Executing an Uncertain Schedule:
Adapting to Reality to Maximize Target Completion Attainment

Annaka Kalton, Technical Lead, Stottler Henke Associates, Inc.
Devin Cline, Stottler Henke Associates, Inc.

Introduction

Effectively executing a complex schedule remains a significant challenge in many domains. Executing to a high-quality schedule improves the likelihood of finishing on time, but simply working to that schedule in scheduled activity order without any adaptation makes end date overruns probable. This probability increases the more ambitious the schedule, the more resource-driven the model, and the higher the degree of activity duration variance.

A more ambitious schedule minimizes the slack and buffer available to protect the target end date. In a more resource-driven model, working to an existing schedule imposes an unnecessary degree of rigidity, because many activities are scheduled where they are because of resource contention, so following the schedule strictly may miss opportunities to regain lost ground. Finally, the higher the activity duration variance, the more likely it is that the actual work will diverge from the planned work.

All of these aggravating issues are present in the modern airplane assembly domain. Schedules tend to be extremely ambitious, because each additional plane represents a large gain in revenue; however, the penalty for missed delivery dates is similarly large. The activity networks are highly constrained, both temporally and in terms of resource requirements; this combination makes producing a good schedule in the first place a challenge, and the resource contentions make working to a static schedule needlessly brittle. Finally, the activities involved in airplane assembly tend to have a high degree of variance. This combination tends to make results extremely poor when the schedule is worked statically. A more dynamic approach would seem advantageous, but there are many ways in which this could be approached; some of these options will be explored in this paper.

In spite of these challenges, there is an advantage to the complex domain of airplane assembly manufacture: because of the high payoff involved a great deal of study has gone into the actual variation involved in different activities. This is useful for two reasons: first, it can be taken into account when actually determining how to execute the schedule; second, it can be used as a basis for a Monte Carlo empirical analysis of different approaches, because the data can be used to construct a more accurate probability distribution reflecting duration variance.

In this paper we will explore several options for how to manage the execution stage of airplane assembly; these options will include traditional assembly management and newer Critical Chain Project Management (Robinson, 2007) buffer-based techniques for assembly management. We will also consider the impact of rescheduling the remaining work more or less frequently, and the way these factors relate in models with different model characteristics.

Execution Management Methods

There are a number of methodologies for managing the execution of a complex schedule. We will focus on the execution methodologies that may require periodic reschedules in order to respond to work-to-date, and then analyze the results in order to determine the appropriate work order. We will focus on two primary categories of execution management: schedule based (working the activities in schedule order), and schedule analysis based (order the activities based on a secondary schedule analysis, and work them in the resulting order). Of the latter category we will primarily explore the range of methods suggested by buffer-based management that grow out of Critical Chain Project Management (CCPM) methodologies.

Schedule Based Management Methods

Traditionally, airplane assembly has been managed based on bar charts: the work is scheduled, and then all work is assigned to a slot representing a group of people (one for each shift) who have the appropriate skills. Optimally, it is equivalent to working to the Gantt chart; depending on the division of labor it can be considerably suboptimal when compared to working the Gantt chart, because if one person completes their work more rapidly, their next activity is already determined, and it may or may not be available to be worked. Another activity that is ready for work will only be worked when the person assigned to it is complete with his previous task. This results in some workers sitting idle when there is work that is within their capabilities waiting to be done – but assigned to someone else.

Originally published as a part of PMICOS 2008 Annual Conference

The advantage of this general family of methodologies is largely psychological: the workers can see where they are in the schedule and can see what is coming next. The major disadvantage is that if one group of workers is performing more efficiently than the rest, their superior performance will not necessarily improve the overall project end date, because they cannot take any extra time to assist sections of the schedule that are running behind. On the other hand, any delay in the schedule propagates itself through the inter-job dependencies, and because of the strict labor division it is difficult to catch up.

These problems are less prevalent if they are working to a Gantt chart (which is less constrained than a bar chart, since jobs are not pre-assigned to a specific person, and can be worked by whatever qualified worker is available at a given time), but is still present. This is because a delay in one job causes a cascading delay through the Gantt chart – even if the activity could have scheduled significantly later because it was resource contention, not temporal constraints, that determined its position. The Gantt chart approach loses much of this additional information, and hence much of the flexibility.

Analysis Based Management Methods

Many more recent execution methods are based on a secondary analysis of the schedule; the idea behind these approaches is that a more robust result could be obtained by analyzing something about the structure of the schedule and the model, and focusing on sections of the schedule that are most likely to cause the schedule's end date to slip.

In this paper we will be focusing on the family of analysis methods derived from CCPM analysis. This family of techniques is focused on protecting the project's critical chain: the tent pole in the schedule that prevents its flow time from being any shorter. It is effectively the schedule's critical path if temporary constraints are added to reflect resource contentions, producing a Partial Order Schedule (POS).

In order to protect the critical chain, a preliminary analysis step buffers the paths into the critical chain: effectively amalgamating contingency time for the series of activities feeding into the chain in order to make sure that delays to said activities do not impact the chain itself. The prioritization used in working the schedule is then based on any incursion into these buffers: the higher the percentage of the incursion, the higher the priority of the activities involved. Should this incursion transmit to the critical chain, causing incursion to the project buffer (the contingency time protecting the actual project end), those incursions become the highest priority.

The intuition behind this methodology is that you should be working the activities that are most risking your project end time; because the buffers represent contingency time usage, you should work those sections of the schedule that are most heavily impacting the available contingency time. The rationale behind the buffers themselves (rather than using a more direct delay metric) is that they gather the contingency for a number of activities together, preventing the direct buffering of each activity that would otherwise be necessary (Newbold, 1998). The problem with the latter technique is that it would look at buffer incursion myopically, not taking into account the larger set of circumstances (see Exhibit 1). The other problem with it is that it tends to over-buffer. Number aggregation theory indicates that the variability of a set of activities is lower overall than the combined individual variabilities (Cook, 1998).

Empirical Comparison of Execution Methods

In order to verify and clarify the strengths and weaknesses of each method we ran a series of empirical comparisons. We ran each of the methods (described in detail below) on two test files using randomized durations. Although the cases tested were not as extensive as could be desired, the results nonetheless give some material for consideration.

Random Duration Generation

The durations for each activity were based on the "safe" and "aggressive" durations that had been determined for the

Exhibit 1. This illustrates some of the problems involved in considering contingency time myopically: in such a situation (left), the system would respond to a small delay in both A and B by giving B precedence because its buffer is so small; however, the total buffer for each chain is actually the same, making such a choice incorrect in many situations. The diagram at right illustrates both that the chains should be equivalent as far as the buffering is concerned, and that amalgamated buffering has the advantage of requiring shorter buffers.

Originally published as a part of PMICOS 2008 Annual Conference

manufacturing process of each activity. The "safe" duration indicated the length of time in which an activity can be completed 85% of the time. The "aggressive" duration indicated the length of time in which an activity can be completed 50% of the time.

Based on previous work on the probability distribution involved in airplane assemble activities (Mattioda, 2002), we used a beta distribution with $\alpha = 1.5$ and $\beta = 3$. This approximates a log distribution, but has the advantage of finite end points (see Exhibit 2). Using this beta distribution (with A and B based on the safe and aggressive durations), we calculated a random duration for each activity. We ran multiple random simulations, but we used the same series of durations across methodologies for each run (e.g. if an activity received a random duration of 1:23 for the first simulation and 1:31 for the second, those durations would be used for the first and second simulations respectively for each of the techniques, the goal being to make the results as comparable as possible.

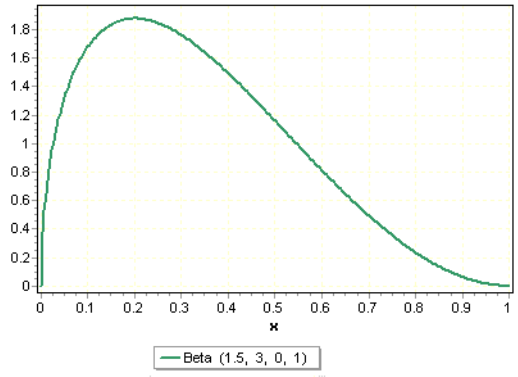

Exhibit 2. The beta distribution used to generate the random durations; its curvature approximates a log distribution but with the advantage of fixed end points.

Monte Carlo Simulation Process

Using the randomly generated durations described above, we simulated the execution process for each methodology by using the following sequence:

1. Generate a schedule.
2. Derive a prioritization based on said schedule.
3. Assign current set of randomly generated durations to all active activities (any activities that have not been completed in a previous cycle).
4. Using that prioritization, simulate the execution for a given time span by scheduling in priority order, taking resource requirements into account.
5. Record resulting actuals for those activities within the current update time frame (e.g. 48 hours).
6. Remove the priorities and revert the durations to standard for all activities beyond the current update timeframe (allowing them to again schedule freely in "planning" mode).
7. If all of the activities have been completed, report results; otherwise, return to step 1 and repeat.

This process effectively simulates the actual execution process, taking advantage of periodic updates to batch the scheduling updates necessary to determine what "actually" schedules when. This is possible because by scheduling in priority order the scheduler simulates a task assigner who selects the highest priority workable activity at any given point based on the current methodology; because it is in fact a scheduling system it enforces all resource requirements, and makes sure that two activities that require the same resource do not schedule at the same time. If they are both workable, the activity with the higher priority will always schedule first, and so be placed earlier in the resulting schedule.

Simulation Test Files

The test files were selected from two very different stages of the assembly process: one is a preparation stage, and so has fewer larger activities and is primarily driven by precedence constraints; the other models the final assembly process, which involves a very high number of activities, and is significantly driven by resource contention (see table below). The goal in the selection of these files was to contrast the strengths and weaknesses of the different approaches. Clearly it would be desirable to use a set of files representing different characteristics independently varied; unfortunately that was not possible because we were limited by the real data available for our use.

Originally published as a part of PMICOS 2008 Annual Conference

	File 1 Assembly Prep	File 2 Final Assembly
activities	106	1,763
resource requirements	525	7,451
constraints	214	4,839
% resource-based decisions	2.1%	36.4%

Detailed Execution Models – Schedule Based

We considered two similar schedule-based execution models. Their key difference was which set of durations – safe vs. aggressive – they made use of in generating the schedule that was then used for prioritization. In purely proportional data (e.g. aggressive duration = .75 x safe duration) these two approaches would have been identical; however, because of the projected activity variability the duration standard made a significant difference.

Schedule Based Model 1 – Aggressive Scheduling. For this model we produced a schedule using the aggressive durations. We then generated a series of priorities by sorting the activities first by scheduled start date and then by duration (such that shorter activities would schedule first in case of a tie), and used these priorities to simulate execution progress. This is equivalent to working an Aggressive duration Gantt chart from left to right as they become workable.

Schedule Based Model 2 – Safe Scheduling. For this model we produced a schedule using the safe durations. We then generated a series of priorities by sorting the activities first by scheduled start date and then by duration, and used these priorities to simulate execution progress. This is equivalent to working a Safe duration Gantt chart from left to right as they become workable.

Detailed Execution Models – Analysis Based

Because of the known limitations of the schedule-based models, our focus was on the analysis based models. We tested 6 different models. Most of them are buffer-oriented analysis models whose goal it is to protect the critical chain. The exception is based on slack, and as such it has a similar effect but without taking advantage of the relationship between safe and aggressive duration; it also distributes attention more evenly throughout the schedule, rather than focusing on the critical chain.

All of the buffer-oriented models used the same basic buffering strategy, described here; the differences lay in details of application and management.

We used the standard CCPM prioritization method (Newbold, 2001). We found the critical chain, and inserted feeder buffers based on the safe and aggressive durations at each of the junctions between non critical chain elements and critical chain elements. Temporary precedence constraints were generated running from the "feeder" element (the element feeding into the critical chain but not on the critical chain) and the buffer, and the buffer and the critical chain element. This allowed gaps to be created in the critical chain if necessary to schedule without initial buffer incursion. This primarily changes the results for models with a large number of low-slack chains (which is very characteristic of the second test file). Once this scheduling had been used to determine appropriate buffer placement we set the buffers' dates and removed the temporary constraints.

Once this initialization stage was complete, in subsequent prioritizations we considered buffer incursion (caused by up-stream slippage of activities) to determine priority. Because this step varied by method it is detailed more fully below. In all of the methods, any project buffer incursion was given higher priority than feeder buffer incursion. In the absence of any incursion information prioritization was based on activity start time + activity slack (giving an estimated latest start time).

Analysis Based Model 3 – Slack Analysis. For this model we produced a schedule using aggressive durations. We then generated a series of priorities by sorting the activities by their slack value (where slack was based on the actual schedule; so resource contention could also impact slack), with start time and duration (see above) used as tiebreaking values. This resulted in giving the priority to those activities with the least flexibility in their schedule location. Note that this resulted in priorities being distributed across activities both early and late in the schedule, but because ordering was determined by workability (all predecessors being done and having resources available), they were still worked in an order correlated with the scheduled order.

Analysis Based Model 4 – Standard CCPM with Default Project Buffer Placement. For this model we produced a standard CCPM buffered schedule. The Project Buffer was positioned at the end of the aggressive-time flow (resulting in immediate project buffer incursion if the critical chain slipped at all). Prioritization was then based directly on buffer incursion percentage (incursion time versus buffer length, where incursion time was considered to

Originally published as a part of PMICOS 2008 Annual Conference

be the projected incursion based on the current schedule – effectively how far the connected section of Gantt chart would overlap with the buffer's time), with all project buffer incursion being considered a higher priority than feeder buffer incursion.

Analysis Based Model 5 – Standard CCPM with Fenced Project Buffer Placement. In general this model operated in the same way as Model 4, except for the project buffer placement. In order to place the project buffer we first scheduled normally using safe durations. This gave us the projected end time for the schedule, and that end time was then used to fence the schedule and place the project buffer (in many cases this results in a gap between the critical chain and project buffer, but not always; if there are many gaps required in the critical chain to accommodate the feeder buffers, this space may be eliminated; this occurs in file 2).

Analysis Based Model 6 – Projected Completion CCPM. Traditionally CCPM does not take remaining chain length into account, which intuitively is problematic; it cannot differentiate between a buffer that is 99% consumed that still has 90% left to work, and a buffer that is 99% consumed that still has 1% left to work. In order to avoid this problem we developed a hybrid that, instead of simply using buffer incursion, uses a computation based on the new projected buffer need for the remaining chain:

$$\text{incursion} = \frac{\text{buffer incursion} + \text{projected buffer need}}{\text{original buffer size}}$$

In those cases when the result is greater than 1 (the projected buffer need is greater than the buffer available), the projected buffer need is used to approximate projected project buffer incursion, which is then used to prioritize the activity more highly. This reflects the idea that aggressive duration + buffer size = current estimated end. As such the prioritization should give the highest priority with the thing that would either push the project's projected end the most or, failing that, push the sub-chain's projected end the most.

Empirical Comparison Results

The results from the tests we conducted are summarized in Exhibits 3 and 4. Note that the results differ considerably between file 1 and file 2. This has to do with the basic file structures: as reflected in the file summary table, file 1 is primarily a temporally constrained file; only a small fraction of the activities are delayed because they are waiting for resources (on average 2.1%). What's more, file 1 is a fairly sparse file; aside from the clear critical chain, most of the activities have a great deal of slack (32% of jobs are within three hours of the critical chain, but only 36% – a four percent increase – are within 6 hours. This indicates a primary chain with a thin layer of dependents, while most jobs are relatively independent).

File 2 has a large number of constraints, but is much more heavily resource driven; on average 36.4% of its scheduling decisions are made because it is waiting for resource availability. It is also a very dense file (only 21% of

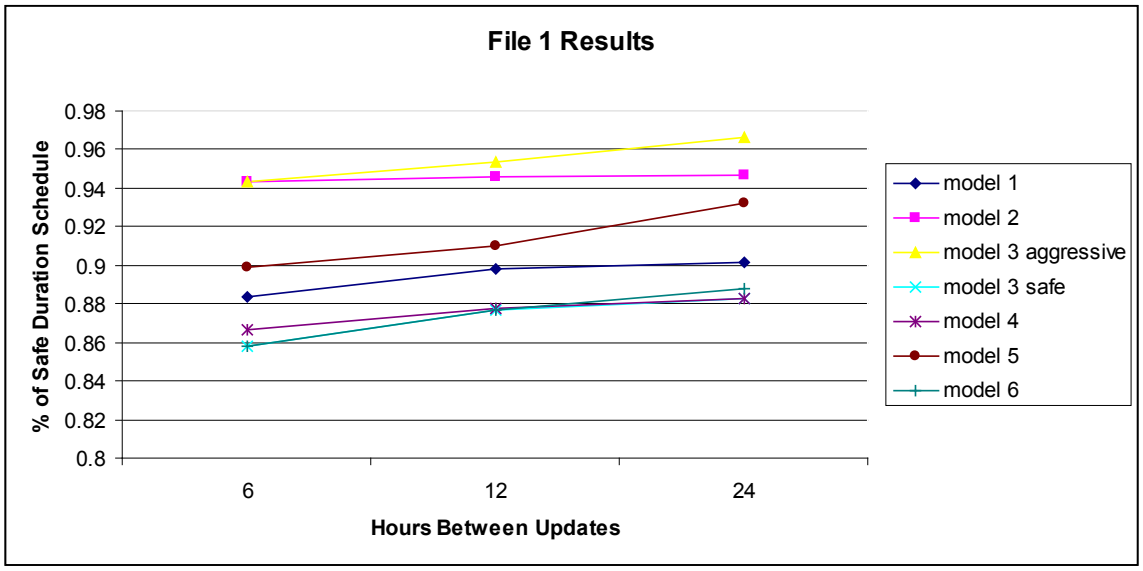

Exhibit 3. This graph shows the file 1 execution results; the lower the %safe duration, the shorter the final execution run. Interestingly the very simple model 3-safe (prioritize activities with lower slack based on a safe duration schedule) performed equivalently with the most sophisticated of the buffer schedules. File 2 differentiates them to a greater extent.

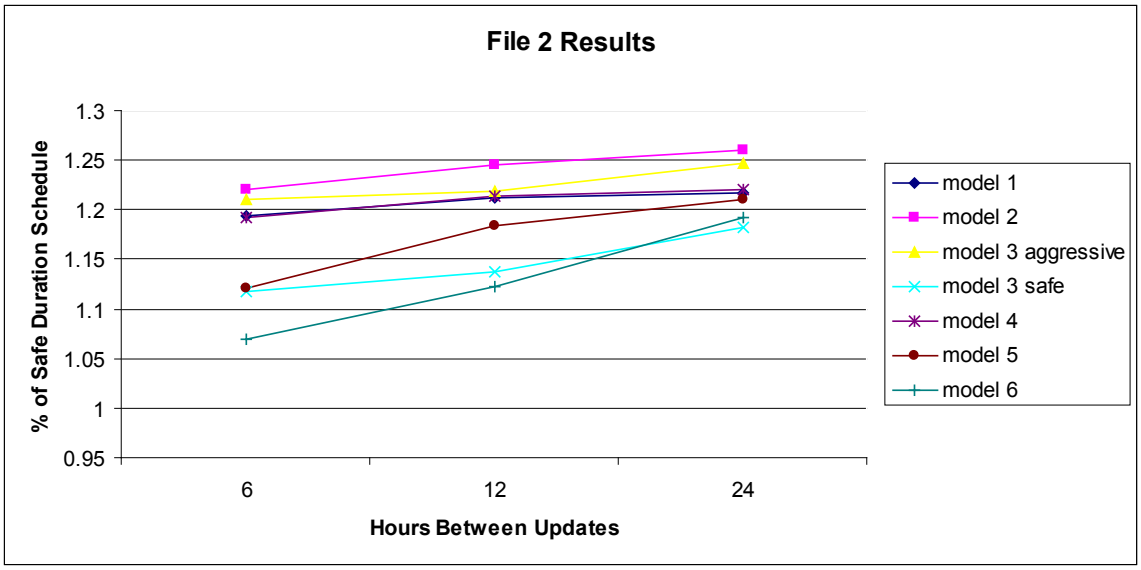

Exhibit 4. This graph shows the file 2 execution results; the lower the %safe duration, the shorter the final execution run. In this file the buffer-based models differentiated themselves to a greater extent, although the very simple slack-based model using a safe schedule was still quite competitive with the best performer. The buffer-based models tended to be more dependent on the update rate. Even the best execution models failed to meet the safe duration schedule flow time, suggesting a need for further improvement.

the jobs are within three hours of the critical chain, but 42% are within 6 hours and 59% are within ten hours).

Based on this characterization it is perhaps not surprising that the buffer management methodologies were little help in file 1; in most cases workability determined what to do next, and in those cases where there was a choice to be made, the slack-based model performed as well as the buffer-based models. One interesting aspect to note with this, however, is the disparity in result quality between model 3 aggressive (using the slack times based on an aggressive schedule) and model 3 safe (using the slack times based on a safe schedule). If the variance had been standard (e.g. aggressive = .75 x safe) this disparity would not have been present. As it is, it is probably indicative of the fact that higher variance tends to show up in the safe durations (e.g. there may be two activities with the same aggressive duration, but radically differing safe durations; this reflects highly differing probabilities, and it is to be expected that better results would be obtained using the more pessimistic view).

In the heavily resource-driven file 2, however, the advantages provided by the buffering models is clear. Model 6 – the project completion based model – performs especially well, in part because it handles differing variances effectively by taking them into account by using the buffer creation to estimate the new end time; the buffer creation process takes the relationship between the aggressive and safe durations into account, allowing the model to keep the schedule on track in spite of differing variabilities.

In all cases more frequent updates resulted in better performance, although this made a bigger difference in the larger file. This makes sense because in between updates the prioritizer is effectively behaving statically; it cannot take intermediate changes into account. The update cycle also had more effect on the buffer-based models, at least in the denser file 2.

The other aspect worth noting is that while all of the techniques finished within the original safe schedule time on file 1, none of the average execution times fell within the original safe schedule time in file 2. This is due in large part to the density of file 2, and suggests that a methodology to take that density into account more explicitly might prove effective.

Related Work

Successful project execution has been addressed by a number of researchers from a variety of disciplines, including project management and planning and scheduling.

CCPM grew out of the Theory of Constraints (TOC), originally developed by E.M. Goldratt as a way of overcoming some of the limitations of Critical Path Method (CPM), which has been the method most widely used in industry (Newbold, 1998). The problem with CPM was that it did not take resource contentions into account in any very useful way, and so tended to fail in highly resource-constrained domains (Swartz, 1999). Since its development to

address management under highly resource constrained circumstances, it has successfully been applied to a variety of domains, including semiconductor development (Harris, 2001) and vibration and noise control systems for aerospace (Lord, 2001). The primary drawback of previous CCPM research is that it has largely been applied to relatively small and simple domains.

Scheduling work with project execution has been more focused on schedule maintenance and consistency (e.g. Cesta and Rasconi, 2003). This is especially important for schedules that have extremely short update turnaround and/or cannot satisfy all activities. In such a situation the goal is not to finish a project at a specific time, but to update the schedule to accommodate the new high-priority activities.

Another angle attempts to both provide efficiency and consistency by rescheduling based on the POS (Policella, 2005). This has major advantages when a new schedule is needed very quickly, or if there is a significant advantage to changing the schedule as little as possible (e.g. if the original schedule is known to a number of people); it can be less helpful in something such as airplane assembly where very often radical changes to the schedule are necessary to respond to delays.

Conclusions

As expected, the buffer model capable of taking remaining work as well as buffer incursion into account overall performed best; however, the good results obtained by the very simple slack-based model would seem to merit further investigation. Given that the slack-based model performed far better with the safe duration schedule (probably because it could better differentiate high-variance activities), it might be interesting to attempt to explicitly take variance into account in the slack computation.

The other result that would merit further research is the fact that in the dense file 2 none of the execution models were able to meet the safe duration schedule. This implies that none of the models are able to adequately capture the way in which the higher density raises the model's risk. It is possible that a different buffering scheme, capable of taking non-chain activities into account in its computations, might be able to better accommodate such a schedule. The key question is whether such a safe schedule is, in fact, attainable for such a file. It seems likely that with the correct triggers to differentiate among many very similar activities it should be possible, but such differentiation would make over-fitting to that schedule model a potential hazard. Regardless, some sort of density accommodation would seem to be highly desirable. Another possible approach would be multiple-layer buffering, rather than simply buffering the critical chain. This could provide better feedback for progress or problems within the dense cloud of activities that make up the model. This also has the advantage of being potentially more easily extensible to new domains, especially if they could be applied based on a density or layering criteria, rather than with a static limit of 1 (the current standard), 2, or 3.

References

Cesta, A. and Rasconi, R. (2003, September). Execution Monitoring and Schedule Revision for O-OSCAR: a Preliminary Report. *Proceedings of Online-2003 (Online Constraint Solving: Handling Change and Uncertainty), A CP-03 Workshop, Kinsale, Ireland, September 29*, C. Beck, K. Brown & G. Verfaillie (Eds.).

Cook, S. (1998). Applying Critical Chain to Improve the Management of Uncertainty in Projects. MS thesis, Sloan School of Management and the Department of Electrical Engineering and Computer Science, MIT, 1998.

Harris Semiconductor (2001). Avraham Y. Goldratee Institute: http://www.goldratt.com/ harriserp.htm.

Lord Corporation. (2001). Multi Project Management. Avraham Y. Goldratt Institute: www.goldratt.com/ lord.htm.

Mattioda, D. (2002). *Use of Critical Chain Scheduling to Increase Aircraft Availability*. Wright-Patterson Air Force Base, Ohio.

Newbold, R. (1998). *Project Management in the Fast Lane; Applying the Theory of Constraints*. Boca Raton Florida: St. Lucie Press.

Newbold, R. (2001). *Project Management for Business and Technology: Principles and Practice*. Upper Saddle River New Jersey: Prentice Hall.

Policella, N. (2005). Scheduling with Uncertainty: A Proactive Approach Using Partial Order Schedules. *AI Communications 18*. IOS Press.

Robinson, H (2007, April*). An Introduction to the Theory Behind Goldratt's Critical Chain Project Management.* Project Management Institutes College of Scheduling Conference (PMICOS) 2007, Vancouver, Canada.

Swartz, S. (1999). *The Effects of Variability and Disruption of Project Stability, Duration, and Net Present Value*. East Lansing Michigan: Michigan State University.

Originally published as a part of PMICOS 2008 Annual Conference