue. The second result is also not realistic – the logic meant something and the preceding activity is probably still required to complete the started activity and certainly required for succeeding activities. This problem is best answered by the use of the Modified Progress Override algorithm suggested by this author.

Other areas of concern, discussed by this author in the 6th Edition of CPM in Construction Management, concern the distortions to the CPM logic model by demands upon a Scheduler to accommodate non-progress-performance issues such as cost and summarizing this unique project for reporting to an upper management reviewing many unique projects. Some of the issues of cost reporting may be alleviated by RDM's Concurrent Lead/Lag code or innovations such as Primavera's Steps function which allow listing of separate uncoupled tasks within an activity. However, much still needs to be done in this area to provide a "bullet-proof" system that can meet the needs of "the

accountants" and "upper management" without impeding the use of the CPM tool in timely and cost efficient delivery of *this* project.

Chapter 5 – Conclusion

As noted by Mr. O'Brien, "[t]he Critical Path Method (CPM) was developed specifically for the planning of construction, ..." and has been adopted most robustly for the planning and scheduling of work upon infrastructure projects. This is more than fortuitous; it is possibly inevitable that the myriad of independent entities involved in infrastructure construction be the prime user of this methodology. One may have difficulty suggesting another endeavor that combines the need for coordination amongst project team members with the transitory nature of such teams – formed but once for each unique project and rarely ever reassembled again for another project.

Infrastructure project design and construction of unique "one off" projects requires a different type of scheduling than other industries. These projects require a combination of precision coordination while providing a maximum of flexibility for a multitude of team members who are expected to weave in and out of this project as well as other projects. Other industries, such as manufacturing and software development, tend to have greater control over team members who are either captive or subordinate, or are willing to undertake greater losses today in return for a long term relationship. Infrastructure construction projects are usually designed and constructed by "the low bidder" who in turn establishes project teams in successive waves of choosing "the lowest qualified bidder" with the emphasis on "lowest."

The Gantt chart, developed for the static and repetitive scheduling required for manufacturing, served well for smaller projects. As projects became more complex, it

was the construction industry that embraced the use of CPM. The full promise of CPM was limited by the power of early computers. But now, as computing power has become more available and affordable, it is time to revisit the original "engineering fixes" of the early CPM programs, replace the duct tape, chewing gum and bailing wire with better adhesives, and once again embrace the full potential of CPM for management of our infrastructure construction projects. And then the Schedulers and Project Managers for these projects should lead by example for all projects, programs and operations requiring planning as well as scheduling.

List of References

1. O'Brien JJ, CPM in Construction Management, 1965, McGraw-Hill

2. Hansen BJ, Practical PERT, 1964, America House

3. Moder JJ, Phillips CR, Project Management with CPM and PERT, 2nd Edition, 1970, Van Nostrand Reinhold

4. O'Brien JJ, CPM in Construction Management, 2nd Edition, 1971, McGraw-Hill

5. O'Brien JJ, CPM in Construction Management, 3rd Edition, 1984, McGraw-Hill

6. Stevens JD, Techniques for Construction Network Scheduling, 1990, McGraw-Hill

7. O'Brien JJ, CPM in Construction Management, 4th Edition, 1993, McGraw-Hill

8. O'Brien JJ, Plotnick FL, CPM in Construction Management, 5th Edition, 1999, McGraw-Hill

9. O'Brien JJ, Plotnick FL, CPM in Construction Management, 6th Edition, 2005, McGraw-Hill

10. Korman R, Daniels S, Off the Critical Path, Engineering News Record, May 26, 2003, Vol. 250 No. 20., p 30 et. seq.

11. Plotnick FL, Introduction to Modified Sequence Logic, Conference Proceedings, PMICOS (first annual) Conference, April 25, 2004, Montreal, Canada

12. Plotnick FL, various presentations, Conference Proceedings, Primavera Systems Annual Conferences (1995 to present)

13. Plotnick FL, Techniques of Project Controls, Lecture Notes for use at Drexel University (1982 to present) and University of Pennsylvania (1990 to 1996)

Vita

Fredric Leigh Plotnick, born in Philadelphia, Pennsylvania on January 15, 1952, is a Professional Engineer (in PA, NJ, FL and MD) and an Attorney and Member of the Bar (of PA, NJ and FL.) He earned a B.S.C.E. majoring in Geotechnical Engineering in 1975, a M.S.C.E. majoring in Construction Management in 1977, and a J.D. in 1980. Mr. Plotnick is the co-author of CPM in Construction Management, 5th and 6th editions, O'Brien and Plotnick, McGraw-Hill (© 1999 and 2005,) as well as of numerous papers and presentations to the technical and professional societies in both engineering and law. He is a past president of the Philadelphia Chapter of the Pennsylvania Society of Professional Engineers in both 1989-90 and 1998-99, and is currently president-elect slated for a third term in 2009-10. He has also been an active member of the American Society of Civil Engineers, the Association for Advancement of Cost Engineers International, the Project Management Institute College of Scheduling, the Philadelphia and Montgomery County Bar Associations, and other technical and professional groups. Mr. Plotnick has been an adjunct member of the faculty of Drexel University continuously since 1979, teaching a variety of courses but focusing upon Engineering and Construction Law, Techniques of Project Controls and Project Scheduling. He has also taught at University of Pennsylvania from 1990 to 1996, assisting the Department of Systems Engineering in obtaining ABET accreditation. Mr. Plotnick has concurrently been employed in industry, including several employers such as Bechtel, Hill International and Fuller Company. Since 1983, Mr. Plotnick has operated a consulting firm, Engineering & Property Management Consultants, Inc., or EnProMaC, with a client base spanning the nation. He is a regular speaker at conferences of Primavera Systems, the developer of the predominant software utilized in his field of engineering. Mr. Plotnick is proud to announce that Primavera Systems has, as of May 2008, implemented into their software elements of the new RDM methodologies discussed in this dissertation.