RDM – RELATIONSHIP DIAGRAMING METHOD

Fredric L. Plotnick, Esq., P.E., Drexel University in Philadelphia, Pennsylvania

Abstract

Recent Enterprise CPM developments focus upon activities and away from the relationships between activities that was the hallmark of the original ADM and PERT methodologies. This paper proposes a system to address these issues, designated as RDM or Relationship Diagramming Method.

RDM documents the reason for a restraint between two activities. It expands the types of relationships, distinguishing partial performance, passage of time and a new "concurrent relationship." The RDM algorithm provides trend durations based upon actual performance to date and an automated assignment of optimistic and pessimistic durations for an integrated Monte Carlo analysis.

ADM was developed in 1956 and PERT in 1958 based largely upon the limitations of computers of that era. PDM was developed using the more, but still limited power of computers in 1964. RDM provides for better planning and scheduling than traditional or current implementations of CPM using the computers of the 21st century.

Key Words: CPM, ADM, PDM, RDM, RDCPM™. These refer to **C**ritical **P**ath **M**ethod of Planning & Scheduling, **A**rrow **D**iagramming **M**ethod variant, **P**recedence **D**iagramming **M**ethod variant, and the proposed **R**elationship **D**iagramming **M**ethod variant and **R**elationship **D**iagramming **CPM** standard. Also PERT or **P**rogram **E**valuation **R**eview **T**echnique.

This paper is an update of last year's presentation on RDM. The new format of RDM, or the Relationship Diagramming Method variant of CPM (Critical Path Methodology) of Planning and Scheduling was introduced in 2005 in the 6th edition of the industry bible, CPM in Construction Management, O'Brien and Plotnick, McGraw-Hill. Further details were presented at the PMI College of Scheduling annual conference by Fredric Plotnick, in April of 2006, as noted in ENR (06/05/06.)

Historical

The original format for CPM, developed fifty years ago in 1957, has since been renamed as ADM or Arrow Diagramming Method (also called AOA for Activity-on-Arrow) to distinguish it from the PDM or Precedence Diagramming Method (also called AON for Activity-on-Node) variant which was first implemented in 1964. Another variant of the same era was PERT or Program Evaluation Review Technique developed by the U.S. Navy for development of the Polaris Missile System, and which focused more upon defined milestones than upon the loosely defined activities between such milestones.

All three formats included weaknesses due to the limitations of computers of that era. The original ADM and PERT formats were designed to operate on computers lacking random access memory (RAM) and limited to linear access memory. Old cartoons illustrating computers with large reel-to-reel magnetic tape (or even punched paper tape) provide an indication of the limitations that the early software designers had to overcome. Many of the arcane rules of older CPM specifications, such as "skip numbering" (or identifying events as 5, 10, 15, rather than 1, 2, 3) are founded in these historical limitations.

The manufacture by IBM of the new computer architecture including random access memory, and thus the ability to rapidly switch between two files, one for activity name and duration, the other for the predecessor-successor restraints between activities, permitted the implementation of PDM. However, computers still had quite limited memory and were relatively slow and expensive. Each computation of the critical path was typically an overnight affair and could cost up to several hundreds of dollars! The advent of the personal computer in the 1980s largely reduced the cost, but in an effort to speed the computation, software designers jettisoned some of the features of the mainframe programs that may increase accuracy of the results in certain circumstances. Thus the ability to distinguish between starting activity B three days after a ten day duration activity A has started (and without regard to continued progress after day one,) from starting activity B when 30% of activity A was complete was dropped.

Introduction

In the research leading to the release of the 6th edition of CPM in Construction Management, Mr. Plotnick reviewed the original mathematics of the early 1950s leading to CPM and PERT, and has reformulated and expanded both the algorithm for calculations and the formats for data entry and reports. Although some portion of this information may be stored and reported from extensions to existing PDM software, it is important to understand that RDM is as different from

PDM as are ADM and PERT. The main distinction is the redefinition of the differences between events, activities and restraints. Additional distinction range from the introduction of new attributes (such as a Just-in-Time start, finish and float, discussed below) to the appreciation that the relationship between two activities is more than restraint between activities (also discussed below.)

Just as CPM and PERT were distinguished from bar-charts and milestone charts by the recording of additional information used to create those charts (and thus automating the process of utilizing that information,) so too RDM is distinguished from the older versions of CPM and PERT by recording more of that information. This additional information may be generally classified into five additional groups of code fields:

- events and event codes
- the reason/why restraint code, additional description as to the reason why a restraint has been placed between two activities,
- duration codes to explain a given duration and how such should be used in calculations,
- additional and expanded types of restraints between activities (and events,) and
- relationship codes to further indicate the relationship between two activities.

Events and Event Codes

The primary fundamental aspect of RDM is the reintroduction of the concept of an *even*t, or point in time, that was the hallmark of both ADM and PERT but was eliminated in the implementation of PDM. However, RDM expands the use of events and attributes of events, and largely redefines the concepts of an activity and a restraint. In ADM, an activity is placed between two events, and an event is properly placed only at the start or finish of one or several activities. In RDM, while an activity is still placed between two events, an event may be placed at the start or finish of only one activity. More significantly, an event may be placed within an activity or be totally independent of any activity. Thus the scope or definition of partial completion of an activity may be noted at such an internal event and true milestones (rather than merely zero duration activities) may be defined.

Events do not have an early start or early finish, nor a late start or late finish. Events, being a point in time, merely have an early occurrence and late occurrence. Historically, these were referenced as T_E and T_L, referring to "time-early" and "time-late," in the early writings on CPM and PERT. A special problem exists in converting "times" to "dates" in most CPM and PERT applications since most calendars are discontinuous. Unless the only calendar is a 24/7/365 calendar, then Monday afternoon at 4:00 P.M. is the same "time" as Tuesday morning at 8:00 A.M. This issue is exacerbated by weekends, holidays and seasonal shutdowns (11/15 to 3/15.) Thus, a code field must be reserved for reporting event times as the preceding (evening) or succeeding (morning) date associated with that time. (As a aid to memory, it may be remembered that Genesis reports the date of creation to be "the evening and the morning of the first day.")

A number of event code fields in RDM are devoted to recording whether an event is preceded or succeeded by an activity, counting the number of restraints preceding and/or succeeding an event, and similar tallies. These may include counting "exclusive predecessor restraints" and "exclusive successor restraints" where such are the sole restraint emanating from or to the event at the other end of the restraint. Where several events "exclusively" share the same predecessor events, these may be designated as concurrent events, similar to the ADM *i*-node common to several activities.

The T_E (time-early) of an event in ADM or PERT, or ES (early start) of an activity in ADM or PDM, is calculated as the latest T_E or ES of all predecessors. So too for most events in RDM. However, RDM also recognizes other choices for independent events not linked (at the beginning or end) of an activity. These choices include:

- do upon the completion of the 1^{st}, 2^{nd}, n^{th}, or last predecessor (note last is traditional default,)
- follow restraint ONE if 3^{rd} event status is (not occurred, occurred,) otherwise restraint TWO,
- follow restraint ONE if 3^{rd} activity status is (not started, started not finished, finished) , otherwise restraint TWO,
- follow restraint ONE if random number w/i set limit, otherwise restraint TWO, and
- follow restraint ONE if random number w/i limit #1, TWO if w/i limit #2, N if w/i limit #N, otherwise default restraint

Note all but the first group of choices indicates multiple possible logic paths and the possibility of loops. The first group of choices also require an alternate event to which restraints from subsequent triggers will lead in order to avoid a logical open end.

Reason/Why Restraint Code, Other Restraint Codes and Restraint Description Fields

The second fundamental aspect of RDM is the systematic recordation of the reason why a restraint has been placed between two activities. This is accomplished by the provision of a *reason/why* code to

each restraint and by further provision of a title or description field and user defined code fields to restraints similar to that provided to activities.

In PDM, the distinction between an activity and restraint may be quite blurred, especially if to the restraint has been assigned a lag duration between activities. The main distinction is that to an activity may be assigned a title or description, while it is not possible to record any information relating to a restraint other than relating to its predecessor, successor and quantum of lag duration. Another distinction is that durations of activities is generally to be measured by performance of the activity (in either days performed, days remaining or percent complete,) while durations of restraints (lags) are generally measured by the passage of units of time (days) from a reported commencement (date.)

RDM permits, encourages and perhaps even requires some level of explanation or description be given to restraints as well as activities. At the very minimum, RDM will request whether the *reason* for the restraint is "physical" or "hard logic," thus indicating that A *must* be performed before B (with examples such as "gravity" or "owner's specification" being the *why* for the restraint,) or that the *reason* for the restraint is "*resource limitations*" (with the specified resource, e.g. crew, craft, access, dollars, etc., being the *why* for the restraint) and that the superintendent or project manager merely chooses to perform A before B (as being the most economical of possible choices.) It is envisioned that RDM software will, if the description field is left blank, optionally provide a text description of "PREDS 1000, 1005; SUCCS 1015, 1020." Other specialized reason/why code values are discussed below.

In ADM, restraints may emanate only from the end of an activity and conclude at the beginning of another activity. In PDM, restraints may emanate from either the beginning or end of an activity and conclude at the beginning or end of another activity. In RDM, a restraint may emanate from the beginning or end or from within a partially performed activity and conclude at the beginning, end or within another activity. Therefore RDM will distinguish between starting B until 5 days after A is reported started, and until 5 days of A have been reported performed (by either reporting of a percent complete or a remaining duration being 5 days less than the original duration provided.) Restraint durations in RDM may thus be deemed independent or dependent upon the activity from which or to which such duration runs.

Duration Codes

RDM permits (requires) more detailed information relating to the durations of both activities and restraints. Durations of activities may be further defined by use of an activity calendar or resource calendar. Durations of restraints independent of measurement of activity progress (being measured solely from a reported activity start or finish date) may also be further defined by use of the activity calendar. Durations of restraints dependent upon the measurement of activity progress (being measured by units of scope performed, estimates of remaining duration, or percent of scope complete) are based on the calendar of the activity upon which such measurement is based.

Durations are also subject to three other assigned characteristics. These are:

- interruptible/continuous,
- performed/clocked/clock-checked, and
- out-of-sequence modified-logic/retained-logic/progress-override.

The *interruptible/continuous* attribute to duration determines how the early start of an activity is to be calculated. If an activity is restrained by a finish-to-finish or start-to-finish type of restraint or by a constraint pushing early finish beyond the calculation of early start plus remaining duration, there is a question whether the early start should remain as calculated based upon its predecessors (and thus the span of time from the start of the activity until its completion, including active and inactive periods of work, will be greater than the original duration entered) or should the early start be delayed until work may be performed continuously. The primary purpose of such designation is to define the early start date to be used by successor start-to-start or start-to-finish restraints emanating from the activity.

However, both early start dates should be calculated and recorded, even though only one may be used for determining the start of subsequent start-to-successor restraints. The knowledgeable practitioner may desire this information to utilize the additional start-float of the activity, or to start between the two early start dates with the hope to encourage the predecessor-to-finish activity to finish early. It is also recommended that software provide some indicator (by use, for example, of a "*" or italicized date text) where two possible early start dates may be reported, and some other indicator (by use, for example, of a "†" or underlined date text) where the activity is succeeded by a start-to-start restraint, and thus it would be important to begin work "ASAP" even if such would require work on the activity to be interrupted.

The *performed/clocked/clock-check* attribute to duration determines how the duration is to be decreased during updates. Most activities have scope where performance may be measured for purposes of

updating. However, some activities, such as curing of concrete or awaiting review of submittals by the engineer, are not subject to measurement and are typically updated by counting the number of days (or other unit of time) from a calculated or reported start date. This can be especially bothersome where the duration is greater than the period of an update, such as a 28 or 56 day cure period, or a 45 day review period. Updating in the field thus requires a manual count of such days and which is often missed and not reported resulting in errors in the updated schedule.

Inclusion of the performed/clocked/clock-check attribute permits the scheduler to identify these type of activity while developing the logic network, thus reducing the effort for updating to that of recording an actual start where the attribute is set to *clocked*. A special case is provided where progress continues automatically with the clock, but a visual check should be made to assure completion, such as for return of an approval of a submittal. The *clock-check* attribute will thus reduce the duration automatically to one day (or other time unit) requiring a manual entry of actual finish date (or zero remaining duration) when such is visually confirmed. Where the attribute is set to *performed*, updating is achieved by visual review and entry of remaining duration or percent complete.

The *out-of-sequence modified-logic/retained-logic/progress-override* attribute to duration also determines how the duration is to be decreased during updates. If an activity starts prior to the finish of its predecessor (or prior to the start of its predecessor where such is by a something-to-start restraint,) there is a question whether the CPM algorithm should calculate that work should stop on the started activity until all predecessor are complete, or may continue without regard to completion of predecessors. The first option is known as *retained logic*, the second as *progress override*. As discussed in previous presentations by the author, initially at the PMI College of Scheduling conference held in Montreal in 2004 and thence elsewhere, a third option to be considered is modified logic, which calculates the early (or continuing) start of the started activity as the datadate, but set the early finish as the latter of DD+RD or the early (continued) start as calculated by the retained logic algorithm. (This "third way" was first introduced in Montreal as "*modified progress override*," was later renamed "*modified retained logic*" in internal discussions of Primavera who are considering the use thereof, but is here again renamed as "*modified-logic*.")

While it has been suggested that the new modified logic algorithm should become the standard and that there should be no further need for the original two choices, there are situations where the older algorithms better approximate the real world. 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. We 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. Assuming a robust software program for implementation, the attribute could be set to progress override, or at least highlighted for review, wherever the reason provided for the restraint is "resource."

Calendars of Durations of Activities and Restraints

Development of a calendar for use in scheduling algorithms is a complex and complicated endeavor. One of the most difficult issues is to set a minimum unit of duration for the calendar. On some projects, including most construction projects, the minimum duration is one day. If the work day begins at 8:00 A.M. and all the predecessors of an activity are expected to be complete by 10:00 A.M., the superintendent will not typically mobilize the labor crew and other resources for that activity until the next morning at 8:00 A.M. Similarly, even if the work is being performed 24/7/365, if such work is being performed in three shifts per day, crews and resources are typically not mobilized for usage in the middle of a shift, but are deferred until the start of the next shift after completion of all predecessors to the new activity to be performed.

For other projects, including most design projects, the minimum unit is one hour. Similar to the discussion for construction, few managers would become distressed if an engineer, receiving data required to provide the next step of a design at 10:17 A.M., will continue to work upon and then button up other work being performed before commencing upon the new design work at 11:00 A.M. Even if the minimum unit is one second, so long as some task may be completed in other than an increment of the minimum unit, this issue of elongation (rather than truncation) of the duration to the next increment must be considered in development of a CPM algorithm calendar. However, in a production line or emergency setting, the period of elongation may be reduced if not eliminated and meaningful work can be commenced immediately upon completion of a predecessor activity.

And as projects grow larger and are combined with other projects into programs and enterprise-wide programs, the issue grow more complex. A fast-track or design-build project calendar must contend with an

hourly (or quarter-hourly) calendar for the designers as well as a daily (or shift) calendar for the construction crafts. While a robust calendar system will be able to accommodate both daily, two and three shift per day, hourly and sub-hourly minimum durations, the transfer between such calendars is best performed at the zero duration point-in-time event.

Types of Activities

Although not presented as a new element instituted with the introduction of RDM, the existence of differing types of activities should be noted at this point. Durations of activities, the performance of which utilize resources, may be via an activity based calendar or upon a resource based calendar or calendars. Thus an activity may be performed on any day (or other time unit) where resources are permitted onto the project site (or the extended site where the activity is to be performed,) or may be limited to days when a resource or multiple resources are available. If the duration is driven by multiple resources, then it must be further determined if progress may continue when any one of multiple resources are available, or only when all required resources are concurrently available.

Types of Restraints

In PERT, there are no activities, only events and restraints (with or without durations) between such events. In ADM, events are connected by either activities of specified duration (which also carry logic between events,) and restraints of zero duration. In PDM, it is generally understood that there are no events, but rather only activities (of specified duration) and restraints (with or without durations) between activities. These restraints are generally understood to emanate from the start or the finish of one activity and connect to the start or finish of another activity. Thus the four types of restraints available in current popular CPM software are usually listed as finish-to-start, start-to-start, finish-to-finish and start-to-finish.

The belief of a limit of four types of restraints between activities is not totally correct under either the papers by Dr. Fondahl (credited with the development of PDM in the 1950s) or the early software programs run on mainframe computers in the late 1960s, 1970s and early 1980s. These programs distinguished between a Start restraint (delaying the start of the successor activity until OD-RD equals the lag duration – which must always be less than the duration of the predecessor activity) and a Begin restraint (measuring the lag duration based upon clocked time units from the reported start date.)

Similarly, the programs distinguished between a Finish restraint (delaying a portion of the successor activity equal to the lag duration – which must thus always be less than the duration of the successor activity) and an End restraint (measuring the lag duration to delay the last moment of the successor activity.) The Start and Finish type restraints could be expressed in percent of predecessor and successor duration respectively. The Begin and End type restraints, however, could only be expressed in time units.

Interestingly, it was the Begin restraint that morphed to the current start-to-start restraint, and the End restraint that morphed to the current finish-to-finish restraint. Certainly, the programming effort required to effectuate such types of restraints is simpler, as there is no need to check that the lag duration is less than the activity duration of the predecessor or successor activity respectively. Therefore, the current restraint choices may all be classified as independent of the duration of the activities.

On the other hand, since the old Start and Finish type of restraints may be expressed as percents of the predecessor and successor activity durations respectively (and must thus be verified as being less than 100% of such durations,) these may be as dependent upon the duration of the activities. Since the old Start restraint type involves partial performance of the predecessor activity duration, RDM has renamed it to a “Partial-to-Start” restraint type. Similarly, since the old Finish restraint type involves deferring a portion of the successor activity duration until the finish of the predecessor activity, RDM has renamed it to a “Finish-to-Partial” restraint type.

Data collection and reporting for these two rediscovered restraint types may use any one of several formats. For the Partial-to-Start restraint duration, measurement may be by:

- time units of the predecessor activity performed before successor activity may begin,
- percent of predecessor activity duration performed before successor activity may begin,
- fraction of predecessor activity quantity performed before successor activity may begin, or
- time units of the predecessor activity remaining to be performed before successor activity may begin.

For the Finish-to-Partial restraint duration, measurement may be by:

- time units of the successor activity duration remaining to be performed after finish of the predecessor activity,
- percent of the successor activity duration remaining to be performed after finish of the predecessor activity,
- fraction of the successor activity duration remaining to be performed after finish of the predecessor activity, or

- maximum time units of the successor activity duration that may be performed prior to finish of the predecessor activity.

Exacerbating the issue are several questions relating to restraint durations. Obviously, the Partial-to-Start and Finish-to-Partial type of restraints are tied to the calendar of the activity of which the restraint duration is a subset of the activity duration. The Start-to-Start and Finish-to-Finish type of restraints must be assigned their own calendar, as well as the Finish-to-Start and Start-to-Finish restraint types.

Partial-to-Start and Finish-to-Partial type of restraints will mimic the activity to which duration is tied with regard to interruptability and means of measurement (performed/clocked/clock-check,) while SS, FF, FS and SF type of restraints will default to continuous and clocked by definition.

In situations of out-of-sequence progress, Partial-to-Start and Finish-to-Partial restraint durations will be linked to the modified-logic/retained-logic/progress-override choice of the respective predecessor or successor activity duration. However, where measurement of the restraint duration is independent of the measured progress of the predecessor or successor activity, such as for SS, FF, FS and SF restraints, the same issues for measurement of restraint durations arise as for activity durations performed out-of-sequence.

When considering all of these issues, there are some six types of restraints between activities with each having three or four sub-types. Note a seventh type (Partial-to-Partial) is listed but would not be used due to the need for two restraint durations and restraint duration codes and difficulties of such implementation. However, a fix is provided for such situations. A similar fix may be provide if two types of restraints are indicated, such as "start activity B 3 days after 50% of activity A is complete." The listed restraints are:

- Finish-to-Start – update restraint duration by clocked time units from predecessor activity
  - update restraint duration always from calculated early finish of activity
  - update restraint duration from actual finish of activity if reported
  - update restraint duration to minimum of ONE until calculated early finish equals datadate
- Start-to-Start – update restraint duration by clocked time units from predecessor activity
  - update restraint duration always from calculated early start of activity
  - update restraint duration from actual start of activity if reported
  - update restraint duration to minimum of ONE until calculated early start equals datadate
- Partial-to-Start – update restraint duration from progress of predecessor activity
  - restraint runs from event within predecessor activity at duration from event at activity start
  - formats: time units complete, percent complete, quantity units complete, time units remaining
- Finish-to-Finish – update restraint duration by clocked time units from predecessor activity
  - update restraint duration always from calculated early finish of activity
  - update restraint duration from actual finish of activity if reported
  - update restraint duration to minimum of ONE until calculated early finish equals datadate
- Finish-to-Partial – update restraint duration from progress of successor activity
  - restraint runs to event within successor activity at duration to event at activity finish
  - formats: time units remaining, percent remaining, quantity units remaining, time units complete
- Start-to-Finish – update restraint duration by clocked time units from predecessor activity
  - update restraint duration always from calculated early start of activity
  - update restraint duration from actual start of activity if reported
  - update restraint duration to minimum of ONE until calculated early start equals datadate
- Partial-to-Partial – theoretical – would require two restraint durations
  - fix is to run Partial-to-Start to independent event, then Finish-to-Partial to successor activity

The decision on how to designate the type and subtype of a restraint, to avoid confusion amongst individuals familiar with the limited choices of PDM, remains an open question. One possible solution involves two codes, one for type of restraint and the other for type of restraint duration (or lag.) These may be expressed (with ## being a number) as: FS##E, FS##A, FS##M, SS##E, SS##A, SS##M, PS##C, PS##P, PS##/##Q,PS##R, FF##E, FF##A, FF##M, FP##R, FP##P, FP##/##Q, FP##C, SF##E, SF##A, SF##M.

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 reason 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, it is doubtful that 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.

The means to communicate, for example, a "handoff" between subcontractors may be by use of an alternate font (say italic and green) for the predecessor and successor activity descriptions and a logic line between bars (on the bar-chart view) or activity boxes (in pure logic view) in an alternate aspect (say dashed and green, with dashed and red indicating a "handoff" that is on the critical path.) Or flagged "P" physical reason restraints may be in a blinking font during the diagnostic review.

Additional Attributes of Events, Activities and Restraints

The advent of PDM, where the calculated early finish of an activity may be driven by a finish-to-finish restraint rather than the early start plus duration, and the calculated late start driven by a start-to-start restraint rather than late finish minus duration, requires the recognition of several new activity attributes. These include a Start Total Float, Finish Total Float and Most Critical Total Float, hereafter notated at "STF", "FTF" and "TF." Many popular CPM software programs will permit the user to calculate one (and only one) of these three choices, while all three are of value to the knowledgeable scheduler.

For most applications, the TF attribute is the one that the scheduler and project team will review. However, the STF and FTF attributes also convey information. A overly high FTF value is indicative of a "hidden" open end, where the activity does not have a finish-to-something successor, and thus need never finish. This open end may be caused by poor initial logic, or may not be discovered until updating the project when work is performed out-of-sequence and choosing a "progress override" calculation algorithm. A overly high STF value may similarly be indicative of a "hidden" open end (assuming use of interruptible activity durations.) The very fact that the STF does not equal the FTF conveys real information to the scheduler (and project team) that the activity completion is driven by restraints and not by activity duration. (Thus increasing resources, providing overtime, etc., will not speed the completion of this activity!)

RDM complements these features with additional attributes. The early start and early finish attributes represent the earliest times that an activity *may* start or finish, based upon the datadate. The late start and late finish attributes represent the latest time by which an activity *must* start or finish if earliest possible project completion is not to be delayed. These attributes are typically denoted as ES, EF, LS and LF. Similarly for an event, the attributes time-early time and time-late time (T_E and T_L) represent the earliest time that an event *may* occur, based upon the datadate, and latest time by which it *must* occur, if not to delay completion of the project.

RDM adds a middle set of attributes for the latest time by which an event or activity *must* occur, start or finish, so that a specified successor event or activity *may* occur, start or finish as early as though this restraint did not exist. These attributes are thus called just-in-time time, just-in-time start and just-in-time finish. These attributes are denoted as T_J, JLS and JLF. The new attributes, in turn, are used to calculate additional just-in-time total float attributes of SJTF

(equal to LS-JLS,) FJTF (equal to LF-JLF) and JTF (equal to the more critical of the two.)

Specialized Reason/Why Codes

The means by which a restraint is designated as not to drive its successor (assuming its successor does have another predecessor which is driving,) is to set the restraint reason code as "J" for "just-in-time." This is a specialized version of the "P" physical restraint, and will be treated as a "P" reason for all other purposes other than during the backward pass calculations. During the backward pass calculation, the normal RDM algorithm of $\{LF_{PRED\ RESTRAINT} = LS_{SUCC\ ACTIVITY}\}$ is changed to $\{LFPRED\ RESTRAINT = ES_{SUCC\ ACTIVITY}.\}$ The resultant T_J, JLS and JLF times calculated will therefore be somewhere between the early and late times. If the restraint carrying the "J" reason is the sole restraint following a preceding event or activity, the JTF of the restraint will equal the free float ("FF") of the restraint or immediate preceding event or activity. However, that is where any similarity of JTF to FF will end; the JTF will continue backward through the string of activities leading to the designated restraint, while the FF attribute is non-zero only for the last activity in the string.

Another set of specialized restraint reason codes relate to automated resource leveling routines and are generated by the RDM leveling algorithm to selectively replace and augment an "R" or "resource" reason. The first is the "S" or "suppressed" reason. When leveling upon a selection of a single resource, or multiple recourses, the first step of the scheduler should be to carefully review all restraints and delete those which carry those resources. If this is not done, the leveling routine is "hard-wired" to allocate the resource when released by a completed activity to a specific activity seeking resources, rather than allowing the computer to choose the most advantageous allocation. The RDM leveling algorithm therefore first converts all restraints from "R" to "S" where the "why" is any of the resources being subjected to the leveling algorithm.

The RDM leveling algorithm then augments the specified logic restraints with additional restraints indicating from which activity *the last* resource was released that now permits the new successor activity to proceed. These added restraints are provided a reason code of "L" or "leveled." In viewing the pure logic diagram or other graphics thereafter, the scheduler may see the logic used by the leveling algorithm. The scheduler may choose to display, or not, the suppressed "S" logic, to see where it has been replaced, or confirmed as being correct (where the "S" restraint is alongside an "L" restraint.) Upon rescheduling or releveling, all "L" restraints are deleted, all "S" restraints are reset to "R" and then may be set again to "S" depending upon the parameters of the scheduler's request for leveling.

Specialized Types of Events

As previously noted, the concept of events is central to a proper logic network and therefore to RDM. A number of limitations of CPM may be remedied by use of specialized types of events. In the original ADM version of CPM and in PERT, where two or more logic restraints or activity arrows (also carrying logic) or any combination thereof enter an event (or node,) processing stops until the early finish of each such restraint or activity has been calculated. Then the early start of the event, and subsequent restraints or activities, is calculated as the latest of such early finishes.

There are other possible outcomes for the merging of two logic paths, as discussed in the GERT literature of the 1950s through the present, but not readily available in commercial software products. Several specialized event types, supported by fully functional RDM, are discussed here.

The first of the specialized event types to be discussed is setting of the early occurrence time based upon the 1^{st}, 2^{nd}, n^{th}, or last early finish of multiple predecessor restraints. In the default setting or event type, the early occurrence of an event having multiple predecessor logic will be the *last* or latest early finish of all such predecessors. However, an event type may be coded to set the early occurrence of the event to the first, second or n^{th} early finish of all such predecessors. Part of the algorithm for implementation of this feature includes the diagnostic to assure that none of the predecessors are "exclusive" or emanating from activities that are not followed by some other successor in such manner to not create an open end.

A second class of special event types is to provide a choice of successor logic strings from an event, rather than indicating that all activities following such event may now be performed. This is the GERT "·OR·" statement to complement the PERT or CPM "·AND·" statement. The choice of logic restraint successor may be by a random number generator (RNG) or by an explicit statement tying such to some attribute of another event. As examples, "A will be followed by B if RNG is ≥90%, by C if RNG is ≥50%, else by D," "A will be followed by B if actual finish reprot4ed for D, otherwise C."

Other specialized event types include those required to implement those discussed above, such as to provide closure to a string of activities that may be abandoned based upon the circumstances encountered, and to reenter a loop, such as where a test, previously failed, is now to be retaken.

Specialized Types of Activities

There are also possible a number of specialized activity types, many of which have been implemented in commercial software products, some of which have not. Activity duration is usually based upon a project calendar indicating when the activity may occur. Often, the determining factor for performance is when the chosen resources for such activity are available. Thus, commercial software, such as that marketed by Primavera Systems, provides a choice between having an activity duration driven by the project calendar or driven by the calendar of a resource. Prior versions of Primavera also provided, when two or more resources may be considered as driving, a choice between the copulative and disjunctive, or between requiring all driving resources to be present to advance progress for one time unit, and requiring any one of several such driving resources to be present. These were known as meeting and independent activity types respectively. However, even if all driving resources are available for a specific time or date, the activity calendar must still permit work to be performed for that time period.

Other specialized activity types may include hammocks, with or without logic checking, steps or tasks, and fragnetted activities. Hammocks traditionally have not carried logic, but rather have been used to summarize a group of activities between two points in time. In ADM and PERT, this was accomplished by drawing the hammock between two event nodes; in PDM this was approximated by insertion of an activity restrained from the start of one activity and restraining the finish of some other activity. Traditionally, a manual or automated check is also made to assure that the hammock spans actual logic and is not merely connecting any two events or activities. Misuse of hammocks in commercial software that does not provide this type of validation check can lead to the humorous instance of update progress on one chain of activities leading to the start of a hammock leading to non-progress on a separate chain of activities leading to the finish of the hammock, causing the hammock to report a negative duration.

Since hammocks are typically used for management overview, rather than day-to-day field operations, the occasional error introduced by the lack of logic validation may perhaps be tolerated. However, the miscommunication to senior management that one milestone leads to another, rather than that it has merely been planned to occur prior to the other, should be noted as a different activity type than a proper hammock.

The concept of "steps" or "task" activities is to split an activity into two or more such tasks which each contribute to the total duration of the activity, but cannot be quantified as to timing within the activity. An example is the card game of "52 pickup" where we can estimate the total duration for picking up all of the card, and we can estimate the duration for picking up any subset of the cards, but we would not normally specify a sequence or order of picking up such cards. Thus, since face card represent $4/13^{th}$ and number cards $9/13^{th}$ of the total (or 31% v 69%,) if we were to report that we have picked up $1/3^{rd}$ of the face cards and $2/3^{rd}$ of the number cards, the step algorithm will compute that 56% of the activity has been complete and calculate the appropriate remaining duration.

The reintroduction of logic validation for hammocks permits another new feature of RDM, that of a hammock that does carry logic and a given duration, but with its remaining duration determined by the reported progress of the activities within the hammock. This is designated a *Pacing Hammock* and treats a group of activities having common (if not immediate) predecessor and successor events, as connected by the pacing hammock, much like the "steps" of a single activity. An example where this may be useful is where several pumps may be rigged, set, piped, powered, etc., onto a common foundation slab. Because the various activities may be performed by differing crews or subcontractors, they must be separately itemized. However, although only one or two pumps are rigged, etc. at a time, the order is not subject to pre-planning and will depend upon field conditions or chance. Thus the total time of these parallel chains of activities will be greater than the duration of any one chain. The algorithm behind the pacing hammock totals the original and remaining duration of all activities which have a common predecessor event and common successor event, compares such as a percentile, and applies that percentile to the original duration of the pacing hammock to determine its remaining duration.

Unlike a traditional hammock (with or without logic validation,) a pacing hammock contains operational information and should be printed on field reports, and such may still be too detailed for management reports. Perhaps a different name should be applied to distinguish the two types of hammock. A similar issue should be noted for the activities that are part of pacing hammock. It may be desired that the detail of these activities, or fragnets, should be generally be masked from reports on the entire project and reserved for detailed "three week look-ahead" and similar reports.

Specialized Types of Restraints

In ADM, the only means to split the scope of one activity into two or more sequential scopes is to physically split the activity into two or more activities. In PDM, a scope may be so split explicitly by a percentage-complete-type of start-to-start restraint (as

with the 1970s MSCS software system) or by implication with currently available commercial software such as Primavera. (Note that Deltek's OpenPlan software currently supports a percentage-complete-type of start-to-start restraint, but does not support a matching finish-to-finish restraint.) RDM accomplishes the split by explicitly using a partial-to-start restraint which creates an event node within the preceding activity (and similarly supports a finish-to-partial restraint which creates an event node within the succeeding activity.) These event nodes may then be further annotated by including an explicit explanation of how the scope has been split in the event description field.

RDM also considers the situation where two (or more) sequential activities are to combined into one continuous activity. This is analogous to setting the activity duration code switch to continuous rather than interruptible. The virtual linking of these activities is accomplished by use of a additional specialized type of restraint, designated as a *Contiguous* restraint type.

A similar result may be accomplished using existing commercial software products by use of a free-float-constraint, assuming that the activity to be deferred until a successor activity may start has no other successors. The RDM activity preceding the contiguous restraint may have other successors, and they too will be deferred until the successor to the contiguous restraint may start.

Another specialized type of restraint introduced in RDM is the *Concurrent* restraint. This indicates that two or more activities must be performed concurrently. As is understood in CPM, the fact that two activities share the same predecessor does not guarantee that they must be performed concurrently, but only that such is possible. Once the common predecessor(s) of the two are finished, each of the two may be performed independently. The concurrent restraint changes this and indicates a common dependency, such as the placement of an MSE wall and backfill of such wall, or the activities of a surgeon and anesthesiologist.

Other specialized types of restraint may be added as users desire, such as a *Duplicate* restraint combining the start-to-start and finish-to-finish restraints and similar to the old MSCS "Z" restraint. However, discussion of support for such extensions should be left for another day.

Where to Next

One of the disappointments of PDM has been the lack of standardization in notation and calculation algoritms such that the same information fed to two software products may provide differing results. In an effort to minimize this type of problem for RDM, a standard on both notation and calculation algorithms is being developed and will be administered via a Certification Trademark RDCPM™. Public comment upon the draft standards, as promulgated in this paper, are welcome, and should be sent to rdcpm@fplotnick.com.

Conclusion

The RDM or Relationship Diagramming Method variant of CPM can provide a robust and mathematically sound improvement to the field of Planning and Scheduling. It is currently being examined for implementation by software vendors, such as Primavera Systems.

Endnotes

1. CPM in Construction Management, 5th and 6th Editions, James J. O'Brien and Fredric L. Plotnick, McGraw-Hill, 1999 and 2005.

About the Author

Fredric L. Plotnick, Esq., PE, is CEO and principal consultant of Engineering & Property Management Consultants, Inc. He has bachelors and masters degrees in civil engineering and is a registered Professional Engineer in Pennsylvania, New Jersey and Florida. He is also an attorney and a member of the Bars of Pennsylvania, New Jersey, and Florida. Mr. Plotnick is an adjunct professor of the departments of Engineering Management, Civil Engineering and Construction Management at Drexel University, Philadelphia, Pennsylvania. He is a past president of the Philadelphia Chapter of the Pennsylvania Society of Professional Engineers, and a past Construction Group chair of the Philadelphia Section of the American Society of Civil Engineers. Mr. Plotnick is currently Director of Academic Liaison and Chair of the Technical Research Track of the annual conference of the PMI Project Management Institute's College of Scheduling, as well as a regular speaker for the PMI College of Performance Management and the AACEi Association for the Advancement of Cost Engineering International. Additional information may be found at www.fplotnick.com