Enhanced Critical Chain Project Management via Advanced Planning & Scheduling Technology

Annaka Kalton, Project Manager, Stottler Henke
Hilbert Robinson, Lead Consultant, Afinitus Group, LLC
Robert Richards, Project Manager, Stottler Henke

Abstract

In the late 1990s, a new project management methodology was developed to overcome many of the shortcomings of Critical Path Project Management (CPPM). This new methodology, *Critical Chain Project Management* (CCPM), is based on methods and algorithms derived from the theory of constraints. The Critical Chain is the sequence of both precedence- and resource-dependent tasks that prevents a project from being completed in a shorter time, given finite resources. If resources (labor/non-labor) are always available in unlimited quantities, then a project's Critical Chain is identical to its critical path. Since the introduction of CCPM, its use has grown considerably and is being applied in more, and more challenging, project management environments.

The Critical Chain method, however, is still in its relative infancy and many limitations have been discovered with the current theory and implementation, especially as it has been applied to ever larger project management tasks. Limitations in theory and its current commercial implementations have limited its suitability for large, dynamic, and complex scheduling environments, such as those that have been the focus of advanced scheduling techniques in the past. Many of these limitations can be surmounted by combining the principles of the CCPM methodology with advanced scheduling techniques. This paper investigates how many of the perceived limitations of the Critical Chain method can be overcome and the results maximized by leveraging the power of advanced planning and scheduling.

By combining CCPM theory with a sophisticated scheduling technology that combines a variety of scheduling techniques and intelligent conflict resolution, a much more robust solution is possible. Resource constraints, including available equipment, space, and human resources, are very important to the implementation of the Critical Chain method because the greater the degree to which it can take these resources into account, the greater the potential improvement over CPPM.

Fortunately, advanced scheduling technology can take into account a variety of resource requirements, and can be adapted to pertinent domain knowledge. This is especially important during the initial Critical Chain analysis step (where the chain length is directly related to schedule quality), and in the execution phase of a CCPM plan, as real-time updates arrive regarding the status of the hundreds or thousands of tasks that make up the plan.

A more sophisticated underlying scheduling framework provides two important benefits: it results in a better Critical Chain (one that can potentially be worked in less time), and it is more flexible and better able to accommodate change. We have been able to greatly enhance the practicality and power of the Critical Chain method by leveraging advanced planning and scheduling techniques. By using sophisticated scheduling software as the underpinnings for Critical Chain reasoning, the Critical Chain method can be applied to projects encompassing thousands of heavily constrained tasks and requiring hundreds of different kinds of resources. Giving the Critical Chain method such a solid scheduling basis also allows it to more easily handle complex situations such as new tasks being inserted during the actual plan execution, as well as other radical changes to the situational model.

Introduction

Large organizations dealing with developing and building complex systems are highly reliant on schedules and project management techniques at almost every level. A number of tools, technologies, and methodologies have been put forward in an attempt to deal with these issues. CPPM emerged as an early leader in project management methodology, but it did not provide nearly as much gain in highly resource constrained domains; CCPM originally emerged to help fill this gap.

Originally published as a part of PMICOS 2007 Annual Conference

The Project Management community was broadly introduced to the Critical Chain Project Management (CCPM) by Eliyahu M. Goldratt's book, *Critical Chain* (Goldratt, 1997). CCPM is further described in both *Project Management In the Fast* Lane (Newbold, 1998) and *Critical Chain Project Management, 2nd Edition* (Leach, 2005). CCPM considers resource constraints more heavily than does Critical Path Project Management. A Gantt chart of a small Critical Chain project is shown in Figure 1; CCPM has the concept of buffers as shown in the figure and further described in the above mentioned references. Many sources, including The Standish Group International, www.standishgroup.com, and The Goldratt Institute, www.goldratt.com, have shown that over 50% of Critical *Path* base projects finish late with the average duration being more than 50% greater than originally planned; CCPM has shown much improved on-time performance.

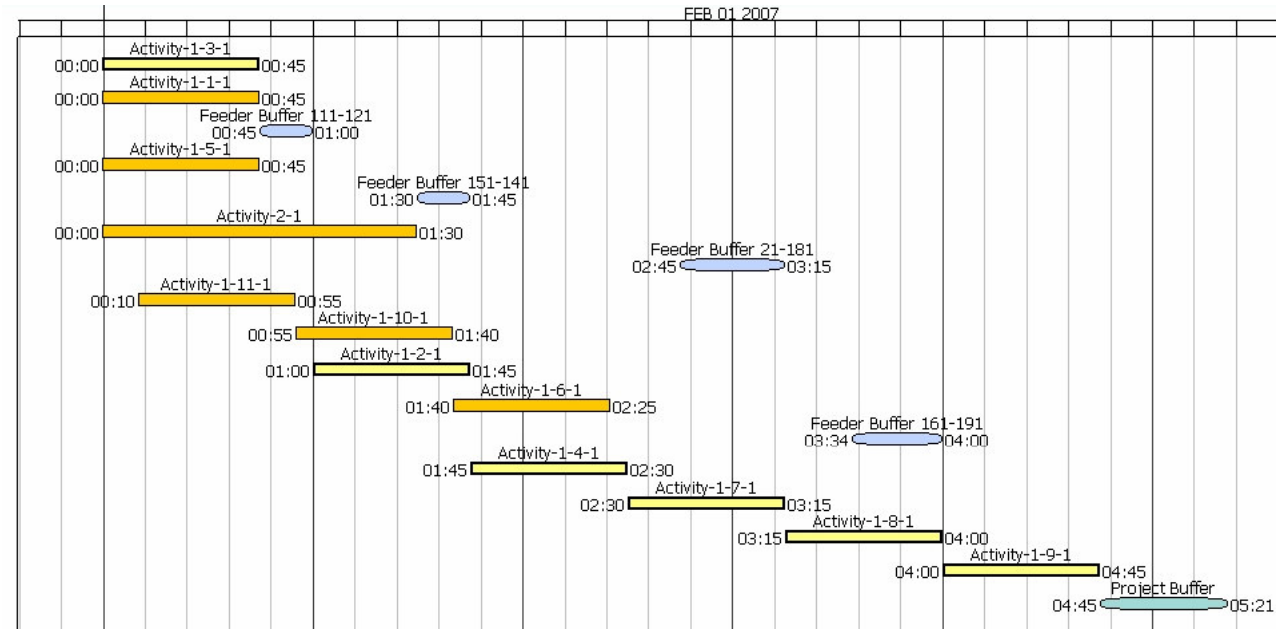

Figure 1. Gantt Chart showing Critical Chain in yellow

Even though CCPM has shown great success to date, many limitations have been discovered with the current theory and implementation, especially as it has been applied to ever larger project management tasks. Because the critical chain takes into account resource contention, the quality of CCPM results are dependent on the underlying scheduling solution, even though the goal of CCPM is not to work to a schedule. The initial schedule length becomes the length of the critical chain - and this is central to the way in which CCPM will determine what should be worked when. Figure 2 illustrates the potential impact of different critical chains, as well as showing the advantages of using the critical chain in such a resource-constrained domain. It includes the critical path, for reference, and two different critical chains, each with elements driven by resource contention shown in a darker shade. Clearly the critical chains differ dramatically. If an inefficient scheduling method is used the unnecessarily long critical chain will give an erroneous impression of the time in which the project could potentially be completed, and so would not be managed as ambitiously as it could successfully be. Of equal importance is the ability to model accurately: a shorter critical chain is not helpful if it is not, in fact, possible to work the project in that time because a number of constraints were neglected or misrepresented.

The goal is the shortest accurate critical chain, because that will permit CCPM's application to result in successful project completion in the shortest amount of time.

Both scheduling engine sophistication and modeling accuracy have historically been problematic when applying CCPM. Many projects can be modeled with labor and precedence dependencies, and these have been the main type handled via the CCPM method to date, because these can be readily modeled with many of the available tools. As the project becomes more resource dependent and the types of resources increase, the current solutions perform less

optimally, often because they lack the flexibility to accurately and effectively model and schedule these additional requirements. For example, many projects depend on certain operations occurring at specific locations (spatial resources), and being done with a certain set of equipment. There may be safety and environmental restrictions that vary depending on what else is occurring, and other domain specific constraints. What is more, there are often contradictory scheduling goals (e.g. to minimize manpower but also minimize flow time); handling these tradeoffs is an extremely domain-specific consideration. Finally, because a complex schedule can be so large (having to take thousands of elements, requirements, and constraints into consideration), the system must be able to use heuristics to render the schedule search feasible. Again, this is a highly domain-specific process.

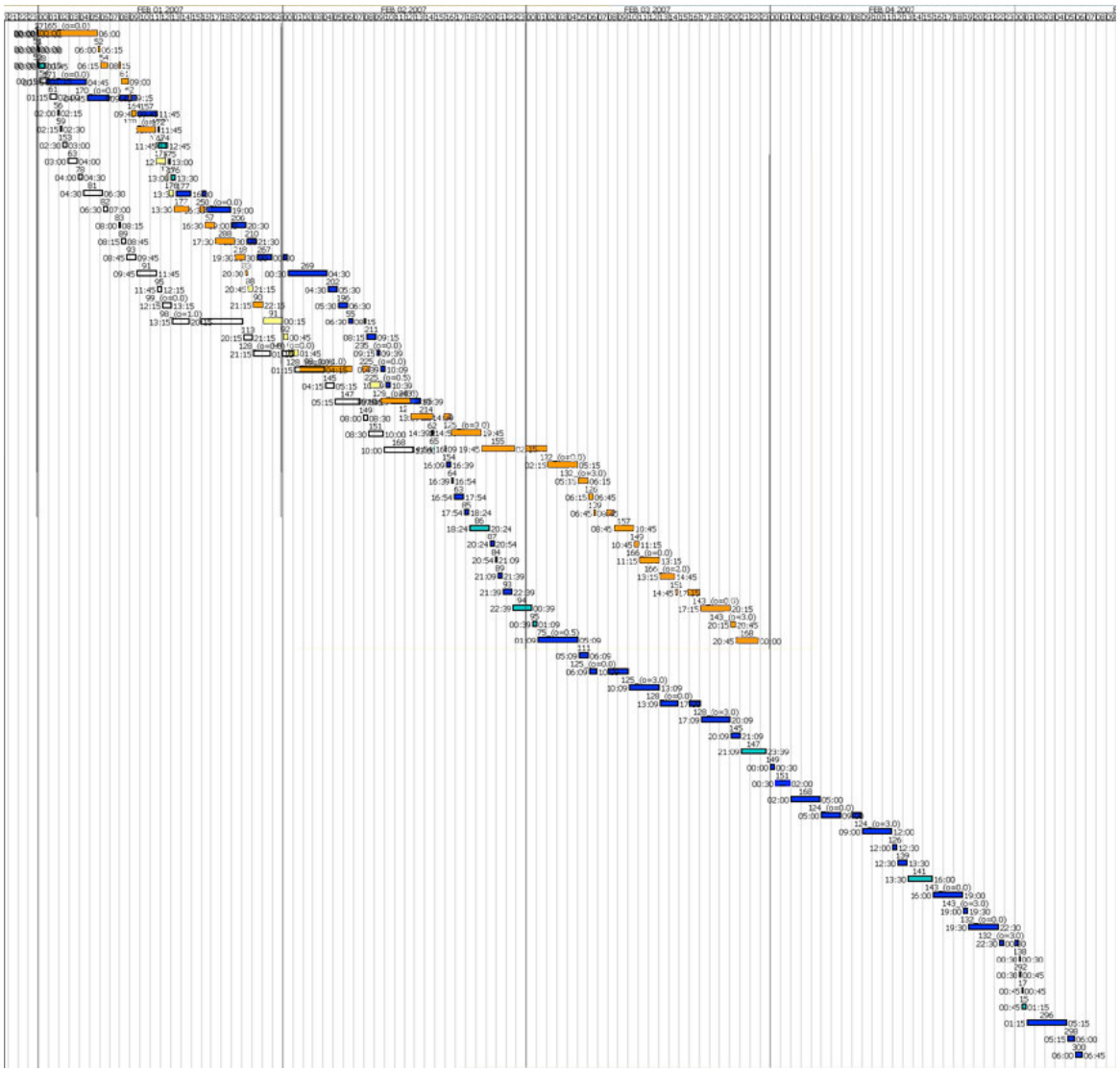

Figure 2. Critical Chain Comparison

For all of these reasons it is important to have a high-quality scheduling engine - capable of handling the modeling complexity and scale - and ideally a scheduling engine which appropriately reflects the preferences for a given domain. In combination these aspects will result in a short, high-quality, workable schedule - which can then be used as the basis for critical chain analysis and the basis for far more flexible CCPM in execution.

Thus it is important to pair a sophisticated scheduling engine with CCPM methodologies to gain the best possible management results.

Originally published as a part of PMICOS 2007 Annual Conference

Current Limitations of CCPM Implementations

In general, larger projects have more constraints than simply labor and precedence. Eli Goldratt's book, *Critical Chain* (Goldratt, 1997) presented Critical Chain using only these constraints; and thus most of the Critical Chain Implementations have focused (solely) on these types of constraints. However, the concepts and benefits of CCPM work just as well when other types of constraints are introduced.

Relationship complexity often goes beyond precedence and concurrency and non-concurrency. Often domain dependent relationships need to be handled correctly and efficiently if an efficient Critical Chain schedule is to be generated. Another concept that occurs in complex projects but is not normally handled is that of variable capacity by time interval; for example, repeating a pattern over a 24 hour cycle (e.g. standard factory staffing will vary from shift to shift), or a change in pattern for two days out of every seven (e.g. most organizations do not work over the weekend, or work to reduced hours). These are fairly standard variations, but some domains may vary in a number of additional ways.

In a domain such as spacecraft preparation, there are a number of additional safety considerations; or in a domain such as airplane assembly, there are a number of space-related issues (only so many workers will fit in a given space, and some actions may permanently eliminate possible work space). Many other domains, such as the building and maintenance of submarines, tanks, helicopters, and oil drilling platforms, offer yet more considerations that cannot be accurately considered in a basic scheduling system.

The challenge is increased when 24 hour operations are introduced, so that the orchestration of many activities with a high degree of variation and intense coordination must occur in near real time. Also near real time schedule maintenance needs to occur per adding and removing work statement components on an emergent basis, both to incorporate rework, as well as to accommodate late breaking changes in the engineering requirements.

Advanced Planning & Scheduling

To illustrate the benefits of advanced scheduling, the lessons learned from a particular scheduler, *Aurora*, are utilized. Aurora evolved out of the needs of NASA and later the United Space Alliance, and finally in application to industry. Aurora evolved over many generations, and the latest generation is the result of a major re-design, where Stottler Henke systematically looked at every planning and scheduling system Stottler Henke had ever developed, and looked at all the decisions that a planning and scheduling has to make and designed and implemented an architecture such that it was easy to customize every one of those decisions. This latest evolution has been chosen by the United Space Alliance as the onboard planner / scheduler for astronauts to use on the Crew Exploration Vehicle, and is also in operational use in other assembly scheduling and project support.

Aurora is used in the planning and scheduling of extremely complex processes involving thousands of operations. Each operation can require a combination of resources (facilities, equipment, personnel). Aurora is adaptable to different domains that each have their own set of additional constraints, examples include the safety limitations and floor plan layout coordination involved in preparing components for the space station; non-concurrency and offset constraints to allow necessary safety and practicality controls over scheduling astronaut time; and space constraints, including addition and removal of these constraints, involved in airplane assembly. Finally, Aurora has evolved to meet the real world challenges of endless changes to the schedule caused by late deliveries, other delays (e.g., launches), and malfunctioning equipment.

Some of the features that have evolved are listed below. Most, if not all, should be part of any advanced scheduler used in complex CCPM projects, because they result in greater inter-domain flexibility and more accurate modeling.

- Tasks, Flows & Resources – Aurora offers three main kinds of scheduling elements: 1) tasks (activities); 2) flows, which represent groups of tasks; and 3) resources. In addition to these primary types there are also resource sets, which can be used to group resources, and constraints, which allow the user to create relationships among elements. Constraints and Resource Sets are described in more detail below.

- Constraints – Aurora provides temporal, resource, capacity change, and spatial constraints to define the relationships among the scheduled elements. Temporal constraints specify what the temporal relationship of two elements should be. Resource constraints indicate that two elements should use the same resource. For example, one could specify that the same worker perform steps one and three of a processing procedure. Capacity change constraints indicate that a given task has an impact on a resource's capacity (e.g. contributes capacity or removes capacity). Finally, spatial constraints allow the user to specify that two elements should (or should not) be next to each other, or in the same spot.
- Resource Sets – Aurora allows for the grouping of related resources into different sets to take advantage of all useful attributes. Aurora allows the user to group resources arbitrarily into resource sets. Elements can then request these sets as part of their resource requirements, reflecting the idea that any properly qualified resource could perform a corresponding task. This also gives the user the freedom to group one resource differently in different situations.
- Resource Requirements – Aurora allows the user to associate resource requirements with any task, flow, or resource. The first reflects the case where a single task requires a resource; the second reflects the case where a full set of tasks needs to use one resource (e.g., a project needs a project manager); the third reflects those occasions when a resource needs another resource to operate properly (a large, specialized piece of lab equipment needs lab space; an engine needs fuel). These requirements may be defined directly in terms of resources, when only one resource can satisfy the requirement; or in terms of resource sets, when any of a group of resources may satisfy the requirement. It also allows you to designate alternate ways in which the set of resources may be satisfied. For example, a training course might require an experienced instructor for the duration, or a regular instructor with half-time support from a specialist.
- Reports – Aurora allows the user to create various textual and graphical reports. For example, the resource display shows what tasks are using a given resource at any given time, see Figure 3. This permits easier analysis of results, which can motivate model correction and improvement, and possibly further domain-specific extension.

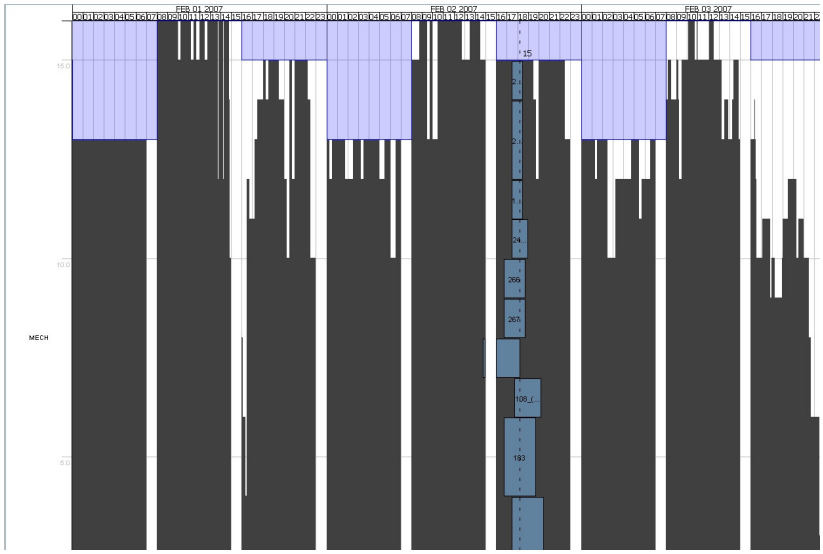

Figure 3. Resource Display

- Calendars – Aurora permits the user to associate a calendar with a task or resource to dictate its standard schedule, and any exceptions that schedule might have. These may include yearly holidays or one-time events. A "five day" activity will likely take very different amounts of calendar time if it is scheduled in late December than if it is scheduled in February. Aurora can dynamically cross-reference these calendars in the course of scheduling; for example, a task that requires a mechanic who is available for two shifts a day will inherit this calendar; if it itself had a daytime only calendar, it would be assigned a calendar reflecting the intersection of the daytime calendar and the mechanic's shift calendar.

- Conflict Viewing – Aurora can usually resolve all conflicts, but sometimes a schedule is over-constrained, resulting in a schedule with one or more conflicts. Any such elements are displayed in red, so the user can see and deal with them. However, there may be instances where they want a global view of all conflicts in the schedule. The user can see these using a conflict display window, see Figure 4. The conflicts are broken down first by resource, and then by time frame.

- Ignoring Conflicts – There are occasions when there is a conflict, but the user knows it is not a "real" conflict - they know that a vendor's delivery is going to be late, or that two classes can, in this situation, use the same room intermittently. In such a situation, the user can specify that a conflict be "ignored". It will no longer be displayed as a conflict, and the conflict manager will not try to resolve it.

- Default Scheduling – Aurora offers a default quality and prioritization scheme that has proven itself across a variety of domains. This works well in most cases, and is useful for those situations where it is undesirable or impossible to construct scheduling heuristics customized to the domain at hand. It considers all possible resource allocations before it begins scheduling, allowing it to pinpoint possible trouble spots - the resources with the most elements vying for their potential use. It then works around these bottle necks, scheduling the elements away from them as much as possible, and leaving only those elements that must use them.

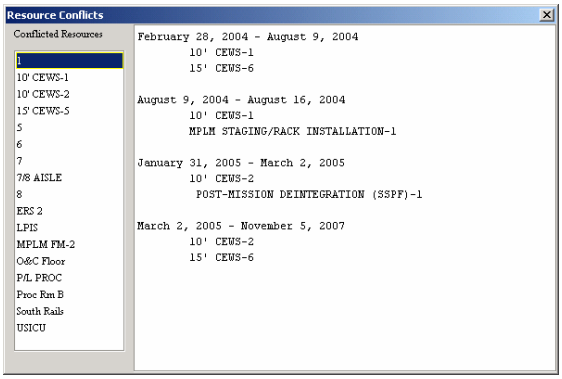

Figure 4. Resource Conflicts

- Customization – Aurora is designed for extensive customization. Needs and priorities vary widely from one company to another, even within a single domain, and the program needs to be able to reflect that. This customizability also allows the program to take expert domain knowledge into account, because this knowledge can easily be encoded into the heuristics that the system relies upon to make its decisions. As mentioned, the entire architecture was developed for customization. Much customization can be performed without programming; however, when customization requires additional programming the design makes this simpler than if customization was not designed in.

  For example, the user interaction (the GUI) is a separate component from the Aurora scheduling engine. This allows for a whole new front-end to be substituted. The front-end could be replaced with that of other commercial Critical Chain solutions, allowing the user to retain a user interface they are familiar with while benefiting from the Aurora scheduling engine. Figure 5 shows Aurora's high-level architecture, revealing the module design and the separation of the front end from the scheduling engine.

- Scheduling Heuristics - In order to find a high-quality schedule quickly, it is necessary to use a battery of heuristics. These heuristics are usually tailored from one domain to another, to get the best solution for a given class of problems. For example, a more temporally-based model (with a lower resource requirement to temporal constraint ratio) can be scheduled effectively by taking the critical path into account, and scheduling elements on or near the critical path earlier in the process; however, this may be quite damaging in a more heavily resource-restricted model. Because of this the best scheduling systems will always require some degree of tuning, and a good engine should permit some degree of adaptation. In Aurora's case, different heuristics impact each stage of the scheduling process, and may be changed independently for a new domain.

To summarize, Aurora is a sophisticated scheduling system that combines a variety of scheduling techniques, intelligent conflict resolution, and decision support to make scheduling faster and easier. The software's scheduling decisions take into account domain knowledge, any number of constraints, and resource requirements. Once Aurora has created a schedule, it displays it in a series of graphical displays that allow the user to see the resource allocations and the temporal relationships among the elements. Aurora focuses on resource requirements and temporal scheduling in combination.

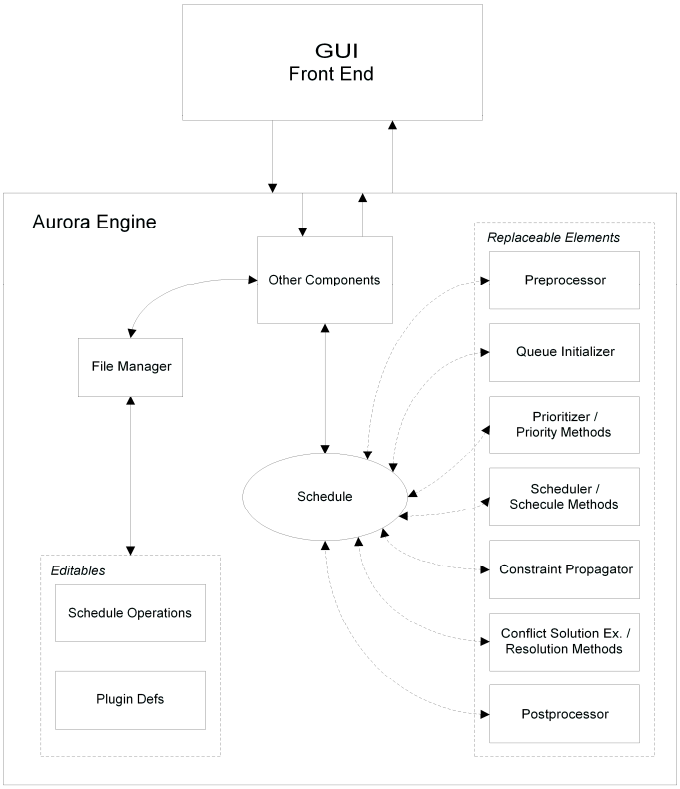

Figure 5. Aurora Scheduling Engine Architecture

Combining CCPM and Advanced Planning & Scheduling

The effects of the underlying scheduling engine on the Critical Chain are shown graphically in Figure 2 and Figure 6. Figure 6 shows the results of four (4) different programs' determination of the Critical Chain elements. Note that every program correctly built a Critical Chain schedule in every regard except global optimality. For larger problems the problem of determining the globally optimal solution is computationally intractable, so all solutions try to find a good approximation of a global solution in a user-acceptable amount of time; this will not necessarily be the best possible solution.

Even in the small project shown in Figure 6, the duration difference between the shortest Critical Chain and the longest is 100 units **of time** versus 110 time units, already a 10% difference. As the size of the project increases the time for the project can differ by over 100% between tools. Furthermore, Aurora has been able to find project durations 50% shorter than other Critical Chain solutions, for the same data set.

This implies that much of the potential for improvements offered by the Critical Chain method are possibly being lost if the project is not being scheduled efficiently. These already important considerations are only for the earlier stages of CCPM. It can become even more important during CCPM execution.

In addition to the challenges of finding a good schedule solution during planning, it is often necessary to incorporated changes to the schedule during execution. In real world situations unexpected events occur during

execution, including insertion of new tasks, deletion of old tasks, and radical changes in the timing of specific tasks. Many of these events require partial re-scheduling as the basis for the next round of analysis, and if these reschedules could be improved, so could the overall execution of the living project.

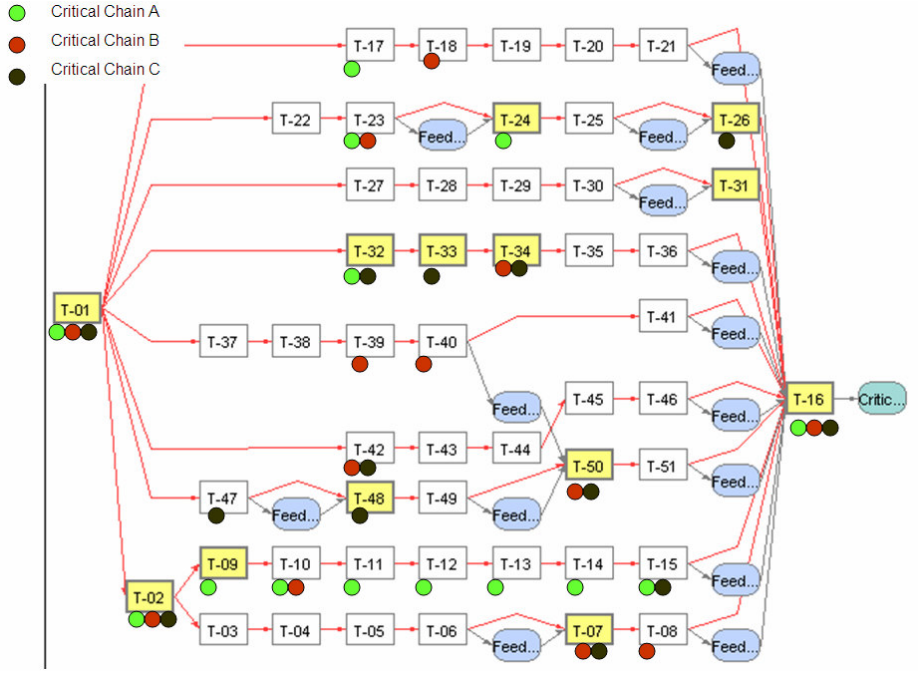

Figure 6. Critical Chain Determination by Four Different Tools

Conclusions

In this paper we have shown that Critical Chain Project Management (described elsewhere) is greatly affected by the underlying scheduling engine - more so as the project becomes larger and includes larger numbers of resource requirements and other non-temporal constraints.

The results of applying the Aurora scheduling engine to Critical Chain problems was used to demonstrate the described effect; however, Aurora is just an example - the primary conclusion is that the underlying scheduling engine can greatly impact the results. History has proven that it is nearly impossible to build a scheduling solution that is best in all situations, so another potentially beneficial approach would be to maintain a pool of possible scheduling engines or engine configurations, and apply all of them to models in a new domain. Because of their differing strengths and weaknesses, some would perform more effectively on some domains than in others. Once all had been applied to a given model or set of models, the best engine for the purpose could then be selected for subsequent CCPM application. It is usually easy to select the best schedule, as it is generally the schedule with the shortest flow time. If possible, the best solution could then be further tailored to maximize the benefit, but the key point is that the scheduling system has a significant impact on the utility of applying CCPM to a given domain, and should be given corresponding weight.

References

Goldratt, Eliyahu M. (1997) *Critical Chain.* Aldershot, Hampshire, United Kingdom: Gower Publishing Limited.
Newbold, Robert C. (1998) *Project Management In the Fast Lane*. Boca Raton, FL: CRC Press.
Leach, Lawrence P. (2005) *Critical Chain Project Management, 2nd Edition*. Norwood, MA: Artech House.

Originally published as a part of PMICOS 2007 Annual Conference